EasyChair Preprint

№ 750

Learning Stabilizable Dynamical Systems via Control Contraction Metrics

Sumeet Singh, Vikas Sindhwani, Jean-Jacques Slotine and Marco Pavone

January 22, 2019

Learning Stabilizable Dynamical Systems via Control Contraction Metrics

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

[1] Dept. of Aeronautics and Astronautics, Stanford University
{ssingh19,pavone}@stanford.edu
[2] Google Brain Robotics, New York
sindhwani@google.com
[3] Dept. of Mechanical Engineering, Massachusetts Institute of Technology
jjs@mit.edu

Abstract. We propose a novel framework for learning stabilizable nonlinear dynamical systems for continuous control tasks in robotics. The key idea is to develop a new control-theoretic regularizer for dynamics fitting rooted in the notion of *stabilizability*, which guarantees that the learned system can be accompanied by a robust controller capable of stabilizing *any* open-loop trajectory that the system may generate. By leveraging tools from contraction theory, statistical learning, and convex optimization, we provide a general and tractable semi-supervised algorithm to learn stabilizable dynamics, which can be applied to complex underactuated systems. We validated the proposed algorithm on a simulated planar quadrotor system and observed notably improved trajectory generation and tracking performance with the control-theoretic regularized model over models learned using traditional regression techniques, especially when using a small number of demonstration examples. The results presented illustrate the need to infuse standard model-based reinforcement learning algorithms with concepts drawn from nonlinear control theory for improved reliability.

Keywords: Model-based reinforcement learning, contraction theory, robotics.

1 Introduction

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

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

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 the 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 optimally act 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 Reinforcement Learning typically offers major improvements in sample efficiency in comparison to state of the art RL methods such as Policy Gradients [3,22] 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

* This work was supported by NASA under the Space Technology Research Grants Program, Grant NNX12AQ43G, and by the King Abdulaziz City for Science and Technology (KACST).

are applied to model complex dynamics, the resulting controllers do not perform at 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 that the learned system can be accompanied by a robust controller capable of stabilizing any trajectory that the system may generate.

Formally, a reference state-input trajectory pair $(x^*(t), u^*(t)),\ t \in [0, T]$ 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.

Problem Statement: 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. Our objective is to solve the 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 expect that for systems that are indeed stabilizable, enforcing such a constraint may drastically *prune the hypothesis space, thereby playing 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.

Related Work: The simplest approach to learning dynamics is to ignore stabilizability and treat the problem as a standard one-step time series regression task [3, 5, 22]. 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 [34, 35]. In recent work on uncertainty-aware model-based RL, policies [3, 22] are optimized with respect to stochastic rollouts from probabilistic dynamics models that are iteratively improved in a model predictive control loop. Despite being effective, these methods are still heuristic in the sense that the existence of a stabilizing feedback controller is not explicitly guaranteed.

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 [9, 10, 13, 19, 26, 29]. 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. In that sense, it is in the spirit of recent model-based RL techniques that exploit control theoretic notions of stability to guarantee model safety during the learning process [1]. However, unlike the work in [1] which aims to maintain a local region of attraction near a known safe operating point, we consider a stronger notion of safety – that of stabilizability, that is, the ability to keep the system within a bounded region of any exploratory open-loop trajectory.

Potentially, the tools we develop may also be used to extend standard adaptive robot control design, such as [32] – 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 [28]. 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.

Statement of Contributions: Stabilizability of trajectories is a complex task in non-linear control. In this work, we leverage recent advances in contraction theory for control design through the use of *control contraction metrics* (CCM) [16] that turns stabilizability constraints into convex Linear Matrix Inequalities (LMIs). Contraction theory [15] is a method of analyzing nonlinear systems in a differential framework, i.e., via the associated variational system [4, 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 [16, 18, 30] that illustrate how to use contraction theory to obtain a *certificate* for trajectory stabilizability and an accompanying tracking controller with exponential stability properties. In Section 2, we provide a brief summary of these results, which in turn will form the foundation of this work.

Our paper makes four primary contributions. First, we formulate the learning stabilizable dynamics problem through the lens of control contraction metrics (Section 3). Second, under an arguably weak assumption on the sparsity of the true dynamics model, we present a finite-dimensional optimization-based solution to this problem by leveraging the powerful framework of vector-valued Reproducing Kernel Hilbert Spaces (Section 4.2). We further motivate this solution from a standpoint of viewing the stabilizability constraint as a novel control-theoretic *regularizer* for dynamics learning. Third, we develop a tractable algorithm leveraging alternating convex optimization problems and adaptive sampling to iteratively solve the finite-dimensional optimization problem (Section 5). Finally, we verify the proposed approach on a 6-state, 2-input planar quadrotor model where we demonstrate that naive 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 for smaller training sets (Section 6).

2 Review of Contraction Theory

The core principle behind contraction theory [15] is to study the evolution of distance between any two *arbitrarily close* neighboring trajectories and drawing conclusions on

the distance between *any* 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.,

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

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 is thereby referred to be *contracting at rate* λ.

