Learning Stabilizable Nonlinear Dynamics with Contraction-Based Regularization

Sumeet Singh[1], Spencer M. Richards[1], Vikas Sindhwani[2], Jean-Jacques E. Slotine[3], and Marco Pavone[1]

[1]Department of Aeronautics and Astronautics, Stanford University [*]
[2]Google Brain Robotics, New York [†]
[3]Department of Mechanical Engineering, Massachusetts Institute of Technology [‡]

December 7, 2020

Abstract

We propose a novel framework for learning stabilizable nonlinear dynamical systems for continuous control tasks in robotics. The key contribution is a control-theoretic regularizer for dynamics fitting rooted in the notion of *stabilizability*, a constraint which guarantees the existence of robust tracking controllers for arbitrary open-loop trajectories generated with the learned system. Leveraging tools from contraction theory and statistical learning in reproducing kernel Hilbert spaces, we formulate stabilizable dynamics learning as a functional optimization with a convex objective and bi-convex functional constraints. Under a mild structural assumption and relaxation of the functional constraints to sampling-based constraints, we derive the optimal solution with a modified representer theorem. Finally, we utilize random matrix feature approximations to reduce the dimensionality of the search parameters and formulate an iterative convex optimization algorithm that jointly fits the dynamics functions and searches for a *certificate of stabilizability*. We validate the proposed algorithm in simulation for a planar quadrotor, and on a quadrotor hardware testbed emulating planar dynamics. We verify, both in simulation and on hardware, significantly improved trajectory generation and tracking performance with the control-theoretic regularized model over models learned using traditional regression techniques, especially when learning from small supervised datasets. The results support the conjecture that the use of stabilizability constraints as a form of regularization can help prune the hypothesis space in a manner that is tailored to the downstream task of trajectory generation and feedback control. This produces models that are not only dramatically better conditioned, but also data efficient.

1 Introduction

The problem of efficiently and accurately estimating an unknown dynamical system,

$$\dot{x}(t) = F(x(t), u(t)), \tag{1}$$

[*]ssingh19@alumni.stanford.edu, {spenrich,pavone}@stanford.edu
[†]sindhwani@google.com
[‡]jjs@mit.edu

from a small set of sampled trajectories, where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the control input, is a central task in model-based Reinforcement Learning (RL). In this setting, a robotic agent strives to pair an estimated dynamics model with a feedback policy in order to act optimally in a dynamic and uncertain environment. The model of the dynamical system can be continuously updated as the robot experiences the consequences of its actions, and the improved model can be leveraged for different tasks, affording a natural form of transfer learning. When it works, model-based RL typically offers major improvements in sample efficiency in comparison to state-of-the-art model-free methods such as policy gradients (Chua et al., 2018; Nagabandi et al., 2017) that do not explicitly estimate the underlying system. Yet, all too often, when standard supervised learning with powerful function approximators such as deep neural networks and kernel methods are applied to model complex dynamics, the resulting controllers do not perform on par with model-free RL methods in the limit of increasing sample size, due to compounding errors across long time horizons. The main goal of this paper is to develop a new control-theoretic regularizer for dynamics fitting rooted in the notion of *stabilizability*, which guarantees the existence of a robust tracking controller for arbitrary open-loop trajectories generated with the learned system.

Problem Statement: The motion planning task we wish to solve is to compute a (possibly non-stationary) policy mapping state and time to control that drives any given initial state to a desired compact goal region, while satisfying state and control input constraints, and minimizing some task specific performance cost (e.g., control effort and time to completion). However, in this work, we assume that the dynamics function $F(x, u)$ is unknown to us and we are instead provided with a dataset of tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$ taken from a collection of observed trajectories (e.g., expert demonstrations) on the robot. Accordingly, the objective of this work is to learn a dynamics model $\hat{F}(\cdot, \cdot)$ for the robot that is subsequently amenable for use within standard planning algorithms.

Approach Overview: Our parameterization of the policy takes the form $u^*(t) + k(x^*(t), x(t))$, where (x^*, u^*) is a nominal open-loop state-input control trajectory tuple, and $k(\cdot, \cdot)$ is a feedback tracking controller. The performance of such a policy however, is strongly reliant upon the quality of the computed state-input trajectory and the tracking controller.

Formally, a reference state-input trajectory tuple $(x^*(t), u^*(t)),\ t \in [0, T]$ for system (1) is termed *exponentially stabilizable at rate* $\lambda > 0$ if there exists a feedback controller $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ such that the solution $x(t)$ of the system:

$$\dot{x}(t) = F(x(t), u^*(t) + k(x^*(t), x(t))),$$

converges exponentially to $x^*(t)$ at rate λ. That is,

$$\|x(t) - x^*(t)\|_2 \leq C \|x(0) - x^*(0)\|_2 \, e^{-\lambda t} \tag{2}$$

for some constant $C > 0$. The *system* (1) is termed *exponentially stabilizable at rate* λ in an open, connected, bounded region $\mathcal{X} \subset \mathbb{R}^n$ if all state trajectories $x^*(t)$ satisfying $x^*(t) \in \mathcal{X},\ \forall t \in [0, T]$ are exponentially stabilizable at rate λ.

In this work, we illustrate that naïve regression techniques used to estimate the dynamics model from a small set of sample trajectories can yield model estimates that are severely ill-conditioned for trajectory generation and feedback control. Instead, this work advocates for the use of a *constrained* regression approach in which one attempts to solve the following problem:

$$\min_{\hat{F}\in\mathcal{H}} \quad \sum_{i=1}^{N} \left\| \hat{F}(x_i, u_i) - \dot{x}_i \right\|_2^2 + \mu \|\hat{F}\|_{\mathcal{H}}^2 \tag{3}$$

$$\text{s.t.} \quad \hat{F} \text{ is stabilizable,} \tag{4}$$

where $\mathcal{H}$ is an appropriate normed function space and $\mu > 0$ is a regularization parameter. Note that we use $(\hat{\cdot})$ to differentiate the learned dynamics from the true dynamics. We demonstrate that for systems that are indeed stabilizable, enforcing such a constraint drastically *prunes the hypothesis space*, and therefore plays the role of a *"control-theoretic" regularizer* that is potentially more powerful and ultimately more pertinent for the downstream control task of generating and tracking new trajectories.

Statement of Contributions: Stabilizability of trajectories is not only a complex task in non-linear control, but also a difficult notion to capture (in an algebraic sense) within a unified control theory. In this work, we leverage recent advances in contraction theory for control design through the use of *Control Contraction Metrics* (CCMs) (Manchester and Slotine, 2017; Singh et al., 2017) that turn stabilizability constraints into convex state-dependent Linear Matrix Inequalities (LMIs). Contraction theory (Lohmiller and Slotine, 1998) is a method of analyzing nonlinear systems in a differential framework, i.e., via the associated variational system (Crouch and van der Schaft, 1987, Chp 3), and is focused on the study of convergence between pairs of state trajectories towards each other. Thus, at its core, contraction explores a stronger notion of stability – that of incremental stability between solution trajectories, instead of the stability of an equilibrium point or invariant set. Importantly, we harness recent results in (Manchester et al., 2015; Manchester and Slotine, 2017; Singh et al., 2017) that illustrate how to use contraction theory to obtain a *certificate* for trajectory stabilizability and an accompanying tracking controller with exponential stability properties. For self-containment, we provide a brief summary of these results in Section 3, which in turn will form the foundation of this work.

Our paper makes the following primary contributions.

- We formulate the learning stabilizable dynamics problem through the lens of CCMs (Section 4). The resulting optimization problem is not only infinite-dimensional, as it is formulated over function spaces, but also infinitely-constrained due to the state-dependent LMI representing the stabilizability constraint.

- Under an arguably weak assumption on the structural form of the true dynamics model and a relaxation of the functional constraints to sampling-based constraints (Section 5), we derive a representer theorem (Scholköpf and Smola, 2001) specifying the form of the optimal solutions for the dynamics functions and the certificate of stabilizability by leveraging the powerful framework of vector-valued Reproducing Kernel Hilbert Spaces (RKHS) (Section 6). We motivate the sampling-based relaxation of the functional constraints from a standpoint of viewing the stabilizability condition as a novel control-theoretic *regularizer* for dynamics learning.

- With theory from randomized matrix feature approximations, we derive a tractable algorithm leveraging alternating convex optimization problems and adaptive sampling to iteratively solve a *finite-dimensional* optimization problem (Section 7).

- We perform an extensive set of numerical simulations on a 6-state, 2-input planar quadrotor model and provide a comprehensive study of various aspects of the iterative algorithm. Specifically, we demonstrate that naïve regression-based dynamics learning can yield estimated models that generate completely unstabilizable trajectories. In contrast, the control-theoretic regularized model generates vastly superior quality trackable trajectories, especially when learning from small supervised datasets (Sections 2.1 and 7.2).
- We validate our algorithm on a quadrotor testbed (Section 8) with partially closed control loops to emulate a planar quadrotor, where we verify that the stabilizability regularization effect in low-data regimes observed in simulations does indeed generalize to real-world noisy data. In particular, with just 150 noisy tuples of $(x, u, \dot{x})$, we are able to stably track a challenging test trajectory, which is generated with the learned model and substantially different from any of the training data. In contrast, a model learned using traditional regression techniques leads to consistently unstable behavior and eventual failure as the quadrotor repeatedly flips out of control and crashes (see Figure 1).

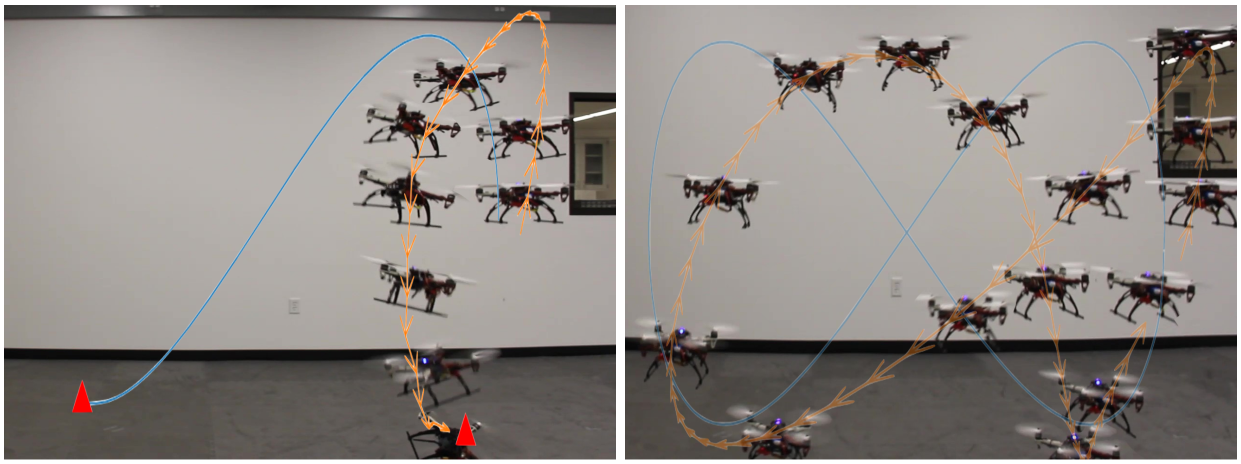


Figure 1: Time-lapse of a quadrotor trying to execute a figure-eight maneuver (blue curve) using a reference trajectory and a Linear-Quadratic Regulator (LQR) feedback tracking controller generated using the learned dynamical system. *Left*: Model learned using traditional ridge-regression; *Right*: Model learned using control-theoretic regularization proposed within this work. The models were trained with the same, extremely limited (150 points) set of $(x, u, \dot{x})$ supervisory tuples. The quadrotor consistently failed and crashed into the floor with the trajectory and controller generated by the model learned with ridge-regression; the red triangles mark the points along the reference and actual trajectories at moment of crash – a separation of 1.6 m. In contrast, despite imperfect tracking (not unexpected given the extremely limited amount of supervision given to the learning algorithm), which leads to a slight graze along the floor at one point during the maneuver, the quadrotor manages to maintain bounded tracking error while using the model learned with control-theoretic regularization.

A preliminary version of this paper was presented at WAFR 2018 (Singh et al., 2018). In this revised and extended version, we include the following additional contributions: (i) rigorous derivation of the stabilizability-regularized finite-dimensional optimization problem using RKHS theory and random matrix features; (ii) extensive additional numerical studies into the convergence behavior of the iterative algorithm and comparison with traditional ridge-regression techniques; and (iii) validation of the algorithm on a quadrotor testbed with partially closed control loops to emulate a planar quadrotor.

Related Work: Model-based RL has enjoyed considerable success in various application domains within robotics such as underwater vehicles (Cui et al., 2017), soft robotic manipulators (Thuruthel et al., 2019), and control of agents with non-stationary dynamics (Ohnishi et al., 2019). While the literature on model-based RL is substantial (see (Polydoros and Nalpantidis, 2017) for a recent review), we focus our attention on five broad categories relevant to the problem we address in this work. Namely, these are: (i) direct regression for learning the full dynamics, where one ignores any control-theoretic notions tied to the learning task and treats dynamics estimation as a standard regression problem; (ii) residual learning, where one only attempts to learn *corrections* to a nominal prediction model that may have been derived, for example, from physics-based reasoning; (iii) uncertainty-aware model-based RL, where one tries to additionally represent the uncertainty in the learned model using probabilistic representations that are subsequently leveraged within the planning phase using robust or stochastic control techniques; (iv) hybrid model-based/model-free methods; and (v) imitation learning, where one learns dynamical representations of stable closed-loop behavior for a set of outputs (e.g., the end-effector on a robotic arm), and assumes knowledge of the robot controlled dynamics to realize the learned closed-loop motion, for instance, using dynamics inversion.

The simplest approach to learning dynamics is to ignore stabilizability and treat the problem as a standard one-step time series regression task (Punjani and Abbeel, 2015; Bansal et al., 2016; Nagabandi et al., 2017; Polydoros and Nalpantidis, 2017). However, coarse dynamics models trained on limited training data typically generate trajectories that rapidly diverge from expected paths, inducing controllers that are ineffective when applied to the true system. This divergence can be reduced by expanding the training data with corrections to boost multi-step prediction accuracy (Venkatraman et al., 2015, 2016). Despite being effective, these methods are still heuristic in the sense that the existence of a stabilizing feedback controller is not explicitly guaranteed. Alternatively, one can leverage strong physics-based priors and use learning to only regress the unmodeled dynamics. For instance, (Mohajerin et al., 2019; Shi et al., 2019; Punjani and Abbeel, 2015) aim to capture the unmodeled aerodynamic disturbance terms as corrections to a prior rigid body dynamics model. (Punjani and Abbeel, 2015) accomplish this for helicopter dynamics using a deep neural network, but then do not use the learned model for control. (Shi et al., 2019) attempt to capture the unmodeled ground-effect forces on quadrotors to build better controllers for near-ground tracking and precision landing. (Mohajerin et al., 2019) leverage a residual RNN in combination with a rigid-body model to generate time-series predictions for linear and angular velocities of a quadrotor as a function of current state and candidate future motor inputs, but do not use the model for closed-loop control. Finally, (Zhou et al., 2017) adopt a different perspective to learning "corrections" in that they attempt to learn the inverse dynamics (output to reference) for a system and pre-cascade the resulting predictions to correct an existing controller's reference signal in order to improve trajectory tracking performance. The approach relies on the existence of a stabilizing controller and the stability of the system's zero dynamics, thereby decoupling the effects of learning from stability. In similar spirit, (Taylor et al., 2019) leverage input-output feedback linearization to derive a Control Lyapunov Function (CLF) for a nominal dynamics model, assume that this function is a CLF for the actual dynamics as well, and regress only the correction terms in the derivative of this CLF. While leveraging physics-based priors can certainly be powerful, especially when the residual errors to be learned are small enough such that the system is feedback stabilizable with a controller derived from the physics model, in this work we are interested in the far more challenging scenario when such priors are unavailable and the full dynamics model must be learned from scratch. While exemplified using quadrotor models that can certainly be accurately stabilized even in the absence of learning, the insights provided in this work shed light on fundamental topics in the context of control-theoretic learning, which hopefully may influence dynamics-learning methods in more complex settings where priors are unavailable or too simple to be useful for adequate control.

An alternative strategy to cope with error in the learned dynamics model is to use uncertainty-aware model-based RL where control policies are optimized with respect to stochastic rollouts from probabilistic dynamics models (Kocijan et al., 2004; Kamthe and Deisenroth, 2018; Deisenroth and Rasmussen, 2011; Chua et al., 2018). For instance, PILCO (Deisenroth and Rasmussen, 2011) leverages a Gaussian Process (GP) state transition model and moment matching to analytically estimate the expected cost of a rollout with respect to the induced distribution. (Kamthe and Deisenroth, 2018) extend this formulation using nonlinear Model Predictive Control (MPC) to incorporate chance constraints. (Chua et al., 2018) leverage an ensemble of probabilistic models to capture both epistemic (i.e., model) and aleatoric (i.e., intrinsic) uncertainty, and compute their control policy in receding horizon fashion through finite sample approximation of the random cost. Probabilistic models such as GPs may also be used to capture the residual error between a nominal physics-based model and the true dynamics. In (Ostafew et al., 2016), a GP is incrementally learned over multiple trials to capture unmodeled disturbances. The 3σ prediction range is subsequently leveraged to formulate chance constraints as a robust nonlinear MPC problem. The goal of (Fisac et al., 2017) and (Berkenkamp et al., 2017) is motivated from a safety perspective, where one wishes to actively learn a control policy while remaining "safe" in the presence of unmodeled dynamics, represented as GPs. The authors in (Fisac et al., 2017) leverage Hamilton-Jacobi reachability analysis to give high-probability invariance guarantees for a region of the state-space within which the learning controller is free to explore. On the other hand, (Berkenkamp et al., 2017) utilize Lyapunov analysis and smoothness arguments to incrementally grow the Lyapunov function's region of attraction while simultaneously updating the GP. For the special case where the underlying dynamics are linear-time-invariant, (Dean et al., 2019) derive high-probability convergence rates for the estimated model and leverage system-level robust control techniques (Wang et al., 2019) for guaranteeing state and control constraint satisfaction.

While utilizing probabilistic prediction models along with a control strategy that incorporates this uncertainty, such as robust or approximate stochastic MPC, can certainly help guard against imperfect dynamics models, large uncertainty in the dynamics can lead to overly conservative strategies. This is true especially when the learned model is not merely a correction or residual term, or if the probabilistic model is computationally intractable to use within planning (e.g., GPs without additional sparsifying simplifications), thereby forcing conservative approximations. Finally, with the exception of the "safe" RL methods mentioned above, the learning algorithms themselves do not incorporate knowledge of the downstream application of the function being regressed, in that learning is viewed purely from a *statistical point-of-view, rather than within a control-theoretic context.*

More recently, hybrid combinations of model-based and model-free techniques have gained attention within the learning community. The authors in (Bansal et al., 2017) use Bayesian optimization to find an optimal linear dynamics model whose induced MPC policy minimizes the task-specific cost. In similar spirit, (Amos et al., 2018) differentiate through the fixed-point solutions of a parametric MPC problem to find optimal MPC cost and dynamics functions in order to minimize the actual task-specific cost. (Nagabandi et al., 2017) use behavioral cloning with respect to an MPC policy generated from a learned dynamics model to initialize model-free policy fine-tuning. The works in (Levine et al., 2016; Finn et al., 2016; Chebotar et al., 2017) leverage subroutines where local time-varying dynamics are fitted around a set of policy rollouts, and then used to perform trajectory optimization via an LQR backward pass. The induced local linear-time-varying policy from this rollout is then used as a supervisory signal for global policy optimization. While these lines of work try to frame dynamics fitting within the downstream context of the task, thereby imbuing the resulting learning algorithm with a more closed-loop flavor, the learned dynamics may be substantially different from the actual dynamics of the robot since, with the exception of the local time-varying dynamics fitting, the true goal is to optimize the task-specific cost. This can yield distorted dynamic models whose induced policies are more cost-optimal than policies extracted from the true dynamics. Thus, while the work presented herein espouses a closed-loop learning ideology, it does so from the control-theoretic perspective of trajectory stabilizability, i.e., the true objective is dynamics fitting which will subsequently be used to derive optimal trajectories and tracking controllers.

Finally, we address lines of work closest in spirit to this work. Learning dynamical systems satisfying some desirable stability properties (such as asymptotic stability about an equilibrium point, e.g., for point-to-point motion) has been studied in the autonomous case, $\dot{x}(t) = f(x(t))$, in the context of imitation learning. In this line of work, one assumes perfect knowledge and invertibility of the robot's *controlled* dynamics to solve for the input that realizes this desirable closed-loop motion (Lemme et al., 2014; Khansari-Zadeh and Khatib, 2017; Ravichandar et al., 2017; Khansari-Zadeh and Billard, 2011; Medina and Billard, 2017). In particular, for a vector-valued RKHS formulation in the autonomous case with constant (identity) contraction metric, see (Sindhwani et al., 2018). Crucially, in our work, we *do not* require knowledge or invertibility of the robot's controlled dynamics. We seek to learn the full controlled dynamics of the robot, under the constraint that the resulting learned dynamics generate dynamically feasible and most importantly, stabilizable trajectories. Thus, this work generalizes existing literature by additionally incorporating the controllability limitations of the robot within the learning problem.

