CHARACTERIZING AND IMPROVING ROBOT LEARNING:
A CONTROL-THEORETIC PERSPECTIVE

by

James Alan Preiss

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2022

Acknowledgments

My path to this dissertation has not been direct. I wish to thank the many people whose support, encouragement, teaching, and inspiration have brought me here.

I am grateful for the opportunity to work with my advisor, Gaurav S. Sukhatme. His guidance has been thoughtful and kind. His interdisciplinary approach to robotics gave me freedom to explore different topics, knowing that he would help isolate the essence of a problem and see how it fits into the big picture. Gaurav often says that the output of a Ph.D. is not the research; it is the researcher. This lesson helped me appreciate the growth and learning from each project, no matter the results.

Many thanks to the USC academic community. I appreciate the insightful questions and suggestions from the members of my qualifying, thesis proposal, and defense committees: Nora Ayanian, Heather Culbertson, Ashutosh Nayyar, and Stefanos Nikolaidis. Nora was also a close collaborator on research projects that do not appear in this dissertation. Shaddin Dughmi and Haipeng Luo taught courses that shaped my research interests. Lizsl De Leon was a boundless source of cheer and wisdom in navigating the Department, School, and University. My studies were generously supported by the USC Viterbi-Graduate School Ph.D. Fellowship.

I was fortunate to collaborate with many fellow researchers: Sébastien M.R. Arnold, Jiajun Bi, Matt Buckley, Tao Chen, Zhenghao Dai, Nicole Fronda, Karol Hausman, Wolfgang Hönig, Marius Kloft, Alexander S. Koumis, T.K. Satish Kumar, Amlesh Sivanantham, Michael Leahy, David Millard, Artem Molchanov, Ragesh K. Ramachandran, Christian Wagner, Chen-Yu Wei, Stephan Weiss, Tao Yao, and Lifeng

Zhou. It was a privilege to learn from people with such diverse backgrounds and interests. I especially thank Karol and Wolfgang for setting an example during the first few papers, Ragesh for introducing me to many new mathematical ideas, Séb, Marius, and Chen-Yu for influencing me ask more theoretical questions, and David and Tao for bringing me into the world of deformable manipulation. To the labmates with whom I did not collaborate directly: Thank you for camaraderie and interesting conversations.

Thanks to Hanna Mazzawi and Eugen Hotaj for hosting my internship at Google Research NYC and broadening my perspective with a new problem. From my time in industry before graduate school, I thank my supervisors Thomas Jansen and Xan Gregg. Their mentorship in software development practice made the implementation side of my work much less intimidating.

From The Evergreen State College, I am grateful to Clyde Barlow, Dawn Rorvik, and Richard Weiss for giving me a wonderful first academic research experience. We worked with numerical algorithms and hardware, as I still do today. Thanks also to David McAvity and Brian Walter for teaching engaging mathematics courses.

My parents Tony and Leah Preiss have always given me unconditional love and encouragement. My brother Sandy is the best friend I could ask for. I thank them for everything. My grandfather Richard Palmer and uncle Dev Palmer were role models in technical fields since childhood. I am lucky to have an extended family filled with nice and interesting people.

I cannot imagine doing this without my partner Alana. She has given emotional support, shared her knowledge of computer science theory, and brightened many days. She reminds me to celebrate good things and take care of myself.

Finally, I dedicate this dissertation to my grandfather Jack Preiss, who pushed me to apply to graduate school when I needed to be pushed.

Table of Contents

List of Tables

List of Figures

Abstract

The interface between machine learning and control has enabled robots to move outside the laboratory into challenging real-world settings. Deep reinforcement learning can scale empirically to very complex systems, but we do not yet understand precisely when and why it succeeds. Control theory focuses on simpler systems, but delivers interpretability, mathematical understanding, and guarantees. We present projects that combine these strengths.

In empirical work, we propose a framework for tasks with complex dynamics but known reward functions. We restrict the use of learning to the dynamics modeling stage, and act based on this model using traditional state-space control. We apply this framework to robotic manipulation of deformable objects.

In theoretical work, we deploy the well-understood linear quadratic regulator (LQR) problem as a test case to "look inside" algorithms and problem structure. First, we investigate how reinforcement learning algorithms depend on properties of the dynamical system by bounding the variance of the REINFORCE policy gradient estimator as a function of the LQR system matrices. Second, we introduce the framework of suboptimal covering numbers to quantify how much a good multi-system policy must change with respect to the dynamics parameters, and bound the covering number for a simple class of LQR systems.

Chapter 1

Introduction

A machine learning system makes decisions based on a data set of observations and improves its performance as the amount of data grows. Machine learning is useful for interacting with systems that are too complicated to model based on first principles. Within the field of robotics, two especially important applications are vision input and complex dynamics. This dissertation focuses on the latter. Some examples of complex dynamics are:

- The contact friction between a foot and the surface upon which it walks.
- A deformable object bending under the influence of boundary conditions.
- The aerodynamics of a helicopter when it is close to the ground.

The dynamics of these phenomena obey the laws of physics, but that does not mean they are easy to predict:

- The distribution of pressure and friction across the foot changes throughout the stride. Each walking surface adds its own complexity, for example sand or ice.
- The deformable object is governed by continuum mechanics, so its true state is infinite-dimensional.
- The helicopter's airflow can be well-approximated in static hover, but is harder to predict when the helicopter is accelerating.

Even when an accurate physics-based model is available, using it in a robot's control loop may be computationally infeasible.

Control theory provides (usually) mathematically principled techniques for realizing a desired behavior of a dynamical system via inputs. However, those techniques often depend on assumptions that are not satisfied by complex systems, such as linearity, smoothness, convexity, simple probability distributions, and so on. Often the techniques struggle with high-dimensional systems. Control theory has developed its own learning methods within the subfields of system identification and adaptive control.

Reinforcement learning (RL) is the branch of machine learning that deals with acting optimally in an unknown dynamical system. Early RL research was mainly confined to finite state and action spaces in discrete time, whereas control theorists focused on continuous spaces in both discrete and continuous time. On the other hand, RL researchers have always been interested in "intelligent" behavior involving long-term planning, whereas control theory has focused on simpler behaviors like stabilizing at an equilibrium or tracking a reference trajectory. A key goal of RL theory, and of machine learning theory in general, is understanding the sample complexity of learning problems: how much data is required to guarantee that a certain performance metric is satisfied? Sample complexity upper bounds are most often derived by proposing an algorithm, while lower bounds are most often derived by carefully constructing worst-case problem instances. Sample complexity analysis is closely related in spirit to computational complexity analysis in computer science.

In the past few years, the theoretical sides of each community have increased their overlap. Especially notable has been the drive for learning-theory-style sample complexity guarantees in control-theory-style dynamical systems, especially linear systems, where many previous results only provided asymptotic guarantees. Mania et al. (2019), Simchowitz and Foster (2020), and others obtained finer-grained characterizations of existing methods in control theory by applying the sample complexity perspective. Conversely, researchers have also used those classic settings as mathematically tractable test cases to gain insight into

existing reinforcement learning algorithms (Fazel et al., 2018; Mohammadi et al., 2019). New lines of inquiry into these systems has also provoked new questions more fundamental than sample complexity (Bu et al., 2019b). A more extensive review of RL theory for continuous systems is given in §5.2.

Empirical successes The overlap of the empirical sides of learning and control have produced spectacular results in recent years. The success of deep neural networks for nonlinear function approximation with high-dimensional data, such as in image classification (Krizhevsky et al., 2012), motivated researchers to experiment with applying them to control tasks. In a landmark result, Mnih et al. (2013) showed that a single reinforcement learning algorithm and deep neural network architecture could reach human-level performance on many different games from the Atari 2600 console. This combination is known as deep reinforcement learning (deep RL). Their method used the Q-learning principle, meaning it could only be applied to settings where computing the maximum in the Bellman optimality operator (2.9) is feasible. Practically, this meant environments with finite (and not combinatorially large) action spaces.

Subsequently, a flurry of deep RL algorithms were proposed targeting similar results for continuous action spaces. The OpenAI Gym (Brockman et al., 2016), especially the locomotion-related environments, emerged as a benchmark. Among these algorithms, those of Schulman et al. (2017) and Haarnoja et al. (2018) were especially successful and became widely used for RL in continuous spaces. However, most of these results were confined to simulation due to high sample complexity, safety issues, or both.

For real physical systems, model-based RL is usually preferred for its sample efficiency. Deisenroth and Rasmussen (2011) used Gaussian process regression to handle model uncertainty in a principled way, but the method is computationally infeasible for high-dimensional systems. The guided policy search family of algorithms uses localized traditional trajectory optimization as a "teacher" for a global neural network policy, and has been applied to tasks with vision input (Levine et al., 2016) and discontinuous dynamics (Chebotar et al., 2017). Kalashnikov et al. (2018) used RL to learn a policy for robotic grasping of a wide range of objects, sidestepping the data efficiency issue by using many robots at once and designing

a "self-resetting" environment. OpenAI et al. (2019) achieved simulation-to-reality transfer in dexterous manipulation of a Rubik's cube, while Hwangbo et al. (2017) and Molchanov et al. (2019) demonstrated it for quadrotor control. We emphasize that these are only a few examples of successful applications of RL to robotics.

RL also became an essential tool in game-playing AI. Silver et al. (2018) achieved superhuman performance on the board game Go using a thoughtful combination of deep reinforcement learning and tree search. (An earlier version reached this milestone but depended on supervised training data from human players.) Later, Schrittwieser et al. (2020) replaced the assumption of known game rules with a learned dynamics model, making the resulting algorithm (MuZero) applicable to a much larger set of RL problems.

New challenges At the same time, researchers were discovering downsides of deep RL. Henderson et al. (2018) pointed out that the policy optimality achieved by deep RL algorithms is unusually sensitive to the initial state (seed) of the algorithm's pseudorandom number generator, as well as hyperparameters and implementation details. In contrast, deep supervised learning accuracy is not sensitive to random seed (Bhojanapalli et al., 2021). Engstrom et al. (2020) and Andrychowicz et al. (2020) showed that some of the performance gains attributed to core algorithmic differences could in fact be explained by minor implementation differences. Agarwal et al. (2021) pointed out that the high computational cost of deep RL experiments leads to small sample sizes, which are often not treated with enough statistical care.

Anecdotally, researchers applying deep RL to a novel task significantly different from the popular benchmarks generally cannot expect it to "just work". Reward shaping, hyperparameter tuning, and comparing several algorithms are the norm. Henderson et al. (2018) reported that the performance ranking of different algorithms is not consistent across different environments.

Non-RL combinations of learning and control Machine learning has useful applications in control beyond RL. For example, Shi et al. (2019, 2021) integrate learned models with first-principles models in

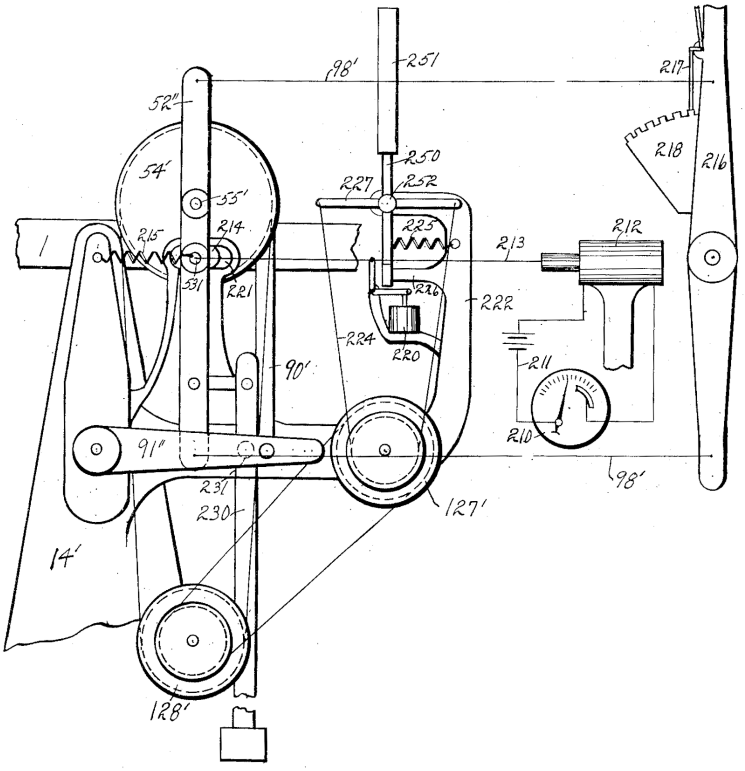


Figure 1.1: Illustration of a subassembly in an early electromechanical autopilot (Sperry, 1921, U.S. Patent 1368226A). Electrical components are used for actuation and as an energy source, but the core control policy is a physical mechanism.

a restricted manner to account for complex dynamics (of quadrotor ground effect and multi-quadrotor downwash, respectively) while preserving a stability guarantee. Ideas from control theory can be used to improve learning: Terzi et al. (2021) derive sufficient conditions that can be enforced upon a recurrent neural network dynamics model to admit guarantees for a state observer, controller, and the model itself. Singh et al. (2021) enforce a stabilizability condition on a learned dynamics model and show that it acts as a regularizer, improving model accuracy over generic regression when the data set is small. Amos et al. (2018) derive analytic derivatives of the outputs of a model-predictive control optimization problem and use them for end-to-end imitation learning and system identification.

Historical remarks The fields of control theory and reinforcement learning are closely related. Historically, control theory developed mainly within the context of mechanical and electrical engineering and

was applied to electrical or physical systems. Control theory predates digital computers: continuous-time controllers such as autopilots were implemented with analog electronics or even mechanisms with moving parts (Sperry, 1921), as shown in the patent illustration of Figure 1.1.

Reinforcement learning developed mainly within the artificial intelligence research community. In addition to practical applications, research was also tied to efforts to understand mechanisms of learning in humans and animals. Sutton and Barto (2018) give a retrospective on the history of reinforcement learning, including its relationship to optimal control.

1.1 Structure of dissertation

The remainder of this dissertation is organized into five chapters.

Chapter 2 provides the setting for our work. This chapter mainly focuses on mathematical foundations, but also discusses some of the key motivating ideas. In particular, §2.6 discusses families of Markov decision processes and families of optimal control problems, including several examples. We use this formalism in Chapters 3 and 6.

Chapter 3 presents our work on a deep reinforcement learning architecture for policies that can adapt online to systems with widely varying dynamics. Similar to adaptive methods in control theory, our framework is built on an online system identification process. We replace online estimation and optimization with pre-trained neural networks to handle complex settings with low computational cost. We also experiment with learned mapping from dynamics parameters to an embedding space. Our experiments highlight some fundamental questions about the multi-system setting and about reinforcement learning itself, which provide motivation for subsequent chapters.

Chapter 4 proposes a more structured alternative to reinforcement learning for systems where the dynamics are complex but the reward is simple and known. We learn a recurrent neural network dynamics model from input-output trajectory data and apply traditional state-space estimation and control to the

abstract internal state of the RNN model. We apply this framework to robotic manipulation of deformable objects. An ablation study demonstrates the benefit of using this closed-loop approach compared to feed-forward planning using the model.

Chapter 5 begins the theoretical portion of this dissertation. We analyze the variance of the REINFORCE policy gradient estimator for linear-quadratic regulator (LQR) systems. We provide upper and lower bounds on the variance as a function of the system parameters. With respect to the dynamics and cost matrices, our bounds are tight and closely match the behavior of the empirical variance. However, we also show that the variance is not directly correlated to the optimality of the policy produced by the REINFORCE algorithm. This challenges the folklore that high-variance gradient estimates are the dominant challenge in policy gradient methods.

Chapter 6 introduces the concept of suboptimal covering numbers as a way to quantify how much an optimal policy for an infinite family of control problems must alter its behavior with respect to the problem parameters. Suboptimal covering numbers are intuitive and have desirable mathematical properties such as parameterization independence. We show matching logarithmic covering number bounds for single-input fully-observable LQR problem families with actuator strength variations. For multi-input problems, we present empirical work testing a conjectured upper bound and use visualizations to inspect the behavior of candidate systems for a lower bound. We discuss work in progress and intermediate results towards proving the matrix-case conjecture, and initial experiments regarding further expansion of the scope of problem families.

Chapter 2

Foundation: Mathematics and Themes

Although the topics of each chapter in this dissertation are diverse, their mathematical foundations have many ideas in common. This chapter defines notation and reviews important definitions and theorems we will use. We assume the reader is familiar with some fundamental definitions in linear algebra, analysis, probability, and differential equations.

We also take the opportunity in this chapter to introduce some common themes that appear repeatedly in our work. In particular, we introduce notion of a structured family of control problems or Markov decision processes, and review concepts such as robustness and adaptivity that are broadly applicable in both traditional and learning-based control.

2.1 Notation

Sets For a set X, 2^X denotes the power set $\{Y : Y \subseteq X\}$.

Linear algebra The notation $A \succ B$ (resp. $A \succeq B$) indicates that the matrix $A - B$ is positive definite (resp. positive semidefinite). The sets of $n \times n$ positive definite and positive semidefinite matrices are denoted by $\mathbb{S}^n_{++}$ and $\mathbb{S}^n_{+}$ respectively. For a matrix $A \in \mathbb{R}^{n \times n}$, the set of its eigenvalues are denoted by $\Lambda(A)$, its spectral radius is denoted by $\rho(A) = \max\{|\lambda| : \lambda \in \Lambda(A)\}$, and the largest real part of its eigenvalues is denoted by $\rho^+(A) = \max\{\operatorname{Re}\lambda : \lambda \in \Lambda(A)\}$. Matrices or vectors of zeros and ones, with

dimension implied by context, are denoted by $\mathbf{0}$ and $\mathbf{1}$. The notation $\mathrm{diag}(x)$ (resp. $\mathrm{diag}(x_1, \ldots, x_n)$) refers to a diagonal matrix with the entries of the vector x (resp. the scalars or blocks $x_1, \ldots, x_n$) on its diagonal.

Functions and images The notation Y^X indicates the set of all functions from X to Y. If $f : X \mapsto Y$ and $X' \subseteq X$, we overload notation and denote the image of X' under f by

$$f(X') = \{f(x) : x \in X'\}.$$

Similarly, we denote the inverse image of $Y' \subseteq Y$ by

$$f^{-1}(Y') = \{x \in X : f(x) \in Y'\}.$$

Sequences The notation $X^{\mathbb{N}}$ indicates the set of all X-valued sequences. The notation $X^{<\mathbb{N}}$ indicates the set of all *finite* X-valued sequences, i.e. $X^{<\mathbb{N}} = \cup_{n=0}^{\infty} X^n$. When a sequence $x_0, x_1, \ldots$ has been defined, the notation $x_{m:n}$ refers to the tuple $(x_m, x_{m+1}, \ldots, x_n)$.

Norms and inner products For $P \succeq 0$, we use the notation

$$\langle x, y \rangle_P = \langle x, Py \rangle$$

for its weighted inner product. For its induced norm, we use

$$\|x\|_P = \sqrt{\langle x, x \rangle_P}.$$

Otherwise, the notation $\|\cdot\|_p$ for $p \geq 1$ refers to the usual p-norm, i.e.

$$\|x\|_p = \left(\sum_{i=1}^{n} x_i^p \right)^{\frac{1}{p}},$$

in which x_i are the individual components of x. For a linear function $F : \mathcal{X} \mapsto \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are vector spaces, the notation $\|F\|_{p,q}$ refers to the p-q operator norm:

$$\|F\|_{p,q} = \max\{\|Fx\|_q : x \in \mathcal{X}, \|x\|_p \leq 1\}.$$

We also apply the operator norm notation to matrices based on the natural isomorphism to linear operators. We denote the Frobenius norm of the matrix A by $\|A\|_F = \sqrt{\operatorname{tr} A^T A}$. We only use the norm notation without subscripts when the particular norm is explicitly stated or clear from the context.

Probability If (X, Σ) is a measurable space, then we use the notation $\Delta(X, \Sigma)$ to denote the set of all probability measures on (X, Σ). In cases where the sigma-algebra has already been stated or is common (e.g. the power set when X is finite; the Borel sigma-algebra when X is a topological space), we often use the notation $\Delta(X)$.

For a function $f : X \mapsto \Delta(Y)$, if the density function (Radon-Nikodym derivative) exists for all measures in $f(X)$, we use the notation $f(\cdot|x)$ to denote the density function of $f(x)$.

Indicator functions Let X denote some set and P a logical predicate on X. We use the notation

$$\mathbb{I}_{[P]}(x) = \begin{cases} 1 : P(x) \\ 0 : \neg P(x). \end{cases}$$

For a subset $S \subseteq X$ and the predicate $P(x) = x \in S$, we use the shorthand $\mathbb{I}_{\{S\}}$ instead of $\mathbb{I}_{[x \in S]}$.

Topology A topological space (X, T) is defined by a set X and a collection of "open" sets $T \subseteq 2^X$ that is closed under arbitrary unions and finite intersections. For arbitrary $Y \subseteq X$, the interior of Y is denoted by $\operatorname{int} Y = \cup\{S \in T : S \subseteq Y\}$.

2.2 Fundamentals

2.2.1 Metrics

Recall that a metric on a set X is a function $d : X \times X \mapsto \mathbb{R}$ satisfying the properties for all $x, y, z \in X$:

- Identity of indiscernibles: $d(x, y) = 0 \iff x = y$.
- Symmetry: $d(x, y) = d(y, x)$.
- Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$.

From these properties, nonnegativity $d(x, y) \geq 0$ can be derived. A *semimetric* on X satisfies nonnegativity, symmetry, and the identity of indiscernibles, but does not satisfy the triangle inequality. Notably, the squared Euclidean distance $\|x - y\|_2^2$ is a semimetric.

2.2.2 Lipschitz and smooth functions

Suppose (X, d_X) and (Y, d_Y) are metric spaces We say $f : X \mapsto Y$ is *L-Lipschitz* for $L > 0$ if, for all $x, x' \in \operatorname{dom} f$,

$$d_Y(f(x), f(x')) \leq L d_X(x, x').$$

Most commonly, we study Lipschitz functions in normed vector spaces where d_X and d_Y are norm-induced metrics.

If the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is differentiable and its gradient is β-Lipschitz, that is,

$$\big\|\nabla f(x) - \nabla f(x')\big\| \leq \beta \big\|x - x'\big\|,$$

we say f is β-*smooth*.

2.2.3 Orthogonal and Euclidean groups

The general linear group, denoted $\mathrm{GL}(n)$, is the group of invertible linear operators on n-dimensional Euclidean space over the field of real numbers, with composition as the group operation.

The subset of $\mathrm{GL}(n)$ that is distance-preserving forms a subgroup, the orthogonal group $\mathrm{O}(n)$. Under the Euclidean topology, the group $\mathrm{O}(n)$ has two connected components. The component that contains the identity is a subgroup, denoted $\mathrm{SO}(n)$, and consists of those elements of $\mathrm{O}(n)$ that preserve orientation. These are the familiar rotations when $n = 2$ or 3.

The notation $\mathfrak{so}(n)$ refers to the Lie algebra associated with $\mathrm{SO}(n)$, which corresponds to the tangent space of $\mathrm{SO}(n)$ at the identity. It is beyond the scope of this dissertation introduce Lie groups and Lie algebras with satisfactory rigor. For our purposes, we can think of each element of $\mathfrak{so}(n)$ as a "rotational velocity".

The *Euclidean group*, denoted $\mathrm{E}(n)$, are the operators that are distance-preserving but not necessarily linear. The *special Euclidean group* $\mathrm{SE}(n)$ is the orientation-preserving subgroup of $\mathrm{E}(n)$. The group $\mathrm{SE}(3)$ describes the full "pose" of a rigid object in three-dimensional space. $\mathrm{SE}(3)$ is isomorphic to $\mathrm{SO}(3) \times \mathbb{R}^3$. The groups $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$ appear often in robotics. The associated Lie algebra $\mathfrak{se}(3)$ is isomorphic to $\mathfrak{so}(3) \times \mathbb{R}^3$, and also appears often in robotics to describe the full "velocity state" of a rigid object in three-dimensional space, sometimes called the *twist*.

There are several ways to represent elements of $\mathrm{SO}(3)$ for computation, including the 3×3 matrices themselves, the unit quaternions, pairs of a unit vector axis and a rotation angle, triples of rotation angles around fixed axes (Euler angles), and skew-symmetric matrices for elements of $\mathfrak{so}(3)$. Although the latter three representations are parameterized by $\mathbb{R}^3$, the incompatible topologies of $\mathbb{R}^3$ and $\mathrm{SO}(3)$ imply that

any mapping from $\mathbb{R}^3$ to $\mathrm{SO}(3)$ must suffer from at least one of the following problems: incompleteness, multiple-covering, or discontinuity.

2.3 Optimization

Mathematical optimization is a generic framework that permeates many aspects of robotics, including planning, control, machine learning, and multi-robot coordination. An optimization problem has the form

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{X}, \end{aligned} \tag{2.1}$$

where $\mathcal{X}$ is some set and $f : \mathcal{X} \mapsto \mathbb{R}$ is some function. In this dissertation we are mainly concerned with *continuous optimization*, where $\mathcal{X}$ is an uncountable set endowed with notions like topology, metric, inner product, etc. (By contrast, in *combinatorial optimization* the set $\mathcal{X}$ is finite, although possibly very large.) If $\mathcal{X} = \mathbb{R}^n$ for some n, we say the problem is *unconstrained*. On the other hand, if $\mathcal{X} \subset \mathbb{R}^n$, we say the problem is *constrained*. For constrained problems we usually define membership in $\mathcal{X}$ by equality and inequality constraints, resulting in the form

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) = \mathbf{0} \\ & h(x) \leq \mathbf{0}, \end{aligned} \tag{2.2}$$

where g and h are arbitrary vector-valued functions.

Local and global optima If $x \in \mathcal{X}$ is a true solution to the generic optimization problem (2.1), we say it is a *global optimum*. On the other hand, we say x is a *local optimum* if there exists an open set $\mathcal{B}$ such that $x \in \mathcal{B}$ and $f(x) \leq f(x')$ for all $x' \in \mathcal{B} \cap \mathcal{X}$. Depending on the structure of the optimization problem, it may

not be computationally feasible to seek a global optimum. An important class of optimization problems for which we can find a global optimum with polynomial query complexity are the convex optimization problems, discussed in §2.4.

2.4 Convexity

A set $\mathcal{X} \subseteq \mathbb{R}^n$ is *convex* if, for any $x, x' \in \mathcal{X}$ and $\theta \in [0, 1]$, it holds that

$$(1 - \theta)x + \theta x' \in \mathcal{X}.$$

Example 2.4.1. Some commonly encountered convex sets are:

- $\mathbb{R}^n$ itself.
- Norm balls $\{x \in \mathbb{R}^n : \|x\|_p \leq r\}$ of any radius $r \geq 0$ for $1 \leq p \leq \infty$. Note that the p-norm ball for $p < 1$ is not convex.
- Polytopes / polyhedra of the form

  $$\{x \in V : Ax \leq b\},$$

  where A is a linear map from V to some other vector space W, $b \in W$, and $\leq$ denotes simultaneous elementwise satisfaction.
- The set of $n \times n$ positive semidefinite matrices $\mathbb{S}^n_+$.
- Degenerate cases: The empty set, the singleton set $\{x\}$ for $x \in \mathbb{R}^n$.

Theorem 2.4.2 (Properties of convex sets)**.** *If $\mathcal{S}$ and $\mathcal{T}$ are convex sets, then …*

- ***Intersections**: $\mathcal{S} \cap \mathcal{T}$ is convex.*

- ***Minkowski sums**: $\{s + t : s \in \mathcal{S}, t \in \mathcal{T}\}$ is convex.*
- ***Cartesian products**: If we define vector space operations for $\mathcal{S} \times \mathcal{T}$ in the natural way, then $\mathcal{S} \times \mathcal{T}$ is convex.*
- ***Affine images:** If f is an affine function, then the image $f(\mathcal{S})$ is convex.*

2.4.1 Convex functions

The *epigraph* of a function $f : \mathcal{X} \mapsto \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^n$, is the set

$$\mathbf{epi}\, f = \{(x, r) : x \in \mathcal{X}, r \geq f(x)\} \subseteq \mathbb{R}^{n+1}.$$

If the epigraph of f is convex, we say f is a convex function. Equivalently, f is convex if and only if it satisfies *Jensen's inequality*:

$$f((1 - \theta)x + \theta x') \leq (1 - \theta)f(x) + \theta f(x')$$

for all $x, x' \in \mathcal{X}$ and $\theta \in [0, 1]$. Jensen's inequality can be generalized to a probabilistic form: if X is a random variable, then

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

We say a function f is *concave* if $-f$ is convex. Convex functions satisfy the following properties:

- If f and g are convex, then $f + g$ is convex. More generally, any nonnegative weighted sum of convex functions is convex.
- $f(Ax + b)$ is convex. (Affine composition)

- If F is a (potentially infinite) set of convex functions on domain $\mathcal{X}$, then the pointwise supremum function $\sup_{f \in F} f(x)$ is convex in x.

2.4.1.1 Subdifferentials

Suppose $\mathcal{X} \subseteq \mathbb{R}^n$ is a convex set and $f : \mathcal{X} \mapsto \mathbb{R}$ is a function. We say that $g \in \mathbb{R}^n$ is a *subgradient of f at* $x_0 \in \mathcal{X}$ if, for all $x \in \mathcal{X}$, we have

$$f(x_0) + \langle g,\ x - x_0 \rangle \leq f(x).$$

The *subdifferential of f at* x_0, denoted by $\partial f(x_0)$, is the set of all subgradients of f at x_0. The subdifferential is a convex set. If $\partial f(x) \neq \emptyset$ for all $x \in \operatorname{int} \mathcal{X}$, then f is convex. Conversely, if f is convex, then $\partial f(x) \neq \emptyset$ for all $x \in \mathcal{X}$. If f is differentiable at x , then $\partial f(x)$ is a singleton set. If $g : \mathbb{R}^n \mapsto \mathbb{R}$ is convex, then $x \in \mathbb{R}^n$ is a local optimum of g if and only if $\mathbf{0} \in \partial g(x)$. These properties are proved by Bubeck (2015).

2.4.1.2 Convex optimization problems

A *convex optimization problem* is a problem of the form

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{X}, \end{aligned} \tag{2.3}$$

where $\mathcal{X}$ is a convex set and f is a convex function. There is much to be said about convex optimization problems (Nesterov, 2003; Boyd and Vandenberghe, 2004; Bubeck, 2015), but the most important facts are the following:

- If x is a local optimum of a convex optimization problem, then it is also a global optimum.

- There exist algorithms to find an approximately optimal solution for convex optimization problems that are polynomial-time in the relevant properties of f and $\mathcal{X}$, even under very weak "membership oracle" access to $\mathcal{X}$ (Abernethy and Hazan, 2016).

2.4.2 Strong convexity

A function f is *μ-strongly convex* with respect to some norm $\|\cdot\|$ if there exists $\mu > 0$ such that, for all $x, y \in \operatorname{dom} f$ and all $g \in \partial f(x)$,

$$f(y) \geq f(x) + \langle g,\ y - x \rangle + \frac{\mu}{2}\|y - x\|^2. \tag{2.4}$$

Informally, strongly convex functions are "at least as convex" as a quadratic function. Strong convexity leads to improved convergence rates of first-order optimization algorithms (Bubeck, 2015).

2.4.3 Quasiconvex functions

Definition 2.4.3. A function $f : \mathcal{D} \mapsto \mathbb{R}$ on the convex domain $\mathcal{D} \subseteq \mathbb{R}^n$ is *quasiconvex* if its sublevel sets $\mathcal{D}_\alpha = \{x \in \mathcal{D} : f(x) \leq \alpha\}$ are convex for all $\alpha \in \mathbb{R}$.

All convex functions are quasiconvex, but not all quasiconvex functions are convex: for example, the function $-e^{-x^2}$ as shown in Figure 2.1.

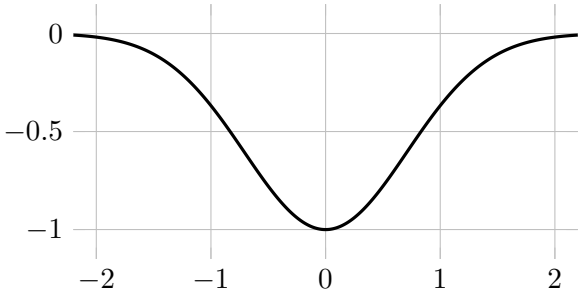


Figure 2.1: The function $-e^{-x^2}$ is quasiconvex but not convex.

Lemma 2.4.4 (Boyd and Vandenberghe (2004), §3.4)**.** *The following facts hold for quasiconvex functions on a convex $\mathcal{D} \subseteq \mathbb{R}$ (equivalently, $\mathcal{D}$ is an interval).*

(a) If $f : \mathbb{R} \mapsto \mathbb{R}$ is continuous, then f is quasiconvex if and only if at least one of the following conditions holds on $\mathcal{D}$:

1. *f is nondecreasing.*
2. *f is nonincreasing.*
3. *There exists $c \in \mathcal{D}$ such that for all $t \in \mathcal{D}$, if $t < c$ then f is nonincreasing, and if $t \geq c$ then f is nondecreasing.*

(b) If $f : \mathbb{R} \mapsto \mathbb{R}$ is twice differentiable and $\dfrac{\mathrm{d}^2 f}{\mathrm{d}x^2} > 0$ for all $x \in \mathcal{D}$ where $\dfrac{\mathrm{d}f}{\mathrm{d}x} = 0$, then f is quasiconvex on $\mathcal{D}$.

