Learning Control-Oriented Dynamical Structure from Data

Spencer M. Richards [1] **Jean-Jacques Slotine** [2] **Navid Azizan** [3] **Marco Pavone** [1]

Abstract

Even for known nonlinear dynamical systems, feedback controller synthesis is a difficult problem that often requires leveraging the particular structure of the dynamics to induce a stable closed-loop system. For general nonlinear models, including those fit to data, there may not be enough known structure to reliably synthesize a stabilizing feedback controller. In this paper, we discuss a state-dependent nonlinear tracking controller formulation based on a state-dependent Riccati equation for general nonlinear control-affine systems. This formulation depends on a nonlinear factorization of the system of vector fields defining the control-affine dynamics, which always exists under mild smoothness assumptions. We propose a method for learning this factorization from a finite set of data. On a variety of simulated nonlinear dynamical systems, we empirically demonstrate the efficacy of learned versions of this controller in stable trajectory tracking. Alongside our learning method, we evaluate recent ideas in jointly learning a controller and stabilizability certificate for known dynamical systems; we show experimentally that such methods can be frail in comparison.[1]

1. Introduction

Data-driven system identification and control algorithms are imperative to the operation of autonomous systems in complex environments. In particular, model-based algorithms equip an autonomous agent with the ability to learn how it and the system it is part of evolve over time. However, for general nonlinear systems including those learned from data, it is not always clear how to synthesize a *stabilizing* tracking controller. Effective control design often leverages specific system structure; some classic examples of this are the linear quadratic regulator (LQR) for linear dynamics, and the computed torque method and its variants for Lagrangian dynamics (Slotine & Li, 1987; Murray et al., 1994). A central goal of *control-oriented learning* (Richards et al., 2021; 2023) and this paper is to jointly learn a dynamics model and additional *control-oriented* structure that naturally encodes or reveals a stabilizing controller design.

Related Work An approach favoured by recent works has been to learn stabilizing controllers for nonlinear system models by simultaneously learning a parametric controller and a parametric control-theoretic certificate, such as a control Lyapunov function (CLF) or control contraction metric (CCM). This paradigm originates in works that learn stability certificates for nonlinear systems of the form $\dot{x} = f(x)$ or $x_{t+1} = f(x_t)$. Convergence of the state to $x = 0$ is guaranteed if a Lyapunov certificate function V can be found such that $\nabla V(x)^\mathsf{T} f(x) < 0$ or $V(f(x)) - V(x) < 0$, respectively, for each $x \neq 0$. Such functional inequalities serve as the cornerstone for methods that learn parametric certificates from data either via gradient descent on a loss function comprising sampled point violations (Richards et al., 2018; Boffi et al., 2020), or formal synthesis and verification (Abate et al., 2021). Similar functional inequalities appear in contraction theory (Lohmiller & Slotine, 1998) to describe the convergence of system trajectories to each other over time, and have been used in imitation learning to regularize fitted dynamics models towards stability (Sindhwani et al., 2018) or intrinsic stabilizability (Singh et al., 2021).

For *controlled* nonlinear systems like $\dot{x} = f(x) + B(x)u$, one can try jointly learning a parametric CLF V and parametric controller $u = k(x)$ by penalizing violations of the inequality $\nabla V(x)^\mathsf{T}(f(x) + B(x)k(x)) < 0$ at sampled states. This concept underlies most prior work on learning certified stabilizing nonlinear controllers (Chang et al., 2019; Chang & Gao, 2021; Dawson et al., 2021; 2022). For

[1]Autonomous Systems Laboratory (ASL), Stanford University, Stanford, CA 94305, USA [2]Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA [3]Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Correspondence to: Spencer M. Richards <spenrich@stanford.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]We provide code to reproduce all of our results at: https://github.com/StanfordASL/Learning-Control-Oriented-Structure.

tracking a trajectory $(\bar{x}(t), \bar{u}(t))$, Sun et al. (2020) jointly learn a CCM and a feedback controller $u = \pi(x, \bar{x}, \bar{u})$, again based on sampled inequality violations. Such approaches aspire to the closed-loop stability promised by satisfaction of this infinite dimensional constraint, yet it is unclear whether penalizing violations at a finite number of points is sufficient to achieve this in practice.

Rather than trying to fit a controller and certificate to data, one can leverage structure in the dynamics to inform stabilizing controller design. Lagrangian dynamics of the form $H(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = u$ with state $x := (q, \dot{q})$ are amenable to feedback linearization (Slotine & Li, 1991) by virtue of their double-integrator form, even when learned from data (Gupta et al., 2020; Richards et al., 2021; Djeumou et al., 2022). Hamiltonian dynamical structure as a physics-based prior in learned models can be exploited to synthesize passivity-based controllers (Zhong et al., 2020; Li et al., 2022). Perhaps the most fundamental example of structure informing control is LQR, which for linear dynamics $\dot{x} = Ax + Bu$ computes an optimal stabilizing controller from a Riccati equation using the system matrices (A, B) and chosen cost matrices (Q, R). Each of these designs is tailored to a subset of control-affine dynamical systems, yet LQR can be extended to general control-affine systems of the form $\dot{x} = f(x) + B(x)u$ with the *state-dependent coefficient (SDC) factorization* $f(x) = A(x)x$ (Cloutier, 1997), which exists as long as f is continuously differentiable and $f(0) = 0$ (Çimen, 2010). A feedback controller can then be implemented by solving the corresponding *state-dependent Riccati equation (SDRE)* in terms of $(A(x), B(x))$ in closed-loop. While such a controller is only locally stabilizing in theory, in practice it has a large region of attraction and has proven effective in automotive (Acarman, 2009), spacecraft (Cloutier & Zipfel, 1999), robotic (Watanabe et al., 2008), and process control (Banks et al., 2002).

Contributions In this work, we study how to jointly identify nonlinear dynamics models and control-oriented structures from data that can be naturally leveraged in stabilizing closed-loop tracking control design. To this end, we study tracking controller for general nonlinear control-affine systems based on SDRE feedback. While SDREs have seen use in fixed-point stabilization, we focus on its extension to exactly characterizing and controlling the error dynamics for *trajectory tracking*. This extension relies on a generalized SDC factorization of the error dynamics that always exists for continuously differentiable dynamics. We propose a method to learn such structure from a finite data set, and thereby enable the use of SDRE-based tracking control. We compare our method of learning control-enabling structure to an adaptation of prior work that tries to jointly learn a dynamics model, controller, and stability certificate. In a variety of simulated nonlinear systems, we demonstrate that our learned controller performs well in closed-loop, and that controllers instead learned alongside dynamics models and parametric certificate functions can be brittle and data inefficient in practice.

2. Problem Statement

In this paper, we are interested in learning to control the nonlinear control-affine dynamical system

$$\dot{x} = f(x) + B(x)u = f(x) + \sum_{j=1}^{m} u_j b_j(x), \qquad (1)$$

with state $x(t) \in \mathbb{R}^n$, control $u(t) \in \mathbb{R}^m$, drift $f : \mathbb{R}^n \to \mathbb{R}^n$, and actuator $B : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ with columns $b_j : \mathbb{R}^n \to \mathbb{R}^n$, $j \in \{1, 2, \ldots, m\}$. In particular, we want to determine a tracking controller of the form $u = \pi(x, \bar{x}(t), \bar{u}(t))$ such that $(x(t), u(t))$ converges to any *dynamically feasible* pair $(\bar{x}(t), \bar{u}(t))$, i.e., satisfying $\dot{\bar{x}} = f(\bar{x}) + B(\bar{x})\bar{u}$. While we know the dynamics take the form of Equation (1), the vector fields $(f, \{b_j\}_{j=1}^m)$ are otherwise *unknown* to us. Instead, we only have access to a finite pre-collected data set $\mathcal{D} := \{(x^{(i)}, u^{(i)}, \dot{x}^{(i)})\}_{i=1}^N$ of input-output measurements of Equation (1).