The tools we develop may also be used to extend standard adaptive robot control design, such as (Slotine and Li, 1987) – a technique which achieves stable concurrent learning and control using a combination of physical basis functions and general mathematical expansions, e.g. radial basis function approximations (Sanner and Slotine, 1992). Notably, our work allows us to handle complex underactuated systems – a consequence of the significantly more powerful function approximation framework developed herein, as well as of the use of a differential (rather than classical) Lyapunov-like setting, as we shall detail.

Notation: Let $\mathbb{S}_j$ be the set of symmetric matrices in $\mathbb{R}^{j\times j}$ and denote $\mathbb{S}_j^{\geq 0}$, respectively $\mathbb{S}_j^{>0}$, to be the set of symmetric positive semi-definite, respectively, positive definite matrices in $\mathbb{R}^{j\times j}$. Given a matrix X, let $\widehat{X} := X + X^T$. We denote the components of a vector $y \in \mathbb{R}^n$ as y^j, $j = 1, \ldots, n$, its Euclidean norm as $\|y\|$, and its weighted norm as $\|y\|_A$ where $\|y\|_A = \sqrt{y^T A y}$ for some $A \in \mathbb{S}_n^{>0}$. Let $\partial_y G(x)$ denote a matrix with $(i,j)^{\text{th}}$ entry given by the Lie derivative of the function $G_{ij}(x)$ along the vector y. Finally, let $\bar{\lambda}(A)$ and $\underline{\lambda}(A)$ denote the maximum and minimum eigenvalues of a square matrix A.

2 Problem Formulation and Solution Methodology

In this section we formally outline the structure of the problem we wish to solve and describe a general solution methodology rooted in model-based RL. To motivate the contributions of this work, we additionally present an attempt at a solution that uses traditional model-fitting techniques, and demonstrate how it fails to capture the nuances of the problem and ultimately yields sub-par results.

Consider a robotic system with state $x \in \mathcal{X}$, where $\mathcal{X}$ is an open, connected, bounded subset of $\mathbb{R}^n$, and control $u \in \mathcal{U}$, where $\mathcal{U}$ is a closed, bounded subset of $\mathbb{R}^m$, governed by the following continuous-time dynamical system:

$$\dot{x}(t) = F(x(t), u(t)),$$

where F is Lipschitz continuous in the state for fixed control, so that for any measurable control function $u(\cdot)$, there exists a unique state trajectory. The motion planning *task* we wish to solve is to find a (possibly non-stationary) policy $\pi : \mathcal{X} \times \mathbb{R} \to \mathcal{U}$ that (i) drives the state x to a compact region $\mathcal{X}_{\text{goal}} \subseteq \mathcal{X}$, (ii) satisfies the state and input constraints, and (iii) minimizes a quadratic cost:

$$J(\pi) := \int_0^{T_{\text{goal}}} 1 + \|\pi(x(t), t)\|_R^2 \, dt,$$

where $R \in \mathbb{S}_m^{>0}$ and T_{goal} is the first time $x(t)$ enters $\mathcal{X}_{\text{goal}}$. While there exist several methods in the literature on how to solve this problem given knowledge of the dynamical system, in this work, we assume that we do not know the governing model $F(x, u)$. The *problem* we wish to address is how to solve the above motion planning task, given a dataset of tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$ extracted from observed trajectories on the robot.

The solution approach presented in this work adopts the model-based RL paradigm, whereby one first estimates a model of the dynamical system $\hat{F}(x, u)$ using some form of regression, and then uses the learned model to solve the motion planning task with traditional planning algorithms. In this work, our strategy to solve the planning task is to parameterize general state-feedback policies as a sum of a nominal (open-loop) input u^* and a feedback term designed to track the nominal state trajectory x^* (induced by u^*):

$$\pi(x(t),\, t) = u^*(t) + k(x^*(t), x(t)). \tag{5}$$

This formulation represents a compromise between the general class of state-feedback control laws (a computationally intractable space over which to optimize) and a purely open-loop formulation (i.e., no tracking). Note that we do *not* present a new methodology for solving the planning task. Specifically, it is assumed that there exists an algorithm for computing (i) the open-loop state and control trajectories $(x^*(t), u^*(t))$ that minimize the open-loop cost:

$$J(u(\cdot)) = \int_0^{T_{\text{goal}}} 1 + \|u(t)\|_R^2 \, dt,$$

and (ii) the feedback tracking controller $k(\cdot, \cdot)$, given a dynamical model. The focus of this paper is on how to design the regression algorithm for computing the model estimate $\hat{F}$.

2.1 Motivating Example

We ground the formalism within the following example that will feature throughout this work.

Example 1 (PVTOL). *Consider the six-state Planar Vertical Take-Off and Landing (PVTOL) system depicted in Figure 2. The system is defined by the state* $(p_x, p_z, \phi, v_x, v_z, \dot{\phi})$*, where* (p_x, p_z) *is the position in the 2D plane,* (v_x, v_z) *is the body-reference velocity, and* $(\phi, \dot{\phi})$ *are the roll and angular rate respectively, and* $u \in \mathbb{R}^2_{>0}$ *are the controlled motor thrusts. The true dynamics are given by:*

$$\dot{x}(t) = \begin{bmatrix} v_x \cos\phi - v_z \sin\phi \\ v_x \sin\phi + v_z \cos\phi \\ \dot{\phi} \\ v_z\dot{\phi} - g\sin\phi \\ -v_x\dot{\phi} - g\cos\phi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ (1/m) & (1/m) \\ l/J & (-l/J) \end{bmatrix} u, \tag{6}$$

where g *is the acceleration due to gravity,* m *is the mass,* l *is the moment-arm of the thrusters, and* J *is the moment of inertia about the roll axis.*

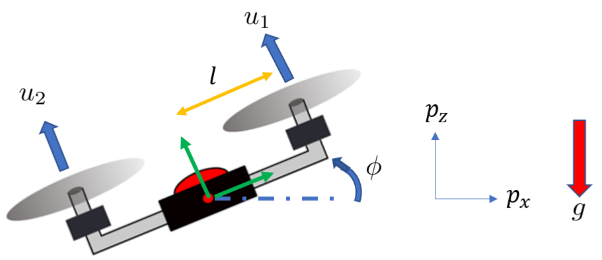


Figure 2: Definition of planar quadrotor state variables: l denotes the thrust moment arm (symmetric), and u_1 and u_2 denote the right and left thrust forces respectively.

The planar quadrotor is a complex non-minimum phase dynamical system that has been heavily featured within the acrobatic robotics literature and therefore serves as a suitable case-study.

2.1.1 Solution Parameterization

The dynamics assume the general control-affine form:

$$\dot{x}(t) = f(x(t)) + B(x(t))u(t), \tag{7}$$

where $f : \mathcal{X} \to \mathbb{R}^n$, and $B : \mathcal{X} \to \mathbb{R}^{n\times m}$ is the input matrix, depicted in column-stacked form as $(b_1, \ldots, b_m)$. Let us define the model estimate also in control-affine form as $\dot{x} = \hat{f}(x) + \hat{B}(x)u$, where $\hat{B} = (\hat{b}_1, \ldots, \hat{b}_m)$. Consider, as a first solution attempt, the following linear parameterization for the vector-valued functions $\hat{f}$ and $\hat{b}_j$:

$$\hat{f}(x) = \Phi_f(x)^T\alpha, \tag{8a}$$

$$\hat{b}_j(x) = \Phi_b(x)^T\beta_j \quad j \in \{1, \ldots, m\}, \tag{8b}$$

where $\alpha \in \mathbb{R}^{d_f}$, $\beta_j \in \mathbb{R}^{d_b}$ are constant vectors to be optimized over, and $\Phi_f : \mathcal{X} \to \mathbb{R}^{d_f\times n}$, $\Phi_b : \mathcal{X} \to \mathbb{R}^{d_b\times n}$ are a priori chosen feature mappings. To replicate the sparsity structure of the PVTOL input matrix, the feature matrix Φ_b has all zeros in its first $n - m$ columns.

The justification for a linear model and the construction of the feature mappings will be elaborated upon later. At this moment, we wish to study the quality of the learned models obtained from solving the following convex optimization problem:

$$\min_{\alpha,\{\beta_j\}} \quad \sum_{i=1}^{N} \|\hat{f}(x_i) + \hat{B}(x_i)u_i - \dot{x}_i\|^2 + \mu_f\|\alpha\|^2 + \mu_b \sum_{j=1}^{m} \|\beta_j\|^2, \tag{9}$$

where $\mu_f, \mu_b > 0$ are given regularization constants. Note that the above optimization corresponds to the ubiquitous ridge-regression problem and is therefore a viable solution approach.

To evaluate the feasibility of this solution approach, we extracted a collection of training tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^{N}$ from simulations of the PVTOL system without any noise (for further details, please see Section 7.2). We learned three models: (i) **N-R**: un-regularized model[1] ($\mu_f = 0$, $\mu_b = 10^{-6}$), (ii) **R-R**: standard ridge-regularized model with $\mu_f = 10^{-4}$, $\mu_b = 10^{-6}$, and (iii) **CCM-R**: control-theoretic regularized model, corresponding to the algorithm proposed within this work and elaborated upon in the remaining paper.

We learned four versions of the model corresponding to varying training dataset sizes with $N \in \{100, 250, 500, 1000\}$. The dimensions of α and β_j were both 576 (corresponding to 96 parameters per state dimension). The feature mappings themselves are described in Section 7.2 and Appendix A. The regularization constants were held fixed for all N.

2.1.2 Evaluation

The evaluation corresponded to the motion planning task of generating and tracking trajectories using the learned models. We gridded the (p_x, p_z) plane to create a set of 120 initial conditions between 4 m and 12 m away from $(0, 0)$, and randomly sampled the other states for the rest of the initial conditions. These conditions were *held fixed* for all models and for all training dataset sizes to evaluate model improvement.

For each model at each value of N, the evaluation task was to (i) solve a trajectory optimization problem to compute a dynamically feasible trajectory for the learned model to go from initial state x_0 to the goal state – a stable hover at $(0, 0)$ at near-zero velocity; and (ii) track this trajectory with a feedback controller computed using Time-Varying LQR (TV-LQR). All simulations without any feedback controller (i.e., open-loop control rollouts) led to the PVTOL crashing. This is understandable since the dynamics fitting objective does not optimize for *multi-step* error. The trajectory optimization step was solved as a fixed-endpoint, fixed-final time optimal control problem using the Chebyshev pseudospectral method (Fahroo and Ross, 2002) with the objective of minimizing $\int_0^T \|u(t)\|_R^2 \, dt$. The final time T for a given initial condition was held fixed between all models. Note that 120 trajectory optimization problems were solved for each model and each value of N.

Figure 3 shows a boxplot comparison of the trajectory-wise RMS full state errors ($\|x(t) - x^*(t)\|$ where $x^*(t)$ is the reference trajectory obtained from the optimizer and $x(t)$ is the actual realized trajectory) for each model and all training dataset sizes. As N increases, the spread of the RMS errors decreases for both R-R and CCM-R models as expected. However, we see that the N-R model generates *several* unstable trajectories for $N \in \{100, 500, 1000\}$, indicating the need for a form of regularization.

[1] A small penalty on μ_b was necessary since the input feature matrix Φ_b was chosen to be constant (and degenerate) to reflect the fact that the input matrix for the PVTOL system is constant.

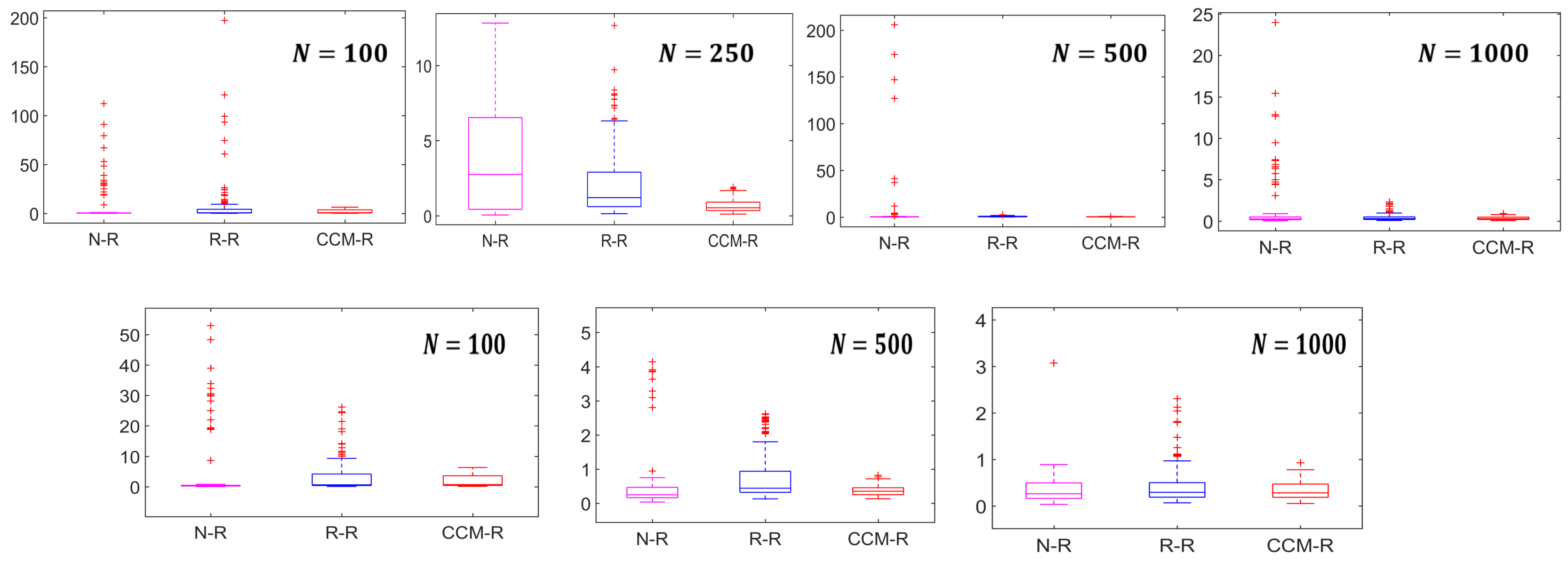


Figure 3: Box-whisker plot comparison of trajectory-wise RMS state-tracking errors over all 120 trajectories for each model and all training dataset sizes. *Top row, left-to-right:* $N = 100, 250, 500, 1000$; *Bottom row, left-to-right:* $N = 100, 500, 1000$ (zoomed in). The box edges correspond to the 25th, median, and 75th percentiles; the whiskers extend beyond the box for an additional 1.5 times the interquartile range; outliers, classified as trajectories with RMS errors past this range, are marked with red crosses. Notice the presence of unstable trajectories for N-R at all values of N and for R-R at $N = 100, 250$. The CCM-R model dominates the other two *at all values of* N, particularly for $N \in \{100, 250\}$.

For $N = 100$ (which is at the extreme lower limit of the necessary number of samples since there are 96 features for each dimension of the dynamics function), both N-R and R-R models generate a large number of unstable trajectories. In contrast, *all* trajectories generated with the CCM-R model were successfully tracked with bounded error. The CCM-R model consistently achieves a lower RMS error distribution than both the N-R and R-R models *for all training dataset sizes*. Most notable, however, is its performance when the number of supervision training samples is small (i.e., $N \in \{100, 250\}$) and there is considerable risk of overfitting. It appears the stabilizability constraints leveraged to compute the CCM-R model have a notable regularizing effect on the resulting model trajectories (recall that the initial conditions of the trajectories are held fixed between the models). In Figure 4, we highlight two trajectories that start from the *same initial conditions* – one generated and tracked using the R-R model, the other using the CCM-R model, for $N = 250$. Overlaid on the plot are snapshots of the vehicle outline itself, illustrating the aggressive flight-regime of the trajectories (the initial bank angle is 40^o). While tracking the R-R model generated trajectory eventually ends in complete loss of control, the system successfully tracks the CCM-R model generated trajectory to the stable hover at $(0, 0)$.

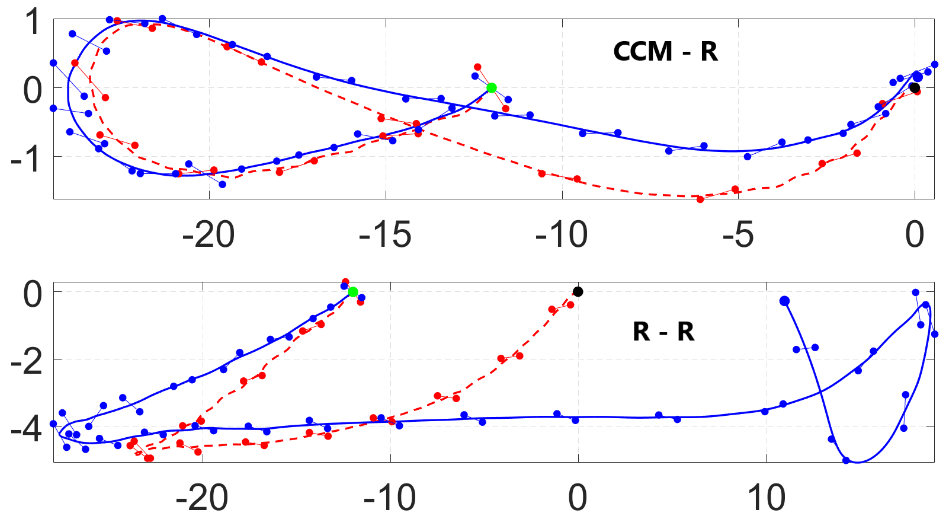


Figure 4: Comparison of reference and tracked trajectories in the (p_x, p_z) plane for R-R and CCM-R models starting at the same initial conditions with $N = 250$. Red (dashed): nominal, Blue (solid): actual, Green dot: start, Black dot: nominal endpoint, blue dot: actual endpoint; *Top:* CCM-R, *Bottom:* R-R. The vehicle successfully tracks the CCM-R model generated trajectory to the stable hover at $(0, 0)$ while losing control when attempting to track the R-R model generated trajectory.

2.1.3 Effect of Regularization

At this point, one might wonder if the choice of the regularization parameter μ_f may be sub-optimal for the R-R model. Traditionally, such parameters are tuned using regression error on a validation dataset. In Figure 5 we plot the mean regression error $\|\hat{f} + \hat{B}u - \dot{x}\|$ over an independently sampled validation dataset of 2000 demonstration tuples, as a function of the regularization parameter μ_f, for all R-R models.

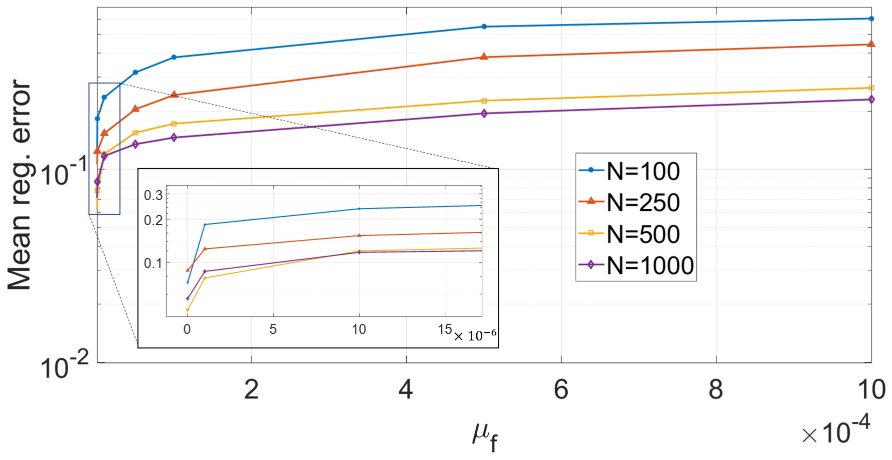


Figure 5: Mean regression error over an independent validation dataset as a function of μ_f for the RR model learned using (9), with varying training set size N. The best out of sample performance is achieved with $\mu_f = 0.0$. The constant μ_b is fixed at 10^{-6}.

The plot illustrates that the best out-of-sample performance is achieved with $\mu_f = 0.0$. However, this corresponds to the N-R model which, as we learned in the previous section, generated several unstable trajectories for all training dataset sizes. This is not a surprising result; the feature mapping used as the basis for the dynamics model corresponds to the randomized matrix approximation of a reproducing kernel (see Section 6.4). Recent results, as in (Liang and Rakhlin, 2018) and references within, corroborate such a pattern and even advocate for "ridgeless" regression.