(c) If $f(x) = \dfrac{p(x)}{q(x)}$, where $p : \mathbb{R} \mapsto \mathbb{R}$ is convex with $p(x) \geq 0$ on $\mathcal{D}$ and $q : \mathbb{R} \mapsto \mathbb{R}$ is concave with $q(x) > 0$ on $\mathcal{D}$, then f is quasiconvex on $\mathcal{D}$.

2.4.4 Convex optimization algorithms

Suppose $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a convex function and that the optimum

$$x^\star = \operatorname*{argmin}_{x \in \mathbb{R}^n} \; f(x)$$

exists. In this section we sketch the main results for optimization of a convex function under the *first-order oracle* model, in which the algorithm can query $\nabla f(x)$ for arbitrary x but has no further structural knowledge of $f(x)$. The first-order oracle falls in the broader class of *black-box optimization* models, which also includes zeroth-order oracles where only $f(x)$ can be queried.

Many black-box optimization algorithms require an initial guess x_0 and generate an infinite sequence of iterates $x_1, x_2, \ldots$ A central object of study for such algorithms is their convergence rate: a bound of the form

$$f(x_k) - f(x^\star) \leq b(k),$$

where the function $b(k) : \mathbb{N} \mapsto \mathbb{R}_{\geq 0}$ depends on properties of f and the distance $R = \|x_0 - x^\star\|$.

Gradient descent The simplest procedure to optimize f is (sub)gradient descent: starting with an initial guess $x_0 \in \mathbb{R}^n$, we follow the recurrence

$$x_{k+1} = x_k - \eta \nabla f(x_k),$$

where $\eta > 0$ is a step size parameter. When f is not differentiable, we can understand $\nabla f(x)$ to denote an arbitrary deterministic selection from $\partial f(x)$.

When f belongs to the following function classes, (sub)gradient descent with the appropriate choice of η leads to the following convergence rates (Bubeck, 2015):

- L-Lipschitz: $O\left(\frac{RL}{\sqrt{k}}\right)$.
- β-smooth: $O\left(\frac{R^2\beta}{k}\right)$.
- β-smooth and μ-strongly convex: $O\left(R^2 \exp\left(-\frac{k}{\beta/\mu}\right)\right)$.

Accelerated gradient descent A computationally inexpensive modification of gradient descent introduced by Nesterov (1983) turns out to yield improved convergence rates. Starting with the initial guess $y_0 = x_0 \in \mathbb{R}^n$, we follow the recurrence

$$\begin{aligned} x_{k+1} &= y_k - \frac{1}{\beta}\nabla f(y_k), \\ y_{k+1} &= x_{k+1} + \frac{\sqrt{\beta} - \sqrt{\mu}}{\sqrt{\beta} + \sqrt{\mu}}(x_{k+1} - x_k). \end{aligned}$$

Accelerated gradient descent leads to the following convergence rates (Bubeck, 2015):

- β-smooth: $O\left(\frac{R^2\beta}{k^2}\right)$.
- β-smooth and μ-strongly convex: $O\left(R^2 \exp\left(-\frac{k}{\sqrt{\beta/\mu}}\right)\right)$.

Note that the ratio $\frac{\beta}{\mu} \geq 1$ according to the definitions of smoothness and strong convexity.

Lower bounds The convergence rates attained by gradient descent and accelerated gradient descent are optimal. For any algorithm in the first-order oracle model that satisfies

$$x_{k+1} \in \operatorname{span}(g_1, \ldots, g_k),$$

where g_k is the (sub)gradient queried at step k, there exists a L-Lipschitz convex function, a β-smooth convex function, and a β-smooth and μ-strongly convex function for which $f(x_k) - f(x^\star)$ is lower-bounded by $\Omega(1/\sqrt{k})$, $\Omega(1/k^2)$, and $\Omega(\exp(-k))$ respectively (Bubeck, 2015).

2.5 Markov decision processes

The formalism of Markov decision processes captures a wide range of problems in which an agent interacts with a dynamical system.

Definition 2.5.1. A discrete-time *Markov decision process (MDP)* is defined by the tuple $(\mathcal{X}, \mathcal{U}, P, r, \mu)$ where:

- $\mathcal{X}$ is the state space,
- $\mathcal{U}$ is the action space,
- $P : \mathcal{X} \times \mathcal{U} \mapsto \Delta(\mathcal{X})$ is the state transition map,
- $r : \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}$ is the reward function, and
- $\mu \in \Delta(\mathcal{X})$ is a distribution over initial state.

The agent interacts with the MDP in the following manner:

- The initial state is sampled: $x_0 \sim \mu$.
- At each time step $t \in \mathbb{N}$, the agent observes the current state $x_t \in \mathcal{X}$ and takes an action $u_t \in \mathcal{U}$.
- The agent receives and observes the reward $r_t = r(x_t, u_t)$.
- The next state is sampled according to $x_{t+1} \sim P(x_t, u_t)$.

The property that

$$P(x_{t+1}|x_t, u_t) = P(x_{t+1}|x_1, u_1, \ldots, x_t, u_t)$$

is called the *Markovian property*.

Any method for choosing actions in a MDP is called a *policy*. A fully general definition of policies includes all functions $\mathcal{X} \times (\mathcal{X} \times \mathcal{U} \times \mathbb{R})^{<\mathbb{N}} \mapsto \Delta(\mathcal{U})$. That is, in addition to the current state, the policy could depend on the full sequence of states, actions, and rewards from previous timesteps. However, the Markovian property implies that history-dependent policies are unnecessary for optimality. A *stationary* policy is a function $\mathcal{X} \mapsto \Delta(\mathcal{U})$ that does not depend on the history or time index. A policy is called

deterministic if it only outputs point distributions. When discussing deterministic policies we state the policy class codomain as $\mathcal{U}$ instead of $\Delta(\mathcal{U})$.

If either of $\mathcal{X}$ and $\mathcal{U}$ are uncountable sets, then many mathematical statements about an MDP require substantial technical delicateness and additional conditions on P, r, and μ to ensure that derived objects of interest exist, are unique, and satisfy properties such as measurability. A thorough treatment of these issues is given by Bertsekas and Shreve (1978). To keep this section simple, we limit our discussion to finite MDPs. However, we emphasize that many of the core ideas, such as Bellman optimality, still hold in uncountable spaces under mild assumptions.

2.5.1 Partially observable Markov decision processes

In the standard MDP, the agent observes the full state x_t. This is often unrealistic. To model incomplete state observations, we introduce the notion of partial observability:

Definition 2.5.2. A discrete-time *partially observable Markov decision process (POMDP)* is defined by the tuple $(\mathcal{X}, \mathcal{U}, \mathcal{Y}, P, r, h, \mu)$ where:

- $\mathcal{X}$, $\mathcal{U}$, P, r, μ are as in Definition 2.5.1,
- $\mathcal{Y}$ is a measurable observation space,
- $h : \mathcal{X} \times \mathcal{U} \mapsto \Delta(\mathcal{Y})$ is the observation function.

The agent interacts with the MDP in the following manner:

- The initial state is sampled: $x_0 \sim \mu$.
- At each time step $t \in \mathbb{N}$, the agent observes a single sample $y_t \sim h(x_t, u_t)$ and takes an action $u_t \in \mathcal{U}$. The agent *does not* observe x_t.
- The agent receives and observes the reward $r_t = r(x_t, u_t)$.

- The next state is sampled according to $x_{t+1} \sim P(x_t, u_t)$.

In the case where $\mathcal{X}, \mathcal{U}, \mathcal{Y}$ are all finite, computing an optimal finite-horizon policy when P, r, h, μ are known is PSPACE-complete (Papadimitriou and Tsitsiklis, 1987). This implies that a vast universe of computational problems can be reduced to policy optimization for a POMDP.

From the perspective of control theory, partially observable settings are the norm. It is often expensive or impossible to equip a physical system with enough sensors to measure all states. The task of estimating state from a history of inputs and outputs is known as *state estimation*. In linear dynamical systems, state estimation becomes mathematically and computationally tractable, as we will discuss in § 2.9.1.7. In nonlinear continuous-state systems it is generally difficult to provide performance guarantees for state estimation, but straightforward extensions of the linear methods often work well in practice when the system dynamics are smooth and the sensors are not too noisy.

Remark 2.5.3. The MDP model is more expressive than it may initially appear. For example, we can model a system where the dynamics change over time by adding a time index variable to the state space.

2.5.2 Trajectories

We refer to a record of interaction with an MDP as a *trajectory* and use the notation

$$\tau = (x_0, u_0, r_0), (x_1, u_1, r_1), \dots$$

We refer to the space of all possible trajectories, with an "ambient" MDP implied by context, as $\mathbb{T}$. Note that $\mathbb{T}$ may be finite-dimensional or a sequence space. A policy $\pi : \mathcal{X} \mapsto \Delta(\mathcal{U})$ induces a distribution over $\mathbb{T}$ governed by

$$x_0 \sim \mu,\ u_t \sim \pi(x_t),\ r_t = r(x_t, u_t),\ x_{t+1} \sim P(x_t, u_t). \tag{2.5}$$

We will denote this distribution by $\boldsymbol{\tau}_\pi$. When we use the notation $\tau \sim \boldsymbol{\tau}_\pi$, it should be understood as a shorthand for the statement (2.5).

2.5.3 Infinite-horizon MDPs

An *infinite-horizon* MDP is one where time goes on forever, $t \to \infty$. For infinite-horizon MDPs, we define the *state-value function*, or simply *value function*, $V^\pi : \mathcal{X} \mapsto \mathbb{R}$ of the policy as the discounted sum

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \;\middle|\; \pi, x_0 = x\right] \tag{2.6}$$

for $\gamma \in (0, 1]$, where the expectation is taken over the randomness of the dynamics P and of the policy π. If the reward r is bounded, then a discount factor $\gamma < 1$ ensures that V^π is bounded. In some special cases (such as linear-quadratic regulators, see §2.9.1.6), it can be shown that V^π is bounded for all policies of interest even though r is unbounded, in which case $\gamma = 1$ is safe to use.

Similarly, we define the *action-value function* or *Q-function* of π as

$$Q^\pi(x, u) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \;\middle|\; \pi, x_0 = x, u_0 = u\right].$$

MDP objective The optimization goal for infinite-horizon MDPs is to solve

$$\underset{\pi \in \Delta(\mathcal{U})^{\mathcal{X}}}{\text{maximize}} \; \underset{s \sim \mu}{\mathbb{E}}[V^\pi(s)], \tag{2.7}$$

2.5.3.1 Bellman equations and operators

From the infinite-horizon value function definition (2.6), we immediately see its recursive nature:

$$\begin{aligned} V^\pi(x) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \;\middle|\; \pi, x_0 = x\right] \\ &= \mathop{\mathbb{E}}_{u\sim\pi,\ x'\sim P(x,u)}\left[r(x,u) + \sum_{t=1}^{\infty} \gamma^t r(x_t, u_t) \;\middle|\; \pi, x'\right] \\ &= \mathop{\mathbb{E}}_{u\sim\pi,\ x'\sim P(x,u)}\left[r(x,u) + \gamma V^\pi(x')\right], \end{aligned} \tag{2.8}$$

where the first step uses linearity of expectation and the last step uses the Markov property. Alternatively, we can define the *Bellman operator for* π, denoted as $\mathcal{T}^\pi : \mathbb{R}^{\mathcal{X}} \mapsto \mathbb{R}^{\mathcal{X}}$, by

$$(\mathcal{T}^\pi V)(x) = \mathop{\mathbb{E}}_{u\sim\pi(x),\ x'\sim P(\cdot|x,u)}\left[r(x,u) + \gamma V(x')\right].$$

Intuitively, $\mathcal{T}^\pi$ takes an arbitrary value function V and returns a value function "more like" the V^π. We can now rewrite the conclusion of eq. (2.8) as

$$\mathcal{T}^\pi V^\pi = V^\pi.$$

In other words, V^π is a fixed point of $\mathcal{T}^\pi$.

This gives us a way to characterize and compute V^π for a given π, but it does not give us a way to characterize or compute an optimal π. For this, we define *optimal value function* and *optimal Q-function* as

$$V^\star(x) = \sup_\pi V^\pi(x),$$

$$Q^\star(x,u) = \sup_\pi Q^\pi(x,u).$$

We define the *Bellman optimality operator*, denoted by $\mathcal{T}^\star : \mathbb{R}^{\mathcal{X}\times\mathcal{U}} \mapsto \mathbb{R}^{\mathcal{X}\times\mathcal{U}}$, by

$$(\mathcal{T}^\star Q)(x,u) = r(x,u) + \gamma \mathop{\mathbb{E}}_{x'\sim P(x,u)} \left[\max_{u'\in\mathcal{U}} Q(x',u')\right]. \tag{2.9}$$

We say a function $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{U}}$ satisfies the Bellman optimality equations if

$$\mathcal{T}^\star Q = Q. \tag{2.10}$$

It can be shown that (Agarwal et al., 2022):

- $Q = Q^\star$ if and only if Q is a fixed point of $\mathcal{T}^\star$ (that is, if it satisfies the Bellman optimality equations).
- For any infinite-horizon MDP, an optimal stationary and deterministic policy exists.
- $\mathcal{T}^\star$ is a γ-contraction mapping with respect to the ∞-norm.

Taken together, these propositions imply that the recurrence

$$Q^{(0)} = 0, \quad Q^{(k+1)} = \mathcal{T}^\star Q^{(k)}$$

will converge to $Q^\star$, and that any policy satisfying $\pi(x) \in \operatorname{argmax}_{u\in\mathcal{U}} Q^\star(x,u)$ is an optimal policy. The algorithm defined by computing this recurrence is known as *Q-value iteration*. Note that Q-value iteration requires full knowledge of P and r.

2.5.4 Finite-horizon objective

In an *episodic* or *finite-horizon* MDP, the agent interacts with the environment for a fixed number of time steps, denoted by $H \in \mathbb{N}$, and is then reset back to an initial state $x_0 \sim \mu$. For the finite-horizon case, the value and Q functions are different for each time step. This implies that optimal policies for finite-horizon

MDPs are time-dependent in general: we must consider the policy class of functions $\mathcal{X} \times \mathbb{N} \mapsto \Delta(\mathcal{U})$. However, in this dissertation we are only concerned with optimizing **stationary** policies, even in finite-horizon settings. Therefore, for the purposes of this dissertation, for each $h \in \{0, \dots, H-1\}$ we define

$$V_h^\pi(x) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(x_t, u_t) \,\middle|\, \pi, x_h = x\right],$$

and

$$Q_h^\pi(x, u) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(x_t, u_t) \,\middle|\, \pi, x_h = x, u_h = u\right],$$

where again the expectations are over the randomness in P and π. For the purposes of this dissertation, the core optimization problem in finite-horizon MDPs is to solve

$$\underset{\pi \in \Delta(\mathcal{U})^{\mathcal{X}}}{\text{maximize}} \; \mathop{\mathbb{E}}_{x \sim \mu}[V_0^\pi(x)]. \tag{2.11}$$

Time-dependent equivalents of the Bellman optimality results described for infinite-horizon MDPs in §2.5.4 exist for finite-horizon MDPs. Stating these results requires more notational bookkeeping and is not used subsequently in this dissertation, so we omit them here. The proofs follow the same spirit, using the additivity of the value function and the Markov property of the dynamics.

Remark 2.5.4. Some classes of stochastic bandit problems (Lattimore and Szepesvári, 2020) can be interpreted as a degenerate case of episodic MDPs with $H = 1$. For contextual bandits, the context becomes the state, while for pure bandits the state space becomes a singleton set.

2.6 Families of MDPs

In this dissertation we will often be interested in discussing a family of MDPs with $\mathcal{X}, \mathcal{U}$ in common but differences in one or more of P, r, μ. When this arises, we use the notation Φ to refer to this family, and $\phi \in \Phi$ for a particular MDP in the family. When working with a family of MDPs, we use the notation

$$P(\cdot|x, u; \phi), \quad r(x, u; \phi), \quad \mu(\cdot|\phi)$$

to denote the "master" dynamics, reward, and initial state distributions that also depend on ϕ. We use the subscripted terms $(P_\phi, r_\phi, \mu_\phi)$ to identify the transition dynamics, reward function, and initial state distribution for a particular $\phi \in \Phi$.

We will also use this notation to refer to families of optimal control problems that do not necessarily fit our discrete-time finite MDP formalism precisely. For example, we will use it to describe families of continuous-time linear dynamical systems in Chapter 6.

We are most interested in cases where Φ is highly structured. We now give a list of such examples.

2.6.1 Dynamics variations

In each of the following examples, only the dynamics P change.

Kinematic-chain robots The family Φ represents a set of robots with a common kinematic topology but with variations in mass, geometry, actuator strength, coefficients of friction, and so on. The state space $\mathcal{X}$ is described by the position in $\mathrm{SE}(3)$ of the kinematic root, the combined rotational and angular velocities in $\mathfrak{se}(3)$ of the kinematic root, and the angles and velocities of the joints, which applies to all robots. The action space $\mathcal{U}$ is torque commands for the joint actuators.

Aircraft The family Φ represents a set of aircraft. For practical purposes, the state is fully described as a rigid body. For a family of quadrotors, the action space $\mathcal{U}$ is four rotor thrusts. For a family of twin-engine airplanes, the action space $\mathcal{U}$ is two engine thrusts and the positions of the ailerons, elevators, and rudder.* Φ represents variations in mass, moments of inertia, thruster configuration, and aerodynamic properties that lead to different dynamics.

Manipulated rigid objects The family Φ represents different objects that may be held in the gripper of a robot. The objects are rigid, so their state is fully described by an element of $\mathrm{SE}(3) \times \mathfrak{se}(3)$. However, the objects have different shapes, masses, and moments of inertia, resulting in different dynamics for the complete system of a force/torque-controlled robot grasping the object.

Deformable objects materials The family Φ represents variations in material properties of a deformable object that may be held in the gripper of a robot. Each variation shares the same shape, so its state is fully described by the same infinite-dimensional continuum state. However, the objects have different material properties leading to varying levels of springiness, compressibility, etc. Again, this results in different dynamics for the coupled robot-object system.

Game opponents The family Φ represents one player's side of a two-player competitive game with varying opponent strategies. The reward r is zero until the game reaches a completed state. Different strategies of the opponent lead to different dynamics.

2.6.2 Reward variations

In each of the following examples, only the reward function r changes.

*These models are simplifications that are only reasonable under the assumption of smooth and Lipschitz control inputs.

Navigation goals The state space $\mathcal{X} \subseteq \mathbb{R}^3$ is some free space in the physical world. The action space $\mathcal{U} = \mathbb{R}^3$ is the desired robot velocity. The dynamics are simple integration combined with collision dynamics. The reward is $d(x, g)$ where $g \in \mathcal{X}$ is a goal state and d is a semimetric on $\mathcal{X}$.

Object arrangement A robot manipulates a set of objects on a desk. The state space $\mathcal{X}$ is the positions of the objects, alongside the robot state. The action space $\mathcal{U}$ includes the ability to move, close, and release the robot's gripper. The reward encodes the user's preferences for how the objects are arranged. For example, the user might want long objects to always be parallel to one of the desk edges. Alternatively, the user might want all objects packed as tightly as possible on one side of the desk.

Driving styles In an autonomous car, the reward r might encode the passenger's preferences regarding speed, aggressiveness, smoothness of motion, avoiding freeways, and so on.

Performance-efficiency tradeoffs Many robotics applications confront the system designer with a tradeoff between performance (time to complete a task, precision of tracking a trajectory, ...) and the amount of energy or other resources used. Changing the balance results in different reward functions.

2.7 Reinforcement learning

Reinforcement learning (RL) refers to the task of learning an optimal policy for an MDP—that is, solving the optimization problem (2.7) or (2.11)—without prior knowledge of the dynamics P, the reward r, or the initial state distribution μ. In the standard formulation of RL, we can only learn information about the MDP by interacting with it through the agent "interface" of Definition 2.5.1. That is, we are placed in the state x_0, we take actions, observe the reward and next state, and the environment eventually resets in the finite horizon case.

On the other hand, computing an optimal policy when P, r, and μ are known is called *solving* the MDP in reference to the Bellman optimality equation $\mathcal{T}^\star Q = Q$ (§ 2.5.3), which can be solved directly using linear programming or value iteration in the finite-state case. In between there is a spectrum of interfaces with the MDP. For example, we may have an "oracle" or "generative model" to sample P from any state and action instead of just the current state. In robotics the reward r is often known and designed by the robotics engineer to achieve some practical goal. We may have the ability to reset the MDP to $x_0 \sim \mu$ as desired.

There are a great many algorithms for reinforcement learning. In this dissertation, we will only explore one family of algorithms in depth: the *policy gradient* family, as defined in § 2.7.2. We analyze a policy gradient method in Chapter 5.

2.7.1 On and off-policy algorithms

Reinforcement learning algorithms can be classified as either *on-policy* or *off-policy*. On-policy algorithms can only make use of data that was generated by the current iterate of the policy being optimized. Off-policy algorithms can use data that was generated by a different behavior policy. Off-policy algorithms tend to be more complicated.

The off-policy data is often, but not always, from an earlier iterate of the policy being optimized. Initializing the store of off-policy data with human (or other "expert") demonstrations is a simple and powerful way to guide RL towards an optimal policy without the need for oracle access to the "expert" policy. The off-policy data may also be generated by a strategic exploration method, whereas in on-policy algorithms the exploratory actions must be part of the policy.

2.7.2 Policy gradient methods

Policy gradient methods are an important class of reinforcement learning algorithms built upon generic techniques for stochastic optimization. We first introduce the generic technique, and then discuss its properties when instantiated for the RL problem.

2.7.2.1 Log-derivative trick

Let $\mathcal{X}$ denote some measurable set and consider a parameterized family of probability distributions over $\mathcal{X}$. We have a parameter space Θ. Each $\theta \in \Theta$ induces a distribution $p_\theta \in \Delta(\mathcal{X})$. Now additionally consider a measurable function $f : \mathcal{X} \mapsto \mathbb{R}$ and the optimization problem

$$\underset{\theta \in \Theta}{\text{minimize}} \underset{x \sim p_\theta}{\mathbb{E}} f(x).$$

It is natural to attempt to solve this optimization problem with gradient descent (introduced in § 2.4.4) over θ. However, without strong restrictions on the form of f and the p_θ, it is generally impossible to obtain analytic derivatives for this objective. More importantly, there are many situations where we do not know the full description of f, but only have a zeroth-order oracle access. This means we can query the value of $f(x)$ for any $x \in \mathcal{X}$, but we cannot compute the gradient $\nabla f(x)$. (For example, evaluating f involves interacting with the real world.) In such situations, we can apply the following trick.

Theorem 2.7.1. *If p_θ is a parameterized family of probability distributions where the probability density exists and is differentiable with respect to the parameter θ, and other sufficient technical conditions are met (L'ecuyer, 1990, and references therein), then*

$$\nabla_\theta \left[\underset{x \sim p_\theta}{\mathbb{E}} f(x) \right] = \underset{x \sim p_\theta}{\mathbb{E}} \left[\nabla_\theta \log p_\theta(x) f(x) \right]. \tag{2.12}$$

Proof. Assuming sufficient conditions hold to apply Leibniz's rule, we have

$$\begin{aligned}
\nabla_\theta \left[\mathop{\mathbb{E}}_{x \sim p_\theta} f(x) \right] &= \nabla_\theta \int_{x \in \mathcal{X}} p_\theta(x) f(x) \mathrm{d}x \\
&= \int_{x \in \mathcal{X}} \nabla_\theta p_\theta(x) f(x) \mathrm{d}x \\
&= \int_{x \in \mathcal{X}} \frac{p_\theta(x)}{p_\theta(x)} \nabla_\theta p_\theta(x) f(x) \mathrm{d}x \\
&= \int_{x \in \mathcal{X}} p_\theta(x) \nabla_\theta \log p_\theta(x) f(x) \mathrm{d}x \\
&= \mathop{\mathbb{E}}_{x \sim p_\theta} \left[\nabla_\theta \log p_\theta(x) f(x) \right].
\end{aligned} \tag{2.13}$$

□

After applying Theorem 2.7.1, the gradient may then be approximated by a finite sample as

$$\nabla_\theta \mathop{\mathbb{E}}_{x \sim p_\theta} f(x) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log p_\theta(x_i) f(x_i), \tag{2.14}$$

where each $x_i \sim p_\theta$. To compute this expression, we require the following:

- Ability to sample from p_θ.
- Ability to compute the log-density gradient $\nabla_\theta \log p_\theta(x)$.
- Zeroth-order oracle for f.

and we **do not** require the following:

- Ability to sample from any other distribution over $\mathcal{X}$.
- First-order oracle for $\nabla_x f$.

2.7.2.2 Policy gradient algorithm

Now we instantiate the log-derivative trick for the RL problem. Suppose our policy class Π is parameterized: we have a parameter space $\Theta \subseteq \mathbb{R}^d$ and a joint parameter/state policy $\varpi : \Theta \times \mathcal{X} \mapsto \Delta(\mathcal{U})$. We further assume that for all of $\Theta \times \mathcal{X}$ the output of ϖ admits a Radon-Nikodym derivative (density function), denoted by $\varpi(\cdot|x, \theta) : \mathcal{U} \mapsto \mathbb{R}_{\geq 0}$, and that $\varpi(\cdot|x, \theta)$ is differentiable with respect to θ. Using trajectory notation (§2.5.2), let

$$R(\tau) = \sum_{t=h}^{H-1} r(x_t, u_t).$$

Let π_θ denote the partial application of ϖ for the parameter θ. We can then state the RL objective as

$$\underset{\theta}{\text{maximize}} \underset{\tau \sim \boldsymbol{\tau}_{\pi_\theta}}{\mathbb{E}} [R(\tau)]. \tag{2.15}$$

We may perform gradient descent on the objective (2.15) by applying Theorem 2.7.1, yielding

$$\nabla_\theta \underset{\tau \sim \boldsymbol{\tau}_{\pi_\theta}}{\mathbb{E}} [R(\tau)] = \underset{\tau \sim \boldsymbol{\tau}_{\pi_\theta}}{\mathbb{E}} \left[\nabla_\theta \log \boldsymbol{\tau}_{\pi_\theta}(\tau) R(\tau)\right]. \tag{2.16}$$

The $R(\tau)$ term is obtained as a direct effect of sampling τ from the MDP. The first term expands to

$$\begin{aligned}
\nabla_\theta \log \boldsymbol{\tau}_{\pi_\theta}(\tau) &= \nabla_\theta \log \left(\mu(x_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|x_t) P(x_{t+1}|x_t, u_t) \right) \\
&= \nabla_\theta \left(\log \mu(x_0) + \sum_{t=0}^{H-1} \log \pi_\theta(u_t|x_t) + \log P(x_{t+1}|x_t, u_t) \right) \\
&= \nabla_\theta \log \mu(x_0) + \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t|x_t) + \nabla_\theta \log P(x_{t+1}|x_t, u_t) \\
&= \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t|x_t).
\end{aligned} \tag{2.17}$$

Critically, the initial state distribution μ and the transition dynamics P do not appear in the final line. This means it is possible to approximate the gradient of the RL objective (2.15) only by interacting with

the MDP using π_θ and evaluating $\nabla_\theta \log \pi_\theta(\cdot|\cdot)$ on the sampled actions. Putting it all together, we see the typical form known as REINFORCE (Williams, 1992):

$$\mathbb{E}[\hat{g}] = \nabla_\theta \mathop{\mathbb{E}}_{\tau \sim \boldsymbol{\tau}_{\pi_\theta}} [R(\tau)], \quad \hat{g} = \left(\sum_{t=0}^{H-1} r_t \right) \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t|x_t) \right). \tag{2.18}$$

This can be approximated by a finite sample of interactions with the MDP and used to perform gradient descent. In general the RL objective is not convex, so the convergence rate guarantees for gradient descent presented in §2.4.4 do not apply. As we can observe from the sampling distribution in (2.18), policy gradient methods are on-policy algorithms (§2.7.1).

The gradient estimator $\hat{g}$ is unbiased, but in practice it can have very high variance. If a sample $\hat{g}$ that is far from $\mathbb{E}[\hat{g}]$ is used in gradient descent, it can potentially cause a drastic and undesirable change in π_θ, and thus also in $\boldsymbol{\tau}_{\pi_\theta}$. Various tricks have been proposed to reduce variance, for example subtracting an action-independent "baseline", also known as a control variate (Greensmith et al., 2004), and using a trust region (Schulman et al., 2015).

Another important limitation of policy gradient algorithms is that they depend on the stochasticity of the policy π_θ to explore the MDP. It is critical that θ be initialized to a policy with high action entropy and that the action distribution does not collapse. One popular method to (heuristically) achieve the latter is entropy regularization, where a bonus is added to the reward that encourages high-entropy conditional action distributions and/or state visitation distributions. Entropy regularization can also yield policies more robust to changes in the MDP dynamics (Eysenbach and Levine, 2021).[†]

By contrast, off-policy RL algorithms and can use exploration procedures that are independent of the current policy iterate. Exploration based on the principle of optimism in the face of uncertainty is especially powerful in off-policy algorithms, where it can deliver sample complexity bounds unavailable under naive

[†]The discussion of Eysenbach and Levine (2021) also provides a good survey of other motivations and results for entropy regularization.

(ϵ-greedy) exploration (Jin et al., 2018). Recently, optimism in the face of uncertainty has been adapted to policy gradient methods (Agarwal et al., 2020).

2.8 Control theory paradigms

Traditionally, the field of control theory is concerned with MDP-like settings in which the state space $\mathcal{X}$ is (some subset of) $\mathbb{R}^n$ and a reasonable model of the transition dynamics P is available. The model is usually either derived from physics or estimated from interacting with the real system.

In the case of physics-derived models there are often parameters that must be supplied, for example spring constants, masses, and so on. However, these parameters can often be identified using scientific experiments "outside" the MDP. For example, we may remove a spring from an assembly and measure its spring constant directly with a weight scale and a ruler.

2.8.1 System identification

The process of estimating a model or parameters from interacting with the real system (i.e. through the MDP "interface") is known as *system identification*. There are two components to system identification: some way to estimate ϕ from a state-action sequence $x_{0:T}, u_{0:T-1}$, and some scheme for choosing the inputs $u_{0:T-1}$ to make the estimation problem easy (or possible at all).

One way to formalize the system identification problem is as maximum-likelihood estimation (MLE) problem, where our goal is to select the ϕ under which the observed sequence was most probable:

$$\underset{\phi \in \Phi}{\text{maximize}} \quad \Pr(x_{1:T} | u_{0:T-1}; x_0, \phi). \tag{2.19}$$

This is equivalent to maximizing the logarithm of the likelihood. Taking the logarithm and using the Markov assumption, we have

$$\log \Pr(x_{0:T}|u_{0:T-1};\phi) = \log \mu(x_0|\phi) + \sum_{t=1}^{T} \log P(x_t|x_{t-1},u_{t-1};\phi).$$

Assuming the whole-family dynamics density $P(\cdot|x,u;\phi)$ is known and is differentiable with respect to ϕ, the maximum-likelihood objective (2.19) is also differentiable with respect to ϕ, so the MLE problem is amenable to numerical optimization. On the other hand, if $P(\cdot|x,u;\phi)$ is unknown or is only accessible via black-box sampling (i.e. a non-differentiable simulator), the problem (2.19) is no longer easy to optimize.

An alternative is to simply learn a function that maps $(x_{0:T}, u_{0:T-1})$ directly to an estimate of ϕ. The field of deep learning (§2.11) provides powerful function classes such as recurrent and convolutional neural networks that are well-suited to representing such a function. Our work in Chapter 3 implements this idea.

2.8.1.1 Persistence of excitation

Persistence of excitation refers to a sufficient condition on input signals to ensure that the solution of the system identification problem (2.19) converges to the true ϕ in the limit of infinite data. For linear systems, it is possible that inputs from a stabilizing linear feedback controller can fail to be persistently exciting. In other words, the goals of 1) minimizing some optimality criterion and 2) correctly identifying ϕ may be in conflict with each other. In this dissertation we are not concerned with the precise definition of persistent excitation for linear systems, but the overall concept is a key motivation for the methods we propose in Chapter 3.

2.8.2 Control with known model

Many published results in control theory provide guarantees under the assumption that an error-free dynamics model is available. In general, there is no reason to believe that these guarantees will be preserved

if the controller is deployed on a system different from the model. In certain cases it is safe, such as the fully observable LQR problem (§ 2.9.1.6) with uncertainty in the input matrix (Safonov and Athans, 1977). On the other hand, introducing partial observability to this same setting can be catastrophic: the famous note of Doyle (1978) gave an example of a system where an arbitrarily small model error renders a nominally optimal (controller, observer) pair unable to even stabilize the system. Recently, this fragility has appeared in complex nonlinear settings as the so-called "sim-to-real" problem in reinforcement learning, which we discuss in Chapter 3.

2.8.3 Robust control

The *robust control* paradigm mainly concerns scenarios in which we know that the true MDP ϕ belongs to some small set Φ, for example some small neighborhood of uncertainty about a nominal system. We seek a policy $\pi : \mathcal{X} \mapsto \mathcal{U}$ that is in some sense adequate for all $\phi \in \Phi$ simultaneously. Traditionally the sense of "adequate" is based on stabilization or system norms. Certain models of uncertainty in the dynamics parameters are also equivalent to robustness against disturbance input signals (Dullerud and Paganini, 2000). In the case of linear dynamical systems, robust control is a mathematically deep topic with connections to functional analysis and convex optimization. Robust control is also highly focused on partially observable settings. Robust methods often consider the full closed loop of state estimator and controller, rather than studying each in isolation. Historically, robust control developed in response to issues of the type observed by Doyle (1978), as mentioned previously.