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 [15] 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 $\lambda > 0$. Thus, the function $V(x, \delta_x)$ may be interpreted as a *differential Lyapunov function*.

2.1 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 dynamics of the form:

$$\dot{x}(t) = f(x(t)) + B(x(t))u(t), \tag{5}$$

where $B : \mathcal{X} \to \mathbb{R}^{n\times m}$ is the input matrix, and denote B in column form as $(b_1, \ldots, b_m)$ and u in component form as $(u^1, \ldots, u^m)$. 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:

$$\dot{\delta}_x = \overbrace{\left(\frac{\partial f(x)}{\partial x} + \sum_{j=1}^{m} u^j \frac{\partial b_j(x)}{\partial x}\right)}^{:=A(x,u)}\delta_x + B(x)\delta_u, \tag{6}$$

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 CCM for the system $\{f, B\}$ is a uniformly positive-definite symmetric matrix function $M(x)$ such that there exists a function $\delta_u(x, \delta_x, u)$ so that the function $V(x, \delta_x) = \delta_x^T M(x)\delta_x$ satisfies

$$\begin{aligned}\dot{V}(x, \delta_x, u) &= \delta_x^T \left(\partial_{f+Bu} M(x) + \widehat{M(x)A(x,u)}\right) \delta_x + 2\delta_x^T M(x)B(x)\delta_u \\ &\leq -2\lambda V(x, \delta_x), \quad \forall (x, \delta_x) \in \mathcal{TX},\ u \in \mathbb{R}^m,\end{aligned} \tag{7}$$

where $\partial_g M(x)$ is the matrix with element (i, j) given by Lie derivative of $M_{ij}(x)$ along the vector g. Given the existence of a valid CCM, one then constructs a stabilizing (in the sense of eq. (2)) feedback controller $k(x^*, x)$ as described in the online version of this work [31].

Some important observations are in order. First, the function $V(x, \delta_x)$ may be interpreted as a differential *control* Lyapunov function, in that, there exists a stabilizing differential controller δ_u that stabilizes the variational dynamics (6) in the sense of eq. (7). Second, and more importantly, we see that 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. Crucially, this is an exact stabilization result, not one based on local linearization-based control. Consequently, one can show several useful properties, such as invariance to state-space transformations [16] and robustness [17,30]. Third, the CCM approach only requires a weak form of controllability, and therefore is not restricted to feedback linearizable (i.e., invertible) systems.

3 Problem Formulation

Using the characterization of stabilizability using CCMs, we can now formalize our problem as follows. Given our dataset of tuples $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$, the objective of this work is to learn the dynamics functions $f(x)$ and $B(x)$ in eq. (5), subject to the constraint that there exists a valid 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 [16], a necessary and sufficient characterization of a CCM $M(x)$ is given in terms of its dual $W(x) := M(x)^{-1}$ by the following two conditions:

$$B_\perp^T \left(\partial_{b_j} W(x) - \widehat{\frac{\partial b_j(x)}{\partial x} W(x)}\right) B_\perp = 0,\ j = 1, \ldots, m \quad \forall x \in \mathcal{X}, \tag{8}$$

$$\underbrace{B_\perp(x)^T \left(-\partial_f W(x) + \widehat{\frac{\partial f(x)}{\partial x} W(x)} + 2\lambda W(x)\right) B_\perp(x)}_{:= F(x; f, W, \lambda)} \prec 0, \quad \forall x \in \mathcal{X}, \tag{9}$$

where $B_\perp$ is the annihilator matrix for B, i.e., $B(x)^T B_\perp(x) = 0$ for all x. In the definition above, we write $F(x; W, f, \lambda)$ since $\{W, f, \lambda\}$ will be optimization variables in our formulation. Thus, our learning task reduces to finding the functions $\{f, B, W\}$ and constant λ that jointly satisfy the above constraints, while minimizing an appropriate regularized regression loss function. Formally, problem (3) can be re-stated as:

$$\min_{\substack{\hat{f}\in\mathcal{H}^f,\hat{b}_j\in\mathcal{H}^B,j=1,\dots,m\\ W\in\mathcal{H}^W\\ \underline{w},\overline{w},\lambda\in\mathbb{R}_{>0}}} \quad \overbrace{\sum_{i=1}^{N}\left\|\hat{f}(x_i)+\hat{B}(x_i)u_i-\dot{x}_i\right\|_2^2+\mu_f\|\hat{f}\|_{\mathcal{H}^f}^2+\mu_b\sum_{j=1}^{m}\|\hat{b}_j\|_{\mathcal{H}^B}^2+}^{:=J_d(\hat{f},\hat{B})}$$

$$+\underbrace{(\overline{w}-\underline{w})+\mu_w\|W\|_{\mathcal{H}^W}^2}_{:=J_m(W,\underline{w},\overline{w})} \tag{10a}$$

$$\text{subject to} \quad \text{eqs. (8), (9)} \quad \forall x\in\mathcal{X}, \tag{10b}$$

$$\underline{w}I_n \preceq W(x) \preceq \overline{w}I_n, \quad \forall x\in\mathcal{X}, \tag{10c}$$

where $\mathcal{H}^f$ and $\mathcal{H}^B$ are appropriately chosen $\mathbb{R}^n$-valued function classes on $\mathcal{X}$ for $\hat{f}$ and $\hat{b}_j$ 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 [30]).