Given that the CCM-R model uses the *same* feature mapping as the R-R and N-R models (i.e., the model capacity of all three models is the same), and is given the same set of demonstration tuples, it appears that traditional model-fitting techniques such as ridge-regression and associated hyper-parameter tuning rules are ill-suited to learn representations of dynamics that are appropriate for planning and control. This motivates the need for *constrained* dynamics learning, where the notion of *model stabilizability* is encoded as a constraint within the learning algorithm (as opposed to the unconstrained optimization in (9)). In the next section, we introduce conditions for nonlinear trajectory stabilizability which can be encoded as algebraic constraints within a model learning algorithm to *prune the hypothesis space in a manner that is tailored to the downstream task of trajectory generation and feedback control.*

3 Review of Contraction Theory

The core principle behind contraction theory (Lohmiller and Slotine, 1998) is to study the evolution of distance between any two *arbitrarily close* neighboring trajectories and draw conclusions upon the distance between *any finitely apart* pair of trajectories. Given an autonomous system of the form: $\dot{x}(t) = f(x(t))$, consider two neighboring trajectories separated by an infinitesimal (virtual) displacement δ_x; formally, δ_x is a vector in the tangent space $\mathcal{T}_x\mathcal{X}$ at x. The dynamics of this virtual displacement are given by:

$$\dot{\delta}_x = \frac{\partial f}{\partial x}\delta_x,$$

where $\partial f/\partial x$ is the Jacobian of f. The dynamics of the infinitesimal squared distance $\delta_x^T\delta_x$ between these two trajectories is then given by:

$$\frac{d}{dt}\left(\delta_x^T\delta_x\right) = 2\delta_x^T\frac{\partial f}{\partial x}\delta_x.$$

Then, if the symmetric part of the Jacobian matrix $\partial f/\partial x$ is *uniformly* negative definite, i.e.,

$$\sup_x \bar{\lambda}\left(\frac{1}{2}\widehat{\frac{\partial f(x)}{\partial x}}\right) \leq -\lambda < 0,$$

where $\widehat{(\cdot)} := (\cdot) + (\cdot)^T$, $\lambda > 0$, one has that the squared infinitesimal length $\delta_x^T\delta_x$ is exponentially convergent to zero at rate 2λ. By path integration of δ_x between *any* pair of trajectories, one has that the distance between any two trajectories shrinks exponentially to zero. The vector field f is thereby referred to be *contracting at rate* λ, and λ is referred to as the *contraction rate*.

Contraction metrics generalize this observation by considering as infinitesimal squared length distance, a symmetric positive definite function $V(x, \delta_x) = \delta_x^T M(x)\delta_x$, where $M : \mathcal{X} \to \mathbb{S}_n^{>0}$ is a mapping from $\mathcal{X}$ to the set of uniformly positive definite $n \times n$ symmetric matrices. Formally, $M(x)$ may be interpreted as a Riemannian metric tensor, endowing the space $\mathcal{X}$ with the Riemannian squared length element $V(x, \delta_x)$. A fundamental result in contraction theory (Lohmiller and Slotine, 1998) is that *any* contracting system admits a contraction metric $M(x)$ such that the associated function $V(x, \delta_x)$ satisfies:

$$\dot{V}(x, \delta_x) \leq -2\lambda V(x, \delta_x), \quad \forall (x, \delta_x) \in \mathcal{T}\mathcal{X},$$

for some positive contraction rate λ. Thus, the function $V(x, \delta_x)$ may be interpreted as a *differential Lyapunov function.*

3.1 Control Contraction Metrics

Control contraction metrics (CCMs) generalize contraction analysis to the controlled dynamical setting, in the sense that the analysis searches *jointly* for a controller design and the metric that describes the contraction properties of the resulting closed-loop system. Consider a control affine dynamical system of the form in (7). To define a CCM, analogously to the previous section, we first analyze the variational dynamics, i.e., the dynamics of an infinitesimal displacement δ_x:

$$\dot{\delta}_x = \overbrace{\left(\frac{\partial f(x)}{\partial x} + \sum_{j=1}^{m} u^j \frac{\partial b_j(x)}{\partial x}\right)}^{:=A(x,u)}\delta_x + B(x)\delta_u, \tag{10}$$

where δ_u is an infinitesimal (virtual) control vector at u (i.e., δ_u is a vector in the control input tangent space, i.e., $\mathbb{R}^m$). A uniformly positive definite matrix-valued function $M(x)$ is a CCM for the system $\{f, B\}$ if there exists a function $\delta_u(x, \delta_x, u)$ such that the function $V(x, \delta_x) = \delta_x^T M(x)\delta_x$ satisfies

$$\begin{aligned}\dot{V}(x, \delta_x, u) &= \delta_x^T \left(\partial_{f+Bu} M(x) + \overbrace{M(x)A(x,u)}\right)\delta_x + 2\delta_x^T M(x)B(x)\delta_u(x, \delta_x, u) \\ &\leq -2\lambda V(x, \delta_x) \quad \forall (x, \delta_x) \in \mathcal{TX},\ u \in \mathcal{U}.\end{aligned} \tag{11}$$

Given the existence of a CCM, one can then construct an exponentially stabilizing (in the sense of (2)) feedback controller $k(x^*, x)$ as described in Appendix B.

Some important observations are in order. First, the function $V(x, \delta_x)$ may be interpreted as a differential CLF, in that there exists a stabilizing differential controller δ_u that stabilizes the variational dynamics (10) in the sense of (11). Second, and more importantly, by stabilizing the variational dynamics (essentially an infinite family of linear dynamics in (δ_x, δ_u)) pointwise, everywhere in the state-space, we obtain a stabilizing controller for the original nonlinear system. This result, proven in (Manchester and Slotine, 2017; Singh et al., 2017), is derived from *integrating* the stabilizing differential controller along curves connecting the desired and actual state trajectories. Informally, the proof shows how the length of these curves can be made to shrink exponentially, by virtue of the exponential stability of the differential controller at all points along the curves. Crucially, this is an exact stabilization result, not one based on local linearization. Consequently, one can show several useful properties, such as invariance to state-space transformations (Manchester and Slotine, 2017) and robustness (Singh et al., 2017; Manchester and Slotine, 2018). Third, the CCM approach only requires a weak form of controllability, and therefore is not restricted to feedback linearizable (i.e., invertible) systems.

4 CCM Constrained Dynamics Learning

Leveraging the characterization of stabilizability via CCMs, we can now formalize our dynamics learning problem as follows. Given a supervised dataset of demonstration tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$, we wish to learn the dynamics functions $f(x)$ and $B(x)$ in (7), subject to the constraint that there exists a CCM $M(x)$ for the learned dynamics. That is, the CCM $M(x)$ plays the role of a *certificate* of stabilizability for the learned dynamics.

As shown in (Manchester and Slotine, 2017), a necessary and sufficient characterization of a CCM $M(x)$ for the system $\{\hat{f}, \hat{B}\}$ is given in terms of its dual $W(x) := M(x)^{-1}$ by the following two conditions:

$$\hat{B}_\perp(x)^T \left(\partial_{\hat{b}_j} W(x) - \overbrace{\frac{\partial \hat{b}_j(x)}{\partial x} W(x)}\right) \hat{B}_\perp(x) = 0,\ j = 1, \ldots, m \quad \forall x \in \mathcal{X}, \tag{12}$$

$$\underbrace{\hat{B}_\perp(x)^T \left(-\partial_{\hat{f}} W(x) + \overbrace{\frac{\partial \hat{f}(x)}{\partial x} W(x)} + 2\lambda W(x)\right) \hat{B}_\perp(x)}_{:=\mathcal{F}_\lambda(x;\hat{f},W)} \preceq 0, \quad \forall x \in \mathcal{X}, \tag{13}$$

where $\hat{B}_\perp$ is the annihilator matrix for $\hat{B}$, i.e., $\hat{B}(x)^T \hat{B}_\perp(x) = 0$ for all x. To see this, perform the change of variables $\eta_x := M(x)\delta_x$ in (11) and examine the case where the coefficient of δ_u is zero. In the definition above, we write $\mathcal{F}_\lambda(x;\hat{f},W)$ since $\{\hat{f},W\}$ will be optimization variables in our formulation while λ is treated as a hyper-parameter. Thus, the learning task reduces to finding the functions $\{\hat{f},\hat{B},W\}$ that jointly satisfy the above constraints, while minimizing an appropriate regularized regression loss function. Formally, problem (3) can be re-stated as:

$$\begin{aligned}
\min_{\substack{\hat{f}\in\mathcal{H}^f,\hat{b}_j\in\mathcal{H}^B,j=1,\dots,m\\ W\in\mathcal{H}^W\\ \underline{w},\overline{w}\in\mathbb{R}_{>0}}} \quad & \overbrace{\sum_{i=1}^{N}\left\|\hat{f}(x_i)+\hat{B}(x_i)u_i-\dot{x}_i\right\|^2+\mu_f\|\hat{f}\|^2_{\mathcal{H}^f}+\mu_b\sum_{j=1}^{m}\|\hat{b}_j\|^2_{\mathcal{H}^B}}^{:=J_d(\hat{f},\hat{B})}+ \\
& +\underbrace{(\overline{w}-\underline{w})+\mu_w\|W\|^2_{\mathcal{H}^W}}_{:=J_m(W,\underline{w},\overline{w})} \qquad (14a)\\
\text{s.t.} \quad & \text{eqs. (12), (13)} \quad \forall x\in\mathcal{X}, \qquad (14b)\\
& \underline{w}I_n \preceq W(x) \preceq \overline{w}I_n \quad \forall x\in\mathcal{X}, \qquad (14c)
\end{aligned}$$

where $\mathcal{H}^f$ and $\mathcal{H}^B$ are appropriately chosen $\mathbb{R}^n$-valued function classes on $\mathcal{X}$ for $\hat{f}$ and $\{\hat{b}_j\}_{j=1}^m$ respectively, and $\mathcal{H}^W$ is a suitable $\mathbb{S}_n^{>0}$-valued function space on $\mathcal{X}$. The objective is composed of a dynamics term J_d consisting of regression loss and regularization terms, and a metric term J_m consisting of a condition number surrogate loss on the metric $W(x)$ and a regularization term. The metric cost term $\overline{w}-\underline{w}$ is motivated by the observation that the state tracking error (i.e., $\|x(t)-x^*(t)\|^2$) in the presence of bounded additive disturbances is proportional to the ratio $\overline{w}/\underline{w}$; see (Singh et al., 2017).

Notice that the coupling constraints (12) and (13) are bi-linear in the decision variables $\hat{f}$ and W. Thus, at a high-level, a solution algorithm must consist of alternating between two convex sub-problems, defined by the objective and decision variable pairs $(J_d,\{\hat{f},\hat{B}\})$ and $(J_m,\{W,\underline{w},\overline{w}\})$. We refer to the first sub-problem as the *dynamics* sub-problem, and the second as the *metric* sub-problem. These sub-problems are described in full in Section 7.

Remark 1. *One may include λ as part of the overall optimization problem; indeed this is the case in the robust planning setting in (Singh et al., 2017). However, doing so would require either (i) adding $-\lambda$ to the dynamics sub-problem cost function, or (ii) line-searching over λ within the metric sub-problem. The first option detracts from the primary objective of the dynamics sub-problem, which is to minimize regression error, while the second option induces needless complexity. Thus, in this work, λ is held fixed as a tuning parameter.*

5 CCM Regularized Dynamics Learning

When performing dynamics learning on a system that is a priori *known* to be exponentially stabilizable with rate at least λ, the constrained problem formulation in (14) follows naturally given the assured *existence* of a CCM. Albeit, the infinite-dimensional nature of the constraints is a considerable technical challenge, falling under the class of *semi-infinite* optimization (Hettich and Kortanek, 1993). Alternatively, for systems that are not globally exponentially stabilizable in $\mathcal{X}$, one can imagine that such a constrained formulation may lead to adverse bias effects on the learned dynamics model.

Thus, in this section we propose a relaxation of problem (14) motivated by the concept of regularization. Specifically, constraints (12) and (13) capture this notion of stability of infinitesimal deviations *at all points* in the space $\mathcal{X}$. They stem from requiring that $\dot{V} \leq -2\lambda V(x, \delta_x)$ in (11) when $\delta_x^T M(x)B(x) = 0$, i.e., when δ_u can have no effect on $\dot{V}$. This is nothing but the standard control Lyapunov inequality, applied to the differential setting. Constraint (12) sets to zero, the terms in (11) affine in u, while constraint (13) enforces this "natural" stability condition.

The simplifications we make are (i) relax constraints (13) and (14c) to hold pointwise over some *finite* constraint set $X_c \subset \mathcal{X}$, and (ii) assume a specific sparsity structure for the input matrix estimate $\hat{B}(x)$. We discuss the pointwise relaxation here; the sparsity assumption on $\hat{B}(x)$ is discussed in the following section.

First, from a purely mathematical standpoint, the pointwise relaxation of (13) and (14c) is motivated by the observation that, as the CCM-based controller is exponentially stabilizing, we only require the differential stability condition to hold locally (in a tube-like region) with respect to the provided demonstrations. By continuity of eigenvalues for continuously parameterized entries of a matrix, it is sufficient to enforce the matrix inequalities at a sampled set of points (Lax, 2007).

Second, while adopting a sampled version of the stability constraint implies that we can no longer claim *exact* stabilizability for the learned model, enforcing the existence of such an "approximate" CCM seems to have an impressive regularization effect on the learned dynamics that is more meaningful than standard regularization techniques used in for instance, ridge or lasso regression. Specifically, problem (14), and more generally, problem (3) can be viewed as the *projection* of the best-fit dynamics onto the set of stabilizable systems. This results in dynamics models that jointly balance regression performance and stabilizablity, ultimately yielding systems whose generated trajectories are notably easier to track. This effect of regularization is apparent in the simulation results presented in Section 2.1, and the hardware testbed results in Section 8.

Practically, the finite constraint set X_c, with cardinality N_c, includes all x_i in the supervised regression training set of $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$ tuples. However, as the LMI constraints are *independent* of u_i and $\dot{x}_i$, the set X_c is chosen as a strict superset of $\{x_i\}_{i=1}^N$ (i.e., $N_c > N$) by randomly sampling additional points within $\mathcal{X}$, drawing parallels with *semi-supervised learning*.

5.1 Sparsity of Input Matrix

While a pointwise relaxation for the matrix inequalities is reasonable, one cannot apply such a relaxation to the exact equality condition in (12). Thus, the second simplification made is the following assumption, reminiscent of control normal form equations.

Assumption 1. *Assume $\hat{B}(x)$ to take the following sparse representation:*

$$\hat{B}(x) = \begin{bmatrix} O_{(n-m)\times m} \\ \mathfrak{b}(x) \end{bmatrix}, \tag{15}$$

where O is the zero matrix and $\mathfrak{b}(x)$ is an invertible $m \times m$ matrix for all $x \in \mathcal{X}$.

For the assumed structure of $\hat{B}(x)$, a valid $\hat{B}_\perp$ matrix is then given by:

$$\hat{B}_\perp = \begin{bmatrix} I_{n-m} \\ O_{m\times(n-m)} \end{bmatrix}, \tag{16}$$

where I is the identity matrix. Therefore, constraint (12) simply becomes:

$$\partial_{\hat{b}_j} W_\perp(x) = 0, \quad j = 1, \dots, m,$$

where $W_\perp$ is the upper-left $(n-m)\times(n-m)$ block of $W(x)$. Assembling these constraints for the (p,q) entry of $W_\perp$, i.e., $w_{\perp_{pq}}$, we obtain:

$$\begin{bmatrix} \frac{\partial w_{\perp_{pq}}(x)}{\partial x^{(n-m)+1}} & \cdots & \frac{\partial w_{\perp_{pq}}(x)}{\partial x^{n}} \end{bmatrix} \mathfrak{b}(x) = 0.$$

Since the matrix $\mathfrak{b}(x)$ in (15) is assumed to be invertible, the *only* solution to this equation is $\partial w_{\perp_{pq}}/\partial x^i = 0$ for $i \in \{(n-m)+1,\ldots,n\}$, and all $(p,q)\in\{1,\ldots,(n-m)\}$. That is, $W_\perp$ cannot be a function of the last m components of x – an elegant simplification of constraint (12).

Let us justify the structural assumption in (15). Physically, this assumption is not as mysterious as it appears. Indeed, consider the standard dynamics form for mechanical systems:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = B(q)u,$$

where $q \in \mathbb{R}^{n_q}$ is the configuration vector, $H \in \mathbb{S}^{>0}_{n_q}$ is the inertia matrix, $C(q,\dot{q})\dot{q}$ contains the centrifugal and Coriolis terms, g are the gravitational terms, $B \in \mathbb{R}^{n_q\times m}$ is the (full rank) input matrix mapping, and $u \in \mathbb{R}^m$ is the input; for fully actuated systems, $\text{rank}(B) = m = n_q$. For underactuated systems, $m < n_q$. By rearranging the configuration vector (Spong, 1998; Olfati-Saber, 2001; Reyhanoglu et al., 1999), one can partition q as (q_u, q_a) where $q_u \in \mathbb{R}^{n_q-m}$ represents the unactuated degrees of freedom and $q_a \in \mathbb{R}^m$ represents the actuated degrees of freedom. Applying this partitioning to the dynamics equation above yields:

$$\begin{bmatrix} H_{uu}(q) & H_{ua}(q) \\ H_{ua}(q) & H_{aa}(q) \end{bmatrix} \begin{bmatrix} \ddot{q}_u \\ \ddot{q}_a \end{bmatrix} + \begin{bmatrix} \tau_u(q,\dot{q}) \\ \tau_a(q,\dot{q}) \end{bmatrix} = \begin{bmatrix} O_{(n_q-m)\times m} \\ \mathfrak{b}(q) \end{bmatrix} u, \tag{17}$$

where $\mathfrak{b} \in \mathbb{R}^{m\times m}$ is an invertible square matrix. As observed in (Reyhanoglu et al., 1999), a substantial class of underactuated systems can be represented in this manner. Of course, fully actuated systems (i.e., $m = n_q$) also take this form. Thus, by taking as state $x = (q,p) \in \mathbb{R}^n$ where $p = H(q)\dot{q}$ is momentum (so that $n = 2n_q$), the dynamics can be written as:

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} H^{-1}(q)p \\ \dot{H}(q)\dot{q} - \tau(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} O_{(n-m)\times m} \\ \mathfrak{b}(q) \end{bmatrix} u. \tag{18}$$

Notice that the input matrix takes the desired normal form in (15). To address the apparent difficult of working with the state representation of (q,p) (when usually only measurements of $(q,\dot{q},\ddot{q})$ are typically available *and* the inertia matrix $H(q)$ is unknown), one may leverage the following result from (Manchester and Slotine, 2017):

Theorem 1 (CCM Invariance to Diffeomorphisms)**.** *Suppose there exists a valid CCM $M_x(x)$ with respect to the state x. Then, if $z = \psi(x)$ is a diffeomorphism, then the CCM conditions also hold with respect to state z with metric $M_z(z) = \Psi(z)^{-T}M_x(x)\Psi(z)^{-1}$, where $\Psi(z) = \partial\psi(x)/\partial x$ is evaluated at $x = \psi^{-1}(z)$.*

Thus, for the class of underactuated systems that does not readily possess an input matrix of the form (15), for instance when using a state-representation $(q,\dot{q})$, one may instead rely on the state representation (q,p) whose associated dynamics in (18) possesses the desired input matrix structure. For such systems, one would solve problem (14) by formulating the regression loss using the state representation $(q,\dot{q})$ and dynamics representation in (17), and using the previous estimate of H within the matrix inequality constraints, formulated with respect to the dynamics representation in (18) and state (q,p). This allows us to borrow several existing results from adaptive control on estimating mechanical system by leveraging the known linearity of $H(q)$ in terms of unknown mass property parameters multiplying known physical basis functions. We leave this extension however, to future work, and from hereon assume the structural form given in (16) for the estimated input matrix.

5.2 Finite-dimensional Optimization

We now present a tractable finite-dimensional optimization for solving problem (14) under the two simplifying assumptions introduced in the previous sections. For ease of presentation, we outline the final optimization formulation here and provide the derivation, which relies extensively on vector-valued Reproducing Kernel Hilbert Spaces (RKHS), in the following section.

Parameterization: Model $\hat{f}$, $\{\hat{b}_j\}_{j=1}^m$ and $\{w_{ij}\}_{i,j=1}^n$ as linear combinations of features. That is,

$$\hat{f}(x) = \Phi_f(x)^T\alpha, \tag{19}$$

$$\hat{b}_j(x) = \Phi_b(x)^T\beta_j \quad j \in \{1,\dots,m\}, \tag{20}$$

$$w_{ij}(x) = \begin{cases} \hat{\phi}_w(x)^T\hat{\theta}_{ij} & \text{if} \quad (i,j) \in \{1,\dots,n-m\}, \\ \phi_w(x)^T\theta_{ij} & \text{else}, \end{cases} \tag{21}$$

where $\alpha \in \mathbb{R}^{d_f}$, $\beta_j \in \mathbb{R}^{d_b}$, $\hat{\theta}_{ij}, \theta_{ij} \in \mathbb{R}^{d_w}$ are constant vectors to be optimized over, and $\Phi_f : \mathcal{X} \to \mathbb{R}^{d_f \times n}$, $\Phi_b : \mathcal{X} \to \mathbb{R}^{d_b \times n}$, $\hat{\phi}_w : \mathcal{X} \to \mathbb{R}^{d_w}$ and $\phi_w : \mathcal{X} \to \mathbb{R}^{d_w}$ are a priori chosen feature mappings. To enforce the sparsity structure in (15), the feature matrix Φ_b must have all zeros in its first $n-m$ columns. The features $\hat{\phi}_w$ are distinct from ϕ_w in that the former are only a function of the first $n-m$ components of x (as per Section 5.1). While one can use any function approximator (e.g., neural nets), we motivate this parameterization from a perspective of RKHS – see Section 6.