2.8.4 Gain scheduling

Gain scheduling describes a technique in which some auxiliary information to identify ϕ is available. A classic example is an airplane autopilot, where the altitude and airspeed strongly affect the ability of the aerodynamic control surfaces to exert moments about the airplane's rotational axes. The altitude and

airspeed can be directly measured with sensors. Therefore, a controller is designed in which the attitude feedback gains depend on the altitude and airspeed sensor inputs (Åström and Wittenmark, 2013). The term "gain" should be understood to mean "control policy" rather than anything more specific like gains for a PID controller.

2.8.5 Adaptive control

In its broadest interpretation, the term *adaptive controller* refers to any controller that changes its behavior based on the (possibly unknown) value of ϕ while it is running. However, in common usage it is restricted to cases where ϕ cannot be directly measured, thus excluding the gain-scheduling controllers. In particular, the so-called *self-tuning regulator* directly attempts to estimate ϕ using system identification techniques (§2.8.1), and then uses the estimate of ϕ to synthesize an optimal controller.

The self-tuning regulator paradigm is quite broad, but in practice it most commonly refers to cases where:

- Structural knowledge of Φ can be exploited in the process of identifying ϕ. For example, in linear dynamical systems, recursive least-squares estimation is especially efficient.
- Synthesizing an optimal policy and/or selecting an optimal action conditioned on the estimate of ϕ can be done in real time.

In Chapter 3 of this dissertation, we present a method for synthesizing adaptive controllers in systems that violate both of these assumptions.

2.8.6 Model-predictive control

Model-predictive control (MPC) is a class of controllers based on solving optimization problems quickly in a real-time loop, rather than storing the control policy as some sort of closed-form function. MPC is

not a subset or disjoint of any of the categories listed above. Known-model, robust, gain-scheduled, and adaptive versions of MPC all exist.

Suppose a deterministic discrete-time dynamics model $x' = f(x, u)$ is known and the current system state is x_t. Model-predictive control poses the optimization problem

$$\underset{u_t,\ldots,u_{t+K-1}}{\text{minimize}} \quad \sum_{\tau=0}^{K-1} \ell_\tau(x_{t+\tau}, u_{t+\tau}) + \ell^R(u_{t:t+K-1}) \tag{2.20}$$

$$\text{subject to} \quad x_{t+\tau+1} = f(x_{t+\tau}, u_{t+\tau}) \; \forall \tau \in \{0, \ldots, K-1\} \tag{2.21}$$

$$u_{t+\tau} \in \mathcal{U} \; \forall \tau \in \{0, \ldots, K-1\}, \tag{2.22}$$

where each ℓ_τ is a task-determined loss (for example, tracking a goal trajectory in x), the term ℓ^R imposes regularization (for example, penalizing large changes in consecutive values of u), and the horizon K is a user-chosen hyperparameter. This is a simple MPC formulation. It is also possible to add state constraints and more complex constraints on the set of admissible input sequences.

2.8.6.1 Receding horizon

Solving the optimization problem (2.20) yields a sequence of inputs $u_t^\star, \ldots, u_{t+K-1}^\star$. Typically, only the first input $u_t^\star$ is actually supplied to the system. Then the sensors and estimation system produce the actual value of x_{t+1}, which may not be equal to $f(x_t, u_t^\star)$ due to modeling error and/or disturbances. We then immediately solve another MPC problem starting from the true x_{t+1}, which is used to decide the input u_{t+1}.

This pattern gives MPC a favorable structure for iterative optimization methods. If the optimization algorithm requires an initial guess, then the guess

$$u_{t+1}^\star, \ldots, u_{t+K-1}^\star, u_{t+K-1}^\star$$

constructed by shifting the previous solution and duplicating the last input is often a nearly optimal for the new optimization problem. Therefore, the optimization algorithm may require far fewer iterations than it would require if starting from scratch, for example with all $u_i = \mathbf{0}$.

2.8.6.2 Linear MPC

An important special case of MPC is linear dynamics (§2.9) with convex losses ℓ_t and convex regularization ℓ^R where each $\mathcal{U}_\tau$ is a convex set. The linear dynamics imply that $x_{t+\tau}$ is an affine function of $u_{t:t+\tau-1}$. Therefore, due to properties of convex functions (§ 2.4.1), the problem (2.20) is a convex optimization problem. If the ℓ_t and ℓ^R are quadratic, then the problem is similar to a linear-quadratic regulator (§2.9.1.6), but MPC can easily handle convex constraints on both actions and states, which cannot fit into the standard LQR framework.

2.9 Linear dynamical systems and control

In this section we collect the background material on linear control theory needed to state our results. We only consider *time-invariant* linear systems, where the coefficient matrices of the recurrences or differential equations are constant with respect to time. For brevity, we will take the term "linear system" and the acronym "LTI system" to mean "linear time-invariant system".

Linear dynamical systems are defined for both discrete and continuous time. We give a brief tour of discrete-time linear control first. We will follow with an abbreviated discussion of continuous-time linear control.

2.9.1 Discrete time

2.9.1.1 Autonomous system

A discrete-time deterministic linear dynamical system follows the recurrence

$$x_{t+1} = Ax_t \tag{2.23}$$

for $x_t \in \mathbb{R}^n,\ A \in \mathbb{R}^{n \times n}$.

2.9.1.2 Stability

Perhaps the most important concept for linear dynamical systems is that of *stability*. For the purposes of this overview, a discrete-time linear dynamical system is stable if

$$\lim_{t \to \infty} \|x_t\| = 0$$

for all initial states x_0 under some norm $\|\cdot\|$. Far more general definitions of stability exist for broader classes of systems (Hinrichsen and Pritchard, 2005). The state at time t is given by

$$x_t = A^t x_0.$$

Recalling that $\rho(A) = \max\{|\lambda| : \lambda \in \Lambda(A)\}$, the spectrum of A controls how quickly the states decay towards zero:

Theorem 2.9.1. *(Hinrichsen and Pritchard, 2005, Lemma 3.3.19) If $\rho(A) < e^{\omega}$, then there exists $M > 0$, depending on ω, such that*

$$\left\|A^t\right\|_{2,2} \leq M e^{\omega t}, \quad t \in \mathbb{N}.$$

If $\rho(A) < 1$ then $\omega < 0$, so we see $\left\|A^t\right\|_{2,2}$ goes to zero. This condition is necessary as well as sufficient: if λ, ν is an eigenvalue/eigenvector pair of A and $|\lambda| \geq 1$, then $\left\|A^t\nu\right\| = \left\|\lambda^t\nu\right\| = \lambda^t\|\nu\| \neq 0$. Therefore, the discrete-time linear dynamical system (2.23) is stable if and only if $\rho(A) < 1$. A matrix with $\rho(A) < 1$ is called *Schur* (not to be confused with the well-known Schur complement of a block matrix).

2.9.1.3 Linear control systems

A linear control system has inputs in addition to state. A discrete-time linear control system follows the recurrence

$$x_{t+1} = Ax_t + Bu_t + w_t, \tag{2.24}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are arbitrary matrices and $w_t \in \mathbb{R}^n$ is a disturbance from the environment. In the *deterministic* case, we have $w_t = \mathbf{0}$ for all t. In the *stochastic* case, we assume w_t is sampled i.i.d. in time according to some probability distribution. Often the distribution is Gaussian: $w_t \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$ for some $\Sigma_x \succeq \mathbf{0}$. If w_t is allowed to be adversarial, sophisticated analysis beyond the scope of this dissertation is required.

2.9.1.4 Controllability

It is clear from the trivial example $B = \mathbf{0}$ that not all linear control systems can be driven to arbitrary states. This is formalized by the notion of *controllability*.

Definition 2.9.2. The discrete-time linear control system (2.24) is *controllable* if, for any initial state $x_0 \in \mathbb{R}^n$ and goal state $x_g \in \mathbb{R}^n$, there exists $H \in \mathbb{N}$ and an input sequence $(u_0, \ldots, u_{H-1}) \in (\mathbb{R}^m)^H$ such that $x_H = x_g$.

Controllability rank conditions Controllability of discrete-time systems can be reduced to simple linear-algebraic questions about the matrices A and B. To see this, we now expand the expression for x_H and introduce the notation

$$\begin{aligned} x_H &= A^H x_0 + A^{H-1}Bu_0 + \cdots + Bu_{H-1} \\ &\triangleq A^H x_0 + \underbrace{\begin{bmatrix} A^{H-1}B & A^{H-2}B & \cdots & AB & B \end{bmatrix}}_{L_c(H)} \underbrace{\begin{bmatrix} u_0^\top & u_1^\top & \dots & u_{H-2}^\top & u_{H-1} \end{bmatrix}^\top}_{u_{0:H-1}}. \end{aligned}$$

From this, we can see that a system can be driven to an arbitrary goal x_g in H timesteps if and only if the system of linear equations

$$L_c(H)u_{0:H-1} = x_g - A^H x_0$$

has a solution in the variables $u_{0:H-1} \in \mathbb{R}^{mH}$. Since x_g is arbitrary, this is true for all x_g and x_0 if and only if $\operatorname{rank}(L_c(H)) = n$. It can be shown that if $\operatorname{rank}(L_c(n)) < n$, then $\operatorname{rank}(L_c(H)) < n$ for all $H > n$, so $\operatorname{rank}(L_c(n)) = n$ is a necessary and sufficient condition for controllability. The matrix $L_c(n)$ is known as the *controllability matrix*. For simplicity, we use the notation $L_c = L_c(n)$.

Since the controllability matrix L_c is "wide" in general, it is sometimes convenient to instead consider the matrix

$$W_c = L_c L_c^\top = \sum_{i=0}^{H-1} A^i B B^\top (A^\top)^i,$$

known as the *controllability Gramian*. Because $\operatorname{rank}(W_c) = \operatorname{rank}(L_c)$, the condition $\operatorname{rank}(W_c) = n$ is also necessary and sufficient for controllability. Note that these rank conditions are often numerically unstable to compute in practice, and more appropriate alternative tests exist.

2.9.1.5 Stabilizing controllers

If we use a linear control policy $u_t = Kx_t$ for some $K \in \mathbb{R}^{m \times n}$, then the system dynamics become

$$x_{t+1} = Ax_t + BKx_t + w_t.$$

In the case where $w_t = \mathbf{0}$, the closed-loop system has the linear dynamics

$$x_{t+1} = (A + BK)x_t.$$

Therefore, our results on stability of linear systems apply: If $\rho^+(A + BK) < 1$, then our control policy stabilizes the system. It can be shown that such a stabilizing K exists if and only if the system is controllable.

2.9.1.6 Linear quadratic regulator (LQR)

If we impose a quadratic cost on a linear dynamical system, we obtain the heavily-studied *linear quadratic regulator* (LQR) problem setup. More specifically, we specify the cost

$$J = \sum_{t=0}^{H} c(x_t, u_t) \triangleq \sum_{t=0}^{H} x_t^\top Q x_t + u_t^\top R u_t, \tag{2.25}$$

where $Q \succeq 0$ and $R \succ 0$. We sometimes use the notation $c_t = c(x_t, u_t)$. The horizon H may be either finite or infinite. It is most natural to think of (2.25) as a penalty on being far from the zero state (first term) and expending control energy (second term). To adapt LQR problems to the standard reward-based MDP formulation, we set $r(x, u) = -c(x, u)$.[‡]

[‡]Many results in reinforcement learning theory depend on boundedness of the reward. These results cannot be directly applied to the LQR setting.

Optimal controller for LQR It can be shown (Lancaster and Rodman, 1995) that, for the infinite-horizon case, the policy that minimizes Equation (2.25) is linear feedback of the form $u_t = Kx_t$, which leads to quadratic total cost, i.e. $J = x_0^\top P x_0$ for some $P \succeq 0$, where P is the unique maximal positive semidefinite solution to the *discrete-time algebraic Riccati equation* (DARE):

$$P = A^\top PA - A^\top PB(R + B^\top PB)^{-1}B^\top PA + Q.$$

The optimal controller is then

$$K = -(R + B^\top PB)^{-1}B^\top PA.$$

We emphasize that this controller is optimal over the class of *all* feedback controllers, not only the linear feedback controllers.

2.9.1.7 Outputs and State Estimation

In many real-world dynamical systems it is not possible to measure every dimension of the state x. For example, many small unmanned aircraft contain an inertial measurement unit (IMU) capable of measuring acceleration and gravity forces and a global positioning system (GPS) receiver capable of measuring position, but no instrument capable of measuring velocity. For linear systems, we consider a sensor or set of sensors that measure some linear mapping of the state

$$y = Cx$$

for $C \in \mathbb{R}^{p \times n}$.

2.9.1.8 Observability

Analogous to the notion of controllability, a LTI system must satisfy certain conditions to ensure that it is possible to estimate x from a history of inputs u and outputs y. For a generic (possibly nonlinear) discrete-time system, we define *observability* by the condition that there exists $H \in \mathbb{N}$ and an input sequence $(u_0, \dots, u_{H-1})$ such that it is possible to determine the initial state x_0 from the known inputs and the observations $y_0, \dots, y_H$.

For a LTI system, expanding the expressions for the observations yields

$$\begin{aligned} \begin{bmatrix} y_0 \\ y_1 \\ y_3 \\ \vdots \\ y_H \end{bmatrix} &= \begin{bmatrix} C & & & & \\ CA & CB & & & \\ CA^2 & CAB & CB & & \\ \vdots & \vdots & & \ddots & \\ CA^{H-1} & CA^{H-2}B & & \ddots & CB \end{bmatrix} \begin{bmatrix} x_0 \\ u_0 \\ u_1 \\ \vdots \\ u_{H-1} \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{H-1} \end{bmatrix}}_{L_o(H)} x_0 + \text{const.} \end{aligned} \tag{2.26}$$

Therefore, determining x_0 reduces to solving a linear system. We see that for LTI systems, the feasibility of determining x_0 does not depend on the chosen inputs at all. (This is not true in general for nonlinear systems.) Instead, it depends only on the coefficient matrix $L_o(H)$. Although this system will generally be overdetermined, in a perfectly modeled and noiseless system one can show that it will always have an exact solution if and only if $\operatorname{rank}(H) = n$ for sufficiently large H. As in the controllability case, we can show that must hold for some $H \leq n$, so the matrix $L_o \triangleq L_o(n)$ is called the *observability matrix*.

2.9.1.9 Luenberger observer

Suppose we wish to design an online recursive estimator of the current state x_t. Let $\hat{x}_t$ denote the current estimate of x_t. A controller (outside our influence) supplies an input u_t, and we observe $y_t = Cx_t$. It is reasonable to impose that our update for $\hat{x}_{t+1}$ takes the form

$$\hat{x}_{t+1} = A\hat{x}_t + Bu_t + f(y_t, \hat{x}_t).$$

Now we additionally impose that $f(y_t, \hat{x}_t) = L(y_t - C\hat{x}_t)$, that is, we update the estimate with a linear function of the "residual" $y_t - C\hat{x}_t$, or the difference between the true measurement and the measurement predicted by our estimate. We then examine the dynamics of the estimate error $e_t = \hat{x}_t - x_t$:

$$\begin{aligned} e_{t+1} &= \hat{x}_{t+1} - x_{t+1} \\ &= A\hat{x}_t + Bu_t + L(y_t - C\hat{x}_t) - (Ax_t + Bu_t) \\ &= A(\hat{x}_t - x_t) + L(y_t - C\hat{x}_t) \\ &= A(\hat{x}_t - x_t) + L(Cx_t - C\hat{x}_t) \\ &= (A + LC)e_t. \end{aligned} \tag{2.27}$$

The error dynamics are linear, so the stability condition for linear systems implies that, if $\rho^+(A+LC) < 1$, then $e_t \to \mathbf{0}$ as $t \to \infty$. We note that $\rho^+(A+LC) = \rho^+(A^\top + C^\top L^\top)$, matching exactly the form $A+BK$ we encounter when describing stabilizing controllers (§ 2.9.1.5). Therefore, linear estimator synthesis is mathematically equivalent to linear control synthesis, with $(A^\top, C^\top)$ playing the same role as (A, B).

2.9.1.10 Kalman filter

For control synthesis, we discussed both arbitrary stabilizing controllers and optimal controllers according to the LQR cost criterion. We might ask if a similar notion of an optimal observer exists for state estimation.

The LQR quadratic cost expressed a tradeoff between regulating the state to zero and consuming energy with control inputs. For state estimation, a quadratic cost would also capture the goal of driving the estimate error to zero, but imposing a cost on the magnitude of the update $L(y_t - C\hat{x}_t)$ does not have a clear meaning.

It turns out that a useful "cost" will arise if we consider the stochastic case, where the dynamics are given by

$$x_{t+1} = Ax_t + Bu_t + w_t, \qquad y_t = Cx_t + v_t,$$

where the dynamics noise w_t and sensor noise v_t are both zero-mean Gaussian random variables with covariances Σ_x and Σ_y respectively.

Whereas the Luenberger observer only maintained an estimate $\hat{x}$ of the state, in the stochastic case we will maintain a Gaussian belief distribution:

$$x_t \sim \mathcal{N}(\hat{x}_t, P_t).$$

From the properties of Gaussian distributions, after supplying the input u_t, the belief distribution should change according to

$$\hat{x}'_t = A\hat{x}_t + Bu_t, \qquad P'_t = AP_tA^\top + \Sigma_x.$$

This is known as the *propagation step*. The key question is how to update the belief distribution after observing the output y_t. This can be treated as a Bayesian inference problem where the current belief is the prior, the measurement y_t is the evidence, and the updated belief distribution is the posterior. Instead of the common maximum a priori (MAP) update, we will seek an update that minimizes the trace of the updated belief covariance. Deriving the update from probabilistic principles without making the assumption that

we update μ with a linear function of the measurement residual—as we did for the Luenberger observer—is complex. It is beyond the scope of this dissertation to derive the update, but it takes the form

$$
\begin{aligned}
e_t &= y_t - C\hat{x}'_{t-1} \\
S_t &= CP'_{t-1}C^\top + \Sigma_y \\
K_t &= P'_{t-1}C^\top S_t^{-1} \\
\hat{x}_t &= \hat{x}'_{t-1} + K_t e_t \\
P_t &= (I - K_t C)P'_{t-1},
\end{aligned}
$$

where e_t is the measurement residual, S_t is the covariance of e_t according to the current belief distribution, K_t is the so-called Kalman gain, and $(\hat{x}_t, P_t)$ are the updated belief distribution parameters. This is called the *update step*.

Although we have coupled the propagation and update steps notationally, they need not be coupled. For instance, it is common in practice to have several propagation steps per update step.

2.9.2 Continuous time

All of the discrete-time definitions and theorems stated above have analogues for continuous-time systems. We quickly review these analogues here. In the remainder of this dissertation we will never need to refer to a discrete-time and continuous-time linear control system simultaneously, so we overload all notation and rely on context to disambiguate.

2.9.2.1 Autonomous system

A continuous-time deterministic linear dynamical system follows the ordinary differential equation (ODE)

$$\dot{x}(t) = Ax(t) \tag{2.28}$$

for $t \in [0, \infty),\ x(t) \in \mathbb{R}^n,\ A \in \mathbb{R}^{n \times n}$.

2.9.2.2 Stability

Again stability is defined by

$$\lim_{t \to \infty} \|x(t)\| = 0$$

for all initial states $x(0)$. The state at time t is given by

$$x_t = e^{tA} x_0,$$

where the matrix exponential is defined by the usual power series. For continuous-time LTI systems $\dot{x} = Ax$, we have that

Theorem 2.9.3. *(Hinrichsen and Pritchard, 2005, Lemma 3.3.19) If $\rho^+(A) < \omega$, then there exists $M > 0$ such that*

$$\lim_{t \to \infty} \left\| e^{tA} \right\|_{2,2} \leq M e^{\omega t}, \quad t \in \mathbb{R}_{\geq 0}.$$

By a similar argument as the discrete case using an eigenvalue-eigenvector pair, the system $\dot{x} = Ax$ cannot be stable if $\rho^+(A) \geq 0$. Therefore, the continuous-time linear dynamical system (2.28) is stable if and only if $\rho^+(A) < 0$. A matrix with $\rho^+(A) < 0$ is called *Hurwitz*.

2.9.2.3 Linear control systems

A continuous-time deterministic linear control system has the state and input spaces $\mathcal{X} = \mathbb{R}^n,\ \mathcal{U} = \mathbb{R}^m$, and follows the ODE

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{2.29}$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ are arbitrary matrices. (We will not consider continuous-time stochastic systems in this dissertation.)

2.9.2.4 Controllability

The continuous-time linear control system (2.29) is controllable if, for any initial state $x_0 \in \mathbb{R}^n$, goal state $x_g \in \mathbb{R}^n$, and time horizon $T > 0$, there exists a continuous function $u(t) : [0, T) \mapsto \mathbb{R}^m$ for which the system reaches state x_g at time T.

It turns out that the exact same definition and rank condition of the controllability matrix as in the discrete-time case (§2.9.1.4) imply controllability in the continuous case. We will not prove this here. The controllability Gramian is defined differently, but it also has full-rank controllability condition.

2.9.2.5 Linear-quadratic regulator

The infinite-horizon continuous-time linear quadratic regulator (LQR) problem is specified by the cost from a particular starting state $x(0) \in \mathbb{R}^n$ by

$$J_{x(0)} = \int_0^\infty \left[x(t)^\top Q x(t) + u(t)^\top R u(t) \right] \mathrm{d}t, \tag{2.30}$$

where $Q \succeq 0$ and $R \succ 0$. Again, it can be shown (Lancaster and Rodman, 1995) that the optimal controller is linear feedback of the form $u(t) = Kx(t)$, which leads to the optimal total cost taking the quadratic form $J_{x(0)} = x(0)^\top P x(0)$, where $P \succeq 0$ is the unique maximal positive semidefinite solution to the *continuous-time algebraic Riccati equation* (CARE):

$$A^\top P + PA - PBR^{-1}B^\top P + Q = \mathbf{0}. \tag{2.31}$$

The optimal controller is then

$$K^\star = -R^{-1}B^\top P. \tag{2.32}$$

2.9.3 Canonical forms

The Laplace transform, its discrete-time counterpart the z-transform, and transfer functions are an important tool for frequency-domain analysis of LTI systems. It is beyond the scope of this dissertation to introduce them. In this section we briefly summarize one tool from frequency-domain analysis that we will use to synthesize a state-space system with specified open-loop eigenvalues.

From the perspective of frequency-domain analysis, a single-input, single-output (SISO) control system is completely characterized by the poles and zeros of its transfer function. However, the mapping from state-space systems to transfer functions is many-to-one. Given a transfer function, one of several particularly useful state-space realizations is the *controllable canonical form* or *reachable canonical form* (Åström and Murray, 2010). If the transfer function is given by

$$\frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1}s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n},$$

then the controllable canonical form (CCF) realization is given by

$$A = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_n & \cdots & -a_2 & -a_1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} b_n - a_n b_0, b_{n-1} - a_{n-1} b_0, \ldots, b_1 - a_1 b_0 \end{bmatrix}.$$

To realize a state-space system for a particular set of open-loop eigenvalues $\lambda_1, \ldots, \lambda_n$, we compute the denominator coefficients $a_{1:n}$ by expanding the characteristic polynomial $(\lambda_1 - x)(\lambda_2 - x)\cdots(\lambda_n - x)$.

2.9.4 Pole placement

Pole placement is a non-optimal control method. A pole placement algorithm $\mathscr{P}$ takes the dynamics matrices A and B and a set of desired closed-loop eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$, and returns a matrix K such that $\Lambda(A + BK) = \{\lambda_1, \ldots, \lambda_n\}$. If the system (A, B) is controllable, then such a K always exists (Sontag, 2013). It is beyond the scope of this dissertation to discuss algorithms for computing pole placement, but many exist.

2.10 Statistical learning

Statistical learning is an umbrella term for machine learning settings in which the learner observes a complete dataset in one instant, as opposed to in some online or interactive protocol.

2.10.1 General statistical learning problem

In the most general statistical learning setting we have some abstract data space Z, an abstract function class $\mathcal{F}$, a loss function $\ell : \mathcal{F} \times Z \mapsto \mathbb{R}$, and an unknown distribution $\mathcal{D} \in \Delta(Z)$. Our ideal goal is to solve the optimization problem

$$\underset{f \in \mathcal{F}}{\text{minimize}} \quad L(f) \triangleq \mathop{\mathbb{E}}_{z \sim \mathcal{D}} \left[\ell(f, z)\right]. \tag{2.33}$$

However, we are only given a data set of N items $D = (z_1, \ldots, z_N) \sim \mathcal{D}^N$, that is, each z_i is sampled i.i.d. from $\mathcal{D}$. A natural learning algorithm to consider is solving the *empirical risk minimization* (ERM) problem:

$$\underset{f \in \mathcal{F}}{\text{minimize}} \quad \sum_{i=1}^{N} \ell(f, z_i). \tag{2.34}$$

The mathematically rich field of statistical learning theory tells us that for certain forms of ℓ, such as binary classification and least-squares regression, a function class $\mathcal{F}$ is learnable if and only if it is learnable by ERM. However, this is not true for the fully general setting (Shalev-Shwartz et al., 2010). In benign settings the learning rate guarantee usually takes the form

$$\sup_{\mathcal{D}\in\Delta(\mathcal{Z})}\left(\mathop{\mathbb{E}}_{D\sim\mathcal{D}^N}[L(f_{\text{ERM}}(D))]-\inf_{f\in\mathcal{F}}L(f)\right)\leq O\left(\frac{1}{\sqrt{N}}\right)$$

In other words, as the sample size goes to infinity, there is no distribution $\mathcal{D}$ that can "trick" the ERM algorithm. The rate $1/\sqrt{N}$ is typical.

2.10.2 Supervised learning

A particularly important class of statistical learning problems is *supervised learning*, in which we have an input space X, an output space Y; we have $Z = X \times Y$; our function class is belongs to $\mathcal{F} \subseteq Y^X$; and our loss function takes the form

$$\ell(f,(x,y)) = \ell_s(f(x),y),$$

in which $\ell_s : Y \times Y \mapsto \mathbb{R}$ is some kind of "comparison" function, often satisfying the properties of a semimetric (§2.2.1). One common example is least-squares regression, where $Y = \mathbb{R}^d$ and we have

$$\ell(f,(x,y)) = \|f(x)-y\|_2^2. \tag{2.35}$$

Another common example classification, where $Y = \{-1,+1\}$ (or any set of two distinct elements) and we have

$$\ell(f,(x,y)) = \mathbb{I}_{[f(x)=y]}. \tag{2.36}$$

A fundamental difference between the least-squares loss (2.35) and the classification loss (2.36) is that only the former is differentiable. The non-differentiability of the classification loss requires extra care in both algorithm design and analysis.

2.10.3 Gradient-based optimization

In this work, we are only concerned with regression problems. The differentiability of the least-squares loss (2.35) lends favorable structure to the problem. Now suppose our function class $\mathcal{F}$ is *parameterized*: we have a parameter space $\Theta \subseteq \mathbb{R}^n$ and a function $F : \Theta \times X \mapsto Y$. We denote the partial application with respect to θ by

$$f_\theta(x) = F(\theta, x)$$

and let $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$. If ℓ_s is differentiable with respect to its first argument and F is differentiable with respect to θ, then the resulting empirical risk minimization objective

$$L_{\text{ERM}}(\theta) = \sum_{(x,y)\in D} \ell_s(f_\theta(x), y) \tag{2.37}$$

is differentiable with respect to θ. This allows us to attack the supervised learning ERM problem with the tools of continuous optimization. In particular, this problem has the following characteristics:

- When the parameter space Θ is high-dimensional, second order optimization algorithms are computationally difficult (Nocedal and Wright, 2006). Second-order algorithms are those that require solving a linear system of the form $Hw = b$ for w, where $H = \nabla_{\theta\,\theta} L_{\text{ERM}}$ is the Hessian of L_{ERM} at some point in Θ. The canonical example is applying Newton's method to the system of equations $\nabla_\theta L_{\text{ERM}} = \mathbf{0}$.
- When the data set size N is large, computing the gradient $\nabla_\theta L_{\text{ERM}}$ or even simply evaluating L_{ERM} can be computationally expensive.

- The parameter space is often unconstrained, i.e. $\Theta = \mathbb{R}^n$.

These characteristics favor the optimization method of *stochastic gradient descent*. In its simplest form, stochastic gradient descent is based on the observation that

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathrm{Uniform}(D)} [\nabla_\theta \ell_s(f_\theta(x), y)] = \nabla_\theta L_{\mathrm{ERM}}(\theta),$$

which follows from linearity of expectation. This suggests we follow Algorithm 1.

Algorithm 1 Stochastic gradient descent for statistical learning

Require: Dataset D, learning rate $\eta > 0$, batch size $1 \leq K \ll N$, initial guess θ

1: **repeat**
2: Sample $(x_1, y_1), \ldots, (x_K, y_K) \sim \mathrm{Uniform}(D)^K$.
3: Compute $\hat{\nabla}_\theta L_{\mathrm{ERM}} = \dfrac{1}{K} \displaystyle\sum_{i=1}^{K} \nabla_\theta \ell_s(f_\theta(x_i), y_i)$.
4: Update $\theta \leftarrow \theta - \eta \hat{\nabla}_\theta L_{\mathrm{ERM}}$.
5: **until** stopping criteria met.

In Algorithm 1, the batch size K balances the variance of the gradient estimate $\hat{\nabla}_\theta L_{\mathrm{ERM}}$ against computational cost. In practice, a size $K > 1$ is almost always used to take advantage of parallel processing. The uniform sample in line 2 of Algorithm 1 is sometimes replaced with another minibatch selection method. For example, we might randomly permute the dataset and then sample each length-K chunk of the permutation in order to improve memory locality. (One complete pass through the dataset in this manner is sometimes called an *epoch*.) It is also common to apply the accelerated gradient descent methods discussed in §2.4.4 to stochastic gradient descent.

2.11 Neural networks

The name *Neural network* is a broad descriptor for function approximation classes built up from recursive composition of linear maps alternating with simple, usually elementwise, nonlinear maps. The latter is often called a *nonlinearity* for short. The pair of a linear map followed by a nonlinearity is referred to as a

layer. This structure is loosely inspired by structures observed in animal brains. Usually the only learnable parameters are those of the linear maps in each layer.

Neural networks have a long history in machine learning (Rosenblatt, 1958), but have recently grown in prominence. Their growth has been influenced by several factors coming together:

- Graphics processing units (GPUs) growing from fixed-function devices into general-purpose massively parallel computers.
- Software libraries for reverse-mode automatic differentiation, also known as *backpropagation*, enabling loss gradient evaluation for arbitrarily complex functions.
- Special-purpose neural network architectures designed to exploit structural properties of high-dimensional data such as images and sequences.
- Huge datasets of user-generated content from the Internet e.g. ImageNet (Deng et al., 2009).

Neural networks have become the *de facto* parametric nonlinear function class for many applications.

2.11.1 Neural network architectures

2.11.1.1 Nonlinearities

Common nonlinearities used in neural networks are:

- *Sigmoid*:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- *Hyperbolic tangent*:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- *Rectified linear unit (ReLU)*:

$$\mathrm{relu}(x) = \max\{x, 0\}$$

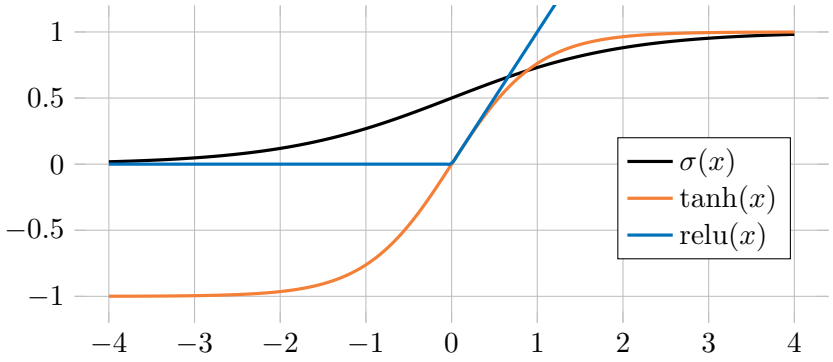


Figure 2.2: Typical nonlinearities used in neural networks.

In the remainder of this chapter, we will occasionally overload the notation $\sigma(x)$ to denote an arbitrary elementwise nonlinearity instead of the sigmoid function specifically. This notational convention is common in the literature on neural networks. Often the final layer lacks the nonlinearity. To avoid complicating definitions, we therefore allow these generic nonlinearities to include the identity map as well.

2.11.2 Fully connected neural network

A fully connected neural network with n layers is a function of the form

$$f(x) = f_n(f_{n-1}(\cdots f_1(x)\cdots)),$$

where each layer f_i takes the form

$$f_i(y) = \sigma_i(W_i y + b_i),$$

in which σ_i is a nonlinearity or an identity function, W_i is a "weight" matrix of appropriate size, and b_i is a "bias" vector of appropriate size. Using identity for the final output σ_n is common.

2.11.3 1D convolutional neural network

One-dimensional convolutional neural networks are used to process sequence data. Consider an input space $X = \mathbb{R}^{n \times d}$, where n is the sequence length and d is the per-step dimensionality. A *1D convolution of width* k maps an input $(x_1, \ldots, x_n) \in X$ to a new sequence $(y_1, \ldots, y_{n-k+1})$ according to

$$y_i = W_0 x_i + W_1 x_{i+1} + \cdots W_{k-1} x_{i+k-1} + b,$$

in which d' is the output per-step dimensionality, each W_i is a $d' \times d$ matrix, and $b \in \mathbb{R}^{d'}$ is a bias vector.

Now consider a single scalar entry of a single vector in the output sequence, for example the first entry of the vector y_1. It is a linear combination of all kd scalar values of the input subsequence $x_1, \ldots, x_k$. Many common and important signal processing operations can be expressed as 1D convolutions, including smoothing, derivative estimation, and finite impulse response filters. Note that a convolution of width k reduces the output sequence length by $k - 1$.