3. Nonlinear Tracking Control

In this section, we overview a number of methods for synthesizing a tracking controller $u = \pi(x, \bar{x}(t), \bar{u}(t))$ for any control-affine nonlinear system of the form in Equation (1). We begin with LQR-based methods, including state-dependent-LQR tracking control. We also discuss tracking controllers that are guaranteed to exponentially stabilize the resulting closed-loop dynamics provided an accompanying certificate function is found, namely a control contraction metric (CCM). For each controller, we highlight the *control-oriented* structure that is required in addition to the dynamics to enable a stabilizing feedback signal. We will then discuss how to jointly learn such structure along with a dynamics model from data in Section 4 to enable closed-loop tracking control.

3.1. Linearized LQR

Perhaps the simplest approach to tracking control is based on linearizing the dynamics in Equation (1) around the current target $(\bar{x}(t), \bar{u}(t))$. Specifically, in this method we first linearize the nonlinear dynamics of the tracking error $e(t) := x(t) - \bar{x}(t)$ given by

$$\dot{e} = f(x) + B(x)u - f(\bar{x}) - B(\bar{x})\bar{u} \qquad (2)$$

to arrive at the approximation

$$\dot{e} \approx \underbrace{\left(\frac{\partial f}{\partial x}(\bar{x}) + \sum_{j=1}^{m} \bar{u}_j \frac{\partial b_j}{\partial x}(\bar{x})\right)}_{=:A(\bar{x},\bar{u})} e + B(\bar{x})(u - \bar{u}). \qquad (3)$$

Then, with $(A(\bar{x},\bar{u}), B(\bar{x}))$ and chosen positive-definite weight matrices (Q, R), we solve the Riccati equation

$$\begin{aligned} P(\bar{x},\bar{u})A(\bar{x},\bar{u}) + A(\bar{x},\bar{u})^\mathsf{T}P(\bar{x},\bar{u}) \\ - P(\bar{x},\bar{u})B(\bar{x})R^{-1}B(\bar{x})^\mathsf{T}P(\bar{x},\bar{u}) = -Q \end{aligned} \tag{4}$$

for the positive-definite solution $P(\bar{x},\bar{u})$. We then compute the tracking controller

$$u = \pi_{\mathrm{LQR}}(x,\bar{x},\bar{u}) \coloneqq \bar{u} - R^{-1}B(\bar{x})^\mathsf{T}P(\bar{x},\bar{u})e. \tag{5}$$

In practice, the "linearized LQR" tracking controller (i.e., LQR control applied to the linearized dynamics) in Equation (5) can be effective as long as $(x(t), u(t))$ remains close to $(\bar{x}(t), \bar{u}(t))$, i.e., as long as the linearized error dynamics in Equation (3) remain a good approximation of original error dynamics in Equation (2). Overall, the linearized LQR tracking controller requires us to be able to evaluate and differentiate the vector fields $(f, \{b_j\}_{j=1}^m)$; no additional structures are required.

3.2. Nonlinear State-Dependent LQR

For general nonlinear systems, the linearized LQR tracking controller presented in the previous section is a good first choice. However, it can fail for nonlinear systems when $(x(t), u(t))$ strays from the target $(\bar{x}(t), \bar{u}(t))$, since then Equation (3) is no longer a good approximation.

In this section, we introduce an *exact* nonlinear factorization of the error dynamics for general control-affine systems that resemble the linearized form in Equation (3). This factorization is based on the theory of SDC forms (Cloutier, 1997; Çimen, 2010; 2012), and thereby enables a feedback law based on solving an associated SDRE.

State-Dependent LQR for Regulation To begin, we first look at the simpler problem of regulating the state $x(t)$ of the system $\dot{x} = f(x) + B(x)u$ to $x = 0$. For now, we assume that $(x, u) = (0, 0)$ is an equilibrium pair, i.e., $f(0) = 0$. If $f : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable, Çimen (2010, Proposition 1) shows

$$\dot{x} = f(x) + B(x)u = A(x)x + B(x)u, \tag{6}$$

where $f(x) \equiv A(x)x$ is an *exact* factorization known as a *state-dependent coefficient (SDC) form* of f. With chosen positive-definite matrices (Q, R), these factorized dynamics naturally enable the controller $u = K(x)x = -R^{-1}B(x)^\mathsf{T}P(x)x$, where $P(x)$ is the positive-definite solution of the *state-dependent Riccati equation (SDRE)*

$$\begin{aligned} P(x)A(x) + A(x)^\mathsf{T}P(x) \\ - P(x)B(x)R^{-1}B(x)^\mathsf{T}P(x) = -Q \end{aligned} . \tag{7}$$

As its name implies, the SDRE is dependent on the *current* state x of the system. This contrasts with the Riccati equation for linearized LQR in Equation (4), which does not depend on x and only depends on the target pair $(\bar{x}, \bar{u})$ due to linearization. Despite using an exact nonlinear factorization of the dynamics, the feedback law $u = -R^{-1}B(x)^\mathsf{T}P(x)x$ is only locally stabilizing in theory and there is no guarantee that it will outperform linearized LQR, especially if only symmetric solutions of Equation (7) are considered (Çimen, 2012, Example 2). Nevertheless, *state-dependent LQR (SD-LQR)* control in practice can induce a large region of attraction, especially relative to linearized control (Çimen, 2012).

Generalized SDC Forms To extend SD-LQR to tracking control for control-affine systems, we leverage a generalization of SDC forms previously introduced by Tsukamoto et al. (2021a;b) and described in Proposition 1 below.

Proposition 1: *Suppose $f : \mathbb{R}^n \to \mathbb{R}^d$ is continuously differentiable. Then a matrix function $A : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{d\times n}$ exists such that*

$$f(x) - f(\bar{x}) = A(\bar{x}, x - \bar{x})(x - \bar{x}) = A(\bar{x}, e)e, \tag{8}$$

for all $x, \bar{x} \in \mathbb{R}^n$ with $e \coloneqq x - \bar{x}$. Furthermore, A can be chosen such that $A(\bar{x}, 0) \equiv \frac{\partial f}{\partial x}(\bar{x})$.

Proof. Consider any curve $r(s) = \bar{x} + R(s)e$ where $R : [0, 1] \to \mathbb{R}^{n\times n}$ is differentiable, $R(0) = 0$, and $R(1) = I$. Then by the fundamental theorem for line integrals,

$$f(x) - f(\bar{x}) \equiv \underbrace{\left(\int_0^1 \frac{\partial f}{\partial x}(\bar{x} + R(s)e)R'(s)\, ds\right)}_{=:A(\bar{x},e)} e. \tag{9}$$

Moreover, $A(\bar{x}, 0) = \int_0^1 \frac{\partial f}{\partial x}(\bar{x})R'(s)\, ds = \frac{\partial f}{\partial x}(\bar{x})$. ■

Proposition 1 describes a factorization of continuously differentiable f that exactly quantifies $f(x) - f(\bar{x})$ between any x and $\bar{x}$. When $x = \bar{x}$, the matrix factor $A(\bar{x}, e)$ reduces to the local Jacobian of f at $\bar{x}$. Much like the linear approximation $\frac{\partial f}{\partial x}(\bar{x})e$, the exact factorization $A(\bar{x}, e)e$ is a function of the chosen "target" $\bar{x}$ and the "error" $e \coloneqq x - \bar{x}$. It is precisely this perspective that now allows us to apply this generalized SDC form to tracking control.