Notice that the coupling constraint (9) is a bi-linear matrix inequality in the decision variables sets $\{\hat{f},\lambda\}$ and W. Thus at a high-level, a solution algorithm must consist of alternating between two convex sub-problems, defined by the objective/decision variable pairs $(J_d,\{\hat{f},\hat{B},\lambda\})$ and $(J_m,\{W,\underline{w},\overline{w}\})$.

4 Solution Formulation

When performing dynamics learning on a system that is a priori *known* to be exponentially stabilizable at some strictly positive rate λ, the constrained problem formulation in (10) follows naturally given the assured *existence* of a CCM. Albeit, the infinite-dimensional nature of the constraints is a considerable technical challenge, broadly falling under the class of *semi-infinite* optimization [7]. Alternatively, for systems that are not globally exponentially stabilizable in $\mathcal{X}$, one can imagine that such a constrained formulation may lead to adverse effects on the learned dynamics model.

Thus, in this section we propose a relaxation of problem (10) motivated by the concept of regularization. Specifically, constraints (8) and (9) 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 eq (7) 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 (8) sets to zero, the terms in (7) affine in u, while constraint (9) enforces this "natural" stability condition.

The simplifications we make are (i) relax constraints (9) and (10c) to hold pointwise over some *finite* constraint set $X_c\in\mathcal{X}$, and (ii) assume a specific sparsity structure for 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 and [31].

First, from a purely mathematical standpoint, the pointwise relaxation of (9) and (10c) 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 [12].

Second, 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 (10), 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 discussed in detail in our experiments in Section 6.

Practically, the finite constraint set X_c, with cardinality N_c, includes all x_i in the regression training set of $\{(x_i, u_i, \dot{x}_i)\}_{i=1}^N$ tuples. However, as the LMI constraints are *independent* of $u_i, \dot{x}_i$, 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.

4.1 Sparsity of Input Matrix Estimate $\hat{B}$

While a pointwise relaxation for the matrix inequalities is reasonable, one cannot apply such a relaxation to the exact equality condition in (8). 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:*

$$\hat{B}(x) = \begin{bmatrix} O_{(n-m)\times m} \\ \mathfrak{b}(x) \end{bmatrix}, \tag{11}$$

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

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

$$B_\perp = \begin{bmatrix} I_{n-m} \\ O_{m\times(n-m)} \end{bmatrix}. \tag{12}$$

Therefore, constraint (8) simply becomes:

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

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:

$$\left[\frac{\partial w_{\perp_{pq}}(x)}{\partial x^{(n-m)+1}} \cdots \frac{\partial w_{\perp_{pq}}(x)}{\partial x^n} \right] \mathfrak{b}(x) = 0.$$

Since the matrix $\mathfrak{b}(x)$ in (11) is assumed to be invertible, the *only* solution to this equation is $\partial w_{\perp_{pq}}/\partial x^i = 0$ for $i = (n-m)+1, \ldots, n$, 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 (8). Due to space limitations, justification for this sparsity assumption is provided in [31].

4.2 Finite-dimensional Optimization

We now present a tractable finite-dimensional optimization for solving problem (10) under the two simplifying assumptions introduced in the previous sections. The derivation of the solution algorithm itself is presented in [31], and relies extensively on vector-valued Reproducing Kernel Hilbert Spaces.

Step 1: Parametrize the functions $\hat{f}$, the columns of $\hat{B}(x)$: $\{\hat{b}_j\}_{j=1}^m$, and $\{w_{ij}\}_{i,j=1}^n$ as a linear combination of features. That is,

$$\hat{f}(x) = \Phi_f(x)^T \alpha, \tag{13}$$

$$\hat{b}_j(x) = \Phi_b(x)^T \beta_j \quad j \in \{1, \dots, m\}, \tag{14}$$

$$w_{ij}(x) = \begin{cases} \hat{\phi}_w(x)^T \hat{\theta}_{ij} & \text{if} \quad (i,j) \in \{1, \dots, n-m\}, \\ \phi_w(x)^T \theta_{ij} & \text{else}, \end{cases} \tag{15}$$

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 (11), the feature matrix Φ_b must have all 0s 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 4.1). While one can use any function approximator (e.g., neural nets), we motivate this parameterization from a perspective of Reproducing Kernel Hilbert Spaces (RKHS); see [31].