A 1D convolutional neural network consists of more than one 1D convolution applied sequentially with nonlinearities in between. After each layer, the "receptive field" of each entry in the sequence grows. We then optimize all of the matrices W and the biases b simultaneously. We have left out many other details such as strides and pooling, which can be found in any tutorial material on convolutional neural networks.

2.11.4 Recurrent neural network

The term *recurrent neural network* (RNN) is used to describe a broad variety of differentiable function classes suitable for representing sequence-to-sequence mappings. In contrast to 1D convolutional networks (§ 2.11.3), RNNs can represent sequence-to-sequence mappings with infinite impulse response. A generic RNN is a discrete-time dynamical system of the form

$$x_{t+1} = f(x_t, u_t), \quad \hat{y}_t = g(x_t, u_t)$$

where x is the internal state, u is the input, $\hat{y}$ is the output and f and g are the dynamics and output functions respectively. The RNN model is parameterized by some real vector θ. Both f and g are differentiable with respect to their arguments and the parameter θ. The particular form of the functions f and g must be carefully chosen to maximize expressiveness while preserving desirable properties for optimization.

It is important to emphasize that the RNN internal state x is an abstract quantity. For example, if we optimize a RNN to evaluate arithmetic expressions, values of x in the learned RNN might contain similar information to the stack in a parser. If we optimize a RNN to imitate the input-output mapping of a Markovian physical system, x might end up being an approximately invertible function of the true system state. However, any such structure would arise implicitly via the optimization objective and would not be enforced.

Depending on the application, the optimization objective for a RNN may be a function of the full output sequence $y_{0:T}$ or only the final output y_T. The RNN parameter θ is typically optimized with stochastic gradient descent (SGD) using backpropagation, so the gradient of the output loss is allowed to flow through the recursive applications of f. This allows the RNN to learn dynamics where the effect of an input does not appear until many time steps later. RNNs have achieved state-of-the-art results on many sequence

modeling tasks, even though the objective is nonconvex and SGD may converge to suboptimal local minima (Lipton, 2015).

2.11.4.1 Long short-term memory

The *long short-term memory* (LSTM) network is a particular kind of recurrent neural network—that is, a particular form of the functions f and g—designed to have favorable properties for optimization with gradient-based methods (Hochreiter and Schmidhuber, 1997). The LSTM is the *de facto* standard RNN architecture due to these properties (Lipton, 2015). The LSTM partitions the internal state x into two vectors $h \in \mathbb{R}^n$ and $c \in \mathbb{R}^n$. The functional form is most clearly expressed using the intermediate values $i, f, q, p \in \mathbb{R}^n$ as

$$\begin{bmatrix} i \\ f \\ q \\ p \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \circ \left(W \begin{bmatrix} h_t \\ u_t \end{bmatrix} + b \right), \quad c_{t+1} = f \odot c_t + i \odot p, \quad h_{t+1} = q \odot \tanh(c_{t+1}), \tag{2.38}$$

where σ denotes the sigmoid nonlinearity (§ 2.11.1.1), the symbol $\odot$ denotes elementwise multiplication, and $\circ$ denotes elementwise function composition. The learnable parameters are the matrix $W \in \mathbb{R}^{4n \times (n+m)}$ and the vector $b \in \mathbb{R}^{n+m}$, where m denotes the dimensionality of u.

In some applications of LSTMs the state h is used directly as the output, i.e. $g(h, c, u) = h$. In other settings, it is appropriate for g itself to be a learned mapping. For example, if the output space is very low-dimensional, it may be useful to select a higher dimensionality of h and learn a projection $y = Ph$, where P is a "wide" matrix. The input can also be pre-processed, replacing u_t in (2.38) with a learned function of u_t. For example, if the input space is very high-dimensional (e.g. a one-hot encoding of English words), it may be useful to use a projection here too.

It is also possible to form a multi-layer LSTM, in which subsequent layers take the internal states h of previous layers as inputs.

Chapter 3

Reinforcement Learning for Universal Policies

In this chapter we work in the MDP family framework described in § 2.6. We propose a method to give reinforcement-learned policies the ability to adapt to unknown dynamics at test time. Our primary motivation is the simulation-reality gap in robotics, where a policy optimized in simulation performs poorly in the real system that the simulator approximates. Adaptivity is also useful for deploying a pre-trained policy into a wide set of real-world scenarios, for example if designing an autonomous driving system that works on both sports cars and passenger vans.

Our method merges ideas from the self-tuning regulator paradigm in adaptive control (§ 2.8.5) with the generality and representation learning ability of deep reinforcement learning. As discussed in § 2.8.5, traditional techniques for adaptive control generally exploit structure in the MDP family Φ to render the processes of estimating ϕ and adapting the policy computationally efficient. Unfortunately, many systems of interest in robotics do not admit analytical solutions for the system identification and policy synthesis problems. In particular, with regard to policy synthesis there are numerous problems in robotics and AI for which the only known satisfactory methods require hours of computation, such as some of the RL successes listed in Chapter 1.

The work presented in this chapter was originally published in Preiss et al. (2018). The presentation has been substantially revised for clarity, but the results have not.

For system identification, if the whole-family system dynamics $P(\cdot|x, u; \phi)$ is only accessible through a non-differentiable simulator, then the only computationally feasible method for online system identification may be the function approximation approach (§ 2.8.1). By using deep neural networks for system identification, we can take advantage of their representation learning ability. We learn an embedding function that maps the system identification parameters in a family of MDPs Φ (see § 2.6) into an abstract embedding space. The embedding retains enough information to support policy specialization to particular MDPs within Φ while potentially being easier to identify from state-action trajectories. Our framework also includes an observability-promoting reward that encourages the policy to balance the task goal with behavior that aids system identification.

Our simulation experiments demonstrate desirable properties of the learned embedding in a toy example, but show only a modest improvement in ultimate optimality of the multi-system policy in a more complex example. The experimental results raise fundamental questions about reinforcement learning and multi-system control that motivate our theoretical work in subsequent chapters.

3.1 Related work

Although policies trained with reinforcement learning (RL) can achieve state-of-the-art performance on some tasks, they are often brittle and fail to generalize beyond the training environment, even when the differences are small (Zhang et al., 2018). An important instance of this problem is sim-to-real transfer for robotics. RL algorithms can often learn policies that exploit bugs in physics simulators, or vision-based policies that work with synthetic rendered inputs but not with real images.

A natural first step to deal with such brittleness is to add randomization to the simulator. In the the *domain randomization* approach, we optimize the objective

$$\underset{\pi \in \Pi}{\text{maximize}} \; \mathop{\mathbb{E}}_{\phi \sim \zeta} J_\phi(\pi), \tag{3.1}$$

where $\Pi \subseteq \Delta(\mathcal{U})^{\mathcal{X}}$ is a policy class, ζ is some distribution over Φ, and J_ϕ is the RL objective for ϕ (encapsulating the horizon and discount). Domain randomization during training can improve robustness (Antonova et al., 2017; Zhu et al., 2018; van Baar et al., 2019; Sadeghi and Levine, 2017; Tobin et al., 2017), but it is limited by its assumption that a single policy can perform adequately in all possible test domains. In this sense, domain randomization is similar in spirit to robust control.

Other deep learning techniques in the robustness spirit include merging ensembles of policies (Parisotto et al., 2016; Teh et al., 2017), adversarial perturbations of state or observations (Pinto et al., 2017; Huang et al., 2017), and learning robust feature spaces (Higgins et al., 2017; Bousmalis et al., 2016).

Fine-tuning (Rusu et al., 2017) and (some) meta-learning approaches (Finn et al., 2017) assume there will be an opportunity to collect data from the test domain and update the policy. In this work, we seek a policy that specializes to a novel test environment rapidly, without using significant data or computational effort at test time. Recurrent neural network policies are also capable of fast adaptation to unobserved quantities but require more complex reinforcement learning algorithms (e.g. Wierstra et al., 2007). Duan et al. (2016b) specifically evaluate RNN policies as a tool for adapting to different tasks, as opposed to more generic POMDP settings. Another possibility is augmenting the MDP states with "memory" and adding actions so that the policy can write to the memory states (Peshkin et al., 1999).

In our setting, we assume that dynamics parameters are known during training but unknown under test. Under the same assumptions, the method most similar to our is that of Yu et al. (2017), who train a policy in simulation that observes ϕ and a neural network to estimate ϕ from a state-action trajectory. Our method builds upon this by adding two contributions: 1) a learned embedding space that represents the system dynamics parameters in a form that is both useful and easy to identify, and 2) an observability reward that encourages the agent to maximize identification accuracy.

3.2 Problem statement

We consider reinforcement learning in a family of Markov decision processes using the notation defined in § 2.6, with variations only in the transition dynamics P. We assume that the system space Φ can be parameterized by a real vector and identify Φ with the parameter, i.e. we assume $\Phi \subseteq \mathbb{R}^{d_\phi}$. We present our results for the case of finite-horizon MDPs with horizon H, but our method is also applicable to infinite-horizon MDPs.

Our learning protocol is separated into training and testing phases. During training, we have access to a simulator for each $\phi \in \Phi$. During testing, the environment selects a particular $\phi \in \Phi$, but does not reveal its choice to us. Our goal is to take near-optimal actions in the MDP ϕ despite not knowing the value of ϕ.

In cases where Φ is large and unstructured, for example the set of all linear dynamical systems or all finite MDPs, the testing phase of our learning protocol has no meaningful difference from the basic RL problem (§ 2.7). At the other extreme, if Φ represents a small neighborhood of uncertainty around a single nominal MDP, then we are in the robust control regime (§ 2.8.3) and it is reasonable to seek a *blind* policy $\pi : \mathcal{X} \mapsto \mathcal{U}$ that is nearly optimal for all $\phi \in \Phi$, for example by using the domain randomization approach (3.1). We consider the cases in between, where Φ is large but still highly structured. In such settings, no *blind* policy can perform adequately over the entirety of Φ. However, it is still possible to gain much sample efficiency at test time compared a generic RL algorithm by doing some kind of meta-learning or other preparation during the training phase.

3.3 Method

A natural approach in this protocol is to learn a policy $\pi : \mathcal{X} \times \Phi \mapsto \Delta(\mathcal{U})$ alongside a system identification function $\mathrm{id}_\phi : \mathcal{X}^{<\mathbb{N}} \times \mathcal{U}^{<\mathbb{N}} \mapsto \Phi$ that maps the history of past states and actions to an estimate of ϕ. At

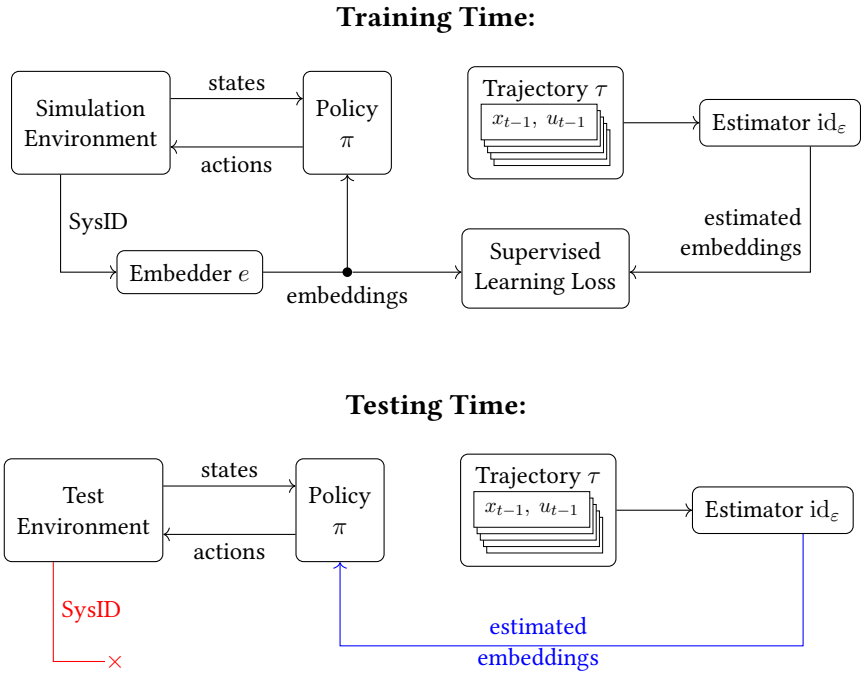


Figure 3.1: Overview of our method. At training time, correct dynamics parameters are available from the simulator. A mapping e from parameters to an abstract embedding space is learned, along with a module id_ε to identify the embedding value from a state-action trajectory τ. The policy is rewarded for behavior that improves system identification accuracy. At testing time, the true dynamics parameters are no longer known, and the estimated embeddings are input directly to the policy.

test time, we act with the policy $u_t \sim \pi\big(x_t,\ \mathrm{id}_\phi(x_{0:t}, u_{0:t-1})\big)$. This method was explored by Yu et al. (2017), in which the authors refer to π as a *universal policy*, and refer to their method as UP-OSI (universal policy with online system identification).

Our method addresses several hypothetical failure modes of UP-OSI. First, UP-OSI requires estimating every dimension of ϕ, even though some may be redundant, difficult to estimate, or unneeded to maximize reward. Second, behavior that maximizes reward in training may be suboptimal for system identification at test time. It is preferable to learn a behavior that balances the primary reward with a secondary goal of making the system identification task as easy as possible. For example, some adaptive control methods for linear systems require a persistently exciting input, but inputs from a stabilizing linear feedback controller may fail to be persistently exciting (§2.8.1.1).

We address the first concern by introducing a learned abstract representation $\mathcal{E} \subseteq \mathbb{R}^{d_\mathcal{E}}$ of the dynamics parameters. (The dimensionality $d_\mathcal{E}$ is a user-chosen hyperparameter.) During training, we learn an embedding function $e : \Phi \mapsto \mathcal{E}$ and a universal policy $\pi_\varepsilon : \mathcal{X} \times \mathcal{E} \mapsto \Delta(\mathcal{U})$ conditioned on an embedding value instead of the environment parameter ϕ. We simultaneously learn an identification function $\mathrm{id}_\varepsilon : \mathcal{X}^{<\mathbb{N}} \times \mathcal{U}^{<\mathbb{N}} \mapsto \mathcal{E}$ to estimate the embedding value from the past states and actions. Then, at test time, we act with the policy $u_t \sim \pi_\varepsilon\big(x_t,\ \mathrm{id}_\varepsilon(x_{0:t}, u_{0:t-1})\big)$.

We address the second concern by augmenting the main RL reward with a term penalizing estimation error of id_ε. This rewards behavior that makes estimating ε easier. Our method is illustrated in Figure 3.1.

3.3.1 Learning algorithms

Simple case: policy and identification decoupled We learn the embedding function e and the universal policy π_ε in an end-to-end fashion using a standard reinforcement learning algorithm. We reduce the universal policy optimization problem to a standard RL problem. First, we choose a distribution over environment parameters $\zeta \in \Delta(\Phi)$. For example, if Φ is bounded, we might choose ζ to be uniform. We then create an augmented MDP with the state space $\mathcal{X}' = \mathcal{X} \times \Phi$, the dynamics $P'((x, \phi), u) = P_\phi(x, u)$, and the initial state following the probability density function $\rho'((x, \phi)) = \rho(x)\zeta(\phi)$. Then, performing RL on our augmented MDP is equivalent to optimizing the objective

$$\underset{\pi_\varepsilon,\ e}{\text{maximize}} \qquad J_R(\pi_\varepsilon, e) \triangleq \mathop{\mathbb{E}}_{\phi\sim\zeta}\left[\mathbb{E}\sum_{t=0}^{H-1} r(x_t, u_t)\right], \tag{3.2}$$

where the inner expectation is with respect to $x_0 \sim \rho,\ u_t \sim \pi_\varepsilon(x_t, e(\phi)),\ x_{t+1} \sim P_\phi(x_t, u_t)$. To optimize the RL objective (3.2), we can use any "off-the-shelf" reinforcement learning algorithm that supports continuous state and action spaces and can optimize the composition of e and π_ε directly.

After RL is complete, we optimize the system identification function id_ε according to the objective

$$\underset{\mathrm{id}_\varepsilon}{\text{minimize}} \; J_{id}(\mathrm{id}_\varepsilon; \pi_\varepsilon, e) \triangleq \mathop{\mathbb{E}}_{\phi \sim \zeta} \left[\mathbb{E} \sum_{t=0}^{H-1} \|\mathrm{id}_\varepsilon(x_{0:t}, u_{0:t-1}) - e(\phi)\|_2^2 \right], \tag{3.3}$$

where the inner expectation is as in (3.2) with respect to the optimal policy obtained from RL. Because this expectation depends on the action distributions of π_ε, the optimal system identification function id_ε can be different for different policies.

Complex case: observability reward As discussed in § 3.3, the optimal policy for the objective (3.2) does not necessarily take actions that make system identification feasible. We can address this by adding a term to the RL objective that penalizes system identification error. Specifically, for a fixed system identification function id_ε, we modify the RL objective to

$$\underset{\pi_\varepsilon,\, e}{\text{maximize}} \quad J_O(\pi_\varepsilon, e; \mathrm{id}_\varepsilon) \triangleq \mathop{\mathbb{E}}_{\phi \sim \zeta} \left[\mathbb{E} \sum_{t=0}^{H-1} r(x_t, u_t) - \alpha \|\mathrm{id}_\varepsilon(x_{0:t}, u_{0:t-1}) - e(\phi)\| \right], \tag{3.4}$$

where $\alpha > 0$ is a user-chosen weight. We refer to the added term as the *observability reward*. Note that the value of the observability reward at time t is completely determined by the actions taken *before* time t. This is no different from the "credit assignment problem" already present in reinforcement learning, and is accounted for by the RL algorithm.

Our algorithm alternates between optimizing the policy and the system identification estimator. In an idealized setting, we would follow Algorithm 2. Note that on line 1 of Algorithm 2, we initialize the policy and embedding function *without* the observability reward. It would also be possible to initialize the system identification function first using some default policy, for example a policy that takes purely random actions independent of state and embedding value. However, we did not explore this possibility in our work.

Algorithm 2 Idealized algorithm

$\pi_\varepsilon^{(0)}, e^{(0)} \leftarrow \operatorname{argmax}\ J_R(\pi_\varepsilon, e)$ ▷ initialize policy for task reward only
for $i \in 1, \ldots, N$ **do**
  $\mathrm{id}_\varepsilon^{(i)} \leftarrow \operatorname{argmin}\ J_{id}(\mathrm{id}_\varepsilon; \pi_\varepsilon^{(i-1)}, e^{(i-1)})$ ▷ update sysID estimator for current policy
  $\pi_\varepsilon^{(i)}, e^{(i)} \leftarrow \operatorname{argmax}\ J_O(\pi_\varepsilon, e; \mathrm{id}_\varepsilon^{(i)})$ ▷ update policy for current sysID estimator
end for

We note a possible failure mode of this approach: If the observability reward weighting α is chosen to be too large, its contribution could dominate the RL reward. As $\alpha \to \infty$, one optimal solution would be for e and id_ε to both be constant functions. The hyperparameter α must be selected to ensure this does not happen.

In practice, it is computationally expensive to solve the optimization problems within each iteration of Algorithm 2. In our experiments, we instead alternate between one iteration of RL for (π_ε, e) followed by a one iteration of gradient descent for supervised learning of id_ε, as shown in Algorithm 3.

Algorithm 3 Practical algorithm

Initialize $\pi_\varepsilon,\ e,\ \mathrm{id}_\varepsilon$ randomly
for $i \in 1, \ldots, N$ **do**
  sample $\phi^1,\ \ldots,\ \phi^B$ i.i.d. from ζ
  collect trajectories $\tau^1,\ \ldots,\ \tau^B$ from π_ε for each ϕ^i
  perform one iteration of RL to optimize π_ε, e for J_O
  perform one iteration of SGD to optimize id_ε for J_{id} on the dataset $\{\tau^i\}_{i=1}^B$
end for

3.3.2 Implementation details

Function classes In our experiments, the universal policy π_ε is parameterized as a fully-connected neural network with 2 hidden layers of 128 units each, using ReLU nonlinearities (§2.11.1.1). The embedding function e is also a fully-connected neural network, containing one hidden layer of 128 units.

We employ a one-dimensional convolutional architecture for the identifier function id_ε, composed of three 1D-convolutional layers, each with 64 filters of width 3 and ReLU activation, followed by a single fully connected layer with 128 units, and a linear output layer. (See § 2.11.3 for more information on

1D-convolutional neural networks.) Convolutions in time match the intuition that differentiation of the state and action trajectories is often required to identify the underlying dynamics parameters in real-world physical systems. If the discrete-time dynamics are derived from integrating continuous-time dynamics, then the finite-difference operations used to approximately recover derivatives are naturally represented as convolutions in the time dimension. For example, if $x_1, x_2, \ldots$ is a sampling of a continuous signal with one unit of time per step, then convolution in time by the kernel $[-1, 1]$ approximates the signal's derivative, and the kernel $[1, -2, 1]$ approximates its second derivative.

Although we defined id_ε in § 3.3 as a function of all states and actions since the episode beginning, the composition of convolutions with a fully-connected layer (as opposed to e.g. convolution followed by taking the mean over time) implies a fixed-length window of state-action inputs. Therefore, we choose a window length K and supply only $x_{t-K:t}, u_{t-K:t-1}$ to id_ε. For steps when $t < K$, we define the initial states and actions in the window as zero vectors. In our experiments $K = 16$.

RL algorithm For all experiments, we use the entropy-regularized Soft Actor-Critic (SAC) reinforcement learning algorithm (Haarnoja et al., 2018). SAC is an off-policy algorithm where a stochastic policy is trained only with TD-learned value function estimates. We observed significantly higher rewards using SAC compared to the on-policy, Monte Carlo policy gradient algorithm PPO (Schulman et al., 2017). We conjecture that the use of TD-learning and a replay buffer is especially helpful in our scenario compared to single-environment training, since the replay buffer helps prevent "forgetting" about areas of the dynamics distribution p_Φ that have not been sampled recently. The replay buffer is used for RL but not to learn the system identification function id_ϕ, to ensure that the (nonstationary) training distribution of state-action trajectories for id_ϕ reflects the same policy behavior that will be observed at test time.

The value function estimators used by SAC are parameterized by fully-connected networks of identical structure to the policy. For *blind* policies, the value function networks are conditioned only on the observed state $x \in \mathcal{X}$ to emulate domain randomization approaches. For *plain* policies, they are conditioned on both

x and $\phi \in \Phi$ to emulate the setup of Yu et al. (2017). For *ours* policies, they are conditioned on x, ϕ, and $\varepsilon \in \mathcal{E}$. Although the embedding function e could theoretically be optimized via the least-squares *value learning* loss of SAC, we optimize it only via the policy gradient.

3.4 Experiments

In this section, we show desirable properties of our learned embedding space $\mathcal{E}$ using a toy example and compare the performance of our architecture against several baselines on a more complicated benchmark problem in robotic locomotion.

3.4.1 Point-Mass Environment

This low-dimensional system allows us to visualize learned embeddings. A 2D point mass with position $p \in \mathbb{R}^2$ follows dynamics

$$\ddot{p} = gu - d\dot{p},$$

where the action $u \in [-1, 1]^2$ is a bounded force input, the parameter $g \in \mathbb{R}_{\neq 0}$ is a gain factor, and the parameter $d \in \mathbb{R}_{\geq 0}$ is a damping factor. The goal is to push the point towards the origin, specified by reward $r = -\|p\|_2$.

To convert these second-order continuous-time dynamics into a discrete-time dynamical system suitable for standard reinforcement learning algorithms, we apply the state space transformation

$$x(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}, \quad \dot{x}(t) = \begin{bmatrix} v(t) \\ gu(t) - dv(t) \end{bmatrix},$$

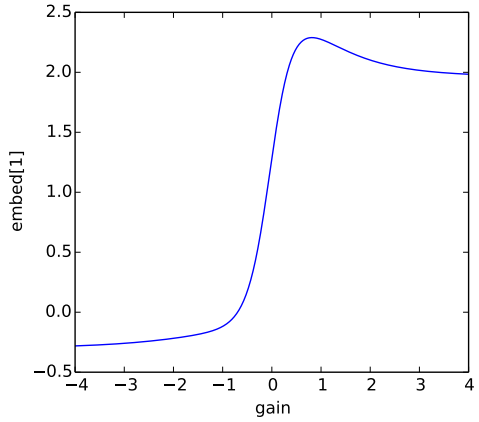


Figure 3.2: Point-mass example system: Learned mapping e from gain g to one dimension of the embedding space $\mathcal{E}$.

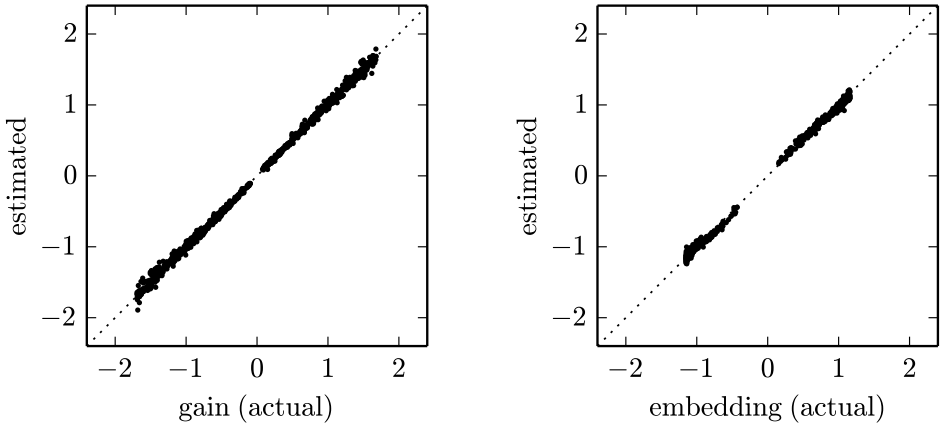


Figure 3.3: Point-mass example system: Comparison of actual vs. estimated gain g (left) and one dimension of the embedding ε (right). Embedding separates parameters requiring disjoint behavior into clusters.

where $v \in \mathbb{R}^2$ is the velocity, and discretize with simple forward Euler integration over the time interval Δ_t to yield the discrete-time system

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix}, \quad x_{t+1} = x_t + \begin{bmatrix} \Delta_t v_t \\ \Delta_t (g u_t - d v_t) \end{bmatrix}.$$

The dynamics parameter ϕ is the gain factor $g \in \pm[0.175, 1.75]$. Due to the unknown sign of g, a policy that ignores the value of ϕ cannot possibly perform well in this environment.

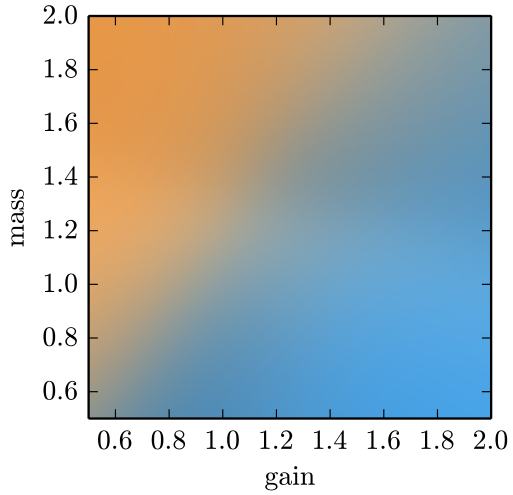


Figure 3.4: Point-mass system with redundant mass parameter: Multidimensional visualization of learned embedding. Axes represent the gain and mass parameters g, m. Colors represent both dimensions of the learned embedding ε: one dimension controls the amount of orange; other dimension controls the amount of blue. Note that lines of constant gain/mass ratio are mapped to approximately the same color. Plot restricted to positive gain domain for clarity.

For our experiments, we select the embedding dimension $d_{\mathcal{E}} = 2$, even though the true space Φ is only one-dimensional, to emphasize that making ε higher-dimensional than necessary is not detrimental. We show results in Figures 3.2 to 3.4.

In Figure 3.2, we show one output dimension of the learned embedding function $e : \Phi \mapsto \mathcal{E}$ after applying our method to the point-mass system. We observe that the embedding "squashes" all positive gains to approximately the same value, and does the same for negative gains.

In Figure 3.3, we evaluate the accuracy of the learned system identification functions in both our method and in the *plain* method. The scatter plots compare the ground truth values of ϕ (resp. ε) against the values estimated by id_ϕ (resp. id_ε). We again see the effect of the "squashing" done by id_ε, resulting in two easily distinguished clusters.

To demonstrate how the learned embedding can eliminate redundant parameters in Φ, we construct a version of the point-mass environment with a mass parameter $m > 0$ alongside the gain g, such that $\ddot{p} = (g/m)u$. The dynamics of all combinations with the same ratio g/m are indistinguishable. This implies that in the *plain* method, the system identification loss will always be large.

In Figure 3.4, we visualize both dimensions of our learned embedding for this version of the point-mass environment. Our framework learns an embedding where all (g, m) combinations with similar g/m ratios map to a similar embedding value.

3.4.2 Half-Cheetah environment

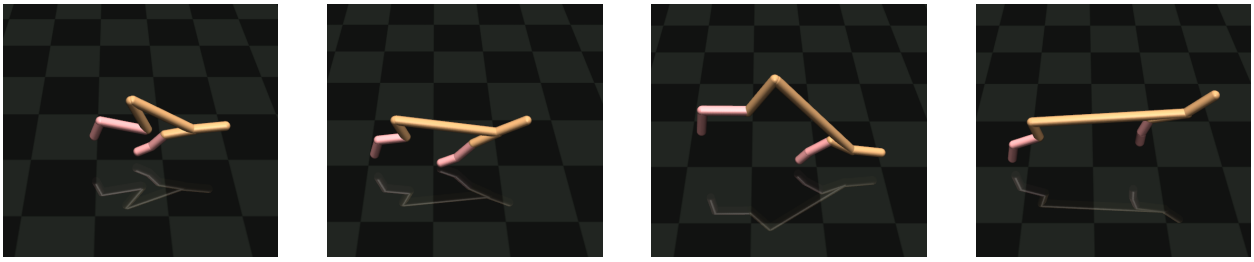


Figure 3.5: Variations of Half-Cheetah environment produced by randomization of kinematic and dynamic properties.

As a more complex benchmark task, we demonstrate results on the *Half-Cheetah* planar locomotion environment from the OpenAI Gym (Brockman et al., 2016). We randomize the length of the torso and the lengths of the three segments in each leg. For each of the six rotational joints, we randomize the beginning and end of the angular range, the gear ratio of the actuator, a velocity-proportional damping constant, and the stiffness of a virtual spring pulling the joint towards a specified resting angle. In total, 37 parameters are randomized.

For joint ranges, limits on either side are shifted by $\omega \sim \mathrm{Uniform}(-0.3, 0.3)$ radians from nominal. All other parameters must take nonnegative values, and some should not be too close to zero. To ensure this, we multiply the nominal value of those parameters by a random nonnegative ratio β^p where $p \sim \mathrm{Uniform}(-1, 1)$. The parameter $\beta > 1$ controls the amount of randomness. For example, if $\beta = 2$, then $1/2 \leq \beta^p \leq 2$. In our experiments, $\beta = 1.75$, which was the largest value we could use without generating too many overly difficult configurations. Some examples of the randomized half-cheetahs are shown in Figure 3.5.

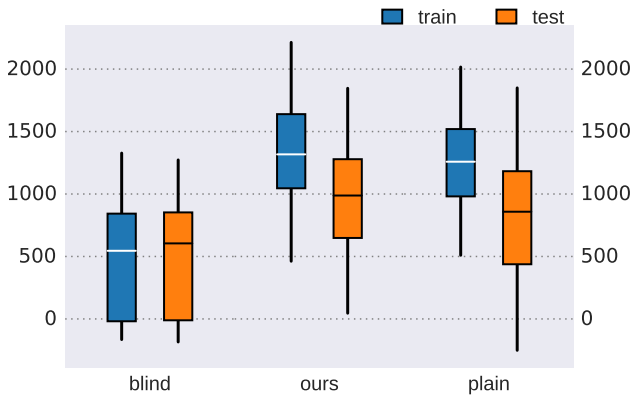


Figure 3.6: Box plot of training and test reward for *blind*, *ours*, and *plain* policies in randomized half-cheetah environment. Distributions are over the random sample of environments. Whiskers indicate full extent (min/max) of rewards.

Due to the architecture of the MuJoCo physics simulator used in this environment (Todorov et al., 2012), it is not practical to sample new random dynamics parameters ϕ for each training iteration. Instead, we construct a "universe" of 256 models initially, and sample a new "universe" at test time.

We select that the embedding space $\mathcal{E}$ is 8-dimensional. We did not observe significant sensitivity to this choice.

Results are shown in Figure 3.6. Variance of rewards is large due to the randomized dynamics. (For example, a random variant with strong damping will require more control effort to reach the same speed.) The *blind* policy fails to achieve high rewards because it is unable to specialize its behavior to the dynamics parameters. Naturally, it also does not suffer a performance loss on the test set. The *plain* policy, conditioned on ϕ instead of ε, achieves similar training rewards but suffers negative rewards in some test environments, while *ours* does not.

In current experiments, we have not yet observed a significant effect of our observability reward term, so we do not include its results in Figure 3.6.

3.5 Discussion

Our experiments raise several questions. First, we observed that our learned embedding space $e : \Phi \mapsto \mathcal{E}$ offered only modest improvement over directly identifying ϕ. This may be because identifying ϕ for the half-cheetah environment is not difficult, despite its high dimensionality. Also, we hypothesized that our embedding would be useful for cases where two $\phi, \phi' \in \Phi$ are indistinguishable, but if they are indistinguishable, then the adaptive policy does not need to differentiate them. Therefore, if the system identification module only outputs ϕ, the least-squares system identification loss will be nonzero but the policy reward will still be high.