State-Dependent LQR for Trajectory Tracking For SD-LQR tracking control, we consider the error dynamics for general control-affine systems. Let $(\bar{x}(t), \bar{u}(t))$ be a dynamically feasible pair that we want to track. Then the dynamics of the tracking error $e \coloneqq x - \bar{x}$ are

$$\begin{aligned} \dot{e} &= f(x) + B(x)u - f(\bar{x}) - B(\bar{x})\bar{u} \\ &= f(x) - f(\bar{x}) + (B(x) - B(\bar{x}))\bar{u} + B(x)(u - \bar{u}) \\ &= \underbrace{\left(A_0(\bar{x}, e) + \sum_{j=1}^m \bar{u}_j A_j(\bar{x}, e)\right)}_{=:A_{\mathrm{SDC}}(\bar{x},\bar{u},e)} e + B(x)v \end{aligned} , \tag{10}$$

where $v := u - \bar{u}$, and $(A_0, \{A_j\}_{j=1}^m)$ are SDC factorizations of the vector fields $(f, \{b_j\}_{j=1}^m)$ such that

$$\begin{aligned} f(x) - f(\bar{x}) &\equiv A_0(\bar{x}, e)e \\ b_j(x) - b_j(\bar{x}) &\equiv A_j(\bar{x}, e)e, \ \forall j \in \{1, 2, \dots, m\} \end{aligned}. \quad (11)$$

An SDRE similar to Equation (7) expressed in terms of $(A_{\text{SDC}}(\bar{x}, \bar{u}, e), B(x))$ and chosen positive-definite weight matrices (Q, R) can be solved for the positive-definite matrix $P_{\text{SDC}}(\bar{x}, \bar{u}, e)$. The associated nonlinear tracking controller is then

$$u = \pi_{\text{SDC}}(x, \bar{x}, \bar{u}) := \bar{u} - R^{-1}B(x)^\mathsf{T} P_{\text{SDC}}(\bar{x}, \bar{u}, e)e. \quad (12)$$

This controller reduces to the linearized LQR controller in Equation (5) if $A_{\text{SDC}}(\bar{x}, \bar{u}, 0)$ is used, since then the SDC factorizations and hence the exact nonlinear error dynamics in Equation (10) reduce to the Jacobians and the linearized error dynamics, respectively, in Equation (3).

Our goal in using SD-LQR tracking control is to enable better tracking performance for highly nonlinear systems that may experience large deviations from the target trajectory, e.g., during fast or aggressive maneuvers. The key trade-off in the use of a more complex controller is the need for additional known control-oriented structure. In this case, this structure comprises the SDC factorizations $(A_0, \{A_j\}_{j=1}^m)$ that are not required in the simpler linearized LQR tracking controller. In Section 4, we will discuss how we can learn $(A_0, \{A_j\}_{j=1}^m)$ from data, and later in Section 5 we will show how this has a powerful regularization effect on learning models of dynamical systems for the purposes of closed-loop control. Before that, in the next section we overview alternative methods that couple a tracking controller with a certificate function guaranteeing closed-loop tracking convergence.

3.3. Exponential Stabilizability via Contraction Theory

Linearized and state-dependent LQR rely on approximate and exact factorized forms, respectively, of the system dynamics to construct tracking control laws. However, neither of these LQR controllers is guaranteed to stabilize the closed-loop error dynamics when the system is nonlinear. In this section, we review a family of tracking controllers that ensure exponential stability, i.e.,

$$\|x(t) - \bar{x}(t)\|_2 \leq \alpha \|x(0) - \bar{x}(0)\|_2 \exp(-\beta t), \quad (13)$$

with overshoot $\alpha > 0$ and decay rate $\beta > 0$, for all $t \geq 0$.

To this end, *contraction theory* (Lohmiller & Slotine, 1998) seeks to construct certifiably stabilizing controllers for any control-affine system of the form Equation (1) by analyzing the stabilizability of the variational dynamics

$$\dot{\delta}_x = \underbrace{\left(\frac{\partial f}{\partial x}(x) + \sum_{j=1}^m u_j \frac{\partial b_j}{\partial x}(x)\right)}_{=:A(x,u)} \delta_x + B(x)\delta_u, \quad (14)$$

where δ_x and δ_u are virtual displacements in the tangent spaces at x and u, respectively. The high-level idea of contraction theory is to stabilize this infinite family of linear variational systems pointwise everywhere with a variational feedback law for δ_u, then path-integrate to get a stabilizing feedback law for u in the original system (Lohmiller & Slotine, 1998; Manchester & Slotine, 2017). Let $M : \mathbb{R}^n \to \mathbb{S}^n_{\succ 0}$ be a uniformly positive-definite matrix-valued function, i.e., such that $\underline{\lambda} I \preceq M(x) \preceq \overline{\lambda} I$ for some constants $\underline{\lambda}, \overline{\lambda} > 0$ and all $x \in \mathbb{R}^n$. Denote the time-derivative of $M(x)$ as $\dot{M}(x, u)$, with ij-th element

$$\dot{M}_{ij}(x, u) := \nabla M_{ij}(x)^\mathsf{T}(f(x) + B(x)u). \quad (15)$$

Then $M(x)$ is a *control contraction metric (CCM)* for the system in Equation (1) if there exist a constant $\beta > 0$ and a variational controller $\delta_u = \delta_\pi(\delta_x, x, u)$ such that

$$\begin{aligned} &\delta_x^\mathsf{T}\Big(\dot{M}(x, u) + M(x)A(x, u) + A(x, u)^\mathsf{T} M(x)\Big)\delta_x \\ &\quad + 2\delta_x^\mathsf{T} M(x)B(x)\delta_\pi(\delta_x, x, u) \leq -2\beta\delta_x^\mathsf{T} M(x)\delta_x \end{aligned} \quad (16)$$

for all δ_x, x, and u. Given a CCM, an exponentially stabilizing tracking controller of the form

$$u = \pi_{\text{CCM}}(x, \bar{x}, \bar{u}) = \bar{u} + k(x, \bar{x}) \quad (17)$$

can be constructed by geodesic integration between x and $\bar{x}$ (Manchester & Slotine, 2017; Singh et al., 2019; 2021), with overshoot $\alpha = \sqrt{\overline{\lambda}/\underline{\lambda}}$, decay rate β, and $k(\bar{x}, \bar{x}) \equiv 0$. Alternatively, a differentiable controller of the form in Equation (17) achieves this same result if

$$\begin{aligned} &\dot{M}(x, u) + \left(A(x, u) + B(x)\frac{\partial k}{\partial x}(x, \bar{x})\right)^\mathsf{T} M(x) \\ &+ M(x)\left(A(x, u) + B(x)\frac{\partial k}{\partial x}(x, \bar{x})\right) \preceq -2\beta M(x) \end{aligned}, \quad (18)$$

for all x, $\bar{x}$, and $\bar{u}$ (Manchester & Slotine, 2017).

The exponential stability of the error dynamics in closed-loop with the tracking controller in Equation (17) is *certified* by the CCM M. Once again we see that attaining better closed-loop performance requires additional control-oriented structure; in this case, this structure comprises the certificate M and the closed-loop contraction condition in Equation (18) that must be satisfied for *all* x, $\bar{x}$, and $\bar{u}$.

4. Jointly Learning Dynamics, Controllers, and Control-Oriented Structure

In the previous section, we introduced a number of tracking controllers for nonlinear control-affine systems. We also highlighted how increasing the complexity of the tracking controller often promises improved closed-loop performance at the cost of requiring knowledge of additional control-oriented components. For linearized LQR, only the vector fields $(f, \{b_j\}_{j=1}^m)$ and their derivatives are needed. For SD-LQR, we also need to know the SDC factorizations $(A_0, \{A_j\}_{j=1}^m)$ of $(f, \{b_j\}_{j=1}^m)$. For CCM-based tracking control, we need to know (f, B) and a CCM M that together satisfy the constraint in Equation (18) for all x, $\bar{x}$, and $\bar{u}$. Even when (f, B) are known, synthesizing SDC factorizations (e.g., via the line integral in Equation (9)) or a CCM is a difficult problem that requires leveraging further structure in the dynamics (e.g., sparsity). This is generally not possible when (f, B) are learned from data for an unknown system using complex parametric function approximators (e.g., neural networks).