Step 2: Given positive regularization constants μ_f, μ_b, μ_w and positive tolerances $(\delta_\lambda, \epsilon_\lambda)$ and $(\delta_{\underline{w}}, \epsilon_{\underline{w}})$, solve:

$$\min_{\alpha, \beta_j, \hat{\theta}_{ij}, \theta_{ij}, \underline{w}, \overline{w}, \lambda} \quad \overbrace{\sum_{k=1}^{N} \|\hat{f}(x_i) + \hat{B}(x_i)u_i - \dot{x}_i\|_2^2 + \mu_f \|\alpha\|_2^2 + \mu_b \sum_{j=1}^{m} \|\beta_j\|_2^2}^{:=J_d}$$
$$\underbrace{+ (\overline{w} - \underline{w}) + \mu_w \sum_{i,j} \|\tilde{\theta}_{ij}\|_2^2}_{:=J_m} \tag{16a}$$

$$\text{s.t.} \quad F(x_i; \alpha, \tilde{\theta}_{ij}, \lambda + \epsilon_\lambda) \preceq 0, \quad \forall x_i \in X_c, \tag{16b}$$

$$(\underline{w} + \epsilon_{\underline{w}}) I_n \preceq W(x_i) \preceq \overline{w} I_n, \quad \forall x_i \in X_c, \tag{16c}$$

$$\theta_{ij} = \theta_{ji}, \hat{\theta}_{ij} = \hat{\theta}_{ji} \tag{16d}$$

$$\lambda \geq \delta_\lambda, \quad \underline{w} \geq \delta_{\underline{w}}, \tag{16e}$$

where $\tilde{\theta}_{ij}$ is used as a placeholder for θ_{ij} and $\hat{\theta}_{ij}$ to simplify notation.

We wish to highlight the following key points regarding problem (16). Constraints (16b) and (16c) are the pointwise relaxations of (9) and (10c) respectively. Constraint (16d) captures the fact that $W(x)$ is a symmetric matrix. Finally, constraint (16e) imposes some tolerance requirements to ensure a well conditioned solution. Additionally, the

tolerances ϵ_δ 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 the next section, we present an iterative algorithm and an adaptive constraint sampling technique to solve problem (16).

5 Solution Algorithm

The fundamental structure of the solution algorithm consists of alternating between the dynamics and metric sub-problems derived from problem (16). We also make a few additional modifications to aid tractability, most notable of which is the use of a *dynamically* updating set of constraint points $X_c^{(k)}$ at which the LMI constraints are enforced at the k^{th} iteration. 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 three minor steps (sub-problems):

1. Finite-dimensional dynamics sub-problem at iteration k:

$$\min_{\substack{\alpha,\beta_j,j=1,\dots,m,\ \lambda \\ s\geq 0}} \quad J_d(\alpha,\beta) + \mu_s\|s\|_1 \tag{17a}$$

$$\text{s.t.} \quad F(x_i;\alpha,\tilde{\theta}_{ij}^{(k-1)},\lambda+\epsilon_\lambda) \preceq s(x_i) I_{n-m} \quad \forall x_i \in X_c^{(k)} \tag{17b}$$

$$s(x_i) \leq \bar{s}^{(k-1)} \quad \forall x_i \in X_c^{(k)} \tag{17c}$$

$$\lambda \geq \delta_\lambda, \tag{17d}$$

   where μ_s is an additional regularization parameter for s – an $N_c^{(k)}$ dimensional non-negative slack vector. The quantity $\bar{s}^{(k-1)}$ is defined as

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

   That is, $\bar{s}^{(k-1)}$ captures the worst violation for the $F(\cdot)$ LMI over the entire constraint set X_c, given the parameters at the end of iteration $k-1$.
2. Finite-dimensional metric sub-problem at iteration k:

$$\min_{\tilde{\theta}_{ij},\underline{w},\overline{w},s\geq 0} \quad J_m(\tilde{\theta}_{ij},\underline{w},\overline{w}) + (1/\mu_s)\|s\|_1 \tag{18a}$$

$$\text{s.t.} \quad F(x_i;\alpha^{(k)},\tilde{\theta}_{ij},\lambda^{(k)}+\epsilon_\lambda) \preceq s(x_i) I_{n-m} \quad \forall x_i \in X_c^{(k)} \tag{18b}$$

$$s(x_i) \leq \bar{s}^{(k-1)} \quad \forall x_i \in X_c^{(k)} \tag{18c}$$

$$(\underline{w}+\epsilon_{\underline{w}}) I_n \preceq W(x_i) \preceq \overline{w} I_n, \quad \forall x_i \in X_c^{(k)}, \tag{18d}$$

$$\underline{w} \geq \delta_{\underline{w}}. \tag{18e}$$

3. Update $X_c^{(k)}$ sub-problem. Choose a tolerance parameter $\delta > 0$. Then, define

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

and set

$$X_c^{(k+1)} := \left\{x_i \in X_c^{(k)} : \nu^{(k)}(x_i) > -\delta\right\} \bigcup \left\{x_i \in X_c \setminus X_c^{(k)} : \nu^{(k)}(x_i) > 0\right\}. \tag{19}$$

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 say K constraint violating points from $X_c \setminus X_c^{(k)}$ (e.g., corresponding to the K worst violators), where K 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 [36] where one is trying to enforce the constraints at *all* points in a compact set $\mathcal{X}$.