Optimization: Given positive regularization constants μ_f, μ_b, μ_w and positive tolerances $\epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}$, solve:

$$\min_{\alpha,\beta_j,\hat{\theta}_{ij},\theta_{ij},\underline{w},\overline{w}} \quad \overbrace{\sum_{i=1}^{N} \|\hat{f}(x_i) + \hat{B}(x_i)u_i - \dot{x}_i\|^2 + \mu_f\|\alpha\|^2 + \mu_b \sum_{j=1}^{m}\|\beta_j\|^2}^{:=\hat{J}_d} \\ + \underbrace{(\overline{w} - \underline{w}) + \mu_w \sum_{i,j} \|\tilde{\theta}_{ij}\|^2}_{:=\hat{J}_m} \tag{22a}$$

$$\text{s.t.} \quad \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i;\alpha,\tilde{\theta}_{ij}) \preceq 0 \quad \forall x_i \in X_c, \tag{22b}$$

$$(\underline{w} + \epsilon_{\underline{w}})I_n \preceq W(x_i) \preceq \overline{w}I_n \quad \forall x_i \in X_c, \tag{22c}$$

$$\theta_{ij} = \theta_{ji}, \ \hat{\theta}_{ij} = \hat{\theta}_{ji} \tag{22d}$$

$$\underline{w} \geq \delta_{\underline{w}}, \tag{22e}$$

where $\tilde{\theta}_{ij}$ is used as a placeholder for θ_{ij} and $\hat{\theta}_{ij}$ to simplify notation, and we use $\hat{J}_d$ and $\hat{J}_m$ to distinguish them from the functional equivalents J_d and J_m in problem (14). We wish to highlight the following key points regarding problem (22). Constraints (22b) and (22c) are the pointwise relaxations of (13) and (14c) respectively; (22d) enforces the symmetry of $W(x)$, and (22e) imposes some tolerance requirements to ensure a well conditioned solution. Additionally, the tolerances ϵ_λ and $\epsilon_{\underline{w}}$ are used to account for the pointwise relaxations of the matrix inequalities. A key challenge is to efficiently solve this constrained optimization problem, given a potentially large number of constraint points in X_c. In Section 7, we present an iterative algorithm and an adaptive constraint sampling technique to solve problem (22).

6 Derivation of Problem (22)

To go from the general problem definition in (14) to the finite dimensional problem in (22), we first must define appropriate function classes for $\hat{f}$, $\hat{b}_j$, and W. We will do this using the framework of RKHS. We summarize the contents of this section here. First, starting from the infinite-dimensional *non-convex* constrained optimization over function spaces, i.e., problem (14), we split the problem into alternating between two infinite-dimensional *convex* sub-problems. Second, we derive the representation of the optimal solution of each sub-problem in Sections 6.2 and 6.3, by leveraging the mild structural assumption on the input matrix and relaxation of the infinite-dimensional constraints to sampling-based constraints. Third, in Section 6.4, we use the universal approximation property of random matrix feature maps to reduce the dimensionality of the representation of the optimal solutions, at last yielding problem (22). We first provide a brief introduction to RKHS theory.

6.1 Reproducing Kernel Hilbert Spaces

Scalar-valued RKHS: Kernel methods (Schölkopf and Smola, 2001) constitute a broad family of non-parametric modeling techniques for solving a range of problems in machine learning. A scalar-valued positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ generates a RKHS $\mathcal{H}_k$ of functions, with the nice property that if two functions are close in the distance derived from the norm (associated with the Hilbert space), then their pointwise evaluations are close at all points. This continuity of evaluation functionals has the far-reaching consequence that norm-regularized learning problems over RKHSs admit finite dimensional solutions via representer theorems. The kernel k may be (non-uniquely) associated with a higher-dimensional embedding of the input space via a feature map, $\phi : \mathbb{R}^n \mapsto \mathbb{R}^D$, such that $k(x, z) = \phi(x)^T\phi(z)$, where D is infinite for universal kernels associated with RKHSs that are dense in the space of square-integrable functions. Standard regularized linear statistical models in the embedding space implicitly provide nonlinear inference with respect to the original input representation. In a nutshell, kernel methods provide a rigorous algorithmic framework for deriving nonlinear counterparts of a whole array of linear statistical techniques, e.g., for classification, regression, dimensionality reduction, and unsupervised learning. For further details, we point the reader to (Hearst et al., 1998).

Vector-valued RKHS: Dynamics estimation is a vector-valued learning problem. Such problems can be naturally formulated in terms of vector-valued generalizations of RKHS concepts. The theory and formalism of vector-valued RKHS can be traced as far back as the work of Laurent Schwartz (Schwartz, 1964), with applications ranging from solving partial differential equations to machine learning. Informally, we say that $\mathcal{H}$ is an RKHS of $\mathbb{R}^n$-valued maps if for any $v \in \mathbb{R}^n$, the linear functional that maps $f \in \mathcal{H}$ to $v^T f(x)$ is continuous for each x.

More formally, denote the standard inner product on $\mathbb{R}^n$ as $\langle \cdot, \cdot \rangle$, and let $\mathbb{R}^n(\mathcal{X})$ be the vector space of all functions $f : \mathcal{X} \to \mathbb{R}^n$ and let $\mathcal{L}(\mathbb{R}^n)$ be the space of all bounded linear operators on $\mathbb{R}^n$, i.e., $n \times n$ matrices. A function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathbb{R}^n)$ is an *operator-valued positive definite kernel* if for all $(x, z) \in \mathcal{X} \times \mathcal{X}$: $K(x, z)^T = K(z, x)$, and for any pair of finite sets $\{x_i\}_{i=1}^N \in \mathcal{X}$ and $\{y_i\}_{i=1}^N \in \mathbb{R}^n$,

$$\sum_{i,j=1}^{N} \langle y_i, K(x_i, x_j) y_j \rangle \geq 0.$$

Given such a K, we define the unique $\mathbb{R}^n$-valued RKHS $\mathcal{H}_K \subset \mathbb{R}^n(\mathcal{X})$ with reproducing kernel K as follows. For each $x \in \mathcal{X}$ and $y \in \mathbb{R}^n$, define the function $K_x y = K(\cdot, x) y \in \mathbb{R}^n(\mathcal{X})$. That is, $K_x y(z) = K(z, x) y$ for all $z \in \mathcal{X}$. Then, the Hilbert space $\mathcal{H}_K$ is defined to be the completion of the linear span of functions $\{K_x y \mid x \in \mathcal{X}, y \in \mathbb{R}^n\}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ between functions $f = \sum_{i=1}^N K_{x_i} y_i$, $g = \sum_{j=1}^{N'} K_{z_j} w_j \in \mathcal{H}_K$, defined as:

$$\langle f, g \rangle_{\mathcal{H}_K} = \sum_{i=1}^{N} \sum_{j=1}^{N'} \langle y_i, K(x_i, z_j) w_j \rangle ,$$

and function norm $\|f\|^2_{\mathcal{H}_K}$ given as $\langle f, f \rangle_{\mathcal{H}_K}$. Analogous to the canonical reproducing property of scalar-valued RKHS kernels, i.e.,

$$h(x) = \langle h, k(x, \cdot) \rangle_{\mathcal{H}_k} \quad \forall h \in \mathcal{H}_k, \tag{23}$$

the matrix-valued kernel K satisfies the following reproducing property:

$$\langle f(x), y \rangle = \langle f, K_x y \rangle_{\mathcal{H}_K} \quad \forall (x, y) \in \mathcal{X} \times \mathbb{R}^n, \forall f \in \mathcal{H}_K, \tag{24}$$

and is thereby referred to as the *reproducing kernel* for $\mathcal{H}_K$. As our learning problem involves the Jacobian of f, we will also require a representation for the derivative of the kernel matrix. Accordingly, suppose the kernel matrix K lies in $\mathcal{C}^2(\mathcal{X} \times \mathcal{X})$. For any $j \in \{1, \ldots, n\}$, define the matrix functions $\frac{\partial}{\partial s^j} K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathbb{R}^n)$ and $\frac{\partial}{\partial r^j} K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathbb{R}^n)$ as

$$\frac{\partial}{\partial s^j} K(z, x) := \left. \frac{\partial}{\partial s^j} K(r, s) \right|_{r=z, s=x} \qquad \frac{\partial}{\partial r^j} K(z, x) := \left. \frac{\partial}{\partial r^j} K(r, s) \right|_{r=z, s=x},$$

where the derivative of the kernel matrix is understood to be element-wise. Define the $\mathcal{C}^1$ function $\partial_j K_x y \in \mathbb{R}^n(\mathcal{X})$ as

$$\partial_j K_x y(z) := \frac{\partial}{\partial s^j} K(z, x) y \quad \forall z \in \mathcal{X}.$$

The following result provides a useful reproducing property for the derivatives of functions in $\mathcal{H}_K$ that will be instrumental in deriving the solution algorithm.

Theorem 2 (Derivative Properties on $\mathcal{H}_K$ (Micheli and Glaunés, 2014))**.** *Let $\mathcal{H}_K$ be a RKHS in $\mathbb{R}^n(\mathcal{X})$ with reproducing kernel $K \in \mathcal{C}^2(\mathcal{X} \times \mathcal{X})$. Then, for all $(x, y) \in \mathcal{X} \times \mathbb{R}^n$:*

(a) $\partial_j K_x y \in \mathcal{H}_K$ *for all* $j = 1, \ldots, n$.

(b) The following derivative reproducing property holds for all $j = 1, \ldots, n$:

$$\left\langle \frac{\partial f(x)}{\partial x^j}, y \right\rangle = \langle f, \partial_j K_x y \rangle_{\mathcal{H}_K} \quad \forall f \in \mathcal{H}_K.$$

As mentioned in Section 4, the bi-linearity of constraint (13) forces us to adopt an alternating solution strategy whereby in the dynamics sub-problem, W is held fixed and we minimize J_d with respect to $\{\hat{f}, \hat{B}\}$. In the metric sub-problem, $\{\hat{f}, \hat{B}\}$ are held fixed and we minimize J_m with respect to $\{W, \underline{w}, \overline{w}\}$.

In the following we derive several useful representer theorems to characterize the solution of the two sub-problems, under the two simplifying assumptions introduced in Section 5.2.

6.2 Dynamics Sub-Problem

Let K^f be the reproducing $\mathcal{C}^2$ kernel for an $\mathbb{R}^n$-valued RKHS $\mathcal{H}_K^f$ and let K^B be another reproducing kernel for an $\mathbb{R}^n$-valued RKHS $\mathcal{H}_K^B$. Define the finite-dimensional subspaces:

$$\mathcal{V}_f := \left\{ \sum_{i=1}^{N_c} K_{x_i}^f a_i + \sum_{i=1}^{N_c} \sum_{p=1}^{n} \partial_p K_{x_i}^f a'_{ip}, \quad a_i, a'_{ip} \in \mathbb{R}^n \right\} \subset \mathcal{H}_K^f \ , \tag{25}$$

$$\mathcal{V}_B := \left\{ \sum_{i=1}^{N_c} K_{x_i}^B c_i + \sum_{i=1}^{N_c} \sum_{p=1}^{n} \partial_p K_{x_i}^B c'_{ip}, \quad c_i, c'_{ip} \in \mathbb{R}^n \right\} \subset \mathcal{H}_K^B \ . \tag{26}$$

Note that all x_i taken from the training dataset of $(x_i, u_i, \dot{x}_i)$ tuples are a subset of X_c.

Theorem 3 (Representer Theorem for f, B)**.** *Suppose the reproducing kernel K^B is chosen such that all functions $g \in \mathcal{H}_K^B$ satisfy the sparsity structure $g^j(x) = 0$ for $j = 1, \ldots, n-m$. Consider then the pointwise-relaxed dynamics sub-problem for problem (14):*

$$\begin{aligned} \min_{\hat{f} \in \mathcal{H}_K^f,\ \hat{b}_j \in \mathcal{H}_K^B, j=1,\ldots,m} \quad & J_d(\hat{f}, \hat{B}) \\ \text{s.t.} \quad & \mathcal{F}_\lambda(x_i; \hat{f}, W) \preceq 0 \quad \forall x_i \in X_c \ . \end{aligned} \tag{27}$$

Suppose the feasible set for the LMI constraint is non-empty. Denote f^* and $b_j^*, j = 1, \ldots, m$ as the optimizers for this sub-problem. Then, $f^* \in \mathcal{V}_f$ and $\{b_j^*\}_{j=1}^M \in \mathcal{V}_B$.

Proof. For a fixed W, the constraint is convex in $\hat{f}$ by linearity of the matrix $\mathcal{F}_\lambda$ in $\hat{f}$ and $\partial \hat{f}/\partial x$. By assumption (1), and the simplifying form for $B_\perp$, the matrix $\mathcal{F}_\lambda$ is additionally independent of B. Then, by strict convexity of the objective functional, there exists unique minimizers $f^*, \{b_j\}^* \in \mathcal{H}_K$, provided the feasible region is non-empty (Kurdila and Zabarankin, 2006).

Since $\mathcal{V}_f$ and $\mathcal{V}_B$ are closed, by the Projection Theorem, $\mathcal{H}_K^f = \mathcal{V}_f \oplus \mathcal{V}_f^\perp$ and $\mathcal{H}_K^B = \mathcal{V}_B \oplus \mathcal{V}_B^\perp$. Thus, any $g \in \mathcal{H}_K^f$ may be written as $g^{\mathcal{V}_f} + g^\perp$ where $g^{\mathcal{V}_f} \in \mathcal{V}_f$, $g^\perp \in \mathcal{V}_f^\perp$, and $\langle g^{\mathcal{V}_f}, g^\perp \rangle_{\mathcal{H}_K^f} = 0$. Similarly, any $g \in \mathcal{H}_K^B$ may be written as $g^{\mathcal{V}_B} + g^\perp$ where $g^{\mathcal{V}_B} \in \mathcal{V}_B$, $g^\perp \in \mathcal{V}_B^\perp$, and $\langle g^{\mathcal{V}_B}, g^\perp \rangle_{\mathcal{H}_K^B} = 0$.

Let $f = f^{\mathcal{V}_f} + f^\perp$ and $b_j = b_j^{\mathcal{V}_B} + b_j^\perp$, and define

$$H^{\mathcal{V}}(x, u) = f^{\mathcal{V}_f}(x) + \sum_{j=1}^{m} u^j b_j^{\mathcal{V}_B}(x)$$

$$H^\perp(x, u) = f^\perp(x) + \sum_{j=1}^{m} u^j b_j^\perp(x).$$

We can re-write $J_d(f,B)$ as:

$$\begin{aligned}J_d(f,B) = \sum_{i=1}^N &\left\langle H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i, H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right\rangle + 2\left\langle H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i, H^{\perp}(x_i,u_i)\right\rangle \\ &+ \left\langle H^{\perp}(x_i,u_i), H^{\perp}(x_i,u_i)\right\rangle + \mu_f\left(\|f^{\mathcal{V}_f}\|^2_{\mathcal{H}_K^f} + \|f^{\perp}\|^2_{\mathcal{H}_K^f}\right) \\ &+ \mu_b\left(\sum_{j=1}^m \|b_j^{\mathcal{V}_B}\|^2_{\mathcal{H}_K^B} + \|b_j^{\perp}\|^2_{\mathcal{H}_K^B}\right).\end{aligned}$$

Leveraging the reproducing property in (24),

$$\begin{aligned}&\left\langle H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i, H^{\perp}(x_i,u_i)\right\rangle \\ &\quad= \left\langle f^{\perp}(x_i) + \sum_{j=1}^m u_i^j b_j^{\perp}(x_i), H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right\rangle \\ &\quad= \underbrace{\left\langle f^{\perp}, K_{x_i}^f\left(H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right)\right\rangle_{\mathcal{H}_K^f}}_{=0} + \underbrace{\left\langle \sum_{j=1}^m u_i^j b_j^{\perp}, K_{x_i}^B\left(H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right)\right\rangle_{\mathcal{H}_K^B}}_{=0}\end{aligned}$$

since $K_{x_i}^f\left(H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right) \in \mathcal{V}_f$, $\mathcal{V}_B^{\perp}$ is closed under addition, and $K_{x_i}^B\left(H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right) \in \mathcal{V}_B$. Thus, $J_d(f,B)$ simplifies to

$$\begin{aligned}&\sum_{i=1}^N \left\|H^{\mathcal{V}}(x_i,u_i)-\dot{x}_i\right\|^2 + \mu_f\|f^{\mathcal{V}_f}\|^2_{\mathcal{H}_K^f} + \mu_b\sum_{j=1}^n \|b_j^{\mathcal{V}_B}\|^2_{\mathcal{H}_K^B} + \\ &\qquad + \left[\sum_{i=1}^N \left\|H^{\perp}(x_i,u_i)\right\|^2 + \mu_f\|f^{\perp}\|^2_{\mathcal{H}_K^f} + \mu_b\sum_{j=1}^m \|b_j^{\perp}\|^2_{\mathcal{H}_K^B}\right].\end{aligned} \tag{28}$$

Now, the (p,q) element of $\partial_f W(x_i)$ takes the form

$$\left\langle \frac{\partial w_{pq}(x_i)}{\partial x}, f(x_i)\right\rangle = \left\langle f, K_{x_i}^f\frac{\partial w_{pq}(x_i)}{\partial x}\right\rangle_{\mathcal{H}_K^f} = \left\langle f^{\mathcal{V}_f}, K_{x_i}^f\frac{\partial w_{pq}(x_i)}{\partial x}\right\rangle_{\mathcal{H}_K^f}.$$

Column p of $\frac{\partial f(x_i)}{\partial x}W(x_i)$ takes the form

$$\begin{aligned}\sum_{j=1}^n w_{jp}(x_i)\frac{\partial f(x_i)}{\partial x^j} = \begin{bmatrix}\sum_{j=1}^n w_{jp}(x_i)\left\langle\frac{\partial f(x_i)}{\partial x^j}, e_1\right\rangle \\ \vdots \\ \sum_{j=1}^n w_{jp}(x_i)\left\langle\frac{\partial f(x_i)}{\partial x^j}, e_n\right\rangle\end{bmatrix} &= \begin{bmatrix}\sum_{j=1}^n w_{jp}(x_i)\left\langle f, \partial_j K_{x_i}^f e_1\right\rangle_{\mathcal{H}_K^f} \\ \vdots \\ \sum_{j=1}^n w_{jp}(x_i)\left\langle f, \partial_j K_{x_i}^f e_n\right\rangle_{\mathcal{H}_K^f}\end{bmatrix} \\ &= \begin{bmatrix}\sum_{j=1}^n w_{jp}(x_i)\left\langle f^{\mathcal{V}_f}, \partial_j K_{x_i}^f e_1\right\rangle_{\mathcal{H}_K^f} \\ \vdots \\ \sum_{j=1}^n w_{jp}(x_i)\left\langle f^{\mathcal{V}_f}, \partial_j K_{x_i}^f e_n\right\rangle_{\mathcal{H}_K^f}\end{bmatrix},\end{aligned}$$

where e_i is the i^{th} standard basis vector in $\mathbb{R}^n$. Thus, $f^\perp$ plays no role in pointwise relaxation of constraint (13) and thus does not affect problem feasibility. Given the assumed structure of functions in $\mathcal{H}_K^B$ as provided in the theorem statement, the matrix in (16) is a valid annihilator matrix for $\hat{B}(x)$. Consequently, $b^\perp$ also has no effect on problem feasibility. Thus, by non-negativity of the term in the square brackets in (28), we have that the optimal f lies in $\mathcal{V}_f$ and all optimal $\{b_j\}_{j=1}^m$ lie within $\mathcal{V}_B$. □

The key consequence of this theorem is the reduction of the infinite-dimensional search problem for the functions f and $b_j, j = 1, \ldots, m$ to a finite-dimensional *convex* optimization problem for the constant vectors $a_i, a'_{ip}, \{c_i^{(j)}, c_{ip}^{(j)'}\}_{j=1}^m \in \mathbb{R}^n$, by choosing the function classes $\mathcal{H}^f = \mathcal{H}_K^f$ and $\mathcal{H}^B = \mathcal{H}_K^B$. The key novelty of this result is in the derivation of a finite-dimensional representation of the solution to a *constrained functional* optimization problem involving *gradients*. Next, we characterize the optimal solution to the metric sub-problem.

6.3 Metric Sub-Problem

By the simplifying assumption in Section 5.1, constraint (12) requires that $W_\perp$ be only a function of the first $(n-m)$ components of x. Thus, define $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as a *scalar* reproducing kernel with associated real-valued scalar RKHS $\mathcal{H}_\kappa$. Additionally, define $\hat{\kappa} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as another scalar reproducing kernel with associated real-valued scalar RKHS $\mathcal{H}_{\hat{\kappa}}$. In particular, $\hat{\kappa}$ is only a function of the first $(n-m)$ components of $x \in \mathcal{X}$ in both arguments. Let κ_x and $\hat{\kappa}_x$ denote $\kappa(x, \cdot)$ and $\hat{\kappa}(x, \cdot)$ respectively.