In this section, we describe our main contribution to learning how to control control-affine dynamical systems when we only have access to a finite labelled data set $\mathcal{D} := \{(x^{(i)}, u^{(i)}, \dot{x}^{(i)})\}_{i=1}^N$ of input-output measurements of Equation (1). Specifically, we describe a few methods for jointly learning a dynamics model and a tracking controller with unconstrained optimization, and focus on how this involves additionally modeling and learning control-oriented structure to enable a particular feedback law.

Learning Dynamics from Data Each method in this section learns a model of the dynamics in Equation (1). To this end, we define the regression loss

$$L_{\text{reg}}^{\text{dyn}}(f, B, \mathcal{D}) = \sum_{(x,u,\dot{x}) \in \mathcal{D}} \|\dot{x} - f(x) - B(x)u\|_2^2. \quad (19)$$

If we instantiate (f, B) with parametric functions, such as neural networks, we can do gradient descent on this loss to fit (f, B) to the data. Thus, a naïve approach and our first baseline for learning how to control Equation (1) is to fit a differentiable model of (f, B) to the data $\mathcal{D}$ and then apply linearized tracking LQR from Section 3.1.

Learning SDC Factorizations (Our Method) For SD-LQR, we need to learn the SDC factorizations denoted by $\mathcal{A} := (A_0, \{A_j\}_{j=1}^m)$. For this, we use the regression loss

$$L_{\text{reg}}^{\text{SDC}}(\mathcal{A}, \mathcal{D}) = \sum_{\substack{(x,u,\dot{x}),\\ (\bar{x},\bar{u},\dot{\bar{x}}) \in \mathcal{D}}} \|\dot{e} - A_{\text{SDC}}(\bar{x}, \bar{u}, e)e - B(x)v\|_2^2, \quad (20)$$

which sums over *pairs* of labelled samples in the data set $\mathcal{D}$. We also need $\mathcal{A}$ to be a set of valid SDC factorizations, for which we define the *unlabelled* data set $\mathcal{D}_{\text{aux}}^{\text{SDC}} = \{(x^{(i)}, \bar{x}^{(i)})\}_{i=1}^{N_{\text{aux}}^{\text{SDC}}}$ and the auxiliary loss

$$\begin{aligned} &L_{\text{aux}}^{\text{SDC}}(f, B, \mathcal{A}, \mathcal{D}_{\text{aux}}^{\text{SDC}}) \\ &= \sum_{(x,\bar{x}) \in \mathcal{D}_{\text{aux}}^{\text{SDC}}} \Bigg(\|f(x) - f(\bar{x}) - A_0(\bar{x}, e)e\|_2^2 \\ &\qquad + \sum_{j=1}^m \|b_j(x) - b_j(\bar{x}) - A_j(\bar{x}, e)e\|_2^2 \Bigg) \end{aligned}. \quad (21)$$

Overall, we can learn $(f, B, \mathcal{A})$ instantiated as parametric functions via gradient descent on the composite loss

$$\begin{aligned} &L^{\text{SDC}}(f, B, \mathcal{A}, \mathcal{D}, \mathcal{D}_{\text{aux}}^{\text{SDC}}) \\ &= L_{\text{reg}}^{\text{dyn}}(f, B, \mathcal{D}) + L_{\text{reg}}^{\text{SDC}}(\mathcal{A}, \mathcal{D}) + L_{\text{aux}}^{\text{SDC}}(f, B, \mathcal{A}, \mathcal{D}_{\text{aux}}^{\text{SDC}}) \end{aligned}. \quad (22)$$

This total loss is *semi-supervised* in that it is a function of both labelled and unlabelled data $\mathcal{D}$ and $\mathcal{D}_{\text{aux}}^{\text{SDC}}$, respectively. Ideally, we would want to constrain $\mathcal{A}$ to be a set of SDC factorizations of (f, B) consistent with Equation (8). Since we cannot straightforwardly enforce Equation (8) by construction, we use the auxiliary loss term in Equation (22) as a penalty-based relaxation, with as many unlabelled samples in $\mathcal{D}_{\text{aux}}^{\text{SDC}}$ as possible. This idea of relaxing pointwise functional constraints with sampling-based penalty terms is a common approach to learning global control-oriented structure (Richards et al., 2018; Singh et al., 2021; Sun et al., 2020; Dawson et al., 2022) and more generally in semi-infinite optimization (Zhang et al., 2010).

Learning CCMs This method is founded on the literature concerning joint learning of dynamics, controllers, and stability certificates (Singh et al., 2021; Sun et al., 2020; Dawson et al., 2022; Zhou et al., 2022). For CCM-based tracking control, we need to learn a dynamics model (f, B), a uniformly positive-definite CCM M, and a feedback controller $u = \bar{u} + k(x, \bar{x})$ such that $k(\bar{x}, \bar{x}) \equiv 0$, that altogether satisfy the inequality in Equation (18) for all x, $\bar{x}$, and $\bar{u}$. We take cues from Sun et al. (2020) to setup a loss function that will allow us to train all three components together with gradient descent, albeit with some adjustments to accommodate our lack of any knowledge of the dynamics (f, B) (which Sun et al. (2020) assume are known).

We first specify the desired overshoot $\alpha > 0$, decay rate $\beta > 0$, and eigenvalue lower bound $\underline{\lambda} > 0$ as hyperparameters, and construct a candidate CCM M as

$$M(x) = \underline{\lambda} I + L(x)L(x)^\top, \quad (23)$$

where $L : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ is any parametric matrix function. This construction ensures $M(x) \succeq \underline{\lambda} I$ for all x. To ensure $k(\bar{x}, \bar{x}) \equiv 0$, we follow Proposition 1 and let $k(x, \bar{x}) = K(x, \bar{x})(x - \bar{x})$ for any parametric function $K : \mathbb{R}^n \to \mathbb{R}^n \to \mathbb{R}^{m \times n}$. With the closed-loop variational

matrix defined by

$$\bar{A}(x,\bar{x},\bar{u}) := \frac{\partial f}{\partial x}(x) + \sum_{j=1}^{m} u_j \frac{\partial b_j}{\partial x}(x) + B(x)\frac{\partial k}{\partial x}(x,\bar{x}), \tag{24}$$

we collect terms of the inequality from Equation (18) in

$$\begin{aligned} C(x,\bar{x},\bar{u}) = \dot{M}(x,u) + \bar{A}(x,\bar{x},\bar{u})^{\mathsf{T}} M(x) \\ + M(x)\bar{A}(x,\bar{x},\bar{u}) + 2\beta M(x), \end{aligned} \tag{25}$$

with $u = \bar{u} + k(x,\bar{x}) = \bar{u} + K(x,\bar{x})(x-\bar{x})$. Finally, with the unlabelled data set $\mathcal{D}_{\text{aux}}^{\text{CCM}} = \{(x^{(i)},\bar{x}^{(i)}),\bar{u}^{(i)})\}_{i=1}^{N_{\text{aux}}^{\text{CCM}}}$, we define the auxiliary loss

$$\begin{aligned} &L_{\text{aux}}^{\text{CCM}}(f,B,M,K,\mathcal{D}_{\text{aux}}^{\text{CCM}}) \\ &= \sum_{(x,\bar{x},\bar{u})\in\mathcal{D}_{\text{aux}}^{\text{CCM}}} \Big(\max(0,\lambda_{\max}(C(x,\bar{x},\bar{u}))) \\ &\qquad + \max\big(0,\lambda_{\max}(M(x)) - \alpha^2\underline{\lambda}\big)\Big) \end{aligned}, \tag{26}$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue operator. Overall, we can learn (f,B,M,K) instantiated as parametric functions via gradient descent on the total loss

$$\begin{aligned} &L^{\text{CCM}}(f,B,M,K,\mathcal{D},\mathcal{D}_{\text{aux}}^{\text{CCM}}) \\ &= L_{\text{reg}}^{\text{dyn}}(f,B,\mathcal{D}) + L_{\text{aux}}^{\text{CCM}}(f,B,M,K,\mathcal{D}_{\text{aux}}^{\text{CCM}}) \end{aligned}. \tag{27}$$