Note that after the first major iteration, we replace the regularization terms in J_d and J_m with $\|\alpha^{(k)} - \alpha^{(k-1)}\|_2^2$, $\|\beta_j^{(k)} - \beta_j^{(k-1)}\|_2^2$, and $\|\tilde{\theta}_{ij}^{(k)} - \tilde{\theta}_{ij}^{(k-1)}\|_2^2$. This is done to prevent large updates to the parameters, particularly due to the dynamically updating constraint set $X_c^{(k)}$. The full pseudocode is summarized below in Algorithm 1.

Algorithm 1 Stabilizable Non-Linear Dynamics Learning (SNDL)

Input: Dataset $\{x_i, u_i, \dot{x}_i\}_{i=1}^N$, constraint set X_c, regularization constants $\{\mu_f, \mu_b, \mu_w\}$, constraint tolerances $\{\delta_\lambda, \epsilon_\lambda, \delta_{\underline{w}}, \epsilon_{\underline{w}}\}$, discard tolerance parameter δ, Initial # of constraint points: $N_c^{(0)}$, Max # iterations: $N_{\max}$, termination tolerance ε.
$k \leftarrow 0$, `converged` $\leftarrow$ **false**, $W(x) \leftarrow I_n$.
$X_c^{(0)} \leftarrow$ RANDSAMPLE$(X_c, N_c^{(0)})$
while $\neg$`converged` $\wedge\, k < N_{\max}$ **do**
  $\{\alpha^{(k)}, \beta_j^{(k)}, \lambda^{(k)}\} \leftarrow$ SOLVE (17)
  $\{\tilde{\theta}_{ij}^{(k)}, \underline{w}, \overline{w}\} \leftarrow$ SOLVE (18)
  $X_c^{(k+1)}, \bar{s}^{(k)}, \nu^{(k)} \leftarrow$ UPDATE $X_c^{(k)}$ using (19)
  $\Delta \leftarrow \max\left\{\|\alpha^{(k)} - \alpha^{(k-1)}\|_\infty, \|\beta_j^{(k)} - \beta_j^{(k-1)}\|_\infty, \|\tilde{\theta}_{ij}^{(k)} - \tilde{\theta}_{ij}^{(k-1)}\|_\infty, \|\lambda^{(k)} - \lambda^{(k-1)}\|_\infty\right\}$
  if $\Delta < \varepsilon$ **or** $\nu^{(k)}(x_i) < \varepsilon \quad \forall x_i \in X_c$ **then**
    `converged` $\leftarrow$ **true**.
  end if
  $k \leftarrow k + 1$.
end while

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. Due to the semi-supervised nature of the algorithm in that the number of constraint points N_c can be significantly larger than the number of supervisory regression tuples

N, it is impractical to enforce constraints at all N_c points in any one iteration. Two key consequences of this are: (i) the matrix function $W(x)$ at iteration k resulting from variables $\tilde{\theta}^{(k)}$ does *not* have to correspond to a valid dual CCM for the interim learned dynamics at iteration k, and (ii) convergence 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. Potential future topics in this regard are: (i) investigate the properties of the converged dynamics for models that are a priori unknown unstabilizable, and (ii) derive sufficient conditions for convergence for both the infinitely- and finitely- constrained versions of problem (10).

Second, as a consequence of this iterative procedure, the dual metric and contraction rate pair $\{W(x), \lambda\}$ do not possess any sort of "control-theoretic" optimality. For instance, in [30], 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. A potential future topic to explore in this regard is how to further optimize $\{W(x), \lambda\}$ for control *performance*.

6 Experimental Results

In this section we validate our algorithms by benchmarking our results on a known dynamics model. Specifically, we consider the 6-state planar vertical-takeoff-vertical-landing (PVTOL) model. 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, $(\phi, \dot{\phi})$ are the roll and angular rate respectively, and 2-dimensional control input u corresponding to the motor thrusts. The true dynamics are given by:

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

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. We note that typical benchmarks in this area of work either present results on the 2D LASA handwriting dataset [9] or other low-dimensional motion primitive spaces, with the assumption of full robot dynamics invertibility. The planar quadrotor on the other hand 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.

6.1 Generation of Datasets

The training dataset was generated in 3 steps. First, a fixed set of waypoint paths in (p_x, p_z) were 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 PD controller; thereby emulating a noisy/imperfect demonstrator. These final simulated paths were sub-sampled

at 0.1s 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.

Due to space constraints, we provide details of the solution parameterization (number of features, etc) in [31].