Define the kernel derivative functions:

$$\partial_j \kappa_x(z) := \frac{\partial \kappa}{\partial r^j} \kappa(r,s)\bigg|_{(r=x,s=z)} \qquad \partial_j \hat{\kappa}_x(z) := \frac{\partial \hat{\kappa}}{\partial r^j} \kappa(r,s)\bigg|_{(r=x,s=z)} \qquad \forall z \in \mathcal{X}.$$

From (Zhou, 2008), it follows that the kernel derivative functions satisfy the following two properties, similar to Theorem 2:

$$\begin{aligned} \partial_j \kappa_x \in \mathcal{H}_\kappa \quad & \forall j = 1, \ldots, n, \; x \in \mathcal{X} \\ \frac{\partial h}{\partial x^j}(x) = \langle h, \partial_j \kappa_x \rangle_{\mathcal{H}_\kappa} \quad & \forall h \in \mathcal{H}_\kappa. \end{aligned}$$

A similar property holds for $\partial_j \hat{\kappa}_x$ and $\mathcal{H}_{\hat{\kappa}}$. Consider then the following finite-dimensional spaces:

$$\mathcal{V}_\kappa := \left\{ \sum_{i=1}^{N_c} a_i \kappa_{x_i} + \sum_{i=1}^{N_c} \sum_{p=1}^{n} a'_{ip} \, \partial_p \kappa_{x_i}, \quad a_i, a'_{ip} \in \mathbb{R} \right\} \subset \mathcal{H}_\kappa \tag{29}$$

$$\mathcal{V}_{\hat{\kappa}} := \left\{ \sum_{i=1}^{N_c} c_i \hat{\kappa}_{x_i} + \sum_{i=1}^{N_c} \sum_{p=1}^{n-m} c'_{ip} \, \partial_p \hat{\kappa}_{x_i}, \quad c_i, c'_{ip} \in \mathbb{R} \right\} \subset \mathcal{H}_{\hat{\kappa}} \tag{30}$$

and the proposed representation for $W(x)$:

$$\begin{aligned} W(x) = & \sum_{i=1}^{N_c} \hat{\Theta}_i \hat{\kappa}_{x_i}(x) + \sum_{i=1}^{N_c} \sum_{j=1}^{n-m} \hat{\Theta}'_{ij} \partial_j \hat{\kappa}_{x_i}(x) \\ & + \sum_{i=1}^{N_c} \Theta_i \kappa_{x_i}(x) + \sum_{i=1}^{N_c} \sum_{j=1}^{n} \Theta'_{ij} \partial_j \kappa_{x_i}(x), \end{aligned} \tag{31}$$

where $\hat{\Theta}, \hat{\Theta}' \in \mathbb{S}_n$ are constant symmetric matrices with non-zero entries only in the top-left $(n - m) \times (n - m)$ block, and $\Theta, \Theta' \in \mathbb{S}_n$ are constant symmetric matrices with zero entries in the top-left $(n - m) \times (n - m)$ block.

Theorem 4 (Representer Theorem for W)**.** *Consider the pointwise-relaxed metric sub-problem for problem* (14)*:*

$$\begin{aligned} \min_{\substack{w_{pq} \in \mathcal{H}_{\hat{\kappa}}, (p,q) \in \{1,\ldots,(n-m)\} \\ w_{pq} \in \mathcal{H}_{\kappa} \text{ else} \\ \underline{w}, \overline{w} \in \mathbb{R}_{>0}}} \quad & J_m(W, \underline{w}, \overline{w}) \\ \text{s.t.} \quad & \mathcal{F}_\lambda(x_i; \hat{f}, W) \preceq 0, \quad \forall x_i \in X_c, \qquad (32) \\ & \underline{w} I_n \preceq W(x_i) \preceq \overline{w} I_n, \quad \forall x_i \in X_c, \qquad (33) \end{aligned}$$

Suppose the feasible set of the above LMI constraints is non-empty. Denote W^* as the optimizer for this sub-problem. Then, W^* takes the form given in (31)*.*

Proof. Notice that while the regularizer term is strictly convex, the surrogate loss function for the condition number is affine. However, provided the feasible set is non-empty, there still exists a minimizer (possibly non-unique) for the above sub-problem.

Since $\mathcal{V}_\kappa$ and $\mathcal{V}_{\hat{\kappa}}$ are closed, by the Projection Theorem, $\mathcal{H}_\kappa = \mathcal{V}_\kappa \oplus \mathcal{V}_\kappa^\perp$ and $\mathcal{H}_{\hat{\kappa}} = \mathcal{V}_{\hat{\kappa}} \oplus \mathcal{V}_{\hat{\kappa}}^\perp$. Thus, any $h \in \mathcal{H}_\kappa$ may be written as $h^{\mathcal{V}_\kappa} + h^\perp$ where $h^{\mathcal{V}_\kappa} \in \mathcal{V}_\kappa$, $h^\perp \in \mathcal{V}_\kappa^\perp$ and $\langle h^{\mathcal{V}_\kappa}, h^\perp \rangle_{\mathcal{H}_\kappa} = 0$. A similar decomposition property holds for $\mathcal{H}_{\hat{\kappa}}$.

Now, consider the following chain of equalities for the $(p, q)^{\text{th}}$ element of $\partial_f W_\perp(x_i)$:

$$\begin{aligned} \partial_f w_{\perp_{pq}}(x_i) &= \sum_{j=1}^{n-m} \frac{\partial w_{\perp_{pq}}(x_i)}{\partial x^j} f^j(x_i) \\ &= \sum_{j=1}^{n-m} \left\langle w_{\perp_{pq}}, \partial_j \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}} f^j(x_i) \\ &= \left\langle w_{\perp_{pq}}, \sum_{j=1}^{n-m} f^j(x_i) \partial_j \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}} \\ &= \left\langle w_{\perp_{pq}}^{\mathcal{V}_{\hat{\kappa}}}, \sum_{j=1}^{n-m} f^j(x_i) \partial_j \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}}, \end{aligned}$$

since $\sum_{j=1}^{n-m} f^j(x_i) \partial_j \hat{\kappa}_{x_i} \in \mathcal{V}_{\hat{\kappa}}$. Trivially, the (p, q) element of $W(x_i)$ takes the form

$$w_{pq}(x_i) = \begin{cases} \left\langle w_{pq}^{\mathcal{V}_{\hat{\kappa}}}, \hat{\kappa}_{x_i} \right\rangle_{\mathcal{H}_{\hat{\kappa}}} & \text{if } (p, q) \in \{1, \ldots, (n - m)\} \\ \left\langle w_{pq}^{\mathcal{V}_\kappa}, \kappa_{x_i} \right\rangle_{\mathcal{H}_\kappa} & \text{else.} \end{cases}$$

Thus, constraints (13) and uniform definiteness at the constraint points in X_c can be written in terms of functions in $\mathcal{V}_\kappa$ and $\mathcal{V}_{\hat{\kappa}}$ alone. By strict convexity of the regularizer, and recognizing that $W(x)$ is symmetric, the result follows. □

Similar to the previous section, the key property here is the reduction of the infinite-dimensional search over $W(x)$ to the finite-dimensional convex optimization problem over the constant symmetric matrices $\Theta_i, \Theta'_{ij}, \hat{\Theta}_i, \hat{\Theta}'_{ij}$ by choosing the function class for the entries of W using the scalar-valued RKHS.

At this point, both sub-problems are finite-dimensional convex optimization problems. Crucially, the only simplifications made are those given in Section 5.2. However, a final computational challenge here is that the number of parameters scales with the number of training (N) and constraint (N_c) points. This is a fundamental challenge in all non-parametric methods. In the next section we present the dimensionality reduction techniques used to alleviate these issues.

6.4 Approximation via Random Matrix Features

The size of the problem using full matrix-valued kernel expansions grows rapidly in $N_c \cdot n$, the number of constraint points times the state dimensionality. This makes training slow for even moderately long demonstrations in low-dimensional settings. The induced dynamical system is slow to evaluate and integrate at inference time. Random feature approximations to kernel functions have been extensively used to scale up training complexity and inference speed of kernel methods (Rahimi and Recht, 2007; Avron et al., 2016) in a number of applications (Huang et al., 2014). The quality of approximation can be explicitly controlled by the number of random features. In particular, it has been shown (Rahimi and Recht, 2008) that any function in the RKHS associated with the exact kernel can be approximated to arbitrary accuracy by a linear combination of a sufficiently large number of random features.

These approximations have only recently been extended to matrix-valued kernels (Minh, 2016; Brault et al., 2016). Given a matrix-valued kernel K, one defines an appropriate matrix-valued feature map $\Phi : \mathcal{X} \to \mathbb{R}^{d\times n}$ with the property

$$K(x, z) \approx \Phi(x)^T \Phi(z),$$

where d controls the quality of this approximation. A canonical example is the Gaussian separable kernel $K_\sigma(x, z) := e^{\frac{-\|x-z\|_2^2}{\sigma^2}} I_n$ with feature map:

$$\Phi(x) = \frac{1}{\sqrt{s}} \begin{bmatrix} \cos(\omega_1^T x) \\ \sin(\omega_1^T x) \\ \vdots \\ \cos(\omega_s^T x) \\ \sin(\omega_s^T x) \end{bmatrix} \otimes I_n,$$

where $\omega_1, \ldots, \omega_s$ are i.i.d. draws from $\mathcal{N}(0, \sigma^{-2} I_n)$ and $\otimes$ denotes the Kronecker product.

By construction, the linear span $\{K_z y \mid z \in \mathcal{X}, y \in \mathbb{R}^n\}$ is dense in $\mathcal{H}_K$. Thus, any function g in the associated RKHS $\mathcal{H}_K$ can be arbitrarily well-approximated by the expansion $\sum_j^{N'} K_{z_j} y_j$, which, in turn, may be approximated as

$$g(x) \approx \sum_{j=1}^{N'} K(x, z_j) y_j \approx \sum_{j=1}^{N'} \Phi(x)^T \Phi(z_j) y_j = \Phi(x)^T \alpha,$$

where $\alpha = \sum_{j=1}^{N'} \Phi(z_j) y_j \in \mathbb{R}^d$. Applying this approximation for the spaces[2] $\mathcal{H}_K^f$, $\mathcal{H}_K^B$, $\mathcal{H}_\kappa$ and $\mathcal{H}_{\hat{\kappa}}$, we obtain finite-dimensional representations of the optimal $f, \{b_j\}, W$ that scale with the order of the matrix feature map, instead of $N_c \cdot n$. Indeed, these are the representations defined in eqs. (19)–(21). To complete the derivation of the cost terms $\hat{J}_d$ and $\hat{J}_m$ in problem (22), we address the functional regularization terms as follows. From the definition of the inner product in $\mathcal{H}_K$:

$$\|g\|^2_{\mathcal{H}_K} = \sum_{i,j=1}^{N'} \langle y_i, K(z_i, z_j) y_j \rangle \approx \sum_{i,j=1}^{N'} \langle \Phi(z_i) y_i, \Phi(z_j) y_j \rangle = \left\langle \sum_{i=1}^{N'} \Phi(z_i) y_i, \sum_{j=1}^{N'} \Phi(z_j) y_j \right\rangle = \|\alpha\|^2 .$$

7 Solution Algorithm

The fundamental structure of the solution algorithm consists of alternating between the dynamics and metric sub-problems derived from problem (22). However, there are three fundamental challenges in solving these problems. First, enforcing the constraints exactly requires a feasible initialization (either with the dynamics or metric function parameters). Given our use of random matrix feature approximations, this is simply intractable. Second, even with an initial feasible guess, due to the alternation, one may lose feasibility within either of the sub-problems stated in Theorems 3 and 4. This is a fundamental numerical challenge of bilinear optimization. Third, since the constraint set X_c includes at least the state samples from the demonstration tuples (plus additional random samples from $\mathcal{X}$), enforcing the LMIs over all constraint points at each iteration is again computationally intractable.

To address these challenges, we present a solution algorithm that (i) eliminates the need for a feasible initial guess, thereby permitting cold-starts, (ii) leverages sub-sampling and *dynamic* updating of the constraint set for each iteration sub-problem to maintain tractability, and (iii) performs feasibility corrections using efficient, unconstrained second-order optimization. We first give the full formulation of the relevant optimization sub-problems and then provide a pseudocode summary of the algorithm at the end.

Let $X_c^{(k)}$ denote the finite sample constraint set at iteration k. In particular, $X_c^{(k)} \subset X_c$ with $N_c^{(k)} := |X_c^{(k)}|$ being ideally much less than N_c, the cardinality of the full constraint set X_c. Formally, each major iteration k is characterized by four minor steps (sub-problems):

1. **Finite-dimensional dynamics sub-problem**:

$$\begin{aligned} \min_{\substack{\alpha, \beta_j, j=1,\dots,m, \\ s \geq 0}} \quad & \sum_{i=1}^{N} \|\hat{f}(x_i) + \hat{B}(x_i) u_i - \dot{x}_i\|^2 + \mu_f \|\alpha - \alpha^{(k-1)}\|^2 \\ & + \mu_b \sum_{j=1}^{m} \|\beta - \beta_j^{(k-1)}\|^2 + \mu_s \|s\|_1 \qquad (34a) \\ \text{s.t.} \quad & \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha, \tilde{\theta}_{ij}^{(k-1)}) \preceq s(x_i) I_{n-m} \quad \forall x_i \in X_c^{(k)} \qquad (34b) \\ & s(x_i) \leq \bar{s}^{(k-1)} \quad \forall x_i \in X_c^{(k)}, \qquad (34c) \end{aligned}$$

where $\{\alpha^{(k-1)}, \beta_j^{(k-1)}, \tilde{\theta}_{ij}^{(k-1)}\}$ are the dynamics and metric parameters obtained from the previous major iteration[3], and $\mu_s \in (0,1)$ is an additional regularization parameter for s, a

[2]To apply this decomposition to $W(x)$, we simply leverage vector-valued feature maps for each entry of $W(x)$.

[3]We set $\alpha^{(0)}, \{\beta_j^{(0)}\}_{j=1}^m, \tilde{\theta}_{ij}^{(0)} = 0$.

non-negative slack vector in $\mathbb{R}^{N_c^{(k)}}$. The quantity $\bar{s}^{(k-1)}$ is defined as

$$\bar{s}^{(k-1)} := \max_{x_i \in X_c} \bar{\lambda}\left(\mathcal{F}^{(k-1)}_{\lambda+\epsilon_\lambda}(x_i)\right), \quad \text{where}$$
$$\mathcal{F}^{(k-1)}_{\lambda+\epsilon_\lambda}(x_i) := \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha^{(k-1)}, \tilde{\theta}^{(k-1)}_{ij}).$$

That is, $\bar{s}^{(k-1)}$ captures the worst violation for the stability LMI over the entire constraint set X_c, given the parameters at the end of iteration $k-1$. Denote $\alpha^{(k)}$, $\{\beta^{(k)}_j\}_{j=1}^m$ to be the optimizers of this sub-problem.

2. **Compute an upper bound $\bar{s}^{(k)\prime}$ on the maximum allowed violation of the stability LMI for the current constraint set $X_c^{(k)}$:**

```
while    $\min_{x_i \in X_c^{(k)}} \underline{\lambda}(W(x_i)) < \delta_{\underline{w}} + \epsilon_{\underline{w}}$    (35a)
    do   $\min_{\tilde{\theta}_{ij}} \sum_{x_i \in X_c^{(k)}} \Big[\psi_w\left(\bar{\lambda}\left((\delta_{\underline{w}} + \epsilon_{\underline{w}})I_n - W(x_i)\right)\right) + \mu'_w \psi_{\mathcal{F}}\left(\bar{\lambda}\left(\mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha^{(k)}, \tilde{\theta}_{ij})\right)\right)\Big] + \mu'_w \sum_{i,j} \|\tilde{\theta}_{ij} - \tilde{\theta}^{(k-1)}_{ij}\|^2,$    (35b)
    $\mu'_w \leftarrow 0.5\mu'_w$    (35c)
```

where $\mu'_w > 0$ is a regularization parameter that is exponentially decayed until the `while` loop termination condition is met. The functions $\psi_w(\cdot)$ and $\psi_{\mathcal{F}}(\cdot)$ are penalty functions. Let $\tilde{\theta}'_{ij}$ be the metric parameters when the termination condition is met. We then set

$$\bar{s}^{(k)\prime} := \max_{x_i \in X_c^{(k)}} \bar{\lambda}\left(\mathcal{F}_{\lambda+\epsilon_\lambda}(x_i; \alpha^{(k)}, \tilde{\theta}'_{ij})\right). \tag{36}$$

The primary objective of this step is to compute an upper bound on the stability LMI over the constraint set $X_c^{(k)}$ with respect to a dual metric that satisfies the positive definiteness condition at all points in this set. This upper-bound will then be used as a constraint within the metric sub-problem, described next. The reason one cannot use $\bar{s}^{(k-1)}$ as in the dynamics sub-problem is because this value is computed over the entire constraint set at the end of the previous iteration, with a metric that is potentially *not* positive definite at all points. Since the metric sub-problem strictly enforces the positive definiteness constraint on W, $\bar{s}^{(k-1)}$ is not a valid upper bound on the stability LMI constraint, and can lead to infeasibility for the metric sub-problem. As long as there exists a set of $\tilde{\theta}_{ij}$ such that the dual metric satisfies the positive definiteness constraint at all points, the `while` loop above will always terminate given a sufficiently small μ'_w. In our implementation, we initialize μ'_w to be equal to μ_w. In Section 7.1 we provide an efficient second-order method for solving this *unconstrained* optimization.

3. **Finite-dimensional metric sub-problem**:

$$\min_{\tilde{\theta}_{ij},\underline{w},\overline{w},s\geq 0} \quad (\overline{w}-\underline{w}) + \mu_w \sum_{i,j} \|\tilde{\theta}_{ij} - \tilde{\theta}_{ij}^{(k-1)}\|^2 + (1/\mu_s)\|s\|_1 \tag{37a}$$

$$\text{s.t.} \quad \mathcal{F}_{\lambda+\epsilon_\lambda}(x_i;\alpha^{(k)},\tilde{\theta}_{ij}) \preceq s(x_i) I_{n-m} \quad \forall x_i \in X_c^{(k)} \tag{37b}$$

$$s(x_i) \leq \bar{s}^{(k)\prime} \quad \forall x_i \in X_c^{(k)} \tag{37c}$$

$$(\underline{w}+\epsilon_{\underline{w}})I_n \preceq W(x_i) \preceq \overline{w} I_n \quad \forall x_i \in X_c^{(k)}, \tag{37d}$$

$$\underline{w} \geq \delta_{\underline{w}}. \tag{37e}$$

The key elements to note here are (i) the use of $\alpha^{(k)}$, i.e., the computed dynamics parameters from the dynamics sub-problem; (ii) the $1/\mu_s$ regularization term on the slack variable to emphasize that the primary objective of the metric sub-problem is to reduce the overall violation of the stability constraint; and (iii) use of the upper-bound $\bar{s}^{(k)\prime}$ computed from Step 2 to force monotonic improvement in constraint violation.

4. **Update $X_c^{(k)}$ sub-problem**.

Choose a tolerance parameter $\delta > 0$. Then, define

$$\nu^{(k)}(x_i) := \max\left\{\bar{\lambda}\left(\mathcal{F}^k_{\lambda+\epsilon_\lambda}(x_i)\right), \bar{\lambda}\left((\delta_{\underline{w}}+\epsilon_\delta)I_n - W(x_i)\right)\right\} \quad \forall x_i \in X_c, \tag{38}$$

and set

$$X_c^{(k+1)} := \left\{x_i \in X_c^{(k)} : \nu^{(k)}(x_i) > -\delta\right\} \bigcup \left\{x_i \in X_c \setminus X_c^{(k)} : \nu^{(k)}(x_i) > 0\right\}. \tag{39}$$

Thus, in the update $X_c^{(k)}$ step, we balance addressing points where constraints are being violated ($\nu^{(k)} > 0$) and discarding points where constraints are satisfied with sufficient strict inequality ($\nu^{(k)} \leq -\delta$). This prevents overfitting to any specific subset of the constraint points. A potential variation to the union above is to only add up to L constraint violating points from $X_c \setminus X_c^{(k)}$ (e.g., corresponding to the L worst violators), where L is a fixed positive integer. Indeed this is the variation used in our experiments and was found to be extremely efficient in balancing the size of the set $X_c^{(k)}$ and thus, the complexity of each iteration. This adaptive sampling technique is inspired by *exchange algorithms* for semi-infinite optimization, as the one proposed in (Zhang et al., 2010) where one is trying to enforce the constraints at *all* points in a compact set $\mathcal{X}$.