Much like in the SD-LQR case, this total loss is semi-supervised, although the auxiliary data set $\mathcal{D}_{\text{aux}}^{\text{CCM}}$ also requires samples of the input $\bar{u}$. This loss function can be viewed as an unconstrained relaxation of the approach from Singh et al. (2021), who instead use pointwise inequalities derived from Equation (16) as exact constraints in an optimization over (f,B,M). However, Singh et al. (2021) only use linear-in-parameter approximators for (f,B,M) to construct a bi-convex program between (f,B) and M, investigate the regularizing effect of fitting (f,B,M) on the predictive capabilities of (f,B) in closed-loop, and do not learn a controller. In contrast, the modified setup described above jointly learns a dynamics model, certificate function, and controller that can each be expressed with complex parametric functions, so that in the next section we can compare with the learning setups for linearized LQR and SD-LQR.

5. Experiments

In this section, we experimentally investigate the three methods described in Section 4 for jointly learning a dynamics model, stabilizing tracking controller, and/or some control-oriented structure enabling the controller, namely:

- *Naïve LQR learning:* Fit a control-affine form (f,B) to labelled data $\mathcal{D} := \{(x^{(i)},u^{(i)},\dot{x}^{(i)})\}_{i=1}^{N}$ via gradient descent on the regression loss in Equation (19). Then perform linearized LQR.
- *CCM learning:* Jointly fit (f,B), a CCM M, and a gain matrix function K to labelled data $\mathcal{D}$ and unlabelled data $\mathcal{D}_{\text{aux}}^{\text{CCM}} = \{(x^{(i)},\bar{x}^{(i)}),\bar{u}^{(i)})\}_{i=1}^{N_{\text{aux}}^{\text{CCM}}}$ via gradient descent on the composite loss in Equation (27). Then apply the controller $u = \bar{u} + K(x,\bar{x})(x-\bar{x})$.
- *SD-LQR learning (our method):* Jointly fit (f,B) and SDC factorizations $\mathcal{A} := (A_0,\{A_j\}_{j=1}^{m})$ to labelled data $\mathcal{D}$ and unlabelled data $\mathcal{D}_{\text{aux}}^{\text{SDC}} = \{(x^{(i)},\bar{x}^{(i)})\}_{i=1}^{N_{\text{aux}}^{\text{SDC}}}$ via gradient descent on the composite loss in Equation (22). Then perform SD-LQR.

We also implement linearized LQR with *known* dynamics as an oracle. We evaluate these methods on two nonlinear benchmark systems:

Spacecraft Our planar spacecraft, based on that of Lew et al. (2022), has mass m with center-of-mass offset at $(d_x,d_y) \in \mathbb{R}^2$, and a rotational moment of inertia J. Its state is $x = (p_x,p_y,\theta,\dot{p}_x,\dot{p}_y,\dot{\theta}) \in \mathbb{R}^6$, where (p_x,p_y) is its position and θ is its heading angle. The control is $u = (F_x,F_y,M) \in \mathbb{R}^3$, where (F_x,F_y) are the applied forces along the inertial x-axis and y-axis, respectively, and M is the applied moment. The control-affine dynamics of the spacecraft are given by

$$f(x) = \frac{1}{m}\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \\ \dot{\theta}^2 d_x \\ \dot{\theta}^2 d_y \\ 0 \end{bmatrix}, \; B(x) = \frac{1}{mJ}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ J+d_y^2 & -d_x d_y & d_y \\ -d_x d_y & J+d_x^2 & -d_x \\ m d_y & -m d_x & m \end{bmatrix}.$$

PVTOL Our planar vertical-take-off-and-landing (PVTOL) vehicle has mass m, rotational moment of inertia J, moment arm length ℓ between the center of mass and each of two rotors, and gravitational acceleration g. Its state is $x = (p_x,p_y,\phi,v_x,v_y,\dot{\phi}) \in \mathbb{R}^6$, where (p_x,p_y) is its position, ϕ is its roll angle, and (v_x,v_y) is its velocity in the body-fixed frame. The control is $u = (F_R,F_L) \in \mathbb{R}^2$, where F_R and F_L are the applied thrusts by the right and left rotors, respectively, along the body-fixed y-axis. The control-affine dynamics of this PVTOL are given by

$$f(x) = \begin{bmatrix} v_x\cos\phi - v_y\sin\phi \\ v_x\sin\phi + v_y\cos\phi \\ \dot{\phi} \\ v_y\dot{\phi} - g\sin\phi \\ -v_x\dot{\phi} - g\cos\phi \\ 0 \end{bmatrix}, \; B(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/m & 1/m \\ \ell/J & -\ell/J \end{bmatrix}.$$

The planar spacecraft is only slightly nonlinear due to the term $\dot{\theta}^2$ introduced by the center-of-mass offset, and so should serve as a relatively easy benchmark for learning-based control. In contrast, the PVTOL is a highly non-

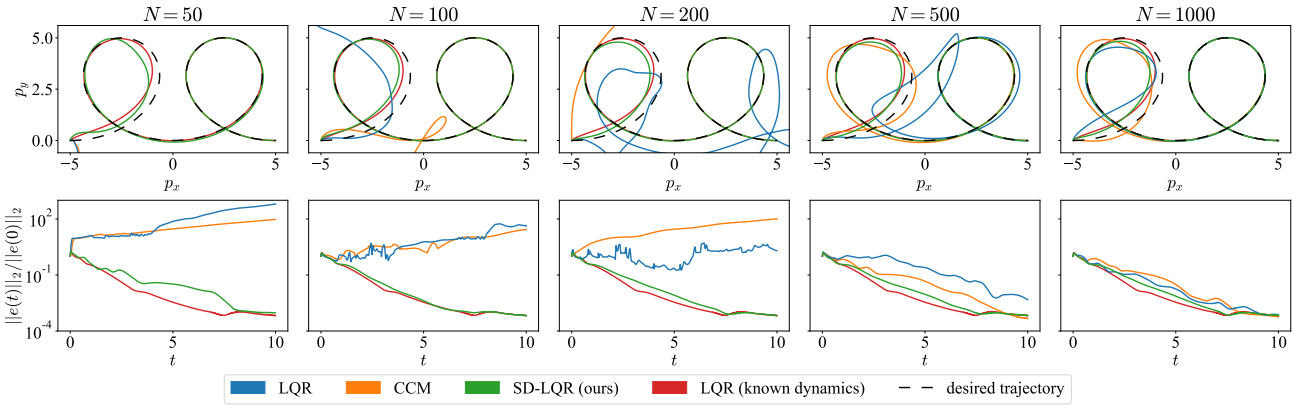

Figure 1. Trajectory tracking results for the PVTOL system on a double loop-the-loop trajectory. The top row qualitative depicts the closed-loop trajectories for each method overlayed with the desired trajectory (black dashed). The bottom row shows the normalized tracking error over time. Plots proceed from left to right with an increasing amount N of labelled training data. Our learned SD-LQR method is the only learning-based approach that successfully tracks the trajectory for all N.

linear, underactuated, non-minimum-phase dynamical system (Hauser et al., 1992), and thus serves as a challenging benchmark.