6.2 Models

Using the same feature space, we trained three separate models with varying training dataset (i.e., $(x_i, u_i, \dot{x}_s)$ tuples) sizes of $N \in \{100, 250, 500, 1000\}$. The first model, **N-R** was an unconstrained and un-regularized model, trained by solving problem (17) without constraints or l_2 regularization (i.e., just least-squares). The second model, **R-R** was an unconstrained ridge-regression model, trained by solving problem (17) without any constraints (i.e., least-squares plus l_2 regularization). The third model, **CCM-R** is the CCM-regularized model, trained using Algorithm 1. We enforced the CCM regularizing constraints for the CCM-R model at $N_c = 2400$ 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 $u, \dot{x}$).

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 (see Section 4), we simulated both the R-R and CCM-R models using the time-varying Linear-Quadratic-Regulator (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.

6.3 Validation and Comparison

The validation tests were conducted by gridding the (p_x, p_z) plane to create a set of 120 initial conditions between 4m and 12m away from $(0, 0)$ and randomly sampling the other states for the rest of the initial conditions. These conditions were *held fixed* for both 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 using the TV-LQR controller. As a baseline, 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 is not optimizing 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 [6] with the objective of minimizing $\int_0^T \|u(t)\|^2 dt$. 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 1 shows a boxplot comparison of the trajectory-wise RMS full state errors ($\|x(t) - x^*(t)\|_2$ 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 $N = 100, 500$ and 1000, indicating the need for *some* form of regularization. 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 training samples is small (i.e., $N \in \{100, 250\}$) when there is

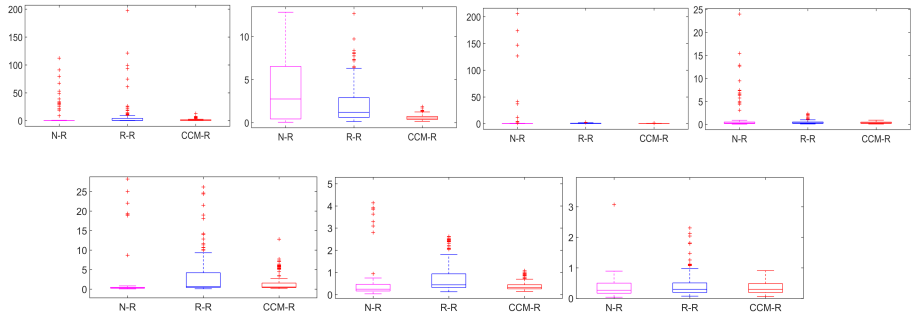


Fig. 1: Box-whisker plot comparison of trajectory-wise RMS state-tracking errors over all 120 trajectories for each model and all training dataset sizes. *Top row, left-to-right:* $N = 100, 250, 500, 1000$; *Bottom row, left-to-right:* $N = 100, 500, 1000$ (zoomed in). The box edges correspond to the 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 $N = 100, 250$.

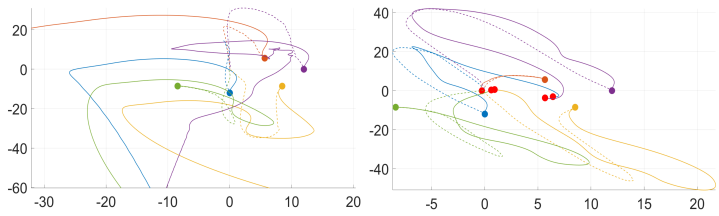
considerable risk of overfitting. It appears the CCM constraints have a notable effect on the *stabilizability* of the resulting model trajectories (recall that the initial conditions of the trajectories and the tracking controllers are held fixed between the models).

For $N = 100$ (which is really at the extreme lower limit of necessary number of samples since there are effectively 97 features for each dimension of the dynamics function), both N-R and R-R models generate a large number of unstable trajectories. In contrast, out of the 120 generated test trajectories, the CCM-R model generates *one* mildly (in that the quadrotor diverged from the nominal trajectory but did not crash) unstable trajectory. No instabilities were observed with CCM-R for $N \in \{250, 500, 1000\}$.

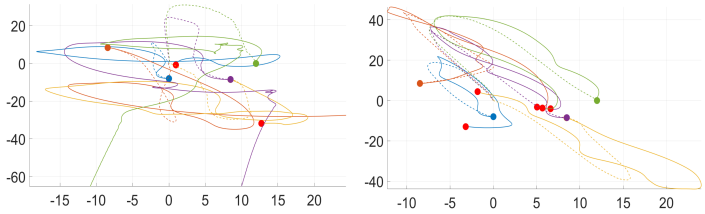
Figure 2a compares the (p_x, p_z) traces between R-R and CCM-R corresponding to the five worst performing trajectories for the R-R $N = 100$ model. Similarly, Figure 2b compares the (p_x, p_z) traces corresponding to the five worst performing trajectories for the CCM-R $N = 100$ model. Notice the large number of unstable trajectories generated using the R-R model. Indeed, it is in this low sample training regime where the control-theoretic regularization effects of the CCM-R model are most noticeable.

Finally, in Figure 3, we highlight two trajectories, starting from the *same initial conditions*, one generated and tracked using the R-R model, the other using the CCM model, for $N = 250$. Overlaid on the plot are the snapshots of the vehicle outline itself, illustrating the quite aggressive flight-regime of the trajectories (the initial starting bank angle is 40°). 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 $(0, 0)$.