Second, we did not observe any effect of adding our observability reward. The simplest explanation is that optimal behavior for the task reward already yields good observability as a side effect.

Third, a result we observed during our experiments that we did not report in § 3.4 because it was tangential to the main work: a performance gap between the universal policy and the average performance of individual "expert" policies trained for a single $\phi \in \Phi$. We investigate this phenomenon in a simplified setting in §3.6.

3.6 Simplified experiment: Universal policy versus experts

In this section we investigate the behavior of RL with a "universal policy" independent of any concern about system identification. We eliminate the testing phase from our protocol and focus purely on the performance during the training phase. Without the need to identify ϕ in the training phase, the system embedding and observability reward proposed in §3.3 are no longer useful, so we eliminate them from our experiments. Therefore, we are working in the same setup as Yu et al. (2017).

We compare the performance of a multi-system "universal policy" neural network against the aggregate performance of a collection of single-system policies. Recalling that ζ is some distribution over Φ, we sample a fixed set of N systems

$$\Phi_s = \phi_1, \ldots, \phi_n \sim \zeta^N.$$

For each $\phi_i \in \Phi_s$, we train an "expert" policy π_i to optimize the RL objective for ϕ_i. We terminate the RL algorithm for π_i after a fixed number T of interactions with the MDP ϕ_i. We also train a multi-system policy π_{multi}, without using any embedding or observability reward, in the augmented MDP as described in § 3.3.1 with the system distribution $\zeta = \text{Uniform}(\Phi_s)$. Note that this distribution is uniform over the sample, not over all of Φ. We train π_{multi} for NT interactions with the augmented MDP. Therefore, the universal policy π_{multi} and the collection of expert policies each consume the same total amount of environment interactions in aggregate.

As a simple test environment, we use a linearization of the "planar quadrotor" (Singh et al., 2021). The system is linearized about its hover state. We randomize the mass and moment of inertia parameters, sampling from a log-uniform distribution over two (decimal) orders of magnitude. In this system, $\mathcal{X} = \mathbb{R}^6$, $\mathcal{U} = \mathbb{R}^2$, and $\Phi \subseteq \mathbb{R}^2$. The reward is negative LQR cost (2.25) with Q and R identity matrices. We train all policies using the implementation of the PPO algorithm (Schulman et al., 2017) from the popular

reinforcement learning library `stable-baselines3` (Raffin et al., 2021). We leave all hyperparameters of the policy class and RL algorithm at their default value.

A goal of multi-system learning is to leverage shared structure between systems to improve sample efficiency. If a multi-system learning method is able to do this, then we expect to see that for some values of T,

$$\frac{1}{N}\sum_{i=1}^{N} J_{\phi_i}(\pi_i) \leq \frac{1}{N}\sum_{i=1}^{N} J_{\phi_i}(\pi_{\text{multi}}),$$

where $J_{\phi_i}(\pi)$ denotes the RL objective for system ϕ_i.

We visualize results for this experiment in Figure 3.7. To account for the different optimal policy costs for different systems, we plot the ratio of the learned policy's cost to the cost of the LQR-optimal linear controller for the infinite-horizon version of the problem. (Note that this normalization is performed only for plotting, not for the RL reward.) Learning curves for each random planar quadrotor are plotted individually. The training steps on the horizontal axis are per-system, so each step corresponds to an equivalent amount of training time for both the multi-system policy and the aggregated single-system policies.

We observe that the multi-system policy does initially achieve lower LQR cost for most systems. However, as training progresses, in most of the systems the "expert" policies eventually become more optimal than the multi-system policy. This suggests that the multi-system setup is able to exploit shared structure to some degree, but there is also some phenomenon inhibiting the multi-system setup from reaching full optimality. It also appears that in the multi-system setting, PPO converged by around 1 million steps (100k steps per system) but became unstable after around 2 million steps (200k steps per system).

Discussion We emphasize that this experiment is not a strong negative result against the "universal policy" architecture. It only shows that applying the simplest possible approach is not enough. Researchers

have proposed more sophisticated approaches for RL for continuous system spaces (for example, Kalashnikov et al., 2021). We include this experiment only as a motivating example for our subsequent theoretical inquiry into reinforcement learning and multi-system control.

There are two broad categories of explanations for this result. One possible explanation is that the augmented MDP is somehow unfavorable for the RL algorithm. Another possible explanation is the policy class. It might be hard to represent a good multi-system policy with a neural network. These two possibilities motivated us to seek a better understanding of RL algorithms and multi-system control at a theoretical level, which we explore in Chapter 5 and Chapter 6 respectively.

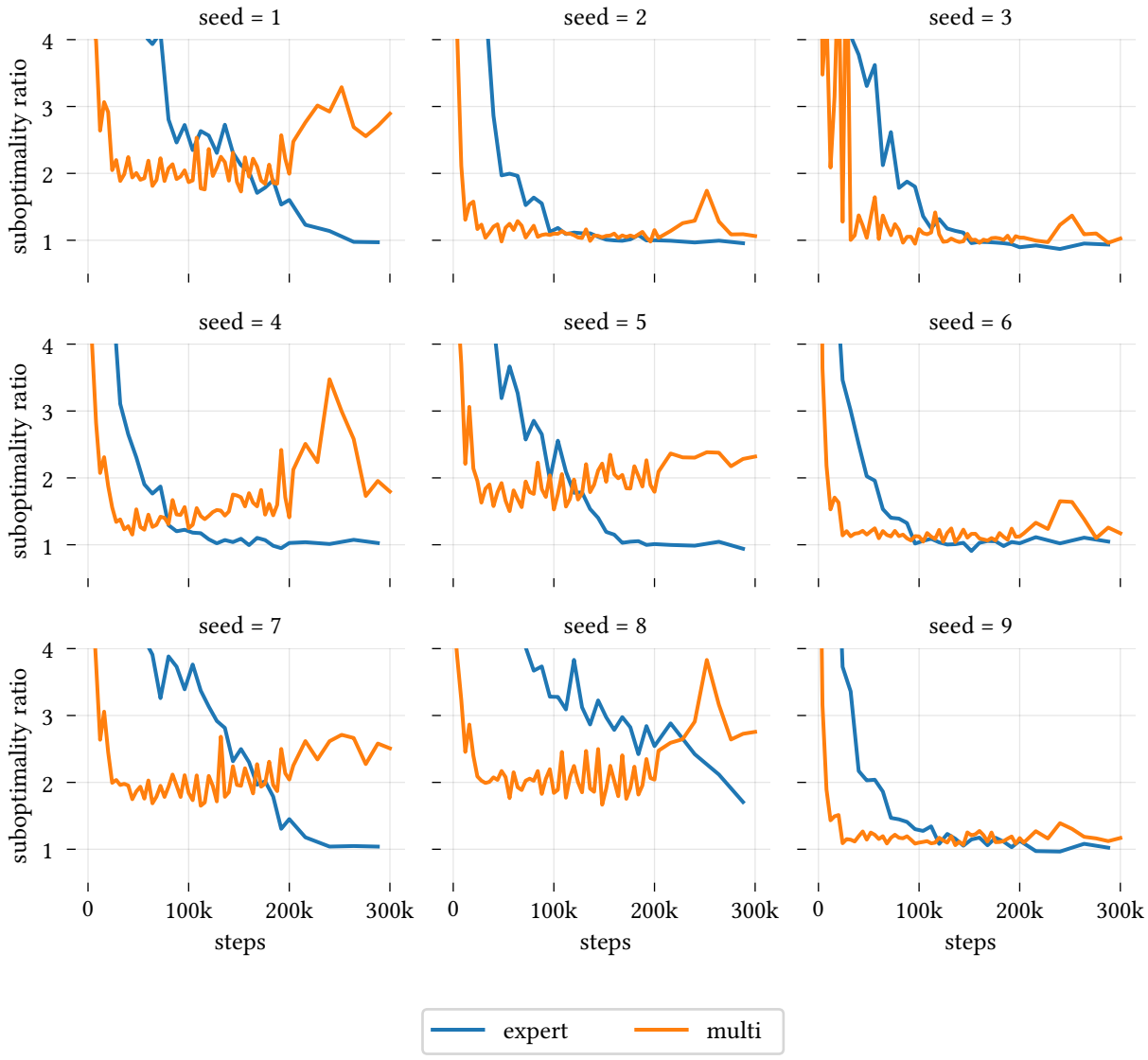


Figure 3.7: Learning curves for multi-system "universal policy" and single-system "expert" policies for nine random linearized planar quadrotor systems.

Chapter 4

Deformable Manipulation using Learned Models

In this chapter we investigate the middle ground between model-free reinforcement learning and traditional control methods based on physics-derived models. Our interest is motivated by the following observation: in MDPs for robotics the reward function is often known, or even designed by an engineer. The main source of complexity in many tasks is the environment dynamics. For example, in the half-cheetah simulated locomotion environment discussed in Chapter 3, the reward is simply forward velocity, which is one of the system states. Therefore, we consider using learning to account for complex dynamics but traditional estimation and control techniques to act optimally within the learned model. We consider the test case of deformable object manipulation, where first-principle physical models are usually high-dimensional.

In the present work, we assume that an engineer can design an experiment that sufficiently explores the state space. After collecting a dataset of trajectories from the real system, we train a recurrent neural network (RNN) to approximate its input-output behavior with a latent state-space model. Unlike finite element models and other physics-derived models of deformable objects, the RNN internal state is low-dimensional enough to enable realtime nonlinear control methods. We demonstrate a closed-loop control scheme with the RNN model using a standard nonlinear state observer and model-predictive controller.

The work presented in this chapter was originally published in Preiss et al. (2022). It is reproduced here with minimal changes.

We apply our method to track a highly dynamic trajectory with a point on the deformable object, in real time and on real hardware. Our experiments show that the RNN model captures the true system's frequency response and can be used to track trajectories outside the training distribution. In an ablation study, we find that the full method improves tracking accuracy compared to an open-loop version without the state observer.

4.1 Introduction and Related Work

Manipulating deformable objects represents a challenging area of robotics. In contrast to typical objects consisting of a single rigid body, deformable objects often admit limited control authority and have dynamics that are difficult to predict. At the same time, many objects of human interest are deformable. Safe, reliable robotic manipulation of these objects is critical for capable general-purpose robots (Arriola-Rios et al., 2020; Zhu et al., 2021).

As with many manipulation problems in robotics, there are multiple ways to model the dynamical behavior of the manipulated object. Broadly speaking, there are two classes of models: Fully data-driven methods based on an expressive function class with many parameters, and analytical methods based on a physical model with fewer parameters that can be identified from data.

Physics-based models of deformable objects have been studied extensively in science and engineering contexts, and many constitutive models have been described for various materials (Holzapfel, 2002). These models are continuous in space and time, and simulation on computer hardware requires a discretization strategy. Finite element methods (FEM) have been used for decades to solve continuum mechanics problems (Wriggers, 2008) and have shown promise for robotic control. Recent work with FEM modeling addresses dynamic control of deformable objects and soft robots with offline trajectory optimization (Zimmermann et al., 2021; Bern et al., 2019; Duenser et al., 2018; Li et al., 2021; Qiao et al., 2020; Heiden et al., 2021a). FEM is appealing as it admits extensive theoretical analysis (Barbič and Popović, 2008; Thieffry

et al., 2018). For real-world problems, it is possible to estimate the parameters of FEM meshes in a principled way from exteroceptive sensors common in robotics (Hahn et al., 2019). Alternative discretizations include the material point method (Sulsky et al., 1994; Hu et al., 2019), (extended) position based dynamics (Macklin et al., 2016), and meshless shape matching (Müller et al., 2005). Additionally, task-specific reduced states can be formulated, which accelerate control, perception and planning (Mcconachie and Berenson, 2018; McConachie et al., 2020).

Deformable objects can be complex to model, and for objects with resonant dynamics, seemingly minor errors in model assumptions or material parameter estimates can cause large deviations in dynamic behavior over time (Bern et al., 2020). For these reasons, purely data-driven methods are an appealing alternative to physics-based models. Machine learning methods have been explored in deformable manipulation (Mirza, 2020), for purely kinematic trajectory tracking (Bern et al., 2020), for cable-driven soft robot actuators (Bruder et al., 2019), for control of pneumatic deformable mechanisms (Gillespie et al., 2018), and for structures actuated by shape-memory alloys (Sabelhaus and Majidi, 2021). For scenarios where controllers can continuously interact with their environment to improve, model-based reinforcement learning has been proposed (Thuruthel et al., 2019). Other data-driven approaches studied in soft robot control include proper orthogonal decompositions (Tonkens et al., 2021).

In this work, we propose a data-driven method for modeling and trajectory-tracking control of a deformable object. Tracking fast trajectories serves as a benchmark for a method's ability to predict and account for dynamics in the manipulated object. Our method models the dynamics with a long short-term memory (LSTM) recurrent neural network (RNN) (Hochreiter and Schmidhuber, 1997; Lipton, 2015), trained in a standard sequence modeling setup using input-output trajectory data from a real physical system. The internal state of the learned RNN is not physically meaningful, but the RNN still forms a discrete-time dynamical system that is compatible with standard methods in state-space nonlinear control.

We therefore apply model-predictive control (Allgöwer and Zheng, 2012) and extended Kalman filtering methods (Kalman, 1960) to track the trajectory using the RNN model.

We implement our method to run online with a real robot, and use it to "draw in the air" along a fast trajectory with the free end of a pool noodle (long foam cylinder) held by the robot arm. To measure how well non-instantaneous dynamics are captured, we compare frequency response plots of the true system and the RNN model. To verify that a nonlinear observer helps compensate for model errors, we perform an ablation study comparing our method to an open-loop variant where the RNN model is assumed to be perfect. We find that the full closed-loop method reduces trajectory tracking error.

RNNs have a long history within the broader field of nonlinear state-space system identification with partial observability (Nelles, 2001). Such models can be used as an intermediate tool for learning a control policy, e.g. in model-based reinforcement learning (Ha and Schmidhuber, 2018), or directly with nonlinear observers and controllers as in our method (Terzi et al., 2021). To our knowledge, our work represents the first application of the RNN/observer/MPC architecture to the task of manipulating deformable objects.

4.2 Problem Setting and Preliminaries

We consider a robot manipulating a deformable object such that a particular point on the object tracks a trajectory. We define a setting with the following assumptions:

Assumption 4.2.1 (Input)**.** The deformable object is attached to the robot's end effector by an unbreakable grasp at a fixed position. The control policy interacts with the robot by commanding the position and attitude of the grasp point.

Assumption 4.2.2 (Output)**.** The robot's perception system can accurately measure the three-dimensional position of a single point on the surface of the deformable object.

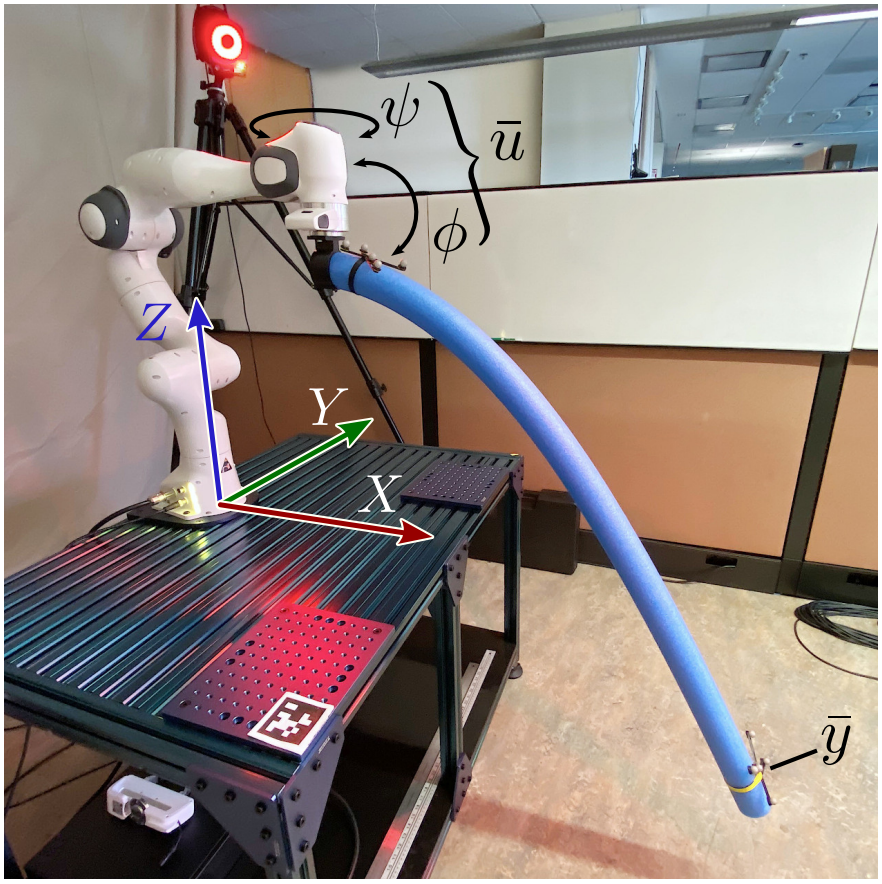


Figure 4.1: Physical testbed for our method. Trajectories are tracked by a Vicon motion capture marker attached to the end of a pool noodle, rigidly held by a Franka Emika Panda robot. Pitch/yaw inputs $\bar{u} = (\phi, \psi)$, tracking point measurement $\bar{y}$, and coordinate axes (X, Y, Z) shown. (The fiducial marker visible in the image is not used.)

Assumption 4.2.3 (Objective)**.** The robot should manipulate the deformable object such that the measured point on its surface tracks a given trajectory in three-dimensional space.

Assumption 4.2.4 (Protocol)**.** Interaction with the environment is divided into two stages. In the *preparation* stage, the robot can interact with the deformable object, perform calculations, and store results. There is no time limit on this stage. In the *testing* stage, a supervisor reveals the trajectory to be tracked. The robot must track the trajectory promptly without slow pre-computations. The robot can use all the results stored from the preparation stage, but cannot interact with the deformable object in any way other than attempting to track the trajectory. This restriction is motivated by safety- and time-constrained tasks, where it may not be feasible for the robot to perform additional exploratory actions.

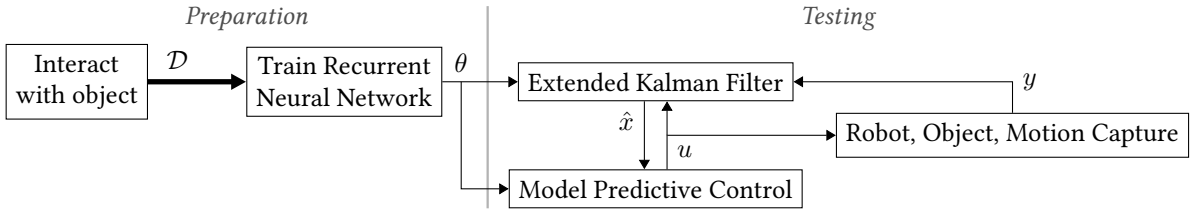


Figure 4.2: Diagram of our system. A small dataset of control inputs and tracked marker locations is obtained from the real system. A recurrent neural network (RNN) model is trained to predict input/output behavior with a latent state. The RNN forms the nonlinear dynamics model for an extended Kalman filter (EKF) and model-predictive controller (MPC) to track a dynamic input trajectory with the deformable object.

Formally, we assume that the coupled system of the robot and deformable object can be described by a continuous-time, deterministic, time-invariant dynamical system

$$\dot{\bar{x}} = \bar{f}(\bar{x}, \bar{u}), \quad \bar{y} = \bar{h}(\bar{x}), \tag{4.1}$$

where the state $\bar{x}$ is an abstract infinite-dimensional quantity representing the full continuum state of the deformable object and the robot, the input $\bar{u} \in \mathrm{SE}(3)$ is the desired pose of the robot end effector, and the output $\bar{y} \in \mathbb{R}^3$ is the position of the measured point. The unknown dynamics $\bar{f}$ encapsulates both the deformable object itself and a low-level controller that attempts to track the end effector pose input $\bar{u}$ by issuing actuator commands, e.g. motor torques, to the robot. The measurement model $\bar{h}$ extracts the position of the measured point from the continuum state.

Assumption 4.2.5. The true system has a unique globally exponentially stable equilibrium state $\bar{x}_0$ corresponding to the steady-state identity input I, i.e. $\bar{f}(\bar{x}_0, I) = \mathbf{0}$. This assumption is realistic for damped elastic objects, such as closed cell foam. We assume that no new plastic deformations to the foam (such as a permanent bend) are introduced during our experimental protocol.

The control policy interacts with the system (4.1) in discrete time steps of fixed length Δ_t. In this paragraph we overload the same notations for discrete-time and continuous-time inputs, states, and outputs; for the remainder of the chapter we will refer to the discrete-time quantities exclusively. At step $k \in \mathbb{N}$,

the discrete-time input $\bar{u}[k]$ is supplied to the continuous-time system as a zero-order hold, resulting in the continuous-time input signal $\bar{u}(t) = \bar{u}[\lfloor t/\Delta_t \rfloor]$ for all $t \geq 0$, and the discrete-time output $\bar{y}[k]$ is sampled such that $\bar{y}[k] = \bar{y}(k\Delta_t)$. Our control task is specified by a discrete-time signal of K goal positions $z[1], \ldots, z[K] \in \mathbb{R}^3$ for the tracked point and the cost function

$$J = \sum_{k=1}^{K} \|z[k] - \bar{y}[k]\|_W^2, \tag{4.2}$$

where the weighting matrix $W \succeq 0$ encodes a (potentially) non-isotropic tracking objective.

Remark The protocol in Assumption 4.2.4 rules out some methods that have been widely used for deformable manipulation. Trajectory optimization methods that build a local dynamics model around a reference trajectory by interacting with the environment, such as guided policy search (Thuruthel et al., 2019), violate the rule against non-task interaction in the testing stage. On the other hand, goal-conditioned reinforcement learning (e.g. Andrychowicz et al., 2017) would be compatible with our protocol.

4.3 Methods

The components of our method are outlined in Figure 4.2. To ensure that our system relies on the whip-like resonant behavior of the deformable object to track trajectories, rather than relying on large translational movements of the robot end effector, we restrict the inputs to only the pitch ϕ and yaw ψ angles of the end effector. We denote this restricted space as $\mathcal{U} \subset \mathrm{SE}(3)$. In practice, we generate training data within a compact subset of pitch and yaw angles, and regularize the MPC problem so that inputs far from the identity I are penalized. Therefore, we parameterize $\mathcal{U}$ by $\mathbb{R}^2$ (in radians) and ignore potential issues of multiple covering (§2.2.3).

4.3.1 Data collection

We begin by collecting a training dataset $\mathcal{D}$ containing N input-output trajectories from the real system. We denote by $\bar{u}[j,k]$ and $\bar{y}[j,k]$ the input and output from the k^{th} time step of the j^{th} trajectory in $\mathcal{D}$. Before starting each trajectory, we apply the identity input and allow the system to settle for several seconds (10 in our experiments) to ensure that the system returns to a state very close to the rest state $\bar{x}_0$. This will happen promptly due to Assumption 4.2.5. We then apply random sinusoidal pitch and yaw inputs. Sinusoidal inputs excite the system enough to demonstrate large-scale dynamics such as resonance that are important for manipulation, but are smooth enough to respect the actuator limits of our robot.

The sinusoidal inputs are chosen to respect a user-selected angular acceleration limit $\dot{\omega}_{\max}$ and absolute angle limits $\phi_{\min}, \phi_{\max}$ and $\psi_{\min}, \psi_{\max}$ to avoid triggering our robot arm's built-in safety stop when actuator limits are reached. For each angle, the sinusoid's frequency ν is sampled log-uniformly between $0.125\,\text{Hz}$ and $3\,\text{Hz}$. The maximum angular acceleration of a sinusoid is given by $2\pi A\nu^2$, where A is the amplitude. Therefore, ν and the acceleration limit induce a maximum amplitude $A \leq A_{\max} = \dot{\omega}_{\max}/2\pi\nu^2$. We sample the amplitude uniformly from $[0.1, A_{\max}]$. Finally, we sample the phase uniformly from $[0, 2\pi)$ and add a constant offset, which is sampled uniformly from the range induced by the amplitude and angle limits.

4.3.2 RNN dynamics model

The true state $\bar{x}$ of the deformable object is an infinite-dimensional continuum of points, which is not representable on a computer without discretization and approximation. Furthermore, the dynamics of the system are not purely dictated by the behavior of the deformable object—they also include the behavior of the low-level controller of the robot arm. We overcome both challenges by representing the system state as the hidden state of a learned RNN. In comparison to other methods of system identification from input/output data, such as linear methods (Brunton and Kutz, 2019), RNNs are a highly expressive class

of nonlinear state-space models. Recall from § 2.11.4 that the RNN is a generic function approximation scheme parameterized by a real-valued vector θ, and consists of a discrete-time dynamics model

$$x[k+1] = f_\theta(x[k], u[k]), \quad y[k] = h_\theta(x[k]), \tag{4.3}$$

where $x \in \mathbb{R}^n$ is the internal state, u is the input, y is the output, and f_θ and h_θ are the dynamics and measurement functions respectively. Both f_θ and h_θ are differentiable with respect to their arguments and the parameter θ. The particular form of the functions f_θ and h_θ must be carefully chosen to maximize expressiveness while preserving desirable properties for optimization, but from the perspective of estimation and control, their exact form is unimportant.

The RNN is trained on the dataset $\mathcal{D}$ to minimize a regression loss on the input-output map over complete sequences:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \sum_{j=1}^{N} \sum_{k=1}^{K_j} \|\bar{y}[j,k] - h_\theta(x[j,k])\|_2^2 \\ \text{subject to} \quad & x[j,0] = \mathbf{0} \\ & x[j,k+1] = f_\theta(x[j,k], \bar{u}[j,k]), \end{aligned} \tag{4.4}$$

where K_j is the length of the j^{th} trajectory in $\mathcal{D}$. The fixed initial state of $\mathbf{0}$ is justified in our setting because each trajectory in $\mathcal{D}$ begins at the rest state $\bar{x}_0$. We use input and output projections, as discussed in § 2.11.4.1, with a $\tanh$ nonlinearity after (resp. before) the input (resp. output) projection. We find an approximate local optimum of (4.4) using stochastic gradient descent.

4.3.3 Model-predictive control with reduced-order model

After finding a value of the RNN parameter θ^* that approximately optimizes the learning objective (4.4), we use the RNN model in a model-predictive control (MPC) framework to optimize the trajectory-tracking

objective (4.2) in an online manner. Assume temporarily (we will return to this assumption in § 4.3.4) that at time step k we know a value $\hat{x}[k]$ of the abstract RNN state that is consistent with the input and output history from previous time steps in the real-world system. We solve the short-horizon optimal control problem

$$\begin{aligned}
\underset{\substack{\bar{u}[k],\ldots,\\ \bar{u}[k+H-1]}}{\text{minimize}} \quad & \sum_{i=1}^{H} \|z[k+i] - h_{\theta^*}(x[k+i])\|_W^2 \\
& + \alpha \sum_{i=0}^{H-1} d(\bar{u}[k+i-1], \bar{u}[k+i]) \\
& + \beta \sum_{i=1}^{H} d(\bar{u}[k+i], I) \qquad (4.5) \\
\text{subject to} \quad & x[k] = \hat{x}[k], \\
& x[k+i+1] = f_{\theta^*}(x[k+i], \bar{u}[k+i]) \; \forall i.
\end{aligned}$$

In the objective (4.5), $H \ll K$ is the MPC horizon. In the second and third terms, $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}_{\geq 0}$ is a semimetric (§ 2.2.1) on the input space $\mathcal{U}$. The first regularization term, weighted by constant $\alpha \geq 0$, encourages smoothness of the control signal. The second regularization term, weighted by constant $\beta \geq 0$, encourages the robot to stay near its rest state. Solving (4.5) yields a sequence of inputs

$$\bar{u}^*[k], \ldots, \bar{u}^*[k+H-1]$$

optimized to track the next H steps of the full goal trajectory. Following the standard moving horizon architecture (§ 2.8.6.1), we apply only the first input $\bar{u}^*[k]$ from the solution to the real system. Then, at time step $k+1$, we solve a new instance of (4.5).

The optimization problem (4.5) is nonconvex, but the solution from the previous time step provides a high-quality initial guess, so local optimization methods typically perform well as long as the initial guess

is not close to a bad local optimum (Allgöwer and Zheng, 2012). We obtain an approximate solution with a few steps of gradient descent with momentum. The momentum state from previous MPC steps is persisted and time-shifted in the same manner as the initial guess.

4.3.4 Estimating the RNN state

In § 4.3.3, we assumed the availability of a RNN state $\hat{x}[k]$ that is consistent with previous inputs and outputs from the real-world system. To obtain $\hat{x}[k]$, it is not sufficient to simply evaluate f_{θ^*} recursively on $\bar{u}[1], \ldots, \bar{u}[k-1]$. Because the RNN model is not perfect, the true outputs $\bar{y}[1], \ldots, \bar{y}[k-1]$ obtained from the real-world system may diverge from the outputs $h_{\theta^*}(x[1]), \ldots, h_{\theta^*}(x[k-1])$ predicted by applying the RNN model in this open-loop manner.

Instead we observe that, although the RNN state has no direct physical meaning, the RNN model is still a nonlinear discrete-time dynamical system with known dynamics. Therefore, its state can be estimated from the input-output history using standard techniques from estimation theory.* In particular, we apply an extended Kalman filter (EKF) to the RNN model. The EKF is defined by linearizing the system about the current state and applying the standard linear Kalman filter covariance propagation and update steps, as discussed in §2.9.1.10. We now review the well-known EKF equations to highlight the role of the RNN.

The EKF maintains a Gaussian-distributed belief over the RNN state. At time k, the belief is distributed according to

$$x[k] \sim \mathcal{N}(\mu[k], \Sigma[k]),$$

*In general, nonlinear state estimation techniques provide no guarantees of optimality or other notions of correctness.

where $\mathcal{N}(\mu, \Sigma)$ is the Gaussian distribution with mean μ and covariance Σ. After sending an input $\bar{u}[k]$ to the system, we then update the belief according to

$$\begin{aligned} \mu[k|k-1] &= f_{\theta^*}(\mu[k-1], \bar{u}[k]), \\ \Sigma[k|k-1] &= F[k]\Sigma[k-1]F[k]^\top + Q[k], \end{aligned} \tag{4.6}$$

where

$$F[k] = \frac{\partial f_{\theta^*}}{\partial x}(\mu[k-1],\ \bar{u}[k])$$

is the Jacobian of the RNN dynamics f_{θ^*} with respect to state. The process noise covariance $Q[k] \succeq \mathbf{0}$ represents noise in the RNN abstract state, so it cannot be derived from a physical model or estimated from data. We simply set Q to a scaled identity matrix in this work. Next, the measurement $\bar{y}[k]$ is captured from the real system. We compute the measurement residual $\gamma[k] = \bar{y}[k] - h_{\theta^*}(\mu[k|k-1])$. According to the current belief, $\gamma[k]$ has covariance

$$S[k] = H[k]\Sigma[k|k-1]H[k]^\top + R[k],$$

where

$$H[k] = \frac{\partial h_{\theta^*}}{\partial x}(\mu[k|k-1])$$

is the Jacobian of the RNN observation function h_{θ^*} with respect to state. The sensor noise covariance $R[k] \succ 0$ represents a physically meaningful quantity that can be derived from a model or estimated from data. In this work we use a motion capture system with isotropic noise, so we set R to a constant scaled identity matrix. We then compute the Kalman gain

$$K[k] = \Sigma[k|k-1]H[k]^\top S[k]^{-1}$$

and update the belief according to

$$\begin{aligned}\mu[k|k] &= \mu[k|k-1] + K[k]\gamma[k],\\ \Sigma[k|k] &= (I - K[k]H[k])\Sigma[k|k-1].\end{aligned} \tag{4.7}$$

We then use the mean of the belief distribution as the initial state for the MPC problem (4.5), that is, $\hat{x}[k] = \mu[k]$.

4.3.5 Implementation

For the RNN reduced-order dynamics model f_θ, we select the long short-term memory (LSTM) architecture (§2.11.4.1). Numerical values of the architectural and training hyperparameters are listed in Table 4.1. We train the LSTM in PyTorch (Paszke et al., 2019). We also solve the model-predictive control problems and implement the EKF in PyTorch due to the ease of differentiating through the RNN model. We run the MPC control loop at $40\,\text{Hz}$.

Table 4.1: Values of user-chosen constants in our experiments.

Q	EKF process covariance	$10^{-6}I$
R	EKF measurement covariance	$10^{-2}I$
H	MPC horizon	25
α	MPC smoothness weight	1.0
β	MPC homing weight	1e-1
—	MPC gradient descent rate	4e-1
—	MPC gradient descent steps	5
—	LSTM layers	1
n	Reduced state dimension	200
—	LSTM SGD steps	1e4
—	LSTM SGD learning rate	1e-3
—	LSTM SGD batch size	10
N	# trajs. in dataset	100

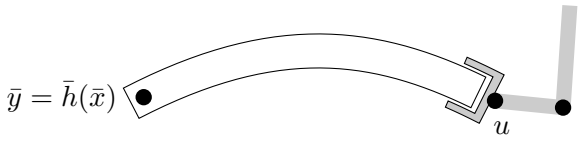


Figure 4.3: Schematic diagram of pool noodle experimental setup showing end effector pose $\bar{u}$ and tracked surface point $\bar{y}$. Details are given in §4.4.