Training Details For each system, we begin by uniformly sampling points $\{(x^{(i)}, u^{(i)})\}_{i=1}^{N}$ from a bounded state-control set $\mathcal{X} \times \mathcal{U} \subset \mathbb{R}^n \times \mathbb{R}^m$, and evaluating the true dynamics to form the labelled data $\mathcal{D}$. Both $\mathcal{X}$ and $\mathcal{U}$ are described in Appendix A, along with other implementation details and hyperparameters. We additionally uniformly sample unlabelled data sets $\mathcal{D}_{\text{aux}}^{\text{CCM}}$ and $\mathcal{D}_{\text{aux}}^{\text{SDC}}$ for use with the CCM and SDC learning methods, respectively, from $\mathcal{X}$ and $\mathcal{U}$. We vary the labelled training set size N to investigate the data efficiency of each method, with a constant number of auxiliary points $N_{\text{aux}}^{\text{CCM}} = N_{\text{aux}}^{\text{SDC}} = 10000$. Each function in $(f, B, M, K, A_0, \{A_j\}_{j=1}^{m})$ is approximated as a feedforward neural network with the same number of fully connected hidden layers, and appropriately shaped input and output dimensions using Python and JAX (Bradbury et al., 2018). For each method, the appropriate subset of these functions is trained via the Adam optimizer (Kingma & Ba, 2015) on the corresponding loss function. Training is performed for 50000 epochs while the loss on a held-out validation set is monitored; for each method, the model parameters corresponding to the lowest validation loss are chosen for testing. This training procedure is repeated for each method across 5 random seeds.

Testing and Results To test the controllers learned with each method, we must first generate dynamically feasible trajectories for tracking. We first evaluate the PVTOL system qualitatively; we leverage its differential flatness (Ailon, 2010) to generate a feasible pair $(\bar{x}(t), \bar{u}(t))$ yielding the double loop-the-loop shape in Figure 1. For a single random seed, we plot the closed-loop trajectory from using each learned controller to track the loop-the-loop. We repeat this test for various sizes N of the labelled training data set $\mathcal{D}$, and plot the trajectories in (p_x, p_y)-space and the normalized tracking error $\frac{\|e(t)\|_2}{\|e(0)\|_2}$ over time. Our learned SD-LQR method is the only learning-based method that succeeds for every size N, while the learned LQR and CCM controllers outright fail for smaller data set sizes. This is initial evidence of the *data efficiency* in learning SDC factorizations for the purpose of control.

For more thorough testing, we want to generate many trajectories in a manner applicable to both the spacecraft and PVTOL. To this end, we generate $N_{\text{test}} = 100$ feasible trajectories $\mathcal{T}_{\text{test}} := \{(\bar{x}^{(k)}(t), \bar{u}^{(k)}(t))\}_{k=1}^{N_{\text{test}}}$ for each system by solving the optimal control problem

$$\begin{aligned} \underset{\bar{x}(\cdot),\bar{u}(\cdot)}{\text{minimize}} \quad & \int_0^T \left(\bar{x}(t)^\mathsf{T} Q \bar{x}(t) + \bar{u}(t)^\mathsf{T} R \bar{u}(t)\right) dt \\ \text{subject to} \quad & \dot{\bar{x}}(t) = f(\bar{x}(t)) + B(\bar{x}(t))\bar{u}(t) \\ & \bar{x}(0) = \bar{x}_0^{(k)} \\ & \bar{x}(T) = 0 \\ & u_{\text{lb}} \preceq \bar{u}(t) \preceq u_{\text{ub}} \end{aligned} \tag{28}$$

for different initial conditions $\bar{x}_0^{(k)}$ sampled uniformly from $\mathcal{X}$, where (Q, R) are positive-definite weight matrices and $(u_{\text{lb}}, u_{\text{ub}})$ are control input bounds. Specifically, we use CasADi (Andersson et al., 2019) to transcribe this problem into a nonlinear multiple shooting optimization that is passed to and solved by the Ipopt solver (Wächter & Biegler, 2006). Then, for each system, test trajectory, and tracking controller, we uniformly sample an initial state $x_0^{(k)} \neq \bar{x}_0^{(k)}$ from $\mathcal{X}$, and simulate the closed-loop system.

Figure 2 displays the normalized tracking error $\frac{\|e(t)\|_2}{\|e(0)\|_2}$ over

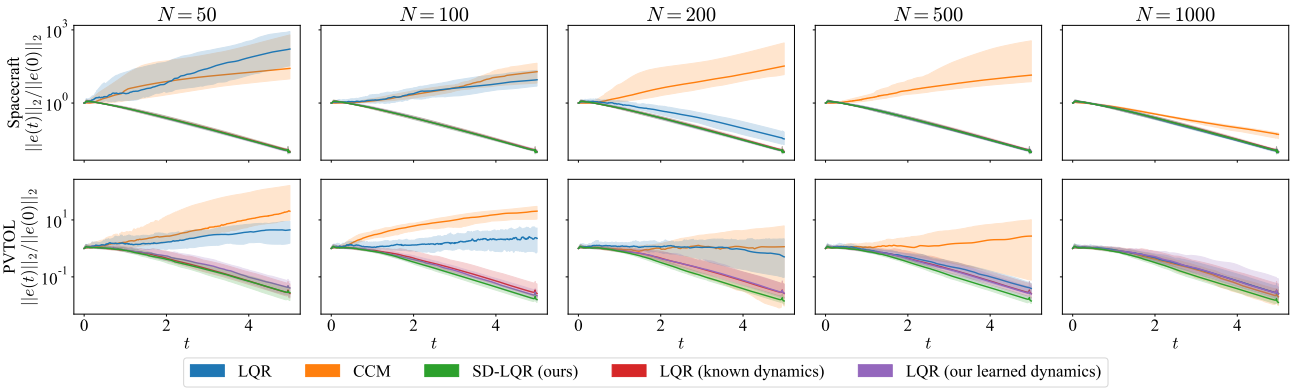

Figure 2. Trajectory tracking results for both the spacecraft and PVTOL systems for $N_{\text{test}} = 100$ trajectories each. The top and bottom rows show the normalized tracking error over time for the spacecraft and PVTOL, respectively. Plots proceed from left to right with an increasing amount N of labelled training data. Colored lines represent the median across all trajectories at each time t, while shaded regions depict interquartile ranges. Our learned SD-LQR method consistently outperforms the considered baseline learning methods.

time for both the spacecraft and the PVTOL, for various training set sizes N. For each method, system, and N, the median normalized tracking error across the test trajectory set $\mathcal{T}_{\text{test}}$ is plotted along with shaded regions denoting the interquartile range over time. Once again we observe the data efficiency of our learned SD-LQR method for both systems. Our method even *outperforms the oracle* LQR controller at higher values of N for the PVTOL, despite having to learn the dynamics. This is likely due to how even the oracle LQR controller is limited by its linear approximation of the error dynamics, while the SD-LQR controller uses a learned model of the full nonlinear error dynamics. The naïve learned LQR method unsurprisingly converges to performance similar to the oracle LQR controller as N increases. Notably, the CCM-based controller has the most trouble overall, thereby highlighting its data inefficiency and brittleness.

As an ablation in Figure 2, with our dynamics model (f, B) learned alongside the SDC factorizations $\mathcal{A}$ via the loss Equation (22), we also perform linearized LQR rather than SD-LQR. We see that even solely using our learned dynamics model greatly improves the performance of linearized LQR across all values of N. This indicates that learning SDC factorizations as a structural prior *regularizes* the dynamics model for closed-loop control. Nevertheless, if they are available, using the SDC factorizations directly in SD-LQR is more practical than repeatedly differentiating the dynamics online for linearized LQR.