An interesting area of future work here is to investigate how to tune the regularization parameters μ_f, μ_b, μ_w. Indeed, the R-R model appears to be extremely sensitive to μ_f, yielding drastically worse results with a small change in this parameter. On the other hand, the CCM-R model appears to be quite robust to variations in this parameter. Standard cross-validation techniques using regression quality as a metric are unsuitable as a tuning technique here; indeed, recent results even advocate for "ridgeless" regres-



(a)



(b)

Fig. 2: (p_x, p_z) traces for R-R (*left column*) and CCM-R (*right column*) corresponding to the 5 worst performing trajectories for (a) R-R, and (b) CCM-R models at $N = 100$. Colored circles indicate start of trajectory. Red circles indicate end of trajectory. All except one of the R-R trajectories are unstable. One trajectory for CCM-R is slightly unstable.

sion [14]. However, as observed in Figure 1, un-regularized model fitting is clearly unsuitable. The effect of regularization on how the trajectory optimizer leverages the learned dynamics is a non-trivial relationship that merits further study.

7 Conclusions

In this paper, 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 nonlinear 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 stabilizable dynamics for complex underactuated and inherently unstable systems. The algorithm was validated on a simulated planar quadrotor system where it was observed that our control-theoretic dynamics learning algorithm notably outperformed traditional ridge-regression based model learning.

There are several interesting avenues for future work. First, it is unclear how the algorithm would perform for systems that are fundamentally unstabilizable and how the resulting learned dynamics could be used for "approximate" control. Second, we will explore sufficient conditions for convergence for the iterative algorithm under the finite- and infinite-constrained formulations. Third, we will address extending the algorithm to work on higher-dimensional spaces through functional parameterization of the control-theoretic regularizing constraints. Fourth, we will address the limitations imposed by the sparsity assumption on the input matrix B using the proposed alternating algorithm proposed in Section 4.1. Finally, we will incorporate data gathered on a physical system

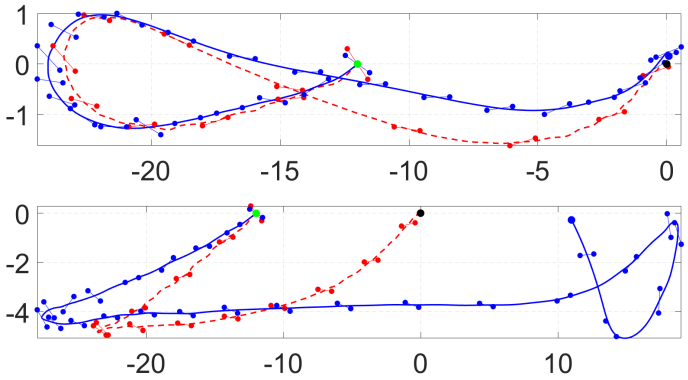


Fig. 3: Comparison of reference and tracked trajectories in the (p_x, p_z) plane for R-R and CCM-R models starting at same initial conditions with $N = 250$. Red (dashed): nominal, Blue (solid): actual, Green dot: start, black dot: nominal endpoint, blue dot: actual endpoint; *Top:* CCM-R, *Bottom:* 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.

subject to noise and other difficult to capture nonlinear effects (e.g., drag, friction, backlash) and validate the resulting dynamics model and tracking controllers on the system itself to evaluate the robustness of the learned models.

References

1. Berkenkamp, F., Turchetta, M., Schoellig, A., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: Conf. on Neural Information Processing Systems (2017)
2. Brault, R., Heinonen, M., d'Alché Buc, F.: Random fourier features for operator-valued kernels. In: Asian Conf. on Machine Learning. pp. 110–125 (2016)
3. Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. arXiv preprint arXiv:1805.12114 (2018)
4. Crouch, P.E., van der Schaft, A.J.: Variational and Hamiltonian Control Systems. Springer (1987)
5. Deisenroth, M.P., Rasmussen, C.E.: PILCO: A model-based and data-efficient approach to policy search. In: Int. Conf. on Machine Learning. pp. 465–472 (2011)
6. Fahroo, F., Ross, I.M.: Direct trajectory optimization by a chebyshev pseudospectral method. AIAA Journal of Guidance, Control, and Dynamics 25(1), 160–166 (2002)
7. Hettich, R., Kortanek, K.O.: Semi-infinite programming: Theory, methods, and applications. SIAM Review 35(3), 380–429 (1993)
8. Huang, P.S., Avron, H., Sainath, T.N., Sindhwani, V., Ramabhadran, B.: Kernel methods match deep neural networks on TIMIT. In: IEEE Int. Conf. on Acoustics, Speech and Signal Processing. pp. 205–209. IEEE (2014)
9. Khansari-Zadeh, S.M., Billard, A.: Learning stable nonlinear dynamical systems with Gaussian mixture models. IEEE Transactions on Robotics 27(5), 943–957 (2011)
10. Khansari-Zadeh, S.M., Khatib, O.: Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors. Autonomous Robots 41(1), 45–69 (2017)
11. Kurdila, A.J., Zabarankin, M.: Convex functional analysis. Springer Science & Business Media, Birkhäuser Basel edn. (2006)
12. Lax, P.: Linear Algebra and its Applications. John Wiley & Sons, 2 edn. (2007)