4.4 Experiments

In all experiments, our deformable body is a thin cylinder of uniform closed-cell polyethylene foam, commonly known as a pool noodle, with length $1.5\,\mathrm{m}$ and diameter $6.5\,\mathrm{cm}$. To attach the object to the robot, we press-fit approximately $4\,\mathrm{cm}$ of one end of the pool noodle into a rigid 3D-printed ring attached to the robot end effector. To track the distal end of the object, we attach a rigid assembly of motion capture markers and track its full pose with a Vicon motion capture system. The measurement $\bar{y}$ has a calibrated offset from the marker assembly to lie on the center line of the pool noodle. Our experimental setup is shown in Figure 4.1.

The robot is a Franka Emika Panda. To track the pose commands $\bar{u} \in \mathrm{SE}(3)$ output by the MPC controller, we apply a proportional-only control law in both the position and in the attitude quaternion to generate desired linear and angular velocities v for the end effector. We compute the kinematic Jacobian from `libfranka` and produce desired joint velocities using the Jacobian pseudoinverse with null space optimization towards the "home" position (Siciliano et al., 2009).

4.4.1 Model frequency response

To validate our RNN model, we compare its empirical frequency response near the internal state $x = \mathbf{0}$ to that of the true physical system at its rest state $\bar{x}_0$. Frequency response is a holistic property of the model that depends on its behavior in multiple parts of the state space. To avoid the complications of frequency-domain analysis for multiple-input/multiple-output systems, we actuate the system only in the yaw axis ψ

and measure only the deflection of the pool noodle tip in the horizontal axis. Yaw inputs, as opposed to pitch inputs, avoid the asymmetric effect of gravity. .

We sample 33 geometrically-spaced frequencies ranging from $0.125\,\mathrm{Hz}$ to $2\,\mathrm{Hz}$. For each frequency, we apply a yaw input with a small amplitude (11.5° peak-to-peak) for $20\,\mathrm{sec}$ and record the trajectory of the pool noodle tip $\bar{y}$. The small input amplitude allows higher-frequency inputs before reaching the robot's actuator limits. We discard approximately the first half of the recording (on the nearest whole cycle boundary) to eliminate transient effects. We then compute the empirical gain and phase by taking the inner product of the output signal with complex exponential functions, analogous to the discrete Fourier transform. Because the input and output have different units, only the relative gain magnitudes between different frequencies are meaningful. We normalize the gains such that the gain of the real system for the slowest input frequency equals 1.

Bode (gain and phase) plots for each system are overlaid in Figure 4.4. In the true physical system, we observe typical behavior of a resonant lowpass filter with a strong resonant peak around $0.8\,\mathrm{Hz}$. The peak gain is approximately 4 times larger than the low-frequency gain. This represents a dramatic phenomenon that a faithful model must capture. In the RNN model, the gain response closely matches the real system. The phase response matches closely below the resonant frequency, but begins to lag behind the true system for high frequencies. We suspect this may be due to an unbalanced training data distribution. Overall, this experimental result suggests that the RNN model has successfully captured the frequency response of the physical system.

4.4.2 MPC tracking

We apply our method to track several test trajectories which attempt to expose the controller's performance with regard to resonant dynamics. The goal trajectories $z[1], \ldots, z[K] \in \mathbb{R}^3$ are specified by the user. As described in eq. (4.2), our tracking cost is non-isotropic. We use the value $W = \mathrm{diag}(0, 1, 1)$ to focus only

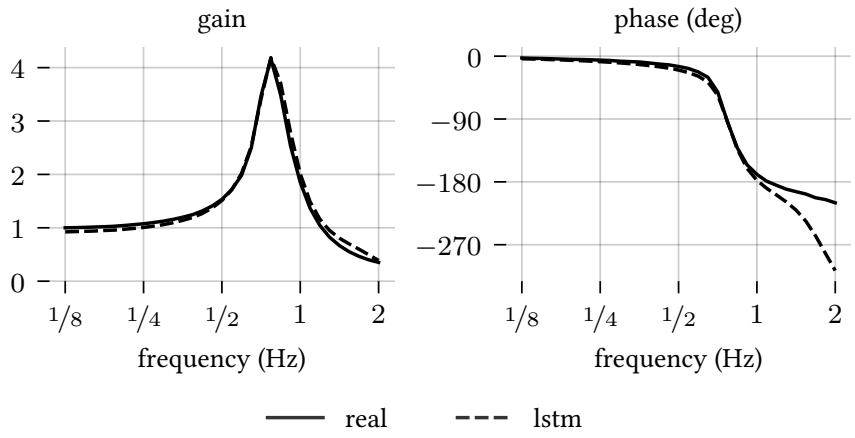


Figure 4.4: Frequency-domain gain and phase response (Bode plots) for real pool noodle and LSTM model.

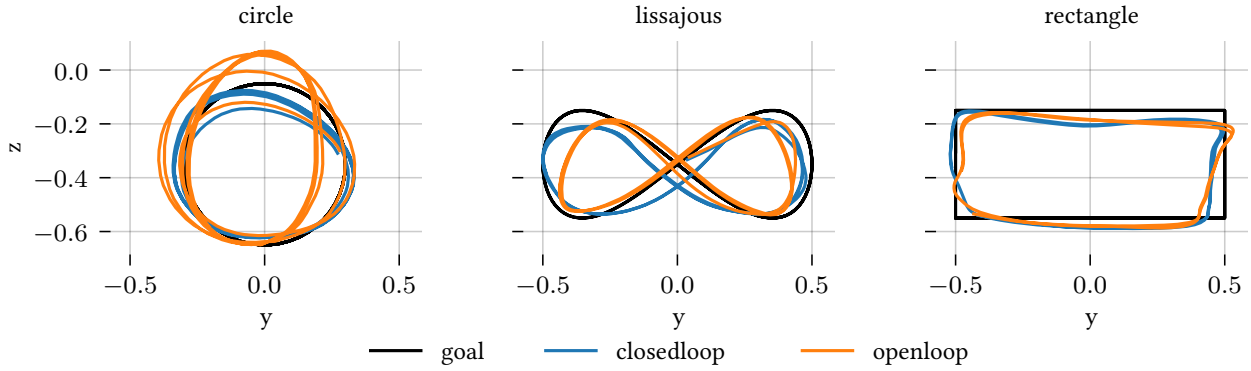


Figure 4.5: Two-dimensional projections of paths traced by pool noodle free end in MPC tracking experiments. See §4.4.2 for details and discussion.

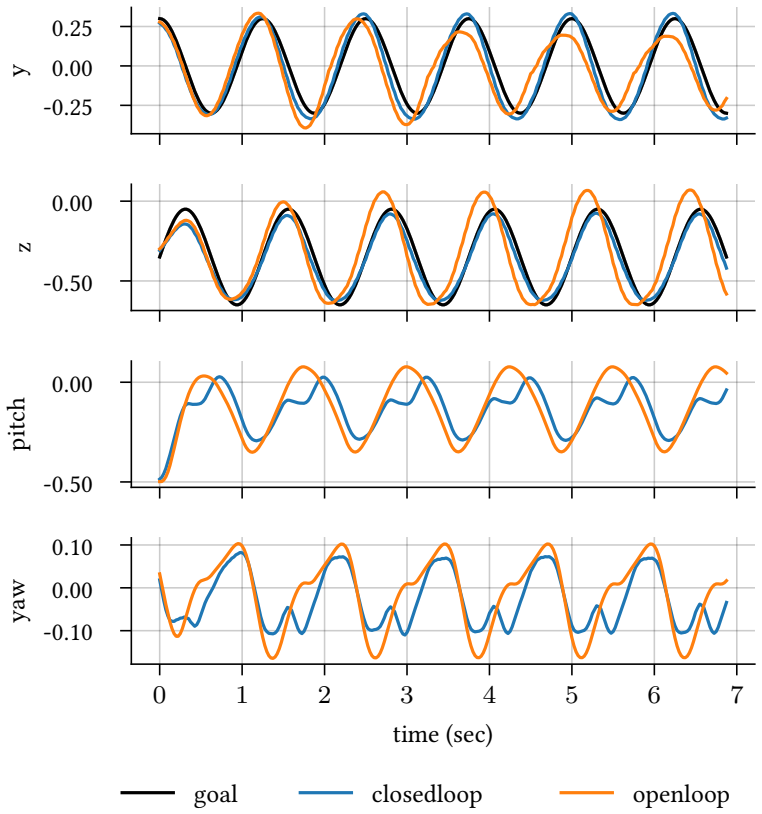


Figure 4.6: Traces of rotation angle inputs (pitch, yaw) and horizontal and vertical components of pool noodle free end (y, z) for MPC tracking of the circle trajectory (see Figure 4.5). The closed-loop variant of our method shows superior tracking performance.

on tracking in the Y- and Z-axes, which represents the view from the front of the robot. Without this non-isotropic weight, the goal trajectory would need to be a carefully designed curve in $\mathbb{R}^3$ to be trackable with zero error.

We show the results from three trajectories in Figure 4.5. The first two trajectories are a circle of diameter $0.6\,\mathrm{m}$ and a figure-8 Lissajous curve of width $1\,\mathrm{m}$ and height $0.4\,\mathrm{m}$. Both trajectories are sinusoidal in each axis, which matches the data collected during our training step. The circle and the vertical axis of the Lissajous curve are set to the system's resonant frequency $0.8\,\mathrm{Hz}$ as determined experimentally. The third trajectory is a rectangle with constant linear velocity along each edge (no stopping at corners). These trajectories are close to the robot's dynamic limits as discussed in § 4.3.1. Improvements in our low-level controller are required to test more demanding trajectories that push the system further into nonlinearity.

To show that the EKF observer provides meaningful feedback to the controller (the "closed-loop" setup), we compare it to a setup that assumes the predicted feedforward state, i.e. the value yielded by applying the RNN dynamics f_{θ^*} to the full sequence of past inputs, is always correct (the "open-loop" setup). The results of this comparison are visualized in Figures 4.5 and 4.6, and the tracking errors are compared numerically in Table 4.2.

Table 4.2: MPC tracking errors

shape	kind	max error (cm)	mean error (cm)
circle	closedloop	12.59	5.59
	openloop	29.15	11.61
lissajous	closedloop	9.19	4.43
	openloop	14.85	4.77
rectangle	closedloop	14.60	6.34
	openloop	14.62	6.01

For the circle trajectory, we observe significantly improved performance in both mean and maximum error from the closed-loop setup. In the traces over time, shown in Figure 4.6, the open-loop solution drifts towards stronger resonance in the vertical axis and weaker resonance in the horizontal axis. In

contrast, the error of the closed-loop solution does not grow over time, indicating that our EKF setup is able to compensate for model error. For the Lissajous trajectory, the closed-loop setup yields a minor improvement in mean tracking error but a significant improvement in maximum error. The rectangle trajectory shows little difference between the open- and closed-loop approaches, but it is noteworthy that both approaches track the rectangle without gross errors, even though it is physically infeasible and not similar to the training data. This result suggests that the LSTM model behaves reasonably for at least one sequence input that is dissimilar to those in the training data.

4.5 Conclusion

We have described and demonstrated a system for optimal control in settings with unknown complex dynamics but a known reward function. Our approach is completely data-driven, and requires a fixed initial data-collection phase without further exploratory actions. We model our dynamical system as an LSTM recurrent neural network and design a nonlinear MPC controller. We use an EKF state observer to account for model error by estimating a value of the LSTM hidden state consistent with past inputs and outputs.

We apply our method to the task of manipulating a deformable object such that a particular point on the object tracks a fast trajectory. We validate our model on a real hardware setup with a robot manipulator holding a foam pool noodle, measured by a motion capture system. Our experiments show that closing the loop with the EKF observer improves tracking performance compared to an open loop control scheme for several of the test trajectories.

In future work, we aim to improve tracking accuracy by investigating other nonlinear state estimation methods and MPC implementations. The EKF is one of several ways to account for modeling error. Other nonlinear state estimators such as particle filters and moving window least-squares estimators could also be adapted to estimate RNN internal states. Alternatively, a pure deep-learning approach might add past

observations as inputs to the RNN. For MPC, a more sophisticated constrained optimization scheme could be applied to the nonlinear MPC problem (4.5) to help enforce smoothness and actuator limits. We are also interested in applying our work to non-position-based control tasks, such as force control for using deformable objects as tools.

We also note that our method resembles the inner loop of a model-based reinforcement learning algorithm. An extended method could use the data gathered at test time to further update the model. This could relax the demand on good state space coverage in the initial training data.

Chapter 5

Variance of Policy Gradient for LQR problems

5.1 Introduction

We concluded Chapter 3 by discussing some possible reasons why reinforcement learning with neural network "universal policies" might lead to a suboptimal multi-system policy. One possibility is the interaction between the RL algorithm and the problem. The RL algorithm was able to learn good policies for a single $\phi \in \Phi$, but when deploying the same algorithm on the multi-system problem, it was unable to learn a multi-system policy that matched the single-system policies in aggregate. One possible reason is that some property of the multi-system MDP hindered the performance of the algorithm. More broadly, it leads us to ask the question: *How do properties of the MDP dynamics impact the performance of RL algorithms?*

For RL in finite MDPs, performance bounds are most commonly given with respect to the number of states, actions, and rewards, and the horizon length (if finite) or discount factor (if infinite-horizon) (Agarwal et al., 2022). In the more challenging setting of undiscounted infinite-horizon MDPs with the average reward criterion, stronger assumptions on the MDP such as connectedness (resp. ergodicity) are required, and regret bounds are given in terms of related quantities such as diameter and/or span (resp. mixing and/or hitting time). Wei et al. (2020, Table 1) review some results of this type alongside their own.

The work presented in this chapter was originally published in Preiss et al. (2019). The related work section has been expanded to include more recent research in this area.

Our results in Chapter 3 are difficult to investigate from a theoretical perspective due to the combination of neural network function approximation, complex deep RL algorithms with many "tricks", and nonlinear dynamics with contact (in the MuJoCo experiments). Adding the multi-system structure increases the complexity further. As we will discuss in §5.2, the theoretical understanding of reinforcement learning in continuous spaces has advanced considerably in the past several years. However, at the time this project was initiated, there were few theoretical guarantees for RL beyond finite MDPs. Therefore, we narrowed our focus to a very simple continuous MDP and a simple RL algorithm. Since the theoretical picture for single-system MDPs is still far from complete, we leave the development of RL theory for multi-system problems to future work.

The experiment in §3.6 used PPO, a member of the policy gradient family of RL algorithms (§2.7.2). Policy gradient methods construct an unbiased estimate of the gradient of the RL objective with respect to the policy parameters, and perform stochastic gradient ascent/descent with this estimate. Policy gradient methods are widely used for RL in continuous spaces because, unlike Q-learning, they do not require evaluating an "argmax" over a continuous $\mathcal{U}$, which is computationally intractable in general. However, the gradient estimate is known to suffer from high variance (Greensmith et al., 2004). The earliest and simplest policy gradient algorithm is REINFORCE (Williams, 1992). More recent algorithms such as TRPO and PPO (Schulman et al., 2015, 2017) extend the basic idea of REINFORCE with techniques to inhibit the possibility of making very large changes in the policy action distribution in a single step. In a benchmark test (Duan et al., 2016a), these algorithms generally learned better policies than REINFORCE, but their additional complexity makes them hard to analyze.

In this chapter, we seek a more detailed understanding of how the policy gradient estimate variance relates to properties of the continuous-space Markov decision process (MDP) that defines the RL problem instance. As an analytically tractable class of continuous MDPs, we select the linear-quadratic regulator (LQR) problem, introduced in §2.9.1.6. Our primary contributions are derivations of bounds on the variance

of the REINFORCE gradient estimate as an explicit function of the dynamics, reward, and noise parameters of the LQR problem instance. We validate our bounds with comparisons to the empirical gradient variance in random problems. We also explore the relationship between gradient variance and sample complexity, but find it to be less straightforward, as the problem parameters that affect variance also affect the optimization landscape. We emphasize that our goal is not to draw a conclusion about the utility of using REINFORCE to solve LQR problems, but rather to use LQR as an example system that is simple enough to allow us to "look inside" the REINFORCE policy gradient estimator.

5.2 Related work

Policy gradient methods for LQR LQR systems are a popular case study for analyzing policy gradient algorithms in continuous spaces. They have also been studied in the context of model-based and value-based RL algorithms, but we restrict our attention to policy gradient methods here. Fazel et al. (2018) showed that, even though the optimization landscape of LQR is neither convex nor smooth, gradient descent using a zeroth-order optimization approximation of the policy gradient enjoys global convergence and sample complexity bounds. The analysis centers on the *gradient domination* (also known as *Polyak-Łojasiewicz*) condition. Malik et al. (2018) and Bu et al. (2019a) strengthened this type of result and generalized to other LQR variants. Bhandari and Russo (2019) generalized the gradient domination technique to other highly structured problems outside the LQR setting, including finite MDPs, an optimal stopping problem, and an inventory control problem. Alternatively, Mohammadi et al. (2019) obtained similar results for the continuous-time case by relating the gradient flow in a classic convex reparameterization of the LQR problem to the nonconvex policy gradient. Sun and Fazel (2021) generalized the reparameterization-based results to a broader family of LQR problems. Cassel and Koren (2021) obtained a nearly optimal regret bound for policy gradient methods with respect to the time horizon, matching a minimax lower bound of Simchowitz and Foster (2020) up to logarithmic factors. In contrast, Tu and Recht (2019) showed

that REINFORCE is strictly less efficient than model-based methods with respect to the state and action dimensionalities in the cheap-control setting.

Most of the aforementioned works study exact policy gradients or policy gradient approximations obtained by parameter-space noise or perturbations, similar to generic derivative-free optimization methods. This is in contrast to the action-space noise used by REINFORCE, as described in §2.7.2. The work of Tu and Recht (2019) is an exception, but their variance analysis only applies to the restricted class of cheap-control LQR problems used in their lower bound. To our knowledge, the work in this chapter represents the only generic variance upper bound for action-space policy gradient methods in the LQR setting.

Nonlinear system classes Outside the LQR setting, researchers have also obtained theoretical guarantees in settings where the state space, and possibly the action space, are continuous. The *linear MDP*, in which transition probabilities and rewards are linear in a feature space, is a popular case for analysis (Jin et al., 2020). Several variations of the linear MDP model exist. Linear MDPs generalize finite MDPs. However, this class cannot express the LQR problem (Song and Sun, 2021), which suggests that it may be of limited relevance for robotics. Mania et al. (2022) gave a finite-sample complexity guarantee for the system identification problem in the expressive *kernelized nonlinear regulator* (KNR) system class. Kakade et al. (2020) proposed a model-based RL algorithm with a regret bound depending on information-theoretic quantities for KNRs, but their algorithm is computationally intractable. Song and Sun (2021) propose a tractable model-based method for KNRs. Boffi et al. (2021) prove a regret bound for a different class of nonlinear systems with a stronger stability assumption. Agarwal et al. (2019) give global convergence results for policy gradient methods in tabular MDPs, and in highly generic settings with smooth function approximation policies they give guarantees in terms of approximation error. Their results highlight the importance of exploration. Agarwal et al. (2020) build upon this framework and address the exploration issue using an ensemble of policies and exploration reward bonuses. Feng et al. (2021) extend this class of methods to handle more nonlinearity. Recently, Jin et al. (2021) and Du et al. (2021) proposed

new expressive classes of tractable MDPs that generalize both linear MDPs and LQRs. These provide a promising setting for future work on provably efficient algorithms for problems that combine features of finite MDPs (like difficult exploration) with features of physical systems (like stabilizing at an equilibrium).

Reinforcement learning theory with deep neural networks Until recently, research into the theoretical properties of deep neural networks focused on static optimization problems like supervised learning. He and Tao (2020) provide a survey on such results. More recently, researchers have begun to study deep neural networks in reinforcement learning context. Fan et al. (2020) analyzed sample complexity for the Deep Q-Network algorithm (DQN, Mnih et al., 2013) under the assumption of an fixed i.i.d. sampling distribution, which they argue is a reasonable approximation of sampling from a large replay buffer generated by ϵ-greedy exploration. Xu and Gu (2020) obtain the same rate for DQN but remove the i.i.d. assumption, learning only from the most recent interaction with the MDP. However, the actions are sampled from a fixed policy, as opposed to an ϵ-greedy policy with respect to the current estimate of $Q^\star$. Therefore, both of these analyses sidestep the issue of exploration. Cai et al. (2019) analyze the closely related task of policy evaluation, i.e. finding a fixed point of the policy Bellman operator instead of the optimality Bellman operator, with neural networks. Yang et al. (2020) study the least-squares value iteration algorithm for both kernels and overparameterized neural networks using optimism in the face of uncertainty for exploration.

Moving from value-based methods to policy-based methods, Wang et al. (2020) proved convergence rates for actor-critic methods with vanilla and natural policy gradients. Their analysis showed that the overparameterized neural networks can approximately satisfy the *compatible function approximation* condition, which leads to unbiased policy gradient estimates. Liu et al. (2019) obtain similar results specifically for the widely-used PPO and TRPO algorithms, but their analysis requires fully solving optimization problems for each update instead of taking a single gradient step. Agazzi and Lu (2021) work in the mean-field regime with continuous-time dynamics to express the policy gradient dynamics as a partial differential equation, and show that all fixed points are global optima.

5.3 Problem setting

In this chapter, $\|\cdot\|$ denotes the $2,2$ operator norm of a matrix or the 2-norm of a vector. We study the REINFORCE policy gradient estimator, as defined in § 2.7.2. We consider a variant of the discrete-time LQR problem introduced in §2.9.1.6 that adds stochasticity to the dynamics but retains noise-free full state observability. Recall that the state space is $\mathcal{X} = \mathbb{R}^n$ and action space $\mathcal{U} = \mathbb{R}^m$. The dynamics are linear dynamics with additive Gaussian noise:

$$x_{t+1} = Ax_t + Bu_t + \epsilon_t^x, \quad \epsilon_t^x \sim \mathcal{N}(\mathbf{0}, \Sigma_x), \tag{5.1}$$

for dynamics matrices $A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m}$, and noise covariance $\Sigma_x \succeq 0$. The initial state x_1 follows an arbitrary zero-mean Gaussian distribution. We consider the finite horizon objective

$$\sum_{t=1}^{H} r_t \triangleq \sum_{t=1}^{H} = -(x_t^T Q x_t + u_t^T R u_t) \tag{5.2}$$

for cost matrices $Q \succeq 0,\ R \succ 0$. (Note that the standard LQR cost has been negated to fit the reward-maximization formulation of RL.)

As discussed in §2.9.1.6, for the deterministic version of the problem, the *infinite-horizon* LQR cost is minimized by a stationary linear policy $u_t = K^\star x_t,\ K^\star \in \mathbb{R}^{m\times n}$, where the value of $K^\star$ depends on the solution of an discrete-time algebraic Riccati equation with coefficients derived from (A, B, Q, R). To apply REINFORCE, the policy must be stochastic, so we consider linear *stochastic* policies

$$u_t = Kx_t + \epsilon_t^u, \quad \epsilon_t^u \sim \mathcal{N}(\mathbf{0}, \Sigma_u), \tag{5.3}$$

for $K \in \mathbb{R}^{m\times n}, \Sigma_u \in \mathbb{S}_{++}^m$. The state noise Σ_x is an immutable property of the system, but the action noise Σ_u is not. Instead, it is usually chosen by the user of the RL algorithm, or learned as a parameter

subject to optimization by the RL algorithm. Genuine noise in the actuators enters the system linearly through the matrix B, so it can be subsumed into Σ_x.

In the RL literature, action noise is usually seen as either 1) a tool for exploring of the state space, 2) a method of regularization to avoid converging on bad local optima, or 3) a consequence of a probabilistic interpretation of the RL problem (Levine, 2018). Its effect on the RL optimization algorithm is less frequently discussed, but in this work we find that it can be significant.

5.4 Main result: Variance bounds on the REINFORCE estimator

In this section, we present bounds on the variance of the REINFORCE estimator for LQR systems. Background on REINFORCE is given in § 2.7.2. The instantiation of REINFORCE for the system defined by eqs. (5.1) to (5.3) using a single trajectory is:

$$\hat{g} = \left(\sum_{t=1}^{H} \Sigma_u^{-1} \epsilon_t^u x_t^\top \right) \left(\sum_{t=1}^{H} r_t \right) \in \mathbb{R}^{m \times n}. \tag{5.4}$$

The estimate $\hat{g}$ is a function of the independent random variables $\{\epsilon_t^u, \epsilon_t^x\}_{t=1}^H$. Although x_t is a linear function of the previous noise variables $\{\epsilon_\tau^u, \epsilon_\tau^x\}_{\tau=1}^{t-1}$, the reward r_t is quadratic in x_t, so the overall form of $\hat{g}$ is a product of a sum and a sum of products of sums. Therefore, while it is possible to apply matrix concentration inequalities (Tropp, 2015) to bound $\|x_t - \mathbb{E}[x_t]\|$ with high probability, it is more difficult to bound the dispersion of $\hat{g}$. Instead, we use a more specialized method to derive a bound on

$$\nu(\hat{g}) \triangleq \sum_{i=1}^{m} \sum_{j=1}^{n} \operatorname{Var}(\hat{g}_{i,j}) = \mathbb{E}\left[\operatorname{tr}\big(\hat{g}^\top \hat{g}\big)\right] - \operatorname{tr}\big(\mathbb{E}[\hat{g}]^\top \mathbb{E}[\hat{g}]\big),$$

which we simplify by bounding $\mathbb{E}\left[\operatorname{tr}\big(\hat{g}^\top \hat{g}\big)\right]$.

Theorem 5.4.1. *If* $\rho(A+BK)<1$, *then* $\mathbb{E}\left[\mathrm{tr}\left(\hat{g}^\top \hat{g}\right)\right] \leq O\left(\bar{n}^4 C_1^2 C_2^2\right)$, *where*

$$C_1 = \mu^2 \left\|\Sigma_u^{-\frac{1}{2}}\right\| \left(\|x_1\| + \sigma H\right) H',$$

$$C_2 = \|R\|\|\Sigma_u\| H + \mu^2 \left(\|Q\| + \|R\|\|K\|^2\right)\left(\|x_1\|^2 + \sigma^2 H\right) H'^2,$$

$\bar{n} \triangleq \max\{n, m\}$, $\sigma \triangleq \left\|\Sigma_x^{\frac{1}{2}}\right\| + \left\|B\Sigma_u^{\frac{1}{2}}\right\|$, $H' \triangleq \min\left\{H, \frac{1}{1-\rho(A+BK)}\right\}$, *and* μ *is a constant bounding the transient behavior of* $\|A+BK\|^t$, *with more details provided in §5.6.*

Proof Sketch.

1. Rewrite $\epsilon_t^u, \epsilon_t^x$ as $\Sigma_u^{\frac{1}{2}}\delta_t^u, \Sigma_x^{\frac{1}{2}}\delta_t^x$, where the δ_t^u, δ_t^x are unit-Gaussian random variables.
2. Bound $\mathrm{tr}\left(\hat{g}^\top \hat{g}\right)$ by P, a polynomial function of the χ-distributed random variables $\{\|\delta_t^u\|, \|\delta_t^x\|\}_{t=1}^H$ with nonnegative coefficients.
3. Bound the sum of the coefficients of P by substituting 1 for all χ random variables.
4. Bound for the expectation of each monomial in P using the moments of the χ distribution.

A detailed proof of Theorem 5.4.1 is given in §5.6.

For a special case of scalar states and actions, we also show a lower bound on $\mathbb{E}[\hat{g}^2]$. Since $\mathbb{E}[\hat{g}] = 0$ at a local optimum, this lower bound corresponds to the variance caused strictly by noise in the system when the policy is already optimal. Here, the matrices A, B, K, Q, R are denoted as a, b, k, q, r, and $\sigma_x,\ \sigma_u$ denote the standard deviation (not variance) of state and action noise. This notation r is different from the notation r_t for reward.

Theorem 5.4.2. *If* $m = n = 1$ *and* $0 \leq a + bk < 1$, *then* $\mathbb{E}[\hat{g}^2] \geq \Omega(c_1^2 c_2^2)$, *with*

$$c_1 = \frac{1}{\sigma_u}\left(|x_1| + \sigma\sqrt{H}\right)\sqrt{h'},$$

$$c_2 = r\sigma_u^2 H + (q + rk^2)(x_1^2 + \sigma^2 H)h',$$

where $\sigma \triangleq \sigma_x + b\sigma_u$ *and* $h' \triangleq \min\left\{H, \frac{1}{1-(a+bk)^2}\right\}$.

A detailed proof of Theorem 5.4.2 is given in §5.7. If we reduce the upper bound of Theorem 5.4.1 to its scalar case, all terms match with the notable exception of the horizon-related terms H and H' (compare to h'), which appear squared in several places in Theorem 5.4.1 compared to the equivalent term in Theorem 5.4.2. There is another gap in the denominators of H' and h' since $\frac{1}{1-x^2} < \frac{1}{1-x}$ on the domain $x \in (0, 1)$. We conjecture our upper bound can be tightened by fully exploiting the independence of the noise variables $\epsilon_x,\ \epsilon_u$.

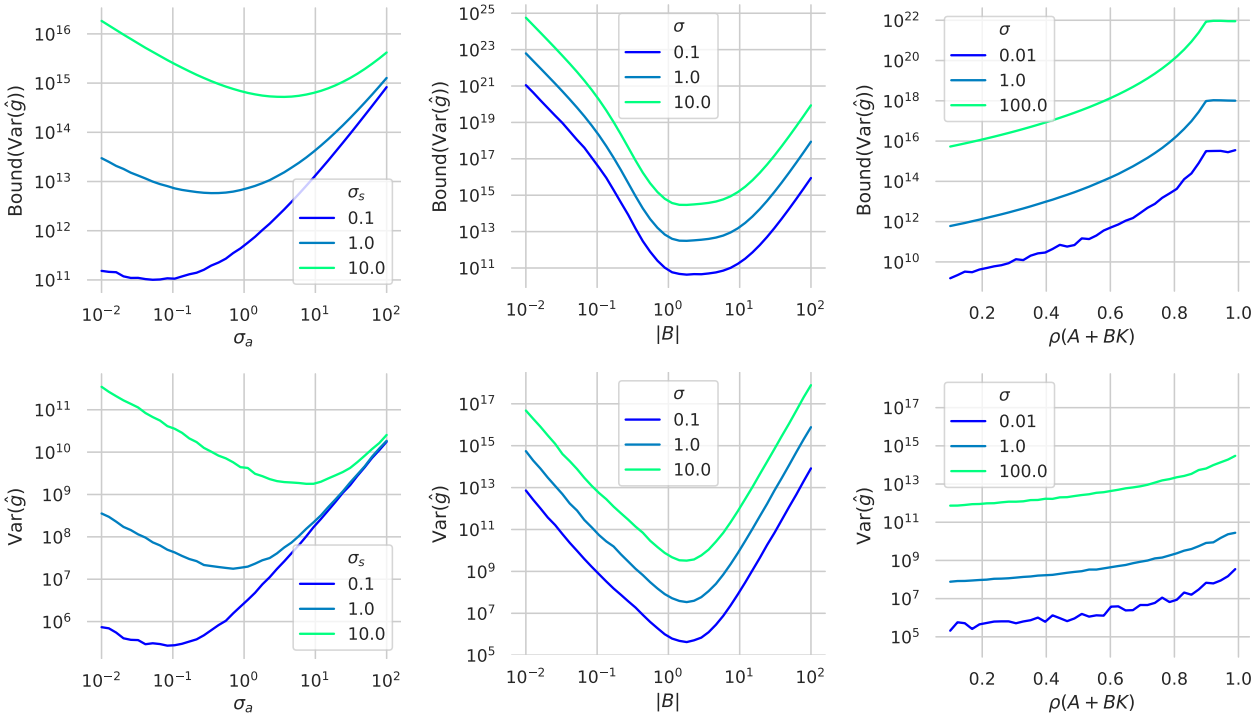
5.5 Experiments

In the first set of experiments, we compare our upper bound of $\nu(\hat{g})$ to its empirical value when executing REINFORCE in randomly generated LQR problems. Our results show qualitative similarity in the parameters for which our upper and lower bounds match. On the other hand, the gap with respect to stability-related parameters is also visible. For each experiment shown here, we repeated the experiment with different random seeds and observed qualitatively identical results.

We generate random LQR problems with the following procedure. We sample each entry in A and B i.i.d. from $\mathcal{N}(0, \sigma{=}n^{-1/2})$ and $\mathcal{N}(0, \sigma{=}m^{-1/2})$ respectively. To construct a random $k \times k$ positive definite matrix, we sample from the $\mathrm{Wishart}(k^{-1}I, k)$ distribution by computing $Q = X^T X$ for X i.i.d. analogous to A. The scale factor k^{-1} ensures that if the vector x is distributed by $\mathcal{N}(0, k^{-1}I)$, such that $\mathbb{E}[\|x\|^2] = 1$, then $\mathbb{E}[x^T Q x] = 1$. We sample Q, R, Σ_x, and Σ_u this way.

In each experiment, we vary some of these parameters systematically while holding the others constant, allowing us to visualize the impact of each parameter on $\nu(\hat{g})$. We plot the upper bound of Theorem 5.4.1 on the top row of Figure 5.1, and the empirical estimate of $\nu(\hat{g})$ on the bottom row. In both cases, since the variance depends on the initial state x_1, we sample $N = 100$ initial states x_1 from $\mathcal{N}(0, n^{-1}I)$ and

plot $\mathbb{E}_{x_1 \sim \mathcal{N}(0, n^{-1}I)} \nu(\hat{g})$. We estimate $\nu(\hat{g})|_{x_1=x}$ for a particular initial state x by sampling 30 trajectories with random $\epsilon^u,\ \epsilon^x$.



(a) Relationship between Σ_x and Σ_u (b) Control authority $|B|$ (c) Stability $\rho(A + BK)$

Figure 5.1: Comparison between our upper bound from Theorem 5.4.1 (top) and the empirically measured variance (bottom) as they relate to various parameters of the LQR problem. Behavior is qualitatively similar for action noise covariance (a) and control authority (b), but less similar for the stability (c) where our bounds are loose. Further discussion is in §5.5.