The use of the modified regularization terms above (i.e., penalizing change in the parameters from the previous iteration as opposed to absolute penalty) is, in the spirit of trust region methods, to prevent large updates to the parameters due to the dynamically updating constraint set $X_c^{(k)}$. At the first major iteration of the problem however, with $\alpha^{(0)}, \{\beta_j^{(0)}\}_{j=1}^m, \tilde{\theta}_{ij} = 0$, this is the same as an absolute regularization term. To ensure non-degeneracy in the stability LMI constraint within the first dynamics sub-problem iteration, we initialize the algorithm with $W(x_i) = I_n$. The full pseudocode for the CCM-Regularized (CCM-R) dynamics learning algorithm is summarized in Algorithm 1.

Some comments are in order. First, convergence in Algorithm 1 is declared if either progress in the solution variables stalls or all constraints are satisfied within tolerance. Using dynamic

Algorithm 1 CCM - Regularized (CCM-R) Dynamics Learning

```
1: Input: Dataset $\{x_i, u_i, \dot{x}_i\}_{i=1}^N$, constraint set $X_c$, regularization constants $\{\mu_f, \mu_b, \mu_w, \mu_s\}$, constraint tolerances $\{\epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}\}$, discard tolerance parameter $\delta$, Initial # of constraint points: $N_c^{(0)}$, Max # iterations: $N_{\max}$, termination tolerance $\varepsilon$.
2: $k \leftarrow 1$, converged $\leftarrow$ false, $W(x) \leftarrow I_n$, $\{\alpha^{(0)}, \{\beta_j^{(0)}\}_{j=1}^m, \tilde{\theta}_{ij}\} \leftarrow 0$.
3: $X_c^{(0)} \leftarrow$ RandSample$(X_c, N_c^{(0)})$
4: while $\neg$converged $\wedge$ $k < N_{\max}$ do
5:     $\{\alpha^{(k)}, \{\beta_j^{(k)}\}\} \leftarrow$ Solve (34)
6:     $\bar{s}^{(k)\prime} \leftarrow$ Solve (35)
7:     $\{\tilde{\theta}_{ij}^{(k)}, \underline{w}, \overline{w}\} \leftarrow$ Solve (37)
8:     $X_c^{(k+1)}, \bar{s}^{(k)}, \nu^{(k)} \leftarrow$ Update $X_c^{(k)}$ using (39)
9:     $\Delta \leftarrow \max\left\{\|\alpha^{(k)} - \alpha^{(k-1)}\|_\infty, \|\beta_j^{(k)} - \beta_j^{(k-1)}\|_\infty, \|\tilde{\theta}_{ij}^{(k)} - \tilde{\theta}_{ij}^{(k-1)}\|_\infty\right\}$
10:    if $\Delta < \varepsilon$ or $\nu^{(k)}(x_i) < \varepsilon \quad \forall x_i \in X_c$ then
11:        converged $\leftarrow$ true.
12:    end if
13:    $k \leftarrow k + 1$.
14: end while
```

sub-sampling for the constraint set at each iteration implies that (i) the matrix function $W(x)$ at iteration k resulting from variables $\tilde{\theta}_{ij}^{(k)}$ does *not* have to correspond to a valid dual metric for the interim learned dynamics at iteration k, and (ii) a termination condition based on constraint satisfaction at all N_c points is justified by the fact that at each iteration, we are solving relaxed sub-problems that collectively generate a sequence of lower-bounds on the overall objective.

Second, as a consequence of this iterative procedure, the dual metric and contraction rate pair $\{W, \lambda\}$ do not possess any sort of "control-theoretic" optimality. For instance, in (Singh et al., 2017), for a known stabilizable dynamics model, both these quantities are optimized for robust control performance. In this work, these quantities are used solely as *regularizers* to *promote* stabilizability of the learned model. Following the recovery of a fixed dynamics model from the algorithm, one may leverage global CCM optimization algorithms as the one presented in (Singh et al., 2017) to compute an optimal *robust* CCM.

7.1 Solving the Upper Bound Sub-Problem

Step 2 in the iterative algorithm outlined in the previous section entails solving a non-smooth, but convex, unconstrained optimization to compute a viable upper bound on the stability LMI for its inclusion as a constraint within the metric sub-problem. While the problem can be formulated as a Semi-Definite Program (SDP) through the use of epigraph LMIs, this would result in yet another computationally intensive step. Instead, recognizing that the purpose of this part of the algorithm is *not* to update any problem variables, one can use faster unconstrained optimization techniques to approximately solve this problem.

Specifically, we employ Newton descent to solve this problem with backtracking line search. The issue however, lies in taking the second order derivatives of the maximum eigenvalue function of an affinely parameterized matrix (recall W is linear in $\tilde{\theta}_{ij}$ and $\mathcal{F}_\lambda$ is linear in $\tilde{\theta}_{ij}$ for fixed α). For a given affinely parameterized symmetric matrix $G(\theta) \in \mathbb{R}^{d\times d}$, the gradient and Hessian of $\bar{\lambda}(G(\theta))$ with respect to θ has the components (Overton and Womersley, 1995):

$$\frac{\partial \bar{\lambda}(G)}{\partial \theta^i}(\theta) = \bar{v}_1^T \frac{\partial G(\theta)}{\partial \theta^i} \bar{v}_1,$$
$$\frac{\partial^2 \bar{\lambda}(G)}{\partial \theta^i \partial \theta^j}(\theta) = 2 \sum_{k>1}^{d} \frac{\bar{v}_1^T \frac{\partial G(\theta)}{\partial \theta^i} \bar{v}_k \cdot \bar{v}_1^T \frac{\partial G(\theta)}{\partial \theta^j} \bar{v}_k}{\lambda_1 - \lambda_k},$$

where we assume the eigenvalue ordering $\lambda_1 \geq \cdots \geq \lambda_d$, and $\bar{v}_1, \ldots, \bar{v}_d$ are the associated normalized eigenvectors. Clearly, the Hessian is undefined when the multiplicity of the maximum eigenvalue is greater than one. While there exist several works within the literature on modified parameterizations of the matrix to allow the use of second order methods, these techniques require knowing the multiplicity of the maximum eigenvalue at the optimal parameters – which is impossible to guess in this case.

Instead, we leverage a stochastic smoothing approximation borrowed from (D'Aspremont and Karoui, 2014), whereby one approximates the maximum eigenvalue of $G(\theta)$ as

$$\max_{i=1,\ldots,k} \bar{\lambda}\left(G(\theta) + \frac{\sigma}{d} z_i z_i^T\right),$$

where $\sigma > 0$ is a small noise parameter, and z_i are i.i.d. samples from $\mathcal{N}(0, I_d)$. In particular, $\bar{\lambda}\left(G(\theta) + \frac{\varepsilon}{n} z_i z_i^T\right)$ is unique with probability one, for any $z_i \sim \mathcal{N}(0, I_d)$ (see (D'Aspremont and Karoui, 2014), Prop. 3.3 and Lemma 3.4). While the algorithm in (D'Aspremont and Karoui, 2014) leverages a finite-sample expectation of the randomized function above and its gradient, for our implementation, we simply leverage the uniqueness property induced by the use of the random rank one perturbation, and compute the gradient and Hessian expressions using the eigenvalue decomposition of the matrix $G(\theta) + (\varepsilon/d)zz^T$, where z is a single Gaussian vector sample. Additionally, we employ early termination of the Newton iterations if the `while` loop termination condition is met by the current iterate.

7.2 Revisiting PVTOL

We now revisit the results presented in Section 2.1 and shed additional light on the performance of the CCM-R model.

The training dataset was generated in three steps. First, a fixed set of waypoint paths in (p_x, p_z) was randomly generated. Second, for each waypoint path, multiple smooth polynomial splines were fitted using a minimum-snap algorithm. To create variation amongst the splines, the waypoints were perturbed within Gaussian balls and the time durations for the polynomial segments were also randomly perturbed. Third, the PVTOL system was simulated with perturbed initial conditions and the polynomial trajectories as references, and tracked using a sub-optimally tuned Proportional-Derivative (PD) controller, thereby emulating a noisy or imperfect demonstrator. These final simulated paths were sub-sampled at a 0.1 s resolution to create the datasets. The variations created at each step of this process were sufficient to generate a rich exploration of the state-space for training.

The matrix feature maps used for all models (N-R, R-R, and CCM-R) are derived from the random matrix feature approximation of the Gaussian separable kernel; the relevant parameters are provided in Appendix A. We enforced the CCM regularizing constraints for the CCM-R model at $N_c = 2426$ points in the state-space, composed of the N demonstration points in the training dataset and randomly sampled points from $\mathcal{X}$ (recall that the CCM constraints do not require samples of $u, \dot{x}$). The contraction rate λ was set to 0.1 and the termination tolerance ε was set to 0.01. All other tolerance parameters are provided in Appendix A.

As the CCM constraints were relaxed to hold pointwise on the finite constraint set X_c as opposed to everywhere on $\mathcal{X}$, in the spirit of viewing these constraints as regularizers for the model, we simulated both the R-R and CCM-R models using the TV-LQR feedback controller. This also helped ensure a more direct comparison of the quality of the learned models themselves, independently of the tracking feedback controller. The results are virtually identical using a tracking MPC controller and yield no additional insight.

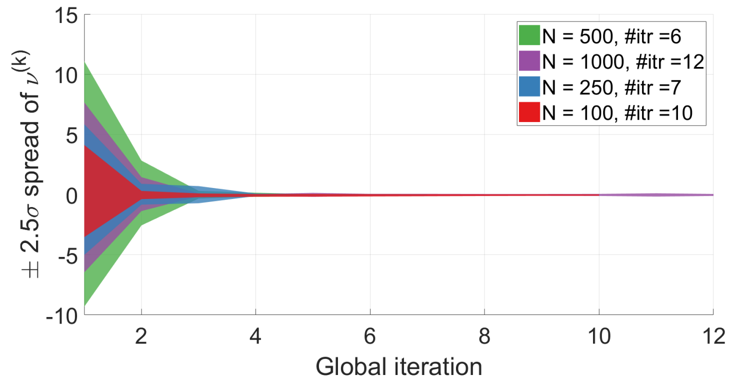
Given the inherently non-convex nature of the original problem (recall the bilinear matrix inequality) and the complex alternations used within the learning algorithm, exact convergence analysis remains elusive. In its place, we provide an empirical study of the convergence behavior. Figure 6 plots the training curves generated by the CCM-R algorithm for varying demonstration dataset sizes N. In particular, we plot (i) the evolution of the distribution of the constraint violation vector ν defined in (38) over the training constraint set X_c, and (ii) evolution of the regression error and fraction of violations (i.e., average number of points with $\nu > 0$) over an independent validation set, as a function of global iteration k. The dynamic constraint set was initialized with 250 points (sampled randomly from the 2426 constraint points in the full training constraint set), and over all iterations, the size of $X_c^{(k)}$ remained below 400 points – a drastic factor of improvement over an intractable brute-force approach. For $N \in \{100, 250, 500\}$, the algorithm terminated with all constraints satisfied below the termination threshold of $\varepsilon = 0.01$. For $N = 1000$, we manually terminated the algorithm after 12 global iterations after observing a stall in the number of violations.

From Figure 6a, we see the effectiveness of the sub-sampling constraint set method in efficiently eliminating all training violations. Several comments are in order regarding Figure 6b. First, as the number of demonstration tuples N goes down, the curves shift further to the right, indicating higher validation error, as expected. Second, the iterates seem to clearly exhibit a two-phase convergent behavior. During the first phase, the fraction of violations on the validation set drops rapidly and approaches a steady-state. During the second phase, the fraction of violations remains roughly constant as the validation error drops monotonically, with decreasing drop rate. The training and final validation errors for all models are summarized in Table 1.

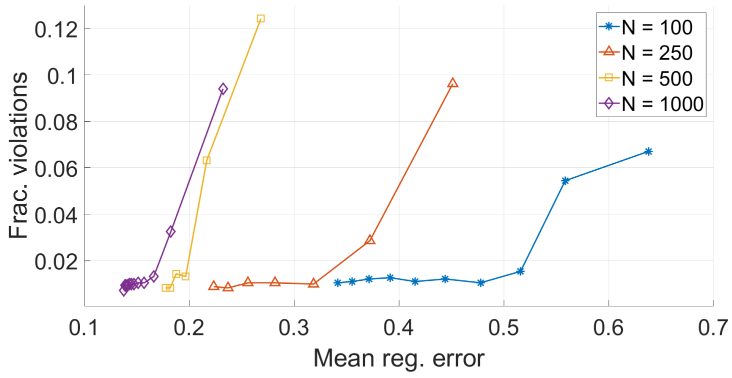
	N-R		**R-R**		**CCM-R**		
N	Train err.	Val err.	Train err.	Val err.	Train err.	Val err.	Frac. viol.
100	0.001	0.072	0.125	0.380	0.012	0.342	0.0008 (0.010)
250	0.003	0.088	0.125	0.243	0.010	0.223	0.0004 (0.009)
500	0.004	0.047	0.092	0.172	0.006	0.178	0.0016 (0.008)
1000	0.01	0.056	0.08	0.146	0.003	0.138	0.0021 (0.007)

Table 1: Comparison of average (over N tuples) training and validation (over 2000 tuples) regression error norms for all 3 models. Also shown are the fraction of violations ($\nu > 0$) on the training (validation) sets for the CCM-R models. The termination threshold for CCM-R training was $\nu < \varepsilon = 0.01$ for all points in the training constraint set.

An interesting trend to observe here is that the training errors for the N-R and CCM-R models are significantly closer (by an order of magnitude) together than R-R and N-R. Consistent with the trend in Figure 5, N-R features the smallest validation errors. Thus, while the CCM-R model manages to perform well in terms of regression error on the training set (almost on par with N-R), the resulting model does not suffer from the overfitting trend observed with the N-R model.



(a) $\pm 2.5\sigma$ spread of the constraint violation vector ν over *training set* as a function of global iteration. `#itr` denotes the total number of global iterations.



(b) Evolution of mean regression error norm and fraction of violations over *validation* set with global iteration number. The markers delineate the iterations. The curves proceed from right to left.

Figure 6: Simulation data training curves for all CCM-R models.

Surprisingly, the final validation errors of the CCM-R models and the R-R models are almost identical. *Despite this however, the CCM-R model significantly outperforms the R-R model, especially when learned from smaller supervised datasets.* What sets the CCM-R model apart is the manner in which the dynamics sub-problem improves the regression error over multiple iterations – through the joint minimization of regression error and stability LMI violations. Effectively, as seen in Figure 6b, the iterates move along a level set of the constraint violations, decreasing regression error while maintaining control over the *stabilizability of the learned model.* This validates the benefit of using a control-theoretic regularization technique that is *tailored* to the motion planning task, and the fragility of *only* using traditional measures of performance (such as regression error) for evaluating learned dynamical models[4].

[4]Code for data generation, training, and evaluation is provided at `https://github.com/StanfordASL/SNDL`.

While the analytical PVTOL model studied in simulation is indeed an example of a system that is CCM stabilizable (see (Singh et al., 2017)), in the next section we deploy the algorithm on a full 3D quadrotor testbed, shown in Figure 7, with partially closed control loops to approximately emulate the PVTOL dynamics.

8 Validation on Quadrotor Testbed

Our quadrotor testbed consists of (i) a standard DJI F330 frame, (ii) a Pixhawk autopilot running the estimator and lower-level thrust and angular rate controllers, and (iii) a companion on-board ODROID-XU4 computer running ROS nodes for motion planning and trajectory tracking controller[5] (which generates the thrust and angular rate setpoints for Pixhawk). There is also an Optitrack motion capture system providing inertial position and yaw estimates at 120 Hz, which is fused with the onboard EKF on the Pixhawk.

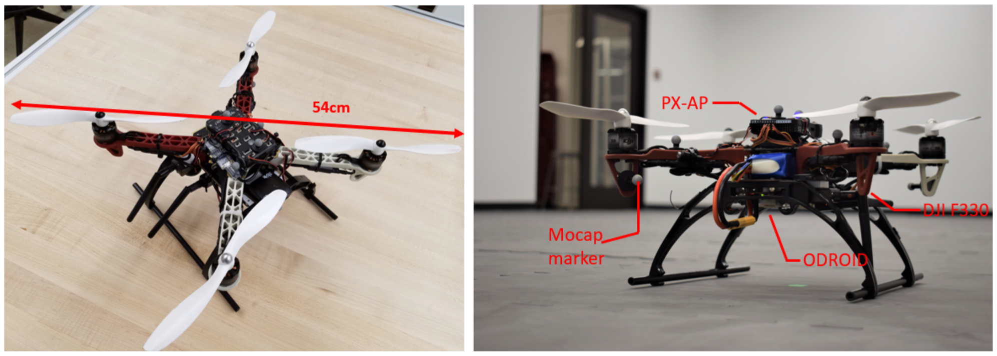


Figure 7: Quadrotor experimental platform, equipped with Pixhawk autopilot (PX-AP) for low-level (thrust and angular rate) control, and ODROID companion computer for planning and trajectory tracking control.

The objective of the flight experiments was to verify whether the performance trends observed in simulation for a relatively simple dynamics model, that was a priori known to be CCM stabilizable, carried over to a far more complex setting where this is unknown. The dynamics to be learned are for the motion of the quadrotor *within the vertical plane*. Accordingly, the training data consisted of samples from trajectories flown within the plane autonomously using a nonlinear trajectory tracking controller. For evaluation, the generated planar trajectories will be tracked using a combination of an *in-plane* controller, derived from the learned dynamics model, and an *out-of-plane* controller derived from known principles (detailed in Section 8.2). The purpose of the out-of-plane controller is simply to keep the quadrotor aligned and fixed within the vertical plane, thereby permitting an evaluation of the quality of the learned planar dynamics.

This is a complex system subject to noise and nonlinear disturbances stemming from aerodynamic effects (e.g., ground-effect) and residual out-of-plane motion, and therefore constitutes a challenging evaluation benchmark.

State and Control Parameterization: We adopt the North-East-Down (NED) body-frame convention and the XYZ-Euler angle sequence parameterized by, in order, roll (ϕ), pitch (θ), and yaw (ψ) angles. For simplicity, we fixed the nominal yaw angle to 0 and inertial X-position to be

[5]The code for these ROS nodes is available for download at `https://github.com/StanfordASL/asl_flight`.

constant so that the planar motion is aligned with the inertial Y-Z plane. The state representation used for the planar dynamics is therefore: $x = (p_y, p_z, \phi, v_y, v_z, \omega_x)$ where (p_y, p_z) are the inertial position in the Y-Z plane, (v_y, v_z) are the body-frame velocities, and ω_x is the body-frame angular rate. The control inputs are set to be the desired net normalized thrust τ and desired body rate $\dot{\phi}_c$. These inputs are fed into the PX4 autopilot, which uses a fast on-board control loop to realize these commands.

The supervisory signal $\dot{x}$ is provided by $(\dot{p}_y, \dot{p}_z, \omega_x, \dot{v}_y, \dot{v}_z, \dot{\omega}_x)$. Note that by kinematics, $\omega_x = \dot{\phi}\cos(\theta)\cos(\psi) + \dot{\theta}\sin(\psi)$. Thus, any non-zero ψ, and/or non-zero $\dot{\theta}$ with non-zero ψ can introduce bias errors into the training data. The signals $\dot{\omega}_x$ and $(\dot{v}_y, \dot{v}_z)$ were obtained by convolving a finite-difference derivative estimate through a moving-average filter.

8.1 Model Training

Training Data Collection: To collect the training data, we created a set of training trajectories within the plane consisting of clockwise and counter-clockwise circles at varying speeds, and "swoop" trajectories (see Figure 8) where the quadrotor swoops towards the ground in a parabolic arc and travels in a straight line close to the ground. The goal of the swoops was to learn data pertaining to ground-effect aerodynamic disturbances. All trajectories were tracked using a CCM-based feedback controller derived from a nominal model of a quadrotor that *neglects any aerodynamic disturbances.* That is, the tracking controller and designed trajectories do not pre-compensate for any aerodynamic disturbances. Instead, we would like for such strategies to evolve directly from the learned model and subsequently derived tracking controller.

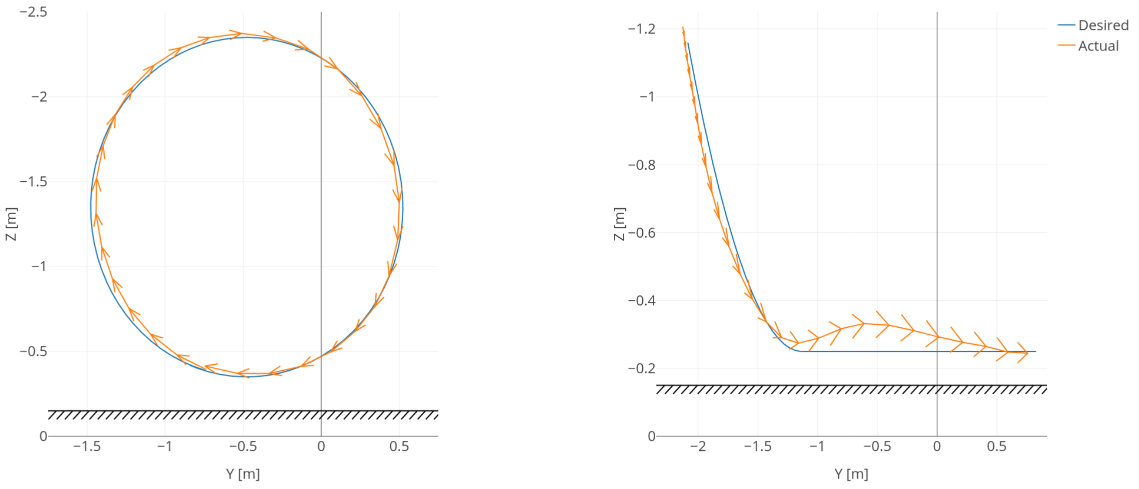


Figure 8: Examples of flown trajectories with a 3D nonlinear tracking feedback controller to create the training dataset. Left: clockwise circle with radius 1 m, period 6 s, and a nominal speed of 1.05 m/s. Right: Swoop trajectory to excite the ground-effect aerodynamic disturbances. Notice the deviation of the quadrotor from the desired trajectory as it approaches the ground. Ideally, a model learned from such data should generate nominal control signals that compensate for such effects.