13. Lemme, A., Neumann, K., Reinhart, R.F., Steil, J.J.: Neural learning of vector fields for encoding stable dynamical systems. Neurocomputing 141(1), 3–14 (2014)
14. Liang, T., Rakhlin, A.: Just interpolate: Kernel ridgeless regression can generalize. arXiv preprint arXiv:1808.00387v1 (2018)
15. Lohmiller, W., Slotine, J.J.E.: On contraction analysis for non-linear systems. Automatica 34(6), 683–696 (1998)
16. Manchester, I.R., Slotine, J.J.E.: Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. IEEE Transactions on Automatic Control (2017), In Press
17. Manchester, I.R., Slotine, J.J.E.: Robust control contraction metrics: A convex approach to nonlinear state-feedback $H-\infty$ control. IEEE Control Systems Letters 2(2), 333–338 (2018)
18. Manchester, I., Tang, J.Z., Slotine, J.J.E.: Unifying classical and optimization-based methods for robot tracking control with control contraction metrics. In: Int. Symp. on Robotics Research (2015)
19. Medina, J.R., Billard, A.: Learning stable task sequences from demonstration with linear parameter varying systems and hidden Markov models. In: Conf. on Robot Learning. pp. 175–184 (2017)
20. Micheli, M., Glaunés, J.A.: Matrix-valued kernels for shape deformation analysis. Geometry, Imaging and Computing 1(1), 57–139 (2014)
21. Minh, H.Q.: Operator-valued bochner theorem, fourier feature maps for operator-valued kernels, and vector-valued learning. arXiv preprint arXiv:1608.05639 (2016)
22. Nagabandi, A., Kahn, G., Fearing, R.S., Levine, S.: Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. arXiv preprint arXiv:1708.02596 (2017)
23. Olfati-Saber, R.: Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles. Ph.D. thesis, Massachusetts Inst. of Technology (2001)
24. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Conf. on Neural Information Processing Systems. pp. 1177–1184 (2007)
25. Rahimi, A., Recht, B.: Uniform approximation of functions with random bases. In: Allerton Conf. on Communications, Control and Computing. IEEE (2008)
26. Ravichandar, H., Salehi, I., Dani, A.: Learning partially contracting dynamical systems from demonstrations. In: Conf. on Robot Learning (2017)
27. Reyhanoglu, M., van der Schaft, A., McClamroch, N.H., Kolmanovsky, I.: Dynamics and control of a class of underactuated mechanical systems. IEEE Transactions on Automatic Control 44(9), 1663–1671 (1999)
28. Sanner, R.M., Slotine, J.J.E.: Gaussian networks for direct adaptive control. IEEE Transactions on Neural Networks 3(6), 837–863 (1992)
29. Sindhwani, V., Tu, S., Khansari, M.: Learning contracting vector fields for stable imitation learning. arXiv preprint arXiv:1804.04878 (2018)
30. Singh, S., Majumdar, A., Slotine, J.J.E., Pavone, M.: Robust online motion planning via contraction theory and convex optimization. In: Proc. IEEE Conf. on Robotics and Automation (2017), Extended Version, Available at http://asl.stanford.edu/wp-content/papercite-data/pdf/Singh.Majumdar.Slotine.Pavone.ICRA17.pdf
31. Singh, S., Sindhwani, V., Slotine, J.J., Pavone, M.: Learning stabilizable dynamical systems via control contraction metrics. In: Workshop on Algorithmic Foundations of Robotics (2018), In Press, Available at: https://arxiv.org/abs/1808.00113
32. Slotine, J.J.E., Li, W.: On the adaptive control of robot manipulators. Int. Journal of Robotics Research 6(3), 49–59 (1987)
33. Spong, M.W.: Underactuated mechanical systems. In: Control Problems in Robotics and Automation. Springer Berlin Heidelberg (1998)
34. Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D., Bagnell, A.: Improved learning of dynamics models for control. In: Int. Symp. on Experimental Robotics. pp. 703–713. Springer (2016)
35. Venkatraman, A., Hebert, M., Bagnell, J.A.: Improving multi-step prediction of learned time series models. In: Proc. AAAI Conf. on Artificial Intelligence (2015)
36. Zhang, L., Wu, S.Y., López, M.A.: A new exchange method for convex semi-infinite programming. SIAM Journal on Optimization 20(6), 2959–2977 (2010)
37. Zhou, D.X.: Derivative reproducing properties for kernel methods in learning theory. Journal of Computational and Applied Mathematics 220(1-2), 456–463 (2008)