Effect of Σ_u In this experiment, we generate a random LQR problem and replace Σ_u with $\sigma_u I$ for σ_u geometrically spaced in the range $[10^{-2}, 10^2]$. Using a scaled identity matrix is common practice when applying RL to a problem where there is no *a priori* reason to correlate the noise between different action dimensions. We evaluate the variance at the value $K = K^\star$, where $K^\star$ is the infinite-horizon optimal controller computed using traditional LQR synthesis, as described in § 2.9.1.6. Using $K^\star$ ensures that $\rho(A + BK) < 1$, a required condition to apply Theorem 5.4.1.

Results are shown in Figure 5.1a. The separate line plots correspond to scaling the random Σ_x by the values $\{0.1, 1, 10\}$, while the x-axis corresponds to the value of σ_u. For each value of σ_x, there appears to

be a unique σ_u that minimizes $\nu(\hat{g})$, and this value of σ_u increases with σ_x. This phenomenon appears in both the bound and empirical variance.

Effect of $\|B\|$ In this experiment, we generate a random problem where $m = n$ and replace B with bI for b geometrically spaced in the range $[10^{-2}, 10^2]$. The resulting system essentially gives the policy direct control over each state. For each B, we compute a separate infinite-horizon optimal K and sample the variance for different x_1. Results are shown in Figure 5.1b. The separate line plots correspond to scaling both the random Σ_x and the random Σ_u by the values $\{0.1, 1, 10\}$. The x-axis corresponds to the value of b. Again, there appears to be a unique $\|B\|$ that minimizes $\nu(\hat{g})$, but its value changes minimally for different magnitudes of $\Sigma_x,\ \Sigma_u$.

Effect of $\rho(A + BK)$ Here we measure the change in variance with respect to the closed-loop spectral radius $\rho(A + BK)$. To synthesize controllers K such that $\rho(A + BK)$ obtains a specified value, we use the pole placement algorithm of Tits and Yang (1996). (Pole placement is discussed in §2.9.4.) We sample a "prototype" set of n eigenvalues with $\lambda_1, \ldots, \lambda_{\lceil n/2 \rceil}$ as complex conjugate pairs $\lambda_i, \lambda_{i+1} = re^{\pm i\varphi}$, where $r \sim \mathrm{Uniform}([0, 1])$ and $\varphi \sim \mathrm{Uniform}([0, 2\pi))$, and sample the remaining real λ_i from $\mathrm{Uniform}([-1, 1])$. Then, for each desired ρ, we compute $K_\rho = \mathscr{P}(A, B, \rho\lambda_1, \ldots, \rho\lambda_n)$. By rescaling the same set of λ_i instead of sampling a new set for each ρ, we avoid confounding effects from changing other properties of K.

Results are shown in Figure 5.1c. Again, we repeat the experiment for different magnitudes of Σ_x and Σ_u. Unlike the previous two experiments, here we see qualitatively different behavior between our upper bound and the empirical variance. The bound begins to increase rapidly near $\rho = 1$, corresponding to the growth of $1/(1 - \rho)$ in the term H', but at $\rho = 0.9$ the H term becomes active in H', and the bound suddenly flattens. In contrast, the empirical variance grows more moderately and does not explode near the

threshold of system instability. This further bolsters our confidence that the upper bound of Theorem 5.4.1 can be tightened to match the $\sqrt{H'}$ and $\sqrt{H}$ terms in the lower bound of Theorem 5.4.2.

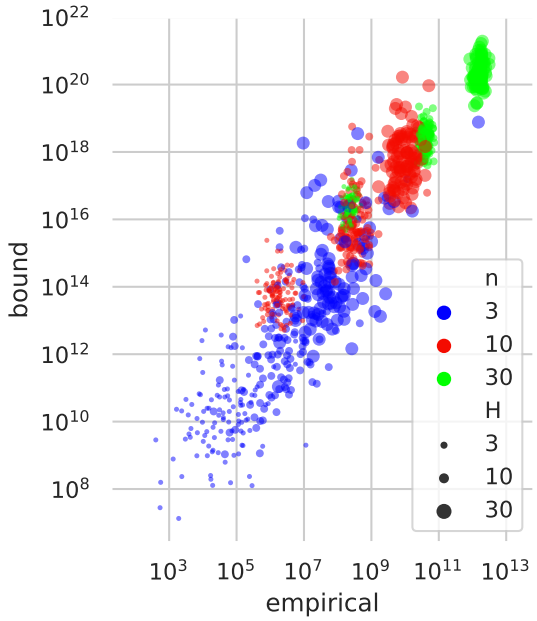


Figure 5.2: Scatter plot of empirical $\nu(\hat{g})$ (x-axis) and upper bound from Theorem 5.4.1 (y-axis) with varying state dimensionality n and time horizon H. Each point represents one random LQR problem.

Dimensionality parameters In all of the preceding experiments, we arbitrarily chose the state and action dimensions $n = 5,\ m = 3$ and time horizon $H = 10$. To visualize the variance for other values of these parameters, we generate 1000 random LQR problems with n and H each varying over the set $\{3, 10, 30\}$. We fix $m = \lceil n/2 \rceil$. Results are shown in Figure 5.2. The overall positive trend with a slope greater than 1 shows that the bound grows superlinearly with respect to the empirical, as expected. One interesting property is the tighter clustering for large values of n. This may be due to several eigenvalue distribution results in random matrix theory which state that, as $n \to \infty$, our random LQR problems tend to become similar up to a basis change (Tao, 2012).

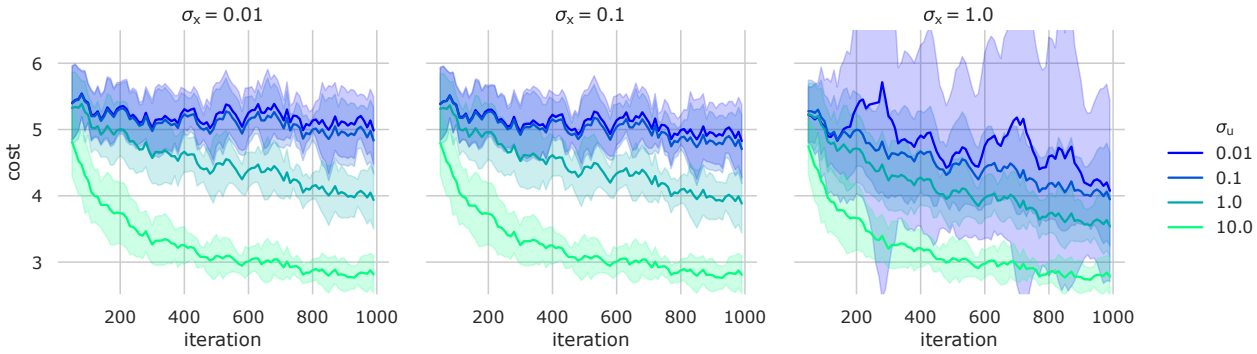


Figure 5.3: Learning curves of REINFORCE for a random LQR problem with varying scales of action noise σ_u and environment noise σ_x. Larger σ_u can improve learning, despite larger variance of $\hat{g}$.

5.5.1 RL policy optimality for varying Σ_u

The results in §5.5 suggest that, for a fixed Σ_x, the magnitude of Σ_u has a significant effect on $\nu(g)$. This is of practical interest because Σ_u is usually under control of the RL practitioner. It is therefore natural to ask if the change in variance corresponds to a change in the rate of convergence of REINFORCE. We test this empirically by executing REINFORCE in variants of one random LQR problem with different values of Σ_u and Σ_x. To avoid a confounding effect from larger Σ_u incurring greater penalty from the $-u_t^T R u_t$ term in r_t, we evaluate the trained policies in a modified version of the problem where $\Sigma_u = \Sigma_x = \mathbf{0}$. It can be shown that the optimal $K^\star$ for the stochastic problem is also optimal for the deterministic problem, so each problem variant should converge to the same evaluation returns in the limit.

We initialize K by perturbing the elements of the LQR-optimal controller with i.i.d. Gaussian noise and scaling the perturbation until $\rho(A + BK) \approx 0.98$. After every 10 iterations of REINFORCE, we evaluate the current policy in the noise-free environment. For each (Σ_u, Σ_x) pair, we repeat this experiment 10 times with different random seeds. The random seed only affects the ϵ_t^u, ϵ_t^x and x_1 samples. The aggregate data are shown in Figure 5.3. Shaded bands correspond to one standard deviation across the separate runs of REINFORCE. The lowercase σ_u, σ_x refer to scaling factors applied to the initial samples of Σ_u, Σ_x in the random LQR problem.

The effect is different than one would predict under the hypothesis that the performance of REINFORCE is mainly dictated by its estimation variance. For all values of Σ_x in the experiment, problems with larger Σ_u converge faster—whereas Figure 5.1a would suggest that the "optimal" value of Σ_u changes with respect to Σ_x. The fact that larger Σ_u tends to make REINFORCE converge faster is not obvious, given the Σ_u^{-1} term in $\hat{g}$ (5.4). Also, when Σ_u is very small and Σ_x is very large, the algorithm becomes unstable and sees large variations across different random seeds. For the middle values $\Sigma_u \in \{0.1, 1.0\}$, we observe that larger Σ_x causes faster convergence.

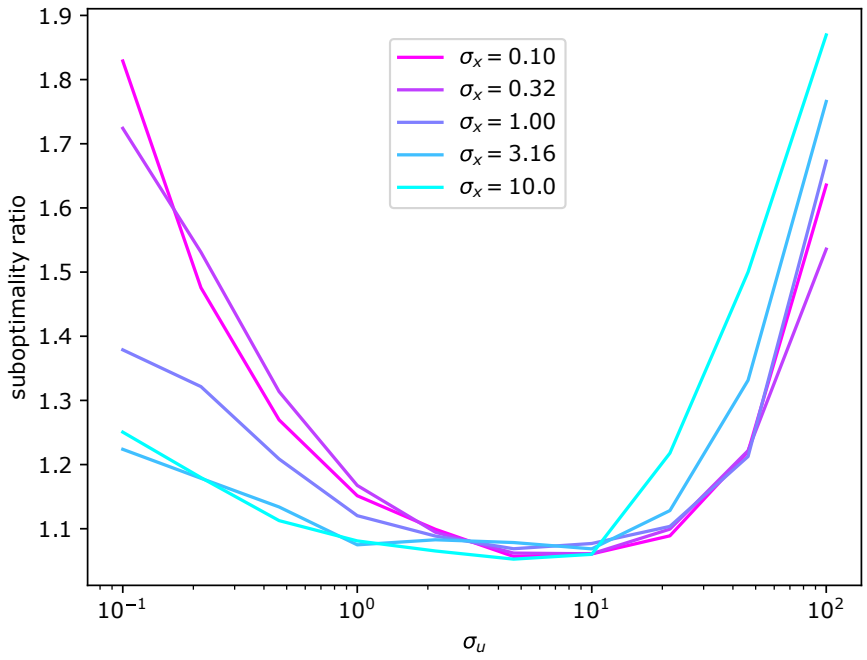


Figure 5.4: Suboptimality ratios of the policy after 1000 iterations of REINFORCE for a random LQR problem with varying scales of action noise σ_u and environment noise σ_x.

An alternate visualization of the same experiment is given in Figure 5.4. Here, as in Figure 5.1, the horizontal axis corresponds to the action variance σ_u and the line colors correspond to different values of the environment dynamics variance σ_x. However, instead of plotting the gradient estimate variance on the vertical axis as in Figure 5.1, here we plot the suboptimality ratio of the final policy after running

REINFORCE for a fixed number of iterations. The ratio is taken relative to the LQR-optimal policy for the infinite-horizon version of the problem.

In this plot we include results for larger values of σ_u than were shown in Figure 5.3. We observe that the suboptimality does increase for large enough values of σ_u, so the effect is not monotonic as Figure 5.3 might suggest. However, the value of σ_u for which the suboptimality ratio is closest to 1 does not move significantly for different values of σ_x. We remark further on these results in §5.8.

5.6 Proof of Theorem 5.4.1

In this section, we provide the detailed derivation of the upper bound stated in Theorem 5.4.1. We define the following notations: let δ_t^x and δ_t^u be independent random vectors that follow $\mathcal{N}(\mathbf{0}, I_n)$ and $\mathcal{N}(\mathbf{0}, I_m)$ respectively for all t. The ϵ_t^x and ϵ_t^u defined in Equations (5.1) and (5.3) can be written as $\epsilon_t^x = \Sigma_x^{\frac{1}{2}} \delta_t^x$ and $\epsilon_t^u = \Sigma_u^{\frac{1}{2}} \delta_t^u$. We will use the following steps to upper-bound $\mathbb{E}\left[\mathrm{tr}\left(\hat{g}^\top \hat{g}\right)\right]$:

1. Bound $\mathrm{tr}\left(\hat{g}^\top \hat{g}\right)$ by P, a polynomial of $d_t^x \triangleq \|\delta_t^x\|$ and $d_t^u \triangleq \|\delta_t^u\|$ for $t = 1, 2, \ldots, H$. We restrict that P should have only nonnegative coefficients. Since we assume δ_t^x and δ_t^u are independent random vectors with standard normal distribution, d_t^x and d_t^u will be independent random variables following the $\chi(n)$ and $\chi(m)$ distributions respectively.

2. Bound the sum of the (already nonnegative) coefficients of P by substituting all of the $d_t^x,\ d_t^u$ with one. More formally, let

$$P(\{d_t^x, d_t^u\}_{t=1}^H) = \sum_i C_i \prod_{t=1}^H (d_t^x)^{\alpha_{t,i}} \prod_{t=1}^H (d_t^u)^{\beta_{t,i}},$$

where $\prod_{t=1}^{H}(d_t^u)^{\alpha_{t,i}}\prod_{t=1}^{H}(d_t^x)^{\beta_{t,i}}$ is the i-th monomial in P, and C_i is its nonnegative coefficient. Then we calculate $\sum_i C_i$ by substituting all d_t^x, d_t^u with 1. We use the notation $\mathbb{1}(P)$ to denote this operation (we define the $\mathbb{1}(\cdot)$ operator analogously for expressions other than P itself):

$$\mathbb{1}(P) = P|_{d_t^x=1,\ d_t^u=1,\ \forall t} = \sum_i C_i.j$$

3. Bound for the expectation of all monomials in P, i.e., find M such that

$$\mathbb{E}\left[\prod_{t=1}^{H}(d_t^x)^{\alpha_{t,i}}\prod_{t=1}^{H}(d_t^u)^{\beta_{t,i}}\right] \le M, \quad \forall i.$$

With the three steps above, we can then bound $\mathbb{E}\left[\mathrm{tr}\left(\hat{g}^\top\hat{g}\right)\right] \le \mathbb{E}[P] \le \sum_i C_i M = M\mathbb{1}(P)$. To calculate M, we can use the known formula for the k-th moment of a χ random variable.

Recall that

$$\hat{g} = \left(\sum_{t=1}^{H}\Sigma_u^{-1}\epsilon_t^a x_t^\top\right)\left(\sum_{t=1}^{H} r_t\right),$$

and thus

$$\mathrm{tr}\left(\hat{g}^\top\hat{g}\right) = \underbrace{\mathrm{tr}\left[\left(\sum_{t=1}^{H}\Sigma_u^{-1}\epsilon_t^u x_t^\top\right)^\top\left(\sum_{t=1}^{H}\Sigma_u^{-1}\epsilon_t^u x_t^\top\right)\right]}_{\mathbf{Term}_1}\underbrace{\left(\sum_{t=1}^{H} r_t\right)^2}_{\mathbf{Term}_2}.$$

Thanks to the property $\mathbb{1}(P) = \mathbb{1}(P_1)\mathbb{1}(P_2)$ for polynomial $P \equiv P_1P_2$, and $\mathbb{1}(P) = \mathbb{1}(P_1) + \mathbb{1}(P_2)$ for $P \equiv P_1 + P_2$, we do not need to directly find P and bound $\mathbb{1}(P)$. Below in Section 5.6.1, we first bound $\|x_t\|$ by a polynomial $\lceil x_t\rceil$ of $\{d_\tau^s, d_\tau^a\}$, and then find $\mathbb{1}(\lceil x_t\rceil)$. In Section 5.6.2 and 5.6.3, we further upper bound $\mathbb{1}(\mathbf{Term}_1)$ and $\mathbb{1}(\mathbf{Term}_2)$ with the help of $\mathbb{1}(\lceil x_t\rceil)$. Then finally we obtain an upper bound for $\mathbb{E}[\mathrm{tr}\left(\hat{g}^\top\hat{g}\right)]$ as $M\mathbb{1}(\mathbf{Term}_1)\mathbb{1}(\mathbf{Term}_2)$.

5.6.1 Bounding $\|x_t\|$

In this section, we bound $\|x_t\|$ from above. Although the spectral radius $\rho(A+BK)$ determines the asymptotic stability of the closed-loop system, it guarantees little about the transient behavior–for example, for any $x > 1$ and $0 < \epsilon < 1$, the matrix

$$A = \begin{bmatrix} \epsilon & x \\ 0 & \epsilon \end{bmatrix}$$

has the properties $\rho(A) < 1,\ \|A\| > x$. Therefore, while the state magnitude $\|x_t\|$ is bounded by the operator norm $\|A + BK\|^{t-1}$, it is too restrictive to require $\|A + BK\| < 1$. Instead, we will use the following result:

Lemma 5.6.1 (Trefethen and Embree (2005))**.** *Let $A \in \mathbb{R}^{n\times n}$ be a matrix with $\rho(A) < 1$. Then there exists $\mu > 0$ such that, for all $k \in \mathbb{N}$,*

$$\|A^k\| \leq \mu\rho(A)^k. \tag{5.5}$$

μ is bounded by the "resolvent condition" $\mu \leq 2enr(A)$, where e is the exponential constant and

$$r(A) = \sup_{z\in\mathbb{C},\ |z|>1} (|z| - 1)\|(zI - A)^{-1}\|. \tag{5.6}$$

The derivation and interpretation of (5.6) is a deep subject related to the matrix *pseudospectrum*, covered extensively by Trefethen and Embree (2005). Intuitively, $r(A)$ is large if a small perturbation $\epsilon \in \mathbb{R}^{n\times n}$ would cause $\rho(\epsilon + A) > 1$.

Expanding the state transition function in Equation (5.1) with the linear stochastic policy in Equation (5.3), we get

$$x_t = (A + BK)^{t-1}x_1 + \sum_{\tau=1}^{t-1}(A + BK)^{t-\tau-1}(\Sigma_x^{\frac{1}{2}}\delta_\tau^x + B\Sigma_u^{\frac{1}{2}}\delta_\tau^u). \tag{5.7}$$

Recall that $\|\cdot\|$ denotes the ℓ_2 norm for vectors and the $\ell_2 - \ell_2$ operator norm for matrices. By Lemma 5.6.1, there exists μ such that $\|(A+BK)^k\| \leq \mu\rho(A)^k$. Let $f = \rho(A+BK)$, $\sigma_x^2 = \|\Sigma_x\|$, $\sigma_u^2 = \|\Sigma_u\|$, and $b = \|B\|$. By repeatedly applying the triangle inequality,

$$\begin{aligned}
\|x_t\| &= \left\|(A+BK)^{t-1}x_1 + \sum_{\tau=1}^{t-1}(A+BK)^{t-\tau-1}(\Sigma_x^{\frac{1}{2}}\delta_\tau^x + B\Sigma_u^{\frac{1}{2}}\delta_\tau^u)\right\| \\
&\leq \left\|(A+BK)^{t-1}x_1\right\| + \sum_{\tau=1}^{t-1}\left\|(A+BK)^{t-\tau-1}(\Sigma_x^{\frac{1}{2}}\delta_\tau^x + B\Sigma_u^{\frac{1}{2}}\delta_\tau^u)\right\| \\
&\leq \mu f^{t-1}\|x_1\| + \mu\sum_{\tau=1}^{t-1} f^{t-\tau-1}(\sigma_x d_\tau^x + b\sigma_u d_\tau^u).
\end{aligned} \tag{5.8}$$

We denote the final bound in (5.8) as $\lceil x_t \rceil$. The bound $\lceil x_t \rceil$ is linear in the random variables $\{d_t^x, d_t^u\}_{t=1}^H$ with only positive coefficients. Furthermore,

$$\begin{aligned}
\mathbb{1}(\lceil x_t \rceil) &= \mu f^{t-1}\|x_1\| + \mu\sum_{\tau=1}^{t-1} f^{t-\tau-1}(\sigma_x + b\sigma_u) \\
&= \mu f^{t-1}\|x_1\| + \mu\sum_{\tau=0}^{t-2} f^{\tau}(\sigma_x + b\sigma_u).
\end{aligned} \tag{5.9}$$

5.6.2 Bounding Term$_1$

In this subsection we show that

$$\operatorname{tr}\left[\left(\sum_{t=1}^H \Sigma_u^{-1}\epsilon_t^u x_t^\top\right)^\top\left(\sum_{t=1}^H \Sigma_u^{-1}\epsilon_t^u x_t^\top\right)\right] \leq \|\Sigma_u^{-1}\|\left(\sum_{t=1}^H d_t^u\|x_t\|\right)^2.$$

Let $\xi_t = \Sigma_u^{-1}\epsilon_t^u$. Then

$$\begin{aligned}
\operatorname{tr}\left[\left(\sum_{t=1}^{H}\xi_t x_t^\top\right)^\top\left(\sum_{t=1}^{H}\xi_t x_t^\top\right)\right] &= \operatorname{tr}\left[\sum_{1\le i,j\le H} x_i\xi_i^\top\xi_j x_j^\top\right] \\
&= \sum_{1\le i,j\le H}\xi_i^\top\xi_j x_i^\top x_j \\
&\le \sum_{1\le i,j\le H}|\xi_i^\top\xi_j||x_i^\top x_j| \\
&= \sum_{1\le i,j\le H}|\delta_i^{u\top}\Sigma_u^{-1}\delta_j^u||x_i^\top x_j| \qquad (5.10)\\
&\le \sum_{1\le i,j\le H}\|\Sigma_u^{-1}\|\|\delta_i^u\|\|\delta_j^u\|\|x_i\|\|x_j\| \\
&= \|\Sigma_u^{-1}\|\left(\sum_{t=1}^{H}\|\delta_t^u\|\|x_t\|\right)^2 \\
&\le \|\Sigma_u^{-1}\|\left(\sum_{t=1}^{H}d_t^u\lceil x_t\rceil\right)^2,
\end{aligned}$$

in which we made use of the circulant property of the trace, the Cauchy-Schwarz inequality, and the fact that $\Sigma^{\frac{1}{2}}\Sigma^{-2}\Sigma^{\frac{1}{2}} = \Sigma^{-1}$ for positive semidefinite Σ.

5.6.3 Bounding Term$_2$

We now bound $C \triangleq \sum_{t=1}^{H} -r_t$ from above. Note that $-r_t \geq 0$, since $Q \succeq 0$ and $R \succ 0$. Let $q = \|Q\|$, $r = \|R\|$, and $k = \|K\|$. Then

$$\begin{aligned} C = \sum_{t=1}^{H} -r_t &= \sum_{t=1}^{H} x_t^\top Q x_t + u_t^\top R u_t \\ &\leq \sum_{t=1}^{H} q\|x_t\|^2 + r\|u_t\|^2 = \sum_{t=1}^{H} q\|x_t\|^2 + r\|Kx_t + \Sigma_u^{\frac{1}{2}}\delta_t^u\|^2 \\ &\leq \sum_{t=1}^{H} q\|x_t\|^2 + r(k\|x_t\| + \sigma_u\|\delta_t^u\|)^2 \\ &\leq \sum_{t=1}^{H} q\|x_t\|^2 + 2rk^2\|x_t\|^2 + 2r\sigma_u^2\|\delta_t^u\|^2 \\ &\leq (q + 2rk^2)\sum_{t=1}^{H} \lceil x_t \rceil^2 + 2r\sigma_u^2 \sum_{t=1}^{H} (d_t^u)^2, \end{aligned} \tag{5.11}$$

where the triangle inequality and the fact $(a+b)^2 \leq 2(a^2+b^2)$ are used above. This bound on C is a quadratic polynomial in the d^x, d^u.

5.6.4 Combining bounds

Combining Section 5.6.2 and Section 5.6.3, we have

$$\operatorname{tr}\left(\hat{g}^\top \hat{g}\right) \leq P = C^2 \|\Sigma_u^{-1}\| \left(\sum_{t=1}^{H} d_t^u \lceil x_t \rceil\right)^2. \tag{5.12}$$

For brevity, let $\alpha = \|\Sigma_u^{-1}\|$, $\beta = q + 2rk^2$, $\gamma = 2r\sigma_u^2$, $\sigma = \sigma_x + b\sigma_u$, and $H' \triangleq \min\left\{H, \frac{1}{1-f}\right\}$. The term H' reflects the stability of the closed-loop system: if highly stable ($f \ll 1$), we have $H' \ll H$, but when approaching instability ($f \to 1$), H' approaches H.

We expand P and substitute $d_t^x = 1,\ d_t^u = 1$ for all t to compute the sum of P's coefficients, using the notation $\mathbb{1}(\cdot)$ for the transformation of replacing all d with 1. We first bound $\mathbb{1}(C^2)$:

$$\begin{aligned}\mathbb{1}(C^2) \le \left(\gamma H + \beta \sum_{t=1}^{H} \mathbb{1}(\lceil x_t \rceil)^2\right)^2 &\le \left(\gamma H + \beta\mu^2 \sum_{t=1}^{H} \left[f^{t-1}\|x_1\| + \sigma H'\right]^2\right)^2 \\ &\le \left(\gamma H + 2\beta\mu^2 \sum_{t=1}^{H} \left[f^{2t-2}\|x_1\|^2 + \sigma^2 H'^2\right]\right)^2 \\ &\le \left(\gamma H + 2\beta\mu^2 (H'\|x_1\|^2 + \sigma^2 H H'^2)\right)^2\end{aligned} \tag{5.13}$$

where the result is obtained by repeatedly applying the fact $(a+b)^2 \le 2(a^2+b^2)$. Next,

$$\begin{aligned}\mathbb{1}(\left(\sum_{t=1}^{H} d_t^u \lceil x_t \rceil\right)^2) &\le \left(\mu \sum_{t=1}^{H} f^{t-1}\|x_1\| + \sigma H'\right)^2 \\ &\le \mu^2 H'^2 (\|x_1\| + \sigma H)^2.\end{aligned} \tag{5.14}$$

Finally,

$$\begin{aligned}\mathbb{1}(P) &\le \alpha\mu^2 H'^2 \left(\gamma H + 2\beta\mu^2 (H'\|x_1\|^2 + \sigma^2 H H'^2)\right)^2 (\|x_1\| + \sigma H)^2 \\ &= 4\|\Sigma_u^{-1}\|\mu^2 H'^2 \left(r\sigma_u^2 H + \mu^2 (q + 2rk^2)(H'\|x_1\|^2 + \sigma^2 H H'^2)\right)^2 (\|x_1\| + \sigma H)^2 \\ &\triangleq \overline{\mathbb{1}(P)}.\end{aligned} \tag{5.15}$$

$\overline{\mathbb{1}(P)}$ is an order-8 polynomial in the d^s, d^a. The formula for the 8th moment of a $\chi(n)$ random variable is

$$\mathbb{E}[X^8] = n(n+2)(n+4)(n+6),$$

so $\mathbb{E}[\mathrm{tr}\left(\hat{g}^\top \hat{g}\right)] \le \overline{\mathbb{1}(P)} \cdot O(\bar{n}^4)$, where $\bar{n} = \max(n, m)$.

(Since we are summing the variances of $O(\bar{n}^2)$ random variables in $\hat{g}$, we would expect scaling of no less than $\bar{n}^2$ compared to the scalar case.)

5.7 Proof of Theorem 5.4.2

In this section, we provide a lower bound for $\mathbb{E}[\hat{g}^2]$ in the scalar case, $m = n = 1$. The matrices A, B, K, Q, R are thus denoted as a, b, k, q, r here (notice that this notation r is different from the notation r_t for reward). Other notations follow the definitions in § 5.6. We will analyze the case when $0 \leq a + bk < 1$.

Lemma 5.7.1.

$$\mathbb{E}\left[\left(\sum_{t=1}^{H} \frac{\epsilon_t^u x_t}{\sigma_u^2}\right)^2 \left(\sum_{t=1}^{H} r_t\right)^2\right] \geq \mathbb{E}\left[\left(\sum_{t=1}^{H} \frac{\epsilon_t^u x_t}{\sigma_u^2}\right)^2\right] \mathbb{E}\left[\left(\sum_{t=1}^{H} r_t\right)^2\right]. \tag{5.16}$$

Proof. When $a+bk \geq 0$, all terms have positive coefficients. Rename the $2H$ random variables $\{\delta_t^x, \delta_t^u\}_{t=1}^H$ as $x_1, \ldots, x_{2H}$. We can see that a monomial on the right-hand side:

$$\mathbb{E}\left[x_1^{\alpha_1} \cdots x_{2H}^{\alpha_{2H}}\right] \mathbb{E}\left[x_1^{\beta_1} \cdots x_{2H}^{\beta_{2H}}\right]$$

corresponds to the monomial on the left-hand side:

$$\mathbb{E}\left[x_1^{\alpha_1+\beta_1} \cdots x_{2H}^{\alpha_{2H}+\beta_{2H}}\right].$$

The x_i are independent zero-mean normal random variables, so the property $\mathbb{E}[x_i^\alpha]\,\mathbb{E}[x_i^\beta] \leq \mathbb{E}[x_i^{\alpha+\beta}]$ holds for any non-negative integers α, β. Combining with the fact that all coefficients are non-negative shows the lemma. □

In the following two subsections, we lower bound the two terms on the right-hand side of Eq. (5.16) separately. Note that the first term can be simplified as

$$\left(\sum_{t=1}^{H}\frac{\epsilon_t^u x_t}{\sigma_u^2}\right)^2=\left(\sum_{t=1}^{H}\frac{\delta_t^u x_t}{\sigma_u}\right)^2=\frac{1}{\sigma_u^2}\left(\sum_{t=1}^{H}\delta_t^u x_t\right)^2.$$

5.7.1 Lower bounding $\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u x_t\right)^2\right]$

By the expansion of x_t in Eq. (5.7), we have

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u x_t\right)^2\right]&=\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u\left((a+bk)^{t-1}x_1+L_t\right)\right)^2\right]\\&\geq\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u(a+bk)^{t-1}x_1\right)^2\right]+\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u L_t\right)^2\right],\end{aligned}$$

where $L_t\triangleq\sum_{\tau=1}^{t-1}(a+bk)^{t-1-\tau}(\sigma_x\delta_\tau^s+b\sigma_u\delta_\tau^a)$. The first term is equal to

$$\sum_{t=1}^{H}(a+bk)^{2t-2}x_1^2=\frac{1-(a+bk)^{2H}}{1-(a+bk)^2}x_1^2.$$

The second term can be further written as

$$\begin{aligned}&\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u\sum_{\tau=1}^{t-1}(a+bk)^{t-1-\tau}(\sigma_x\delta_\tau^s+b\sigma_u\delta_\tau^a)\right)^2\right]\\&=\mathbb{E}\left[\sum_{t=1}^{H}(\delta_t^u)^2\left(\sum_{\tau=1}^{t-1}(a+bk)^{t-1-\tau}(\sigma_x\delta_\tau^s+b\sigma_u\delta_\tau^a)\right)^2\right]\\&=\mathbb{E}\left[\sum_{t=1}^{H}\left(\sum_{\tau=1}^{t-1}(a+bk)^{t-1-\tau}(\sigma_x\delta_\tau^s+b\sigma_u\delta_\tau^a)\right)^2\right]\\&=\mathbb{E}\left[\sum_{t=1}^{H}\sum_{\tau=1}^{t-1}(a+bk)^{2t-2-2\tau}(\sigma_x^2+b^2\sigma_u^2)\right]\\&=(\sigma_x^2+b^2\sigma_u^2)\mathbb{E}\left[\sum_{t=1}^{H}\frac{1-(a+bk)^{2t-2}}{1-(a+bk)^2}\right]\end{aligned}$$

$$= (\sigma_x^2 + b^2\sigma_u^2)\left(\frac{H}{1-(a+bk)^2} - \frac{1-(a+bk)^{2H}}{(1-(a+bk)^2)^2}\right).$$

In the above several equalities, we use the independence among δ_t^u, δ_t^x. Combining two terms, we get

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{t=1}^{H}\delta_t^u x_t\right)^2\right] &\geq \frac{x_1^2 + H(\sigma_x^2+b^2\sigma_u^2)}{1-(a+bk)^2} - \frac{(a+bk)^{2H}}{1-(a+bk)^2}x_1^2 - (\sigma_x^2+b^2\sigma_u^2)\frac{1-(a+bk)^{2H}}{(1-(a+bk)^2)^2} \\
&= \left(\frac{1-(a+bk)^{2H}}{1-(a+bk)^2}x_1^2 + \left(H - \frac{1-(a+bk)^{2H}}{1-(a+bk)^2}\right)\frac{\sigma_x^2+b^2\sigma_u^2}{1-(a+bk)^2}\right) \\
&\approx \begin{cases} \frac{x_1^2+H(\sigma_x^2+b^2\sigma_u^2)}{1-(a+bk)^2} & \text{when } H \gg \frac{1}{1-(a+bk)^2} \\ H\left(x_1^2 + H(\sigma_x^2+b^2\sigma_u^2)\right) & \text{when } H \ll \frac{1}{1-(a+bk)^2} \end{cases} \\
&\approx \min\left\{H, \frac{1}{1-(a+bk)^2}\right\}\left(x_1^2 + H(\sigma_x^2+b^2\sigma_u^2)\right) \\
&= H'\left(x_1^2 + H(\sigma_x^2+b^2\sigma_u^2)\right).
\end{aligned}$$

5.7.2 Lower bounding $\mathbb{E}\left[\left(\sum_{t=1}^{H} r_t\right)^2\right]$

5.7.3

We first lower bound this term by

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{t=1}^{H} r_t\right)^2\right] \geq \mathbb{E}\left[\sum_{t=1}^{H} r_t\right]^2 &= \mathbb{E}\left[\sum_{t=1}^{H} qx_t^2 + r(kx_t + \sigma_u\delta_t^u)^2\right]^2 \\
&= \mathbb{E}\left[\sum_{t=1}^{H}(q+rk^2)x_t^2 + 2rkx_t\sigma_u\delta_t^u + r\sigma_u^2\delta_t^{a2}\right]^2 \\
&= \left((q+rk^2)\,\mathbb{E}\left[\sum_{t=1}^{H} x_t^2\right] + Hr\sigma_u^2\right)^2
\end{aligned}$$

$$
\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{H} x_t^2\right] &= \mathbb{E}\left[\sum_{t=1}^{H} \left((a+bk)^{t-1}x_1 + L_t\right)^2\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{H} \left((a+bk)^{2t-2}x_1^2 + 2(a+bk)^{t-1}x_1 L_t + L_t^2\right)\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{H}(a+bk)^{2t-2}x_1^2 + \sum_{t=1}^{H}\left(\sum_{\tau=1}^{t-1}(a+bk)^{t-1-\tau}(\sigma_x\delta_t^x + b\sigma_u\delta_t^u)\right)^2\right] \\
&\quad \text{(the middle term } \mathbb{E}[(a+bk)^{t-1}x_1 L_t] \text{ is zero because } L_t \text{ is a sum of zero-mean RVs)} \\
&= \frac{1-(a+bk)^{2H}}{1-(a+bk)^2}x_1^2 + \mathbb{E}\left[\sum_{t=1}^{H}\sum_{\tau=1}^{t-1}(a+bk)^{2t-2-2\tau}(\sigma_2^2 + b^2\sigma_u^2)\right] \\
&= \frac{1-(a+bk)^{2H}}{1-(a+bk)^2}x_1^2 + (\sigma_x^2 + b^2\sigma_u^2)\left(\frac{H}{1-(a+bk)^2} - \frac{1-(a+bk)^{2H}}{(1-(a+bk)^2)^2}\right) \\
&\approx \min\left\{H, \frac{1}{1-(a+bk)^2}\right\}\left(x_1^2 + H(\sigma_x^2 + b^2\sigma_u^2)\right) \\
&= H'\left(x_1^2 + H(\sigma_x^2 + b^2\sigma_u^2)\right).
\end{aligned}
$$

The second-to-last approximation is obtained similarly as in the previous subsection.