We collected data from 3 clockwise and 3 counter-clockwise 1 m radius circles at periods of 10, 8 and 6 seconds (corresponding to nominal speeds of 0.63, 0.79, and 1.05 m/s), and two symmetric swoop trajectories. Figure 9 shows the yaw and inertial X tracking errors for the fastest clockwise training circle. Both these quantities pertain to the out-of-plane motion and need to be small in order to minimize any induced bias within the planar motion, which is significantly coupled with the out-of-plane motion. This is indeed observed to be the case, validating the suitability of the collected data to train a planar dynamics model.

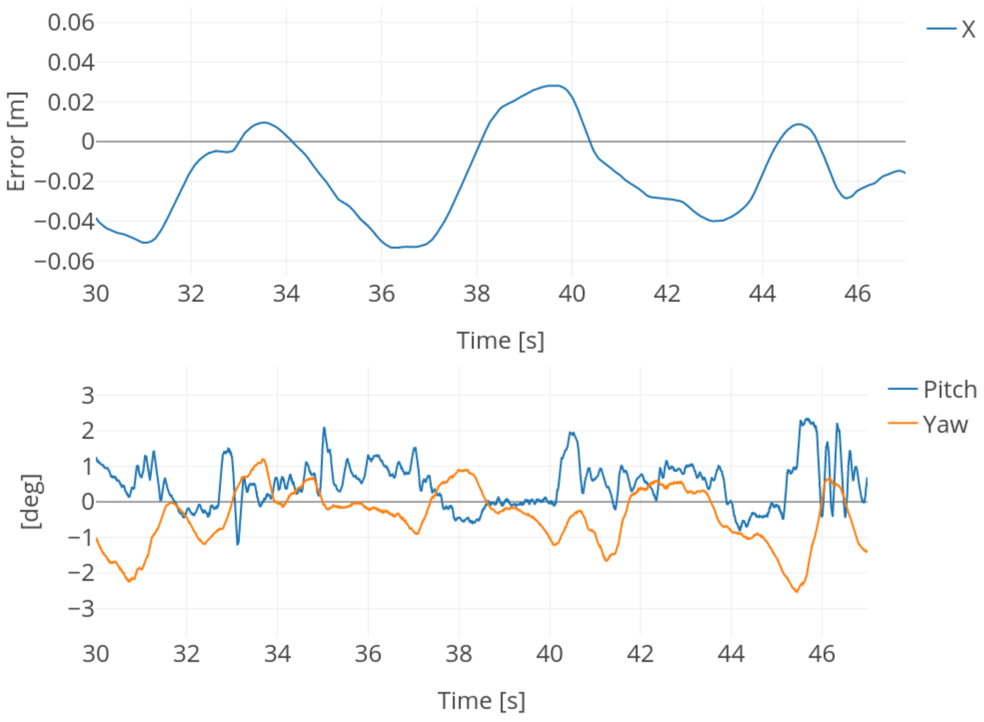


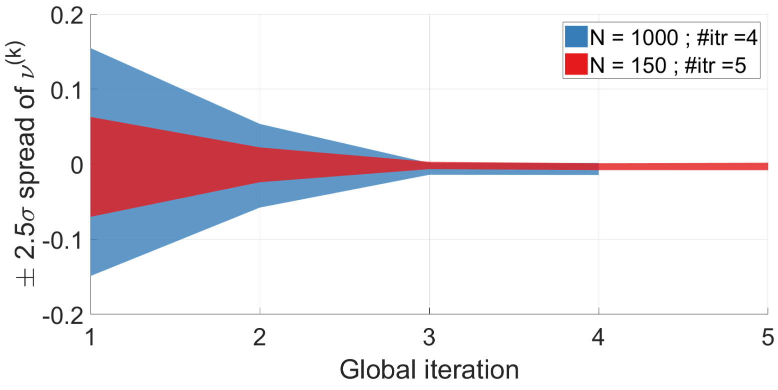
Figure 9: Out-of-plane errors, i.e., inertial X motion and pitch and yaw angles. All errors are sufficiently small, validating the use of the remaining data to train a planar dynamics model.

Figure 8 shows a trace of a swoop training trajectory. Note the clear influence of ground-effect on the Z-tracking error as the quadrotor dips towards the ground and settles into the straight line path. These are exactly the sort of effects we wish to learn.

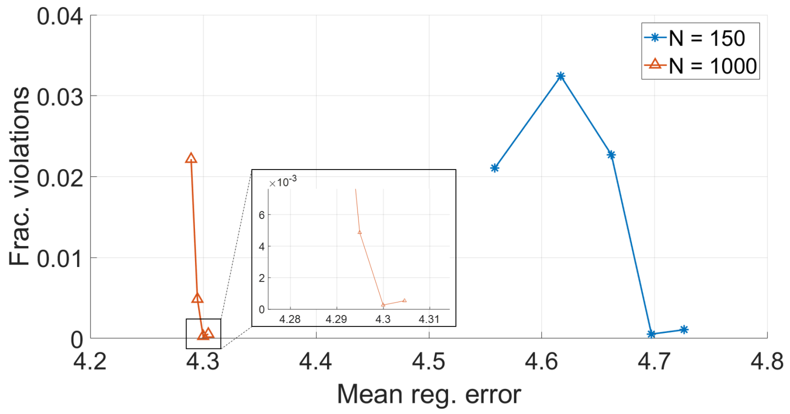
The complete dataset consists of approximately 2 minutes of flight time, logged at 250 Hz. After filtering the finite-difference derivative estimates, we further sub-sampled the data to avoid aliasing effects, and extracted training and validation datasets of 1000 and 3000 $(x, u, \dot{x})$ tuples respectively.

Models Learned: To investigate the hypothesized trend where the control-theoretic regularization from the CCM approach dominates for small supervised training dataset sizes, we learned R-R and CCM-R models at $N = 150$ and $N = 1000$. For the CCM-R models, the full training constraint set X_c comprised of the state samples from the demonstration tuples, plus up to 1600 additional state samples from the collected trajectories, and 500 independently sampled Halton points within the state-space. In total, for both the $N = 150$ and $N = 1000$ models, a total of 3100 constraint points were used.

The dimensionality of the feature space for $\hat{f}, \{\hat{b}_j\}_{j=1}^{2}$, and W were held the same as from the PVTOL simulations (corresponding to $\alpha, \beta_j \in \mathbb{R}^{576}$, and $\tilde{\theta}_{ij} \in \mathbb{R}^{72}$). The constant μ_f was held fixed at 10^{-3} for both R-R and CCM-R models and both N, and μ_b was set to 10^{-2}. The need for a higher regularization constant for β_j stemmed from the high noise content in the $\dot{\phi}_c$ and $\dot{\omega}_x$ signals. The constant μ_w was set to 10^{-3} for $N = 150$ and 10^{-4} for $N = 1000$, as in the PVTOL simulations. Figure 10 gives the training curve plots for the $N = 150$ and $N = 1000$ CCM-R models, and Table 2 summarizes the final measures of performance. The contraction rate λ was again set to 0.1 and all other tolerances from the simulations were held the same. CCM-R models converged in 5 iterations for $N = 150$ and 4 iterations for $N = 1000$.



(a) $\pm 2.5\sigma$ spread of the constraint violation vector ν over *training set* as a function of global iteration. #`itr` denotes the total number of global iterations.



(b) Evolution of mean regression error norm and fraction of violations over *validation* set with global iteration number. The markers delineate the iterations. The curves proceed from left to right.

Figure 10: Testbed data training curves for both CCM-R models.

	R-R		**CCM-R**		
N	Train err.	Val err.	Train err.	Val err.	Frac. viol.
150	4.077	4.549	0.399	4.726	0.0013 (0.0011)
1000	4.149	4.289	0.167	4.305	0.0048 (0.0005)

Table 2: Comparison of average (over N tuples) training and validation (over 3700 tuples) regression error norms for all 3 models. Also shown are the fraction of violations ($\nu > 0$) on the training (validation) sets for the CCM-R models. The termination threshold for CCM-R training was $\nu < \varepsilon = 0.01$ for all points in the training constraint set.

From Table 2, we note that the final regression validation errors for R-R and CCM-R are again close together, as in the simulation case. However, the training curves in Figure 10 appear to go in the opposite direction, i.e., validation regression error for CCM-R grows with iterations, starting from around 4.558 (almost identical to the R-R model), and growing to a final value of 4.726 as the number of violations converges to 0. A potential reason could be that the noise in the demonstration tuples (i.e., from $\dot{x}$) forces more drastic updates in the parameters in order to find a compatible dual metric W. Nevertheless, as will be seen in the evaluation results, this small sacrifice in validation error performance works in the CCM-R model's favor.

8.2 Out-of-Plane Control for Enabling Evaluation

The out-of-plane errors in the training data were quite small (since the tracking controller was a full 3D state feedback controller). The learned planar dynamics will be used to generate a planar desired trajectory and LQR feedback tracking controller. To ensure that the quadrotor remains within the plane however, we need to separately close the loops on inertial X and yaw motion.

The nominal (ignoring disturbances) equation for p_x is given by:

$$\ddot{p}_x = -\tau \sin(\theta),$$

which is completely decoupled from the planar dynamics. In order to regulate p_x to an arbitrary constant $\bar{p}_x$ (without loss of generality, assume $\bar{p}_x = 0$), given a desired $\tau > 0$ computed using the planar control loop, we choose a desired inertial acceleration $\ddot{p}_{x,des} := -k_x p_x - k_v \dot{p}_x$, where $k_x, k_v > 0$ are control gains. Inverting the dynamics above, we obtain a desired pitch command θ_c:

$$\sin(\theta_c) = -\frac{\ddot{p}_{x,des}}{\tau} = \frac{k_x p_x + k_v \dot{p}_x}{\tau}.$$

From the above equation, we deduce a desired nominal pitch rate $\dot{\theta}_{des}$:

$$\dot{\theta}_{des} \cos(\theta_c) = \frac{\tau(k_x \dot{p}_x + k_v \ddot{p}_x) - \dot{\tau}(k_x p_x + k_v \dot{p}_x)}{\tau^2} \approx \frac{k_x \dot{p}_x}{\tau},$$

where we neglect the $\dot{\tau}$ and $\ddot{p}_x$ terms. The actual pitch rate command $\dot{\theta}_c$ sent to the quadrotor is then given by:

$$\dot{\theta}_c = \dot{\theta}_{des} + k_\theta(\theta_c - \theta),$$

which is a combination of the feedforward desired pitch rate plus a proportional feedback term on the error with respect to the desired pitch angle with gain $k_\theta > 0$. Finally, to keep yaw ψ at zero, we implemented a simple proportional controller to generate a desired yaw rate command $\dot{\psi}_c$:

$$\dot{\psi}_c = -k_\psi \psi,$$

with gain $k_\psi > 0$. In the next section we provide plots of p_x, θ, and ψ to check how well the planar assumption holds.

8.3 Evaluation

Test Trajectory Generation: The evaluation task is similar to that used for simulations, except that instead of initializing the quadrotor at random initial configurations and asking it to stabilize to a hover (an arguably dangerous test), we designed a desired nominal trajectory in the Y-Z plane consisting of segments of a continuous, smooth figure-eight, denoted as $(p_y^{\text{ref}}(t), p_z^{\text{ref}}(t))$. The nominal state and control trajectory for the quadrotor was then computed as the solution to the following trajectory optimization problem:

$$(x^*(\cdot), u^*(\cdot)) = \underset{x(\cdot),u(\cdot)}{\operatorname{argmin}} \int_0^T \begin{bmatrix} p_y(t) - p_y^{\text{ref}}(t) \\ p_z(t) - p_z^{\text{ref}}(t) \end{bmatrix}^T Q \begin{bmatrix} p_y(t) - p_y^{\text{ref}}(t) \\ p_z(t) - p_z^{\text{ref}}(t) \end{bmatrix} + \begin{bmatrix} \tau(t) - g \\ \dot{\phi}_c(t) \end{bmatrix}^T R \begin{bmatrix} \tau(t) - g \\ \dot{\phi}_c(t) \end{bmatrix} dt,$$

where g is the gravitational acceleration, $T = 10$ s, and Q, R are cost weighting matrices in $\mathbb{S}_2^{>0}$. To avoid the initial jump in desired velocity for a quadrotor starting from rest, we designed the trajectory in three segments (see Figure 11). The quadrotor accelerates along half of the figure-eight path to full speed (point A to B in Figure 11) in the first segment. This is followed by a full figure-eight loop (back to B) in the second segment. Finally the quadrotor decelerates along the second half of the figure-eight back to zero velocity during the third segment. Each segment was designed to overlap with the desired figure-eight shape, and the desired speed was modulated along the path using a smooth time-varying phase function for the first and third segments. These two segments along with the speed along the trajectory are depicted in Figure 11.

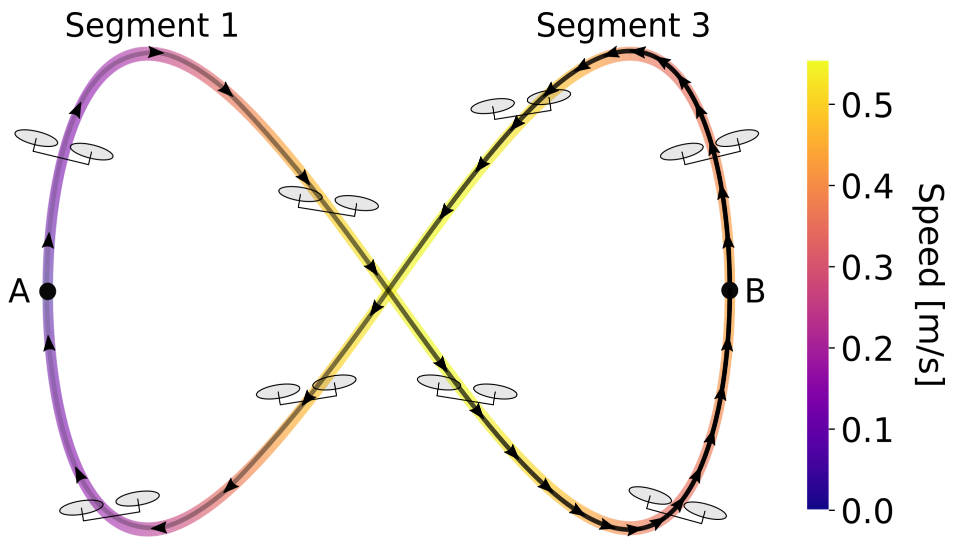


Figure 11: Illustration of the desired Y-Z plane trajectory to be flown for evaluation of the learned models. Shown here are the first (accelerating from A to B) and third (decelerating from B to A) segments of the trajectory with the indicated speed profile. The middle segment (not shown) is a complete figure-eight maneuver that overlaps with the above path. The actual reference state/control trajectories for each model are computed using trajectory optimization where the cost to be minimized is a combination of control effort and the deviation from the desired figure-eight path above. All computed trajectories, projected onto the Y-Z plane overlap with the desired figure-eight maneuver.

For each model tested, three trajectory optimization problems were solved corresponding to the three segments, each with the same time-span of 10 s. The middle segment therefore represents the most challenging and aggressive part of the trajectory, executed following the acceleration segment. The planar tracking controller was set as the steady-state LQR controller computed from the TV-LQR solution (to avoid having to additionally compute and store the true time-varying gain solution from the backward Riccati differential equation).

Results: We present tracking results[6] for all 4 models: (i) CCM-R $N = 1000$, (ii) R-R $N = 1000$, (iii) CCM-R $N = 150$, and (iv) R-R $N = 150$. Figure 12 plots tracking errors for the $N = 150$ case and Table 3 summarizes the tracking performance for all models in terms of RMS and maximum translational tracking errors. The quadrotor is unable to track the R-R $N = 150$ model generated trajectory and quickly becomes unstable as it accelerates into the middle segment of the trajectory. A human pilot needed to take control just as the quadrotor crashed into the ground. At the point of the takeover, the net Y tracking error was approximately 1.5 m.

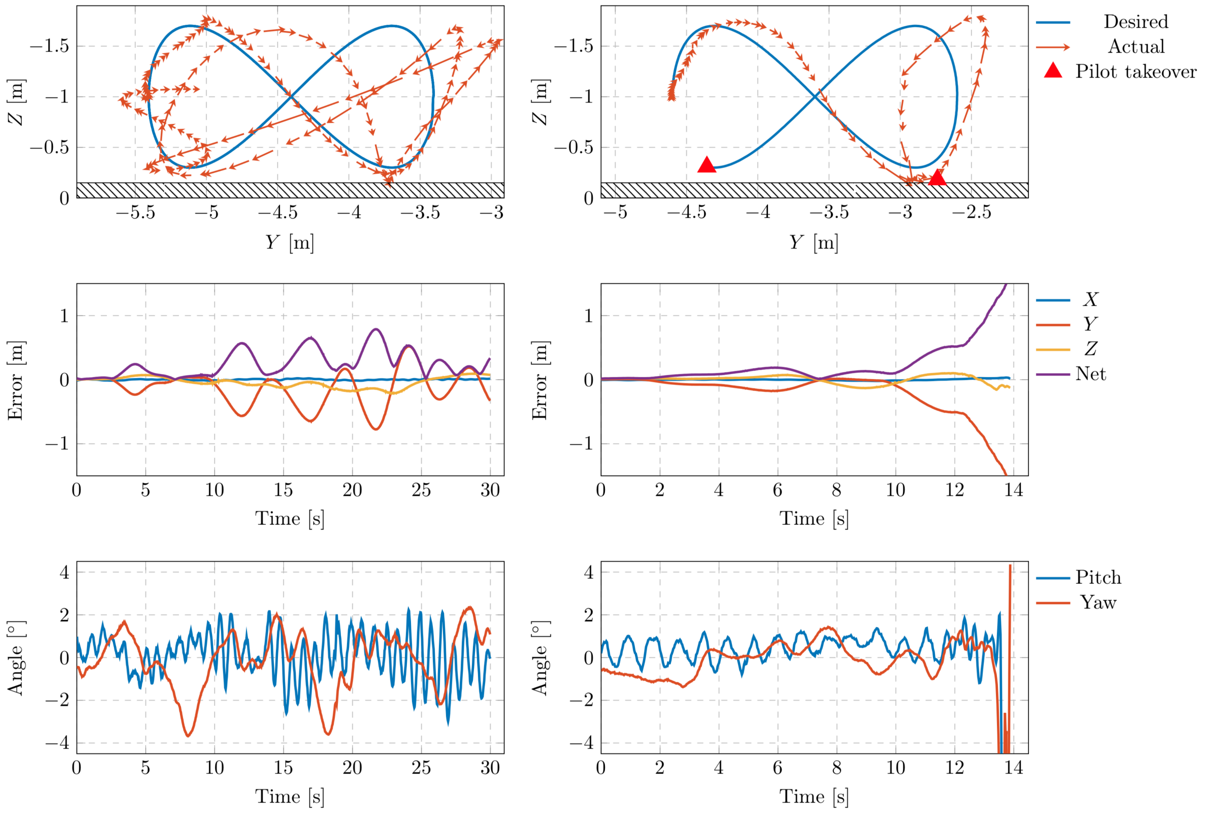


Figure 12: Visualization of tracking results for the learned models CCM-R $N = 150$ (left column) and R-R $N = 150$ (right column). The top row shows the desired and actual trajectory traces in the inertial Y-Z plane, while the middle row shows the error over time in the inertial X, Y, and Z coordinates. The quadrotor is unable to track the R-R $N = 150$ model generated trajectory and quickly becomes unstable as it accelerates into the middle segment of the trajectory. A human pilot needed to take control just as the quadrotor crashed into the ground. At the point of the takeover, the net Y tracking error was 1.5 m. Tracking the CCM-R $N = 150$ model generated trajectory, the quadrotor achieves bounded tracking errors and successfully completes the trajectory. The bottom row shows the pitch and yaw angles over time, which remain close to zero, thereby ensuring our planar motion assumption is valid.

[6]Videos of the flight experiments can be found at https://youtu.be/SK1tsYrXXUY.