To complete our assessment, we repeat the training procedure and tests underlying Figure 2 with 5 random seeds, and aggregate the results in Figure 3. Moreover, we repeat this entire process with *trajectory* data. Specifically, instead of uniformly sampling points $\{(x^{(i)}, u^{(i)})\}_{i=1}^{N}$, we solve the optimal control problem Equation (28) to generate N_{traj} dynamically feasible trajectories, different from those in the test set and each beginning at a different initial condition. We then collect data "on-policy" by simulating linearized LQR with the true dynamics to track these trajectories and generate a dataset. Each trajectory is 5 seconds long with samples collected at 100 Hz, such that $N = 500N_{\text{traj}}$. However, samples along a trajectory are clearly temporally correlated, so we use N_{traj} rather than N as a measure of the size of the dataset when training on trajectory data. For both uniformly sampled data and trajectory data, we consider the average root mean squared (RMS) error

$$\text{RMS}(\mathcal{T}_{\text{test}}) := \frac{1}{N_{\text{test}}} \sum_{k=1}^{N_{\text{test}}} \sqrt{\frac{1}{T} \int_0^T \frac{\|e^{(k)}(t)\|_2^2}{\|e^{(k)}(0)\|_2^2} \, dt} \tag{29}$$

across all test trajectories for each random seed and training set size N (or N_{traj}). In Figure 3, we plot the median and interquartile range of $\text{RMS}(\mathcal{T}_{\text{test}})$ across random seeds as a function of N and N_{traj}. From these plots, we can see an even starker contrast between the performance of our learned SD-LQR method and the others. For the spacecraft, our method is close in performance to that of the oracle LQR, which is not surprising given that the spacecraft dynamics are only mildly nonlinear. For the highly nonlinear PVTOL, our method begins outperforming the oracle LQR at only $N = 100$ and $N_{\text{traj}} = 5$. Meanwhile, both the learned linearized LQR and CCM controllers struggle until more training data is used, which highlights their data inefficiency compared to our method. Moreover, in the ablation wherein we apply linearized LQR to our dynamics model (f, B) learned alongside SDC factorizations $\mathcal{A}$, much of the performance gained does not require using the SDC factorizations directly. Once again, this indicates the usefulness of SDC factorizations as a *regularizing* structural prior when they are learned alongside the model (f, B).

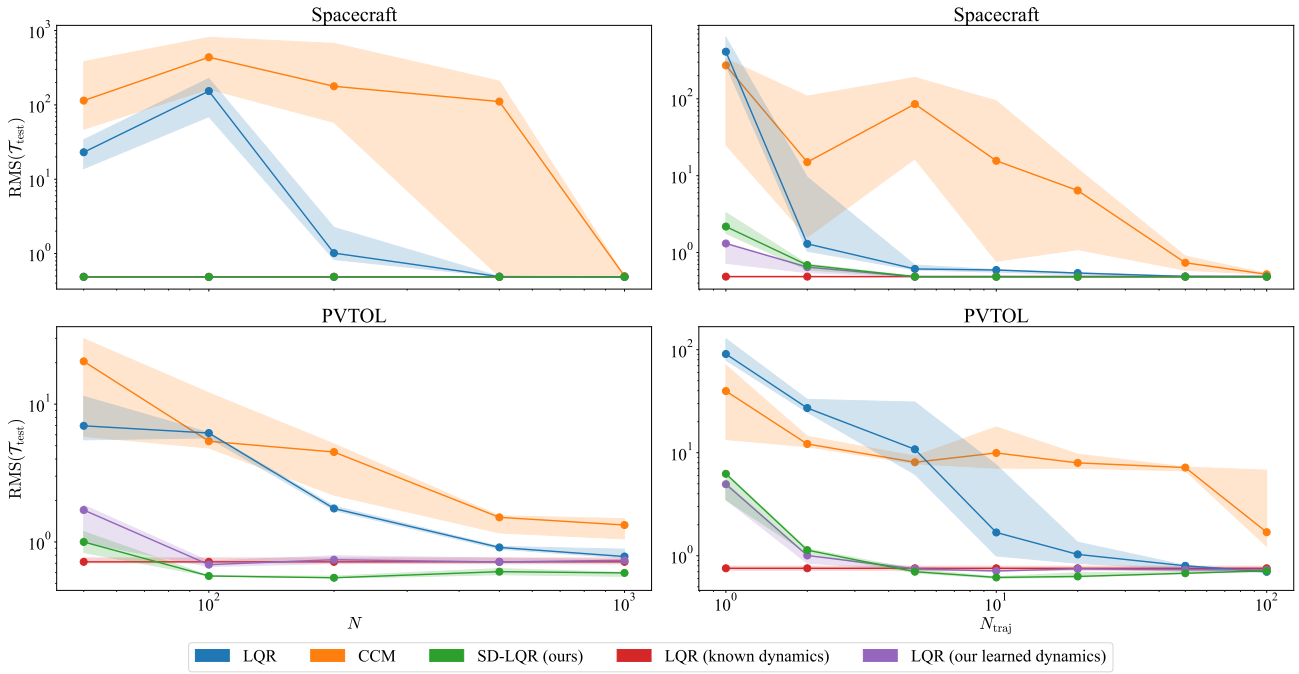

Figure 3. RMS tracking error as a function of the labelled training data set size N (left) and N_{traj} (right), averaged across $N_{\text{test}} = 100$ test trajectories (see Equation (29)). Colored lines denote medians across 5 random seeds, while shaded regions depict interquartile ranges. When training is done on uniformly sampled data, our learned SD-LQR method outperforms all other methods, even the oracle LQR on the PVTOL system. When training is done on trajectory data, performance is still at its best when using our dynamics model (f, B) learned alongside SDC factorizations $\mathcal{A} := (A_0, \{A_j\}_{j=1}^m)$. For $N_{\text{traj}} \in \{1, 2\}$, linearized LQR with our learned model seems to do better than SD-LQR, although the performance difference is small.

6. Conclusions and Future Work

In this paper, we studied how to jointly learn a dynamics model and a stabilizing tracking controller from only a finite data set of input-output measurements of an unknown dynamical system. We highlighted the importance of not only learning the dynamics, but also control-oriented structure that enables performant controller design. For this purpose, we proposed learning SDC factorizations of the dynamics for the purpose of state-dependent LQR tracking controller. Inspired by the literature, we compared our method to naïvely learning a model for linearized LQR, and to methods that couple learned controllers with learned certificate functions. Overall, we found that our method outperformed the baselines in terms of data efficiency and tracking capability. Moreover, we observed that including SDC factorizations in the learning problem regularizes the dynamics model to perform better during closed-loop LQR-based control.

Future Work We view this paper in part as a critique of methods that try to enforce closed-loop stabilizability guarantees by penalizing sampled violations of certificate conditions like Equation (18). As we have demonstrated, such methods can be data inefficient and brittle in learning good controllers, although the performance guarantees they are meant to certify (e.g., exponential stability) are attractive. Unlike these methods, our method learns *intrinsic* structure in the dynamics to enable control, rather than simultaneously learning a parametric controller. Thus, an interesting avenue for future research lies in building system models that are *intrinsically stabilizable*. This could build off of existing work in parameterizing dynamics models in part by stability certificates such that they are stable by construction (Manek & Kolter, 2019; Revay et al., 2021), albeit for the controlled case.

Acknowledgements

We thank Masha Itkina for her invaluable feedback. We also thank the reviewers for their thoughtful input. This research was supported in part by 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 NASA or NSERC entity.

References

Abate, A., Ahmed, D., Giacobbe, M., and Peruffo, A. Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2021. doi: 10.1109/LCSYS.2020.3005328.

Acarman, T. Nonlinear optimal integrated vehicle control using individual braking torque and steering angle with on-line control allocation by using state-dependent Riccati equation technique. *Vehicle System Dynamics*, 47 (2):157–177, 2009. doi: 10.1080/00423110801932670.