In contrast, while the tracking performance for the CCM-R model is rather far from what could be achieved with an accurate model, all errors remain bounded. There is a point where the quadrotor grazes the ground as it passes through the fastest part of the middle segment, traveling downwards. The margin of error (distance between the lowest point on the trajectory and the ground) is 15 cm, which is a challenging tracking constraint to meet for a quadrotor operating using a model trained with just 150 samples of supervision. Despite this, the quadrotor recovers and successfully completes the remaining portion of the trajectory. Figure 1 shows a time-lapse of the quadrotors during this middle segment of the maneuver.

From Table 3, as expected for the $N = 1000$ case, the tracking numbers for CCM-R and R-R are on par, with remaining differences at the noise level. For the $N = 150$ case, we only present numerics up until 14 s, which is when the R-R case crashes. For this range, CCM-R significantly outperforms R-R, let alone the fact that the quadrotor operating with the CCM-R model manages to complete the entire 30 s trajectory while maintaining stability. The results confirm the trend observed within simulations that, at low sample regimes, the stabilizability constraints enforced during the CCM-R model learning process have a dramatic context-driven regularization effect.

	CCM-R $N = 1000$		**R-R $N = 1000$**		**CCM-R $N = 150$**		**R-R $N = 150$**	
Error	RMS	Max.	RMS	Max.	RMS	Max.	RMS	Max.
Y	0.150	0.445	0.126	0.290	0.208	0.567	0.423	1.618
Z	0.066	0.167	0.076	0.196	0.047	0.090	0.068	0.157
Net	0.165	0.446	0.148	0.308	0.213	0.570	0.429	1.623

Table 3: Numerical tracking results for all learned models. These include both root mean square (RMS) and maximum error values (in meters) over the entire trajectory. At $N = 1000$ training points, both the CCM-R and R-R models yield similar tracking performance with remaining differences at the noise level. The tracking numbers for $N = 150$ are presented up until 14 s, which is when the crash occured with the R-R model. CCM-R clearly outperforms R-R during those first 14 s, let alone the fact that the quadrotor operating with the CCM-R model successfully completes the entire 30 s trajectory while maintaining bounded errors.

9 Conclusions and Future Work

We presented a framework for learning *controlled* dynamics from demonstrations for the purpose of trajectory optimization and control for continuous robotic tasks. By leveraging tools from non-linear control theory, chiefly, contraction theory, we introduced the concept of learning *stabilizable* dynamics, a notion which guarantees the existence of feedback controllers for the learned dynamics model that ensures trajectory trackability. Borrowing tools from reproducing kernel Hilbert spaces and convex optimization, we proposed a bi-convex semi-supervised algorithm for learning the dynamics, and provided a substantial numerical study of its performance and comparison with traditional regression techniques. In particular, we validated the algorithm in simulations for a planar quadrotor system, and on a full quadrotor hardware testbed with partially closed control loops to emulate planar quadrotor dynamics. The results lend credence to the hypothesis that enforcing stabilizability constraints during the learning process can have a dramatic regularization effect on the learned dynamics in a manner that is tailored to the downstream task of trajectory generation and feedback control. This effect is most apparent when learning from small supervised training datasets, where we showed, both in simulation and on hardware, the quadrotor losing control and crashing when using a model learned with traditional ridge-regression. In contrast, the quadrotor is able to maintain bounded tracking performance when using the stabilizability regularized model in the low sample regime, and is on par with the ridge regularized model in the large sample regime.

9.1 Challenges and Extensions

There are several exciting future directions that we would like to pursue, categorized below.

Multi-Step Error: An immediate theoretically sound extension of this work to incorporate *multi-step* regression error would be to integrate the nonlinear features and the control signal in time by leveraging the linearity in parameters of the estimated model. This would result in an inner-product between time-varying features and the parameters, thereby retaining linearity in parameters, with the supervisory signal now being state samples along the trajectory, as opposed to the time-derivative of the state. A similar transformation is central to composite adaptive control for robotic systems (Slotine and Li, 1989; Wensing and Slotine, 2018). While such a transformation would require the full control time signal for each sampled trajectory (as opposed to state-control samples), it eliminates the need for numerical estimation of the state time-derivative along the trajectory, thereby improving the signal-to-noise ratio.

Leveraging Model Priors: The problem studied in this work concerned learning the full dynamics model from scratch. However, one can often leverage physics-based modeling to form a baseline, reducing the role of learning to estimate the residual dynamics. Instead of adopting an additive error formulation as is common in the literature, a variation of contraction theory – partial contraction (Wang and Slotine, 2005) – may be utilized to *smoothly interpolate* between the prior model and the model learned from data using an auxiliary dynamical system, termed the "virtual dynamics." This system has the special property in that the prior model and the learned model are both particular solutions to the virtual dynamics. Using similar stabilizability constraints, one can construct a model and feedback controller that forces the solution of the learned dynamics model (representing the true dynamics) to converge exponentially to the solution of the prior model, which can be designed to satisfy desirable performance criteria.

Notion of Stabilizability: The form of control-theoretic regularization employed in this work was founded upon conditions for exponential stabilizability of smooth systems via state feedback. It is currently unknown what the algorithm would produce for systems that are not inherently stabilizable or at least not globally exponentially stabilizable. As the stabilizability condition is approximated with sampling in our approach, thereby playing the role of a regularizer, we anticipate that the most probable cause of algorithm termination would be a stall, with the resulting model likely satisfying the stabilizability conditions in regions of the state-space where the true system is also stabilizable. In all other regions, the model would simply accrue non-zero slack penalty. The implications for resulting task performance is an area that merits further study.

More broadly, it is of interest to investigate not only other, more general forms of stabilizability (e.g., boundedness), but also extend the analysis to hybrid mechanical systems. This would enable tackling more challenging problems within the manipulation and locomotion domains where prior models can be severely inaccurate due to the complexity of modeling contact physics.

Computational and Experimental: By far the largest hurdle in being able to extend our algorithm to higher-dimensional systems (e.g., visual and contact domains) lies in addressing the LMI constraints. There are three key approaches that may be used to address this challenge. The first involves leveraging distributed optimization techniques such as ADMM (Boyd et al., 2011) to distribute and parallelize the constraints (which are encoded within the cost in sum-separable form). A second approach is to collapse the pointwise LMI constraints into a single LMI that is the weighted average of all the $\mathcal{F}_\lambda$ matrices, where the weighting scales with the maximum eigenvalue of each of the $\mathcal{F}_\lambda$ matrices. This is a non-convex constraint that can be solved using alternation by fixing the weights from the previous iteration. Indeed, this is equivalent to re-writing all the LMI constraints in Lagrangian form and performing primal-dual descent on the complementarity condition. While the above methods can allow scaling in terms of *number of constraint points*, in order to scale with respect to problem dimensionality, one must resort to first-order unconstrained gradient descent methods, where non-degeneracy in the contraction metric can be achieved by parameterizing the dual metric W in Cholesky form LL^T. This opens the possibility of using more expressive function approximators such as deep-neural-networks, albeit, at the expense of a fully non-convex formulation.

We believe that the dynamics learning framework presented in this work provides compelling motivation from both a stability and data efficiency standpoint, for incorporating control-theoretic notions within learning as a means of context-driven hypothesis pruning.

References

Amos B, Rodriguez IDJ, Sacks J, Boots B and Kolter JZ (2018) Differentiable MPC for end-to-end planning and control. In: *Conf. on Neural Information Processing Systems*.

Avron H, Sindhwani V, Yang J and Mahoney MW (2016) Quasi-Monte Carlo feature maps for shift-invariant kernels. *Journal of Machine Learning Research* 17(120): 1–38.

Bansal S, Akametalu AK, Jiang FJ, Laine F and Tomlin CJ (2016) Learning quadrotor dynamics using neural network for flight control. In: *Proc. IEEE Conf. on Decision and Control*.

Bansal S, Calandra R, Xiao T, Levine S and Tomlin CJ (2017) Goal-driven dynamics learning via Bayesian optimization. In: *Proc. IEEE Conf. on Decision and Control*.

Berkenkamp F, Turchetta M, Schoellig A and Krause A (2017) Safe model-based reinforcement learning with stability guarantees. In: *Conf. on Neural Information Processing Systems*.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J et al. (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3(1): 1–122.

Brault R, Heinonen M and d'Alché Buc F (2016) Random Fourier features for operator-valued kernels. In: *Asian Conf. on Machine Learning*. pp. 110–125.

Chebotar Y, Hausman K, Zhang M, Sukhatme G, Schaal S and Levine S (2017) Combining model-based and model-free updates for trajectory-centric reinforcement learning. In: *Int. Conf. on Machine Learning*.

Chua K, Calandra R, McAllister R and Levine S (2018) Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114* .

Crouch PE and van der Schaft AJ (1987) *Variational and Hamiltonian Control Systems*. Springer.

Cui R, Yang C, Li Y and Sharma S (2017) Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. *IEEE Transactions on Systems, Man, & Cybernetics: Systems* 47(6): 1019–1029.

D'Aspremont A and Karoui NE (2014) A stochastic smoothing algorithm for semidefinite programming. *SIAM Journal on Optimization* 24(3): 1138–1177.

Dean S, Tu S, Matni N and Recht B (2019) Safely learning to control the constrained linear quadratic regulator. In: *American Control Conference*. In press.

Deisenroth MP and Rasmussen CE (2011) PILCO: A model-based and data-efficient approach to policy search. In: *Int. Conf. on Machine Learning*. pp. 465–472.

Fahroo F and Ross IM (2002) Direct trajectory optimization by a Chebyshev pseudospectral method. *AIAA Journal of Guidance, Control, and Dynamics* 25(1): 160–166.

Finn C, Levine S and Abbeel P (2016) Guided cost learning: Deep inverse optimal control via policy optimization. In: *Int. Conf. on Machine Learning*.

Fisac JF, Akametalu AK, Zeilinger MN, Kaynama S, Gillula J and Tomlin CJ (2017) A general safety framework for learning-based control in uncertain robotic systems. Available at https://arxiv.org/abs/1705.01292.

Hearst MA, Dumais ST, Osuna E, Platt J and Scholkopf B (1998) Support vector machines. *IEEE Intelligent Systems and their Applications* 13(4): 18–28.

Hettich R and Kortanek KO (1993) Semi-infinite programming: Theory, methods, and applications. *SIAM Review* 35(3): 380–429.

Huang PS, Avron H, Sainath TN, Sindhwani V and Ramabhadran B (2014) Kernel methods match deep neural networks on TIMIT. In: *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, pp. 205–209.

Kamthe S and Deisenroth MP (2018) Data-efficient reinforcement learning with probabilistic model predictive control. In: *AI & Statistics*.

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

Khansari-Zadeh SM and Khatib O (2017) Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. *Autonomous Robots* 41(1): 45–69.

Kocijan J, Murray-Smith R, Rasmussen CE and Girard A (2004) Gaussian process model based predictive control. In: *American Control Conference*.

Kurdila AJ and Zabarankin M (2006) *Convex functional analysis*. Birkhäuser Basel edition. Springer Science & Business Media.

Lax P (2007) *Linear Algebra and its Applications*. 2 edition. John Wiley & Sons.

Lemme A, Neumann K, Reinhart RF and Steil JJ (2014) Neural learning of vector fields for encoding stable dynamical systems. *Neurocomputing* 141(1): 3–14.

Levine S, Finn C, Darrell T and Abbeel P (2016) End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(1): 1–40.

Liang T and Rakhlin A (2018) Just interpolate: Kernel ridgeless regression can generalize. *arXiv preprint arXiv:1808.00387v1* .

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

Manchester I, Tang JZ and Slotine JJE (2015) Unifying classical and optimization-based methods for robot tracking control with control contraction metrics. In: *Int. Symp. on Robotics Research.*

Manchester IR and Slotine JJE (2017) Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control* In Press.

Manchester IR and Slotine JJE (2018) Robust control contraction metrics: A convex approach to nonlinear state-feedback H_∞ control. *IEEE Control Systems Letters* 2(2): 333–338.

Medina JR and Billard A (2017) Learning stable task sequences from demonstration with linear parameter varying systems and hidden Markov models. In: *Conf. on Robot Learning.* pp. 175–184.

Micheli M and Glaunés JA (2014) Matrix-valued kernels for shape deformation analysis. *Geometry, Imaging and Computing* 1(1): 57–139.

Minh HQ (2016) Operator-valued Bochner theorem, Fourier feature maps for operator-valued kernels, and vector-valued learning. *arXiv preprint arXiv:1608.05639* .

Mohajerin N, Mozifian M and Waslander S (2019) Deep learning a quadrotor dynamic model for multi-step prediction. In: *Proc. IEEE Conf. on Robotics and Automation.*

Nagabandi A, Kahn G, Fearing RS and Levine S (2017) Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596* .

Ohnishi M, Wang L, Notomista G and Egerstedt M (2019) Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Transactions on Robotics* Early access.

Olfati-Saber R (2001) *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles.* PhD Thesis, Massachusetts Inst. of Technology.

Ostafew C, Schoellig AP and Barfoot TD (2016) Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. *Int. Journal of Robotics Research* 35(13): 1547–1563.

Overton ML and Womersley RS (1995) Second derivatives for optimizing eigenvalues of symmetric matrices. *SIAM Journal on Matrix Analysis and Applications* 16(3): 697–718.

Polydoros AS and Nalpantidis L (2017) Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems* 86(2): 153–173.

Punjani A and Abbeel P (2015) Deep learning helicopter dynamics models. In: *Proc. IEEE Conf. on Robotics and Automation.*

Rahimi A and Recht B (2007) Random features for large-scale kernel machines. In: *Conf. on Neural Information Processing Systems*. pp. 1177–1184.

Rahimi A and Recht B (2008) Uniform approximation of functions with random bases. In: *Allerton Conf. on Communications, Control and Computing*. IEEE.

Ravichandar H, Salehi I and Dani A (2017) Learning partially contracting dynamical systems from demonstrations. In: *Conf. on Robot Learning.*

Reyhanoglu M, van der Schaft A, McClamroch NH and Kolmanovsky I (1999) Dynamics and control of a class of underactuated mechanical systems. *IEEE Transactions on Automatic Control* 44(9): 1663–1671.

Sanner RM and Slotine JJE (1992) Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks* 3(6): 837–863.

Scholköpf B and Smola AJ (2001) *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT Press.

Schwartz L (1964) Sous-espaces hilbertiens d'espaces vectoriels topologiques et noyaux associés (noyaux reproduisants). *Analyse Mathématique* 13(1): 115–256.

Shi G, Shi X, O'Connell M, Yu R, Azizzadenesheli K, Anandkumar A, Yue Y and Chung S (2019) Neural lander: Stable drone landing control using learned dynamics. In: *Proc. IEEE Conf. on Robotics and Automation.*

Sindhwani V, Tu S and Khansari M (2018) Learning contracting vector fields for stable imitation learning. *arXiv preprint arXiv:1804.04878* .

Singh S, Majumdar A, Slotine JJE and Pavone M (2017) Robust online motion planning via contraction theory and convex optimization. In: *Proc. IEEE Conf. on Robotics and Automation*. Extended Version, Available at http://asl.stanford.edu/wp-content/papercite-data/pdf/Singh.Majumdar.Slotine.Pavone.ICRA17.pdf.

Singh S, Sindhwani V, Slotine JJE and Pavone M (2018) Learning stabilizable dynamical systems via control contraction metrics. In: *Workshop on Algorithmic Foundations of Robotics.*

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

Slotine JJE and Li W (1989) Composite adaptive control of robot manipulators. *Automatica* 25(4): 509–519.

Spong MW (1998) Underactuated mechanical systems. In: *Control Problems in Robotics and Automation*. Springer Berlin Heidelberg.

Taylor AJ, Dorobantu VD, Le HM, Yue Y and Ames AD (2019) Episodic learning with control Lyapunov functions for uncertain robotic systems. Available at https://arxiv.org/abs/1903.01577.

Thuruthel TG, Falotico E, Renda F and Laschi C (2019) Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. *IEEE Transactions on Robotics* 35(1): 124–134.

Venkatraman A, Capobianco R, Pinto L, Hebert M, Nardi D and Bagnell A (2016) Improved learning of dynamics models for control. In: *Int. Symp. on Experimental Robotics*. Springer, pp. 703–713.

Venkatraman A, Hebert M and Bagnell JA (2015) Improving multi-step prediction of learned time series models. In: *Proc. AAAI Conf. on Artificial Intelligence*.

Wang W and Slotine JJE (2005) On partial contraction analysis for coupled nonlinear oscillators. *Biological Cybernetics* 92(1): 38–53.

Wang YS, Matni N and Doyle JC (2019) A system level approach to controller synthesis. *IEEE Transactions on Automatic Control* Early Access.

Wensing PW and Slotine JJ (2018) Cooperative adaptive control for cloud-based robotics. In: *Proc. IEEE Conf. on Robotics and Automation*.

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

Zhou DX (2008) Derivative reproducing properties for kernel methods in learning theory. *Journal of Computational and Applied Mathematics* 220(1-2): 456–463.

Zhou S, Helwa MK and Schoellig AP (2017) Design of deep neural networks as add-on blocks for improving impromptu trajectory tracking. In: *Proc. IEEE Conf. on Decision and Control*.

Appendix

A Solution Parameters for PVTOL

Notice that the true input matrix for the PVTOL system satisfies Assumption 1. Furthermore, it is a constant matrix. Thus, the feature mapping Φ_b is therefore just a constant matrix with the necessary sparsity structure.

The feature matrix for f was generated using the random matrix feature approximation to the Gaussian separable matrix-valued kernel with $\sigma = 6$ and $s = 8n = 48$ sampled Gaussian directions, yielding a feature mapping matrix Φ_f with $d_f = 576$ (96 features for each component of f). The scalar-valued reproducing kernels for the entries of W were taken to be the Gaussian kernel with $\sigma = 15$. To satisfy condition (12), the kernel for $w_{ij},\ (i,j) \in \{1,\ldots,(n-m)\}$ was only a function of the first $n-m$ components of x. A total of $s = 36$ Gaussian samples were taken to yield feature vectors ϕ_w and $\hat{\phi}_w$ of dimension $d_w = 72$. Furthermore, by symmetry of $W(x)$ only $n(n+1)/2$ functions were actually parameterized. Thus, the learning problem in (22) comprised of $d_f + d_w n(n+1)/2 + 4 = 2092$ parameters for the functions, plus the extra scalar constants $\underline{w}, \overline{w}$.

The learning parameters used were: model N-R (all N): $\mu_f = 0, \mu_b = 10^{-6}$, R-R (all N): $\mu_f = 10^{-4}, \mu_b = 10^{-6}$, CCM-R (all N): $\mu_f = 10^{-3}, \mu_b = 10^{-6}$; $N \in \{100, 250, 500\} : \mu_w = 10^{-3}$, $N = 1000 : \mu_w = 10^{-4}$. Note that a small penalization on μ_b was necessary for all models due to the fact that feature matrix Φ_b is rank deficient. Contraction rate: $\lambda = 0.1$. Tolerance parameters: constraints: $\{\epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}\} = \{0.1, 0.1, 0.1\}$; discard tolerance $\delta = 0.05$. Note that the tolerance parameters have little effect on the resulting model since the stability conditions are *homogenous in* $W(x)$.

B CCM Controller Synthesis

Let $\Gamma(p,q)$ be the set of smooth curves $c : [0,1] \to \mathcal{X}$ satisfying $c(0) = p, c(1) = q$, and define $\delta_c(s) := \partial c(s)/\partial s$. At each time t, given the nominal state/control pair (x^*, u^*) and current actual state x:

1. Compute a curve $\gamma \in \Gamma(x^*, x)$ defined by:

$$\gamma \in \underset{c \in \Gamma(x^*,x)}{\text{argmin}} \int_0^1 \delta_c(s)^T M(c(s)) \delta_c(s) ds, \tag{40}$$

   and let $\mathcal{E}$ denote the minimal value.

2. Define:

$$\begin{aligned}\mathcal{E}_d(k) :=& 2\delta_\gamma(1)^T M(x)(f(x) + B(x)(u^* + k)) \\ &- 2\delta_\gamma(0)^T M(x^*)(f(x^*) + B(x^*)u^*).\end{aligned} \tag{41}$$

3. Choose $k(x^*, x)$ to be any element of the set:

$$\mathcal{K} := \{k : \mathcal{E}_d(k) \leq -2\lambda\mathcal{E}\}. \tag{42}$$

By existence of a CCM $M(x)$ (equivalently, its dual $W(x)$), the set $\mathcal{K}$ is always non-empty. The resulting $k(x^*, x)$ then ensures that the solution $x(t)$ indeed converges towards $x^*(t)$ exponentially (Singh et al., 2017).

From an implementation perspective, note that having obtained the curve γ, constraint (42) is simply a linear inequality in k. Thus, one can analytically compute a feasible feedback control value, e.g., by searching for the smallest (in any norm) element of $\mathcal{K}$; for additional details, we refer the reader to (Manchester and Slotine, 2017; Singh et al., 2017). This controller was not used for the CCM-R experiments in this paper however since the contraction conditions are only enforced at the discrete sampled constraint set, as opposed to over the continuous state space $\mathcal{X}$.