Ailon, A. Simple tracking controllers for autonomous VTOL aircraft with bounded inputs. *IEEE Transactions on Automatic Control*, 55(3):737–743, 2010.

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

Banks, H. T., Beeler, S. C., Kepler, G. M., and Tran, H. T. Reduced order modeling and control of thin film growth in an HPCVD reactor. *SIAM Journal on Applied Mathematics*, 62(4):1251–1280, 2002.

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

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

Çimen, T. Systematic and effective design of nonlinear feedback controllers via the state-dependent Riccati equation (SDRE) method. *Annual Reviews in Control*, 34(1):32–51, 2010. doi: 10.1016/j.arcontrol.2010.03.001.

Çimen, T. Survey of state-dependent Riccati equation in nonlinear optimal feedback control synthesis. *AIAA Journal of Guidance, Control, and Dynamics*, 35(4): 1025–1047, 2012. doi: 10.2514/1.55821.

Chang, Y.-C. and Gao, S. Stabilizing neural control using self-learned almost Lyapunov critics. In *Proc. IEEE Conf. on Robotics and Automation*, 2021.

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

Cloutier, J. R. State-dependent Riccati equation techniques: An overview. In *American Control Conference*, 1997.

Cloutier, J. R. and Zipfel, P. H. Hypersonic guidance via the state-dependent Riccati equation control method. In *IEEE Conf. on Control Applications*, 1999.

Dawson, C., Qin, Z., Gao, S., and Fan, C. Safe nonlinear control using robust neural Lyapunov-barrier functions. In *Conf. on Robot Learning*, 2021.

Dawson, C., Gao, S., and Fan, C. Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods. Available at https://arxiv.org/abs/2202.11762, 2022.

Djeumou, F., Neary, C., Goubault, E., Putot, S., and Topcu, U. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics & Control Conference*, 2022.

Gupta, J. K., Menda, K., Manchester, Z., and Kochenderfer, M. J. Structured mechanical models for robot learning and control. In *Learning for Dynamics & Control Conference*, 2020.

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

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

Lew, T., Sharma, A., Harrison, J., Bylard, A., and Pavone, M. Safe active dynamics learning and control: A sequential exploration-exploitation framework. *IEEE Transactions on Robotics*, 38(5):2888–2907, 2022.

Li, Z., Duong, T., and Atanasov, N. Safe autonomous navigation for systems with learned SE(3) Hamiltonian dynamics. In *Learning for Dynamics & Control Conference*, 2022.

Lohmiller, W. and Slotine, J.-J. E. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

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

Manek, G. and Kolter, J. Z. Learning stable deep dynamics models. In *Conf. on Neural Information Processing Systems*, 2019.

Murray, R. M., Li, Z., and Sastry, S. S. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1 edition, 1994.

Revay, M., Wang, R., and Manchester, I. R. Recurrent equilibrium networks: Unconstrained learning of stable and robust dynamical models. In *Proc. IEEE Conf. on Decision and Control*, 2021. doi: 10.1109/CDC45484.2021.9683054.

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

Richards, S. M., Azizan, N., Slotine, J.-J., and Pavone, M. Adaptive-control-oriented meta-learning for nonlinear systems. In *Robotics: Science and Systems*, 2021.

Richards, S. M., Azizan, N., Slotine, J.-J., and Pavone, M. Control-oriented meta-learning. *Int. Journal of Robotics Research*, 2023. In press.

Sindhwani, V., Tu, S., and Khansari, M. Learning contracting vector fields for stable imitation learning. Available at https://arxiv.org/abs/1804.04878, 2018.

Singh, S., Landry, B., Majumdar, A., Slotine, J.-J. E., and Pavone, M. Robust feedback motion planning via contraction theory. *Int. Journal of Robotics Research*, 2019. Submitted.

Singh, S., Richards, S. M., Sindhwani, V., Slotine, J.-J. E., and Pavone, M. Learning stabilizable nonlinear dynamics with contraction-based regularization. *Int. Journal of Robotics Research*, 40(10–11):1123–1150, 2021.

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

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

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

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

Tsukamoto, H., Chung, S.-J., and Slotine, J.-J. E. Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control*, 52:135–169, 2021b.

Wächter, A. and Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

Watanabe, K., Iwase, M., Hatakeyama, S., and Maruyama, T. Control strategy for a snake-like robot based on constraint force and verification by experiment. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2008.

Zhang, L., Wu, S.-Y., and López, M. A. A new exchange method for convex semi-infinite programming. *SIAM Journal on Optimization*, 20(6):2959–2977, 2010.

Zhong, Y. D., Dey, B., and Chakraborty, A. Symplectic ODE-Net: Learning Hamiltonian dynamics with control. In *Int. Conf. on Learning Representations*, 2020.

Zhou, R., Quartz, T., De Sterck, H., and Liu, J. Neural Lyapunov control of unknown nonlinear systems with stability guarantees. In *Conf. on Neural Information Processing Systems*, 2022.

A. Hyperparameters and Implementation Details

Physical Parameters For the spacecraft, we set its mass to $m = 0.5$, rotational moment of inertia to $J = 0.005$, and its center-of-mass offset to $(d_x, d_y) = (0.1, 0.1)$. For the PVTOL, we set the its mass to $m = 0.5$, arm length to $\ell = 0.25$, rotational moment of inertia to $J = 0.005$, and gravitational acceleration to $g = 9.81$.

Hyperparameters Each function in $(f, B, M, K, A_0, \{A_j\}_{j=1}^m)$ is approximated as a feedforward neural network with two hidden layers and 128 hidden $\tanh$ activation units per layer. We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-3} and otherwise default hyperparameters. Training is performed for 50000 epochs while the loss on a held-out validation set of size $0.10N$ is monitored, where N is the size of the labelled training data set. For each method, the model parameters corresponding to the lowest validation loss are chosen for testing.

For the CCM-based learning method, since Equation (18) is homogeneous in $M(x)$, we choose $\underline{\lambda} = 0.1$ without loss of generality. Additionally, we fix the overshoot $\alpha = 10$ and the decay rate $\beta = 0.5$ in the auxiliary loss Equation (27). For both the CCM and SDC learning methods, we use $N_{\text{aux}}^{\text{CCM}} = N_{\text{aux}}^{\text{SDC}} = 10000$ unlabelled samples.

Sampling For sampling states and inputs, we draw uniformly from bounded sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$, respectively. For the spacecraft, we use

$$\begin{aligned} \mathcal{X} &= \{x \in \mathbb{R}^6 \mid -c \preceq x \preceq c,\ c \coloneqq (1, 1, \pi, 0.2, 0.2, 0.25)\} \\ \mathcal{U} &= \{u \in \mathbb{R}^3 \mid -c \preceq u \preceq c,\ c \coloneqq (1, 1, 0.1)\} \end{aligned}. \tag{30}$$

For the PVTOL, we use

$$\begin{aligned} \mathcal{X} &= \{x \in \mathbb{R}^6 \mid -c \preceq x \preceq c,\ c \coloneqq (10, 10, \pi/3, 2, 1, \pi/3)\} \\ \mathcal{U} &= \{u \in \mathbb{R}^2 \mid (0.1mg, 0.1mg) \preceq u \preceq (2mg, 2mg)\} \end{aligned}, \tag{31}$$

where m and g are the vehicle mass and gravitational acceleration, respectively. We also use $\mathcal{U}$ to define the control bounds in the optimal control problem for generating test trajectories.

Testing When generating test trajectories with the optimal control problem in Section 5, we use $Q = I$ and $R = 0.01I$ in the cost function for both systems. For simulating the linearized and state-dependent LQR tracking controllers, we use $Q = I$ and $R = I$ in their corresponding Riccati equations.