5.7.4 Combining

Combining the results in previous two subsections, we get the final lower bound on $\mathbb{E}[\hat{g}^2]$:

$$
\begin{aligned}
\mathbb{E}[\hat{g}^2] &= \mathbb{E}\left[\left(\sum_{t=1}^{H}\frac{\delta_t^u x_t}{\sigma_u}\right)^2\left(\sum_{t=1}^{H} r_t\right)^2\right] \geq \mathbb{E}\left[\left(\sum_{t=1}^{H}\frac{\delta_t^u x_t}{\sigma_u}\right)^2\right]\mathbb{E}\left[\left(\sum_{t=1}^{H} r_t\right)^2\right] \\
&\geq \Omega(c_1^2 c_2^2),
\end{aligned}
$$

where (recall $\sigma \triangleq \sigma_x + b\sigma_u$)

$$
\begin{aligned}
c_1 &= \frac{1}{\sigma_u}\left(|x_1| + \sigma\sqrt{H}\right)\sqrt{H'}, \\
c_2 &= r\sigma_u^2 H + (q + rk^2)(x_1^2 + \sigma^2 H)H'.
\end{aligned}
$$

5.8 Discussion

In this chapter, we derived bounds on the variance of the REINFORCE policy gradient estimator in the stochastic linear-quadratic control setting. Our upper bound is fully general, while our lower bound applies to the scalar case at a stationary point. The bounds match with respect to all system parameters except the time horizon H and closed-loop spectral radius $\rho(A + BK)$. We compared our bound prediction to the empirical variance in a variety of experimental settings, finding a close qualitative match in the parameters for which the bounds are tight.

Our experiments in § 5.5.1 plotting the empirical convergence rate of REINFORCE suggest that the effect of action noise Σ_u on the overall RL performance is not fully captured by its effect on the variance. An interesting direction for future work would be to investigate the role of Σ_u more closely and attempt to disentangle its effect on gradient magnitude, variance, exploration, and regularization. Such an analysis could lead to improved variance reduction methods or algorithms that manipulate Σ_u to speed up the RL optimization.

Chapter 6

Suboptimal Coverings

6.1 Introduction

In this chapter, we present work motivated by one of the questions in the discussion of §3.5: *How hard is it to represent a good multi-system policy?* We propose the α-suboptimal covering number to characterize multi-system control problems where the set of dynamical systems and/or cost functions is infinite, analogous to the cardinality of finite system sets. We study suboptimal covering numbers for continuous-time linear-quadratic regulator problems (§ 2.9.2.5) and construct a class of multi-system LQR problems amenable to analysis. For the scalar case, we show logarithmic dependence on the "breadth" of the space. For the matrix case, we present empirical results and intermediate theoretical results towards an equivalent theorem.

In the multi-system control paradigm described in § 2.6, we did not specify any particular properties of the set of MDPs Φ. Let us consider the cardinality of the system set, using the example of a mobile robot as motivation. If the system set is finite, like selecting between "map an environment" and "deliver a package", then its size is naturally quantified by the number of systems. If the system set is infinite, like delivering packages with arbitrary mass and inertial properties, then its size is not so easily quantified. Even if Φ is equipped with a metric or measure, these structures may be only weakly linked to the diversity of behavior required for good performance on all systems.

Most work presented in this chapter was originally published in Preiss and Sukhatme (2021). The efforts towards the matrix case in § 6.7 are new.

Suppose a multi-system policy that maps state and system parameters directly to actions is to be selected from a parameterized family of functions. As the system space expands from a singleton set, we expect to need a more expressive class of functions to represent a good multi-system policy. In this work, we propose the *α-suboptimal covering number* to capture this idea. For a system space Φ and a suboptimality ratio $\alpha > 1$, we define $N_\alpha^{\text{cov}}(\Phi)$ as the size of the smallest set of single-system policies $\mathcal{C}$ such that for every $\phi \in \Phi$, at least one $\pi \in \mathcal{C}$ has a cost ratio no greater than α relative to the optimal policy for ϕ. If the policies in $\mathcal{C}$ are parameterized functions, then $\mathcal{C}$ provides an upper bound on the number of parameters needed to represent an α-suboptimal multi-system policy. In switching-based adaptive control, where ϕ is unknown, a smaller $\mathcal{C}$ reduces the computational complexity of the controller and can reduce the number of switches per unit of time (Hespanha et al., 2000).

To study suboptimal covering numbers in a concrete setting, we consider continuous-time LQR problems (§2.9.2.5). LQR problems are a common setting to analyze learning algorithms because detailed properties are known (see §5.2 for examples). This has led to new inquiries into their fundamental properties (Bu et al., 2019b). Our work follows the latter spirit. We construct a family of well-behaved multi-system LQR problems where Φ is controlled by a "breadth" parameter $\theta \in [1, \infty)$, and for which $N_\alpha^{\text{cov}}(\Phi_\theta)$ is finite and increasing in θ. For the special case of a scalar LQR problem, we derive matching logarithmic upper and lower bounds on $N_\alpha^{\text{cov}}(\Phi_\theta)$ as a function of θ.

As an effort towards analogous bounds for the matrix case, we present empirical results intended to shed light on the problem structure. For the upper bound, we analyze properties of a logical extension of our scalar cover. For the lower bound, we visualize suboptimal neighborhoods for two choices of "extremal" systems, revealing surprising topological behavior for one choice. Finally, we present some intermediate theoretical tools that may be useful for the matrix case, and more suboptimal neighborhood visualizations for a generalization of our multi-system LQR family to include variations in the state dynamics matrix A as well as the input matrix B.

This chapter is an initial step towards a comprehensive theory. In addition to a more complete picture of deterministic LQR systems, ideas of α-suboptimal coverings could be applied to a wide range of multi-system problems. We also hope they will lead to insights about function class expressiveness in learning-based multi-system control.

6.2 Problem setting

In this section we define α-suboptimal covering numbers with respect to an abstract multi-system control problem, independent of distinctions such as continuous vs. discrete time and stochastic vs. deterministic dynamics. We then instantiate these definitions for a particular class of LQR problems.

Notation We consider a family of MDPs as defined in § 2.6, with state space $\mathcal{X}$, action space $\mathcal{U}$, and system space Φ. We also require a class of reference policies $\Pi_{\mathrm{ref}} \subseteq \mathcal{U}^{\mathcal{X}}$ and a strictly positive objective function $J : \Phi \times \mathcal{U}^{\mathcal{X}} \mapsto \mathbb{R}_{>0}$. The partial application of J for $\phi \in \Phi$ is denoted by $J_\phi : \mathcal{U}^{\mathcal{X}} \mapsto \mathbb{R}$. The optimal reference cost for an system is denoted by $J_\phi^\star = \inf_{\pi \in \Pi_{\mathrm{ref}}} J_\phi(\pi)$.

Definition 6.2.1. Consider a multi-system optimal control problem $(\mathcal{X}, \mathcal{U}, \Phi, \Pi_{\mathrm{ref}})$ and a suboptimality ratio $\alpha > 1$. Given a policy $\pi : \mathcal{X} \mapsto \mathcal{U}$, we define its *$\alpha$-suboptimal neighborhood* as

$$\mathscr{N}_\alpha(\pi) = \left\{ \phi \in \Phi : \frac{J_\phi(\pi)}{J_\phi^\star} \leq \alpha \right\}.$$

A set of policies $\mathcal{C} \subseteq \mathcal{U}^{\mathcal{X}}$ is an *α-suboptimal cover* of Φ if

$$\bigcup_{\pi \in \mathcal{C}} \mathscr{N}_\alpha(\pi) = \Phi.$$

(Note that $\mathcal{C}$ need not belong to Π_{ref} – these definitions are still meaningful in the "improper" case.) The *α-suboptimal covering number* of Φ, denoted $N_\alpha^{\text{cov}}(\Phi)$, is the size of the smallest finite α-suboptimal cover of Φ if one exists, or ∞ otherwise.

Standard LQR problem In this chapter, we analyze suboptimal coverings for continuous-time, deterministic, infinite-horizon, LQR problems, as defined in §2.9.2.5. Whereas in (2.30) the LQR cost was defined for a particular initial state $x(0)$, in this chapter we define the overall policy cost for a stabilizing controller $K \in \mathbb{R}^{m\times n}$ by

$$\begin{aligned} J(K) &= \mathop{\mathbb{E}}_{x(0)\sim\mathcal{N}(\mathbf{0},I)}\left[J_{x(0)}\right] \\ &= \mathop{\mathbb{E}}_{x(0)\sim\mathcal{N}(\mathbf{0},I)}\int_0^\infty \left[x(t)^\top Q x(t) + u(t)^\top R u(t)\right] \mathrm{d}t, \end{aligned} \tag{6.1}$$

in other words the expected cost when the initial state is distributed by a unit Gaussian. With this definition, if P solves the algebraic Riccati equation (2.31) and K is the optimal controller $K = -R^{-1}B^\top P$, then $J(K) = \mathrm{tr}[P]$. Furthermore, if K is an arbitrary stabilizing controller, we have (Mohammadi et al., 2019):

$$J(K) = \mathrm{tr}[(Q + K^\top R K)W], \tag{6.2}$$

where

$$W = \int_0^\infty e^{t(A+BK)^\top} e^{t(A+BK)} dt. \tag{6.3}$$

W can be computed by solving the Lyapunov equation

$$(A+BK)^\top W + W(A+BK) + I = \mathbf{0}.$$

Multi-dynamics LQR A fully general formulation of multi-system LQR would allow variations in each of (A, B, Q, R), but this creates redundancy. Any LQR problem where $Q \succ 0$ is equivalent via change of

coordinates to another LQR problem where $Q = I$ and $R = I$. To reduce redundancy, we consider only *multi-dynamics* LQR problems where $Q = I_{n\times n}$ and $R = I_{m\times m}$ in this work. The reference policy class is linear: $\Pi_{\text{ref}} = \mathbb{R}^{m\times n}$.

One way to then define a multi-dynamics LQR problem is by a simple product $\Phi = \mathbf{A}\times\mathbf{B}$ for some sets $\mathbf{A} \subseteq \mathbb{R}^{n\times n}$ and $\mathbf{B} \subseteq \mathbb{R}^{n\times m}$. However, it is not obvious how to design $\mathbf{A}$ and $\mathbf{B}$. To support an asymptotic analysis of $N_\alpha^{\text{cov}}(\Phi)$, the system space Φ should have a real-valued "breadth" parameter θ that sweeps from a single system to sets with arbitrarily large, but finite, covering numbers. Matrix norm balls are a popular representation of dynamics uncertainty in the robust control literature, but they can easily contain uncontrollable pairs, and removing the uncontrollable pairs can lead to an infinite covering number. For example, in the scalar problem

$$\mathbf{A} = \{a\},\ \mathbf{B} = [-\theta, 0) \cup (0, \theta],$$

where $a \neq 0$, it can be shown that no α-suboptimal cover is finite.

These properties are worrying, but the example $\mathbf{B}$ is pathological. The zero crossing is analogous to reversing the direction of force applied by an actuator in a physical system. Allowing B to become arbitrarily close to zero means the system can become arbitrarily close to uncontrollable. A more relevant multi-dynamics problem is variations in mass or actuator strength, whose signs are fixed. We formalize this idea with the following definition.

Definition 6.2.2. Fix $A \in \mathbb{R}^{n\times n}$ and a *breadth* parameter $\theta \geq 1$. Let $\mathbf{A} = \{A\}$ and let

$$\mathbf{B} = \{U\Sigma V^\top : \Sigma \in \mathbf{\Sigma}\}, \quad \text{where} \quad \mathbf{\Sigma} = \{\text{diag}(\sigma) : \sigma \in [\tfrac{1}{\theta}, 1]^d\}.$$

The matrices $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{m\times d}$ each have rank d, where $0 < d \leq \min\{n, m\}$. The tuple (A, U, V, θ) fully defines a *multi-system LQR problem in decomposed dynamics form*, or *DDF problem* for brevity.

We will abuse notation and associate Φ with both $\mathbf{A} \times \mathbf{B}$ and $\mathbf{\Sigma}$ when the meaning is clear from context. The continuity of the LQR cost (6.2) with respect to B and the compactness of Φ for any θ imply that $N_\alpha^{\text{cov}}(\Phi_\theta)$ is always finite. Variations in A are redundant in the scalar case where we focus our theoretical work in this chapter. The definition can be extended to include them in future work.

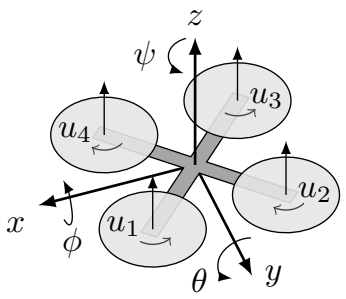


Figure 6.1: Quadrotor helicopter with position states x, y, z, attitude states θ, ϕ, ψ, and per-rotor thrust inputs u_1, u_2, u_3, u_4. The linearized dynamics at hover, subject to variations in mass, geometry, etc., can be expressed in decomposed dynamics form—see §6.2.

Linearized quadrotor example As an example of a realistic DDF problem, we consider the quadrotor helicopter illustrated in Figure 6.1. Near the hover state, its full nonlinear dynamics are well approximated by a linearization. The state is given by $x = (\mathbf{x}, \mathbf{v}, \mathbf{r}, \boldsymbol{\omega})$, where $\mathbf{x} \in \mathbb{R}^3$ is position, $\mathbf{v} \in \mathbb{R}^3$ is linear velocity, $\mathbf{r} \in \mathbb{R}^3$ is attitude Euler angles, and $\boldsymbol{\omega} \in \mathbb{R}^3$ is angular velocity. The inputs $u \in \mathbb{R}^4_{\geq 0}$ are the squared angular velocities of the propellers.

Many factors influence the response to inputs, including geometry, mass, moments of inertia, motor properties, and propeller aerodynamics. These can be combined and partially nondimensionalized into

four control authority parameters to form $\phi \in \Phi$. The hover state occurs at $x = \mathbf{0},\ u \propto \mathbf{1}$, where the constant input counteracts gravity. The linearized dynamics are given by

$$\dot{x} = \underbrace{\begin{bmatrix} \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & G & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{A} x + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \hat{e}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sigma_z & & & \\ & \sigma_\theta & & \\ & & \sigma_\phi & \\ & & & \sigma_\psi \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}}_{V^\top} u, \quad G = \begin{bmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where g is the gravitational constant and $\hat{e}_z = [0\ 0\ 1]^\top$. The parameters $(\sigma_z,\ \sigma_\theta,\ \sigma_\phi,\ \sigma_\psi)$ denote the thrust, roll, pitch, and yaw authority constants respectively. Since we use the convention $\sigma \in [\frac{1}{\theta}, 1]$, the maximum value of each constant can be varied by scaling the columns of U.

6.3 Related work

Suboptimal coverings are closely related to several topics in control theory. Robust control synthesis under parametric uncertainty (Dullerud and Paganini, 2000) can be interpreted as seeking a policy that performs well on all of Φ without observing the particular $\phi \in \Phi$. Most problem formulations in robust synthesis admit problem instances with no solution; the goal is to find a robust policy *if* one exists. Gain-scheduled control considers a multi-system setup identical to ours, while adaptive control control adds the complication that ϕ is not known to the policy.

Adaptive and gain-scheduled policies of the *self-tuning* type synthesize a single-system policy after estimating ϕ, but this relies on the assumption that control synthesis can be computed quickly (Åström and Wittenmark, 2013). In contrast, methods of the *multi-model* type use a precomputed set of policies (Murray-Smith and Johansen, 1997). All multi-model methods impose some kind of coverage condition on the policy set. In a *stable cover*, each $\phi \in \Phi$ is stabilized by at least one policy. Researchers often

focus more on the switching rule than the policy set. For example, Fu and Barmish (1986); Stilwell and Rugh (1999); Yoon et al. (2007) non-constructively assert the existence of a finite cover by continuity and compactness arguments. To address the need for small covers, Anderson et al. (2000); McNichols and Fadali (2003); Tan et al. (2004); Fekri et al. (2006); Du et al. (2012) propose constructive algorithms for various classes of Φ, sometimes with arguments for minimality but without bounds on the covering number. Jalali and Golmohammad (2012) bound stability covering numbers in terms of worst-case Vinnicombe metric distances and sensitivity function norms across the system set. The most closely related work to ours is from Fu (1996), who shows a tight bound of 2^n for the stability covering number of a relatively broad Φ. This result is complementary to ours: suboptimality is a stronger criterion than stability, but our class of Φ is more restrictive. We are not aware of prior work that bounds covering numbers in a setup based on local suboptimality, as opposed to a single global performance measure.

Multi-system control is also a popular topic in deep learning research, where it is often motivated by ideas of lifelong skill acquisition in robotics. Domain randomization methods follow the spirit of robust control (Peng et al., 2017), but usually optimize for the average case instead of a worst-case guarantee. Many methods where the policy observes ϕ use architectural constructs that can only be applied to finite system sets (Yang et al., 2017; Parisotto et al., 2016; Devin et al., 2017). A common approach for infinite system spaces is to treat ϕ as a vector input alongside the system state. Yu et al. (2017) and Chen et al. (2018) use this approach for dynamics parameters; Schaul et al. (2015) use it for navigation goals. There is evidence that policy class influences these methods: in a recent benchmark (Yu et al., 2019), the concatenated-input architecture that supports infinite system spaces trails the multi-head architecture that only supports finite system spaces. Other investigations into the difficulty of learning policies for multi-system control include methods to condition the multi-system optimization landscape (Yu et al., 2020) or balance disparate cost ranges (van Hasselt et al., 2016).

6.4 Theoretical results

In this section we show logarithmic upper and lower bounds on the growth of $N_\alpha^{\mathrm{cov}}(\Phi_\theta)$ in θ for scalar DDF problems. We present several intermediate results in matrix form because they are needed for our empirical results later. We begin with a key lemma in the framework of *guaranteed cost control* (GCC) from Petersen and McFarlane (1994), simplified for our use case.

Lemma 6.4.1 (GCC synthesis, Petersen and McFarlane (1994))**.** *Given the multi-system LQR problem defined by* $\mathbf{A} = \{A\}$, $\mathbf{B} = \{B_1\Delta + B_2 : \|\Delta\| \leq 1\}$, *where* $B_1, B_2 \in \mathbb{R}^{m\times p}$ *are arbitrary for arbitrary* p, *and the state cost matrix is* $Q \succ \mathbf{0}$, *if there exists* $\tau > 0$ *such that* $P \succ \mathbf{0}$ *solves the Riccati equation*

$$A^\top P + PA + P\left(\tfrac{1}{\tau}B_1B_1^\top - \tfrac{1}{1+\tau}B_2B_2^\top\right)P + Q = \mathbf{0}, \tag{6.4}$$

then the controller

$$K = -\frac{1}{1+\tau}B_2^\top P$$

has cost $J_B(K) \leq \mathrm{tr}(P)$ *for all* $B \in \mathbf{B}$. *Also,* $\mathrm{tr}(P)$ *is a convex function of* τ.

We use the notation $P, \tau, K = \texttt{GCC}(A, B_1, B_2, Q)$ to indicate that P, τ solve (6.4) and K is the corresponding controller. It is straightforward to show that any DDF problem can be expressed in the form required by Lemma 6.4.1 with additional constraints on Δ.

In the original presentation, Petersen and McFarlane (1994) treat B_1 as given, so they accept that (6.4) may have no solution. (For example, any time $B_2 = \mathbf{0}$ and A has unstable eigenvalues, due to uncontrollability.) Our application requires constructing values of B_1, B_2 that guarantee a solution, motivating the following lemma. We abbreviate the reference text Lancaster and Rodman (1995) as Lan95.

Lemma 6.4.2 (existence of α-suboptimal GCC)**.** *For the DDF problem (A, U, V, θ), if $B \in \mathbf{B}$ and $\alpha > 1$, then there exists $\tau > 0$ such that the GCC Riccati equation* (6.4) *with $B_1 = \tau B$ and $B_2 = B$ has a solution (P, τ) satisfying $\operatorname{tr}[P] \leq \alpha J_B^\star$.*

Proof. For this proof, it will be more convenient to write the algebraic Riccati equation as

$$A^\top P + PA - PDP + Q = \mathbf{0}, \tag{6.5}$$

where $D \succeq \mathbf{0}$. Let $\mathcal{D} = \{D \succeq \mathbf{0} : (A, D) \text{ is controllable}\}$. Controllability of (A, B) implies that $BB^\top \in \mathcal{D}$ (Lan95, Corollary 4.1.3). Let Ric_+ denote the map from $\mathcal{D}$ to the maximal solution of (6.5), which is continuous (Lan95, Theorem 11.2.1), and let

$$\mathcal{D}_\alpha = \{D \in \mathcal{D} : \operatorname{tr}[\mathrm{Ric}_+(D)] < \alpha J_B^\star\}.$$

The set $\mathcal{D}_\alpha$ is open in $\mathcal{D}$ by continuity and is nonempty because it contains $BB^\top$. Now define $B_1(\tau) = \tau B$ for $\tau \in (0, \frac{1}{2})$. The equivalent of D in the GCC Riccati equation (6.4) becomes

$$D(\tau) = -\tfrac{1}{\tau} B_1(\tau) B_1(\tau)^\top + \tfrac{1}{1+\tau} B_2 B_2^\top = \tfrac{1-\tau-\tau^2}{1+\tau} BB^\top.$$

As a positive multiple of $BB^\top$, we know $D(\tau) \in \mathcal{D}$, and because $\lim_{\tau \to 0} D(\tau) = BB^\top$, the set of τ for which $D(\tau) \in \mathcal{D}_\alpha$ is nonempty. Any such τ and $B_1(\tau)$ provide a solution. □

Finally, the following comparison result will be useful in several places.

Lemma 6.4.3 (ARE comparison lemma)**.** *Given two algebraic Riccati equations*

$$A^\top P + PA - PBB^\top P + Q = \mathbf{0} \quad \textit{and} \quad \tilde{A}^\top P + P\tilde{A} - P\tilde{B}\tilde{B}^\top P + \tilde{Q} = \mathbf{0},$$

with maximal solutions P and $\tilde{P}$, let $X = \begin{bmatrix} Q & A^\top \\ A & -BB^\top \end{bmatrix}$ and $\tilde{X} = \begin{bmatrix} \tilde{Q} & \tilde{A}^\top \\ \tilde{A} & -\tilde{B}\tilde{B}^\top \end{bmatrix}$. If $X \succeq \tilde{X}$, then $P \succeq \tilde{P}$.

Proof. Lan95, Corollary 9.1.6. □

6.4.1 Scalar upper bound

We are now ready to bound the covering number for scalar systems. The first lemma bounding $J^\star_{a,b}$ will be useful for the lower bound also. We then construct a cover inductively.

Lemma 6.4.4. *In a scalar LQR problem, if $a > 0$ and $0 < b \leq 1$, then the optimal scalar LQR cost satisfies the bounds*

$$\frac{2a}{b^2} < J^\star_{a,b} < \frac{2a+1}{b^2}.$$

Proof. The closed-form solution for the scalar Riccati equation is

$$J^\star_{a,b} = \frac{a + \sqrt{a^2 + b^2}}{b^2}.$$

The lower bound is immediately visible. The upper bound follows from observing that $a^2 + b^2 \leq (a+1)^2$.

□

Lemma 6.4.5. *If $p, \tau, k = \texttt{GCC}(a, b_1, b_2, q)$, then for any $\beta \in (0, 1)$, there exists $k' \in \mathbb{R}$ such that*

$$\beta^{-2}p, \tau, k' = \texttt{GCC}\left(a, \beta b_1, \beta b_2, \beta^{-2} q\right).$$

Proof. In the scalar system, the GCC matrix Riccati equation (6.4) reduces to the quadratic equation

$$\left(\tfrac{1}{\tau} b_1^2 - \tfrac{1}{1+\tau} b_2^2\right) p^2 + 2ap + q = 0. \tag{6.6}$$

Substituting $p' = \beta^{-2}p$ into (6.6) and multiplying by β^{-2} yields a new instance of (6.6) with the parameters $b'_1 = \beta b_1,\ b'_2 = \beta b_2,\ q' = \beta^{-2}q$, for which p' is a solution with τ unchanged. □

Theorem 6.4.6. *For the scalar DDF problem defined by* $\mathbf{A} = \{a\}$, *where* $a > 0$, *and* $\mathbf{B} = \left[\frac{1}{\theta}, 1\right]$, *if* $\alpha \geq \frac{2a+1}{2a}$, *then* $N_\alpha^{\text{cov}}(\mathbf{B}) = O(\log \theta)$.

Proof. We construct a cover from the upper end of $\mathbf{B}$. By Lemma 6.4.4, the condition $\alpha \geq \frac{2a+1}{2a}$ implies that $J^\star_{b=1} < \alpha 2a < \alpha J^\star_{b=1}$. Therefore, by Lemmas 6.4.1 and 6.4.2, there exists $\beta \in (0, 1)$ and p, τ, k such that

$$p, \tau, k = \texttt{GCC}\left(a, \frac{1-\beta}{2}, \frac{1+\beta}{2}, 1\right)$$

and $p \leq \alpha 2a$. Proceeding inductively, suppose that for $N \geq 1$, we have covered $[\beta^N, 1]$ by the intervals $\mathbf{B}_n = [\beta^{n+1}, \beta^n]$ for $n \in \{0, \ldots, N-1\}$, and each $\mathbf{B}_n$ has a controller k_n such that

$$\beta^{-2n}p, \tau, k_n = \texttt{GCC}\left(a, \frac{\beta^n - \beta^{n+1}}{2}, \frac{\beta^n + \beta^{n+1}}{2}, \beta^{-2n}\right).$$

Then the existence of the desired $\mathbf{B}_N, k_N$ follows immediately from Lemma 6.4.5.

By Lemma 6.4.3, for each $\mathbf{B}_n$ the GCC state cost $q_n = \beta^{-2n} \geq 1$ is an upper bound on the cost if we replace q_n with 1 to match the DDF problem. Therefore, for each interval $\mathbf{B}_n$, for all $b \in \mathbf{B}_n$,

$$\alpha J^\star_b \geq \alpha J^\star_{\beta^n} > \beta^{-2n}\alpha 2a \geq \beta^{-2n}p \geq J_b(k_n),$$

where first inequality is due to Lemma 6.4.3, the second is due to Lemma 6.4.4, the third is by construction of p, and last is due to the GCC guarantee of k_n. Hence, $\mathbf{B}_n \subseteq \mathscr{N}_\alpha(k_n)$. We cover the full $\mathbf{B}$ when $\beta^N \leq \frac{1}{\theta}$, which is satisfied by $N \geq -\log\theta / \log\beta$. □

6.4.2 Scalar lower bound

For the matching lower bound, we begin by deriving a simplified overestimate of $\mathscr{N}_\alpha(k)$. We then show that the true $\mathscr{N}_\alpha(k)$ is still a closed interval moving monotonically with k. Finally, we argue that the gaps between consecutive elements of a cover grow at most geometrically, while the range of k values in a cover must grow linearly with θ.

Lemma 6.4.7. *For a scalar DDF problem with $a \geq 1$, $\mathbf{B} = [\frac{1}{\theta}, 1]$, for any $k < 0$, if $\alpha \geq 3/2$, then $\mathscr{N}_\alpha(k) \subseteq \frac{1}{|k|}[c_1 - c_2,\ c_1 + c_2]$, where c_1 and c_2 are constants depending on α and a.*

Proof. Beginning with the closed-form solution for $J_b(k)$, which can be derived from (6.2), we define

$$J_b(k) = \frac{1+k^2}{-2(a+bk)} \geq \frac{k^2}{-2(a+bk)} \triangleq \underline{J_b}(k). \tag{6.7}$$

By Lemma 6.4.4, we have

$$J_b^\star < \frac{3a}{b^2} \triangleq \overline{J_b^\star},$$

so $\tilde{r} = \underline{J_b}(k) / \overline{J_b^\star}$ is a lower bound on the suboptimality of k. Computing $\partial^2 \tilde{r} / \partial b^2$ shows that $\tilde{r}$ is strictly convex in b on the domain $a + bk < 0$, so the α-sublevel set of $\tilde{r}$ is the closed interval with boundaries where $\tilde{r} = \alpha$. This equation is quadratic in b with the solutions

$$b = -\frac{a(3\alpha \pm \sqrt{9\alpha^2 - 6\alpha})}{k}.$$

The resulting interval contains $\mathscr{N}_\alpha(k)$. □

Lemma 6.4.8. *For a scalar DDF problem, if $\alpha > 1$ and $k < -1$, then $\mathscr{N}_\alpha(k)$ is either empty or a closed interval $[b_1, b_2]$, with b_1 and b_2 positive and nondecreasing in k.*

Proof. The result follows from the quasiconvexity of both the suboptimality ratio $J_b(k)/J_b^\star$ and the cost $J_b(k)$. Showing these requires some tedious calculations and is deferred to §6.6. □

Theorem 6.4.9. *For a scalar DDF problem defined by* $a = 1$ *and* $\mathbf{B} = [\frac{1}{\theta}, 1]$*, if* $\alpha \geq 3/2$*, then* $N_\alpha^{\text{cov}}(\mathbf{B}) = \Omega(\log \theta)$.

Proof. From the closed-form solution

$$k_{a,b}^\star = -\frac{a + \sqrt{a^2 + b^2}}{b},$$

we observe that $k_b^\star < -1$ for all $b \in \mathbf{B}$. This, along with the quasiconvexity of $J_b(k)$ in k, implies that there exists a minimal α-suboptimal cover $\mathcal{C}$ for which all $k_i < -1$. Suppose $\mathcal{C} = k_1, \ldots, k_N$ is such a cover, ordered such that $k_i < k_{i+1}$. Then by Lemma 6.4.8, $\mathscr{N}_\alpha(k_i)$ and $\mathscr{N}_\alpha(k_{i+1})$ must intersect, so their overestimates according to Lemma 6.4.7 certainly intersect, therefore satisfying

$$\frac{c_1 + c_2}{-k_{i+1}} \geq \frac{c_1 - c_2}{-k_i} \implies \frac{k_{i+1}}{k_i} \leq \frac{c_1 + c_2}{c_1 - c_2} \implies \frac{k_N}{k_1} \leq \left(\frac{c_1 + c_2}{c_1 - c_2}\right)^{N-1}.$$

By Lemma 6.4.7, to cover $b = 1$ controller k_1 must satisfy $k_1 \geq -(c_1 + c_2)$, and to cover $b = \frac{1}{\theta}$, controller k_N must satisfy $k_N \leq -\theta(c_1 - c_2)$. Along with the previous result, this implies

$$\left(\frac{c_1 + c_2}{c_1 - c_2}\right)^{N-1} \geq \theta \frac{c_1 - c_2}{c_1 + c_2} \implies N \geq \frac{\log \theta}{\log \frac{c_1 + c_2}{c_1 - c_2}}.$$

Recalling that c_1 and c_2 only depend on a and α, the $\Omega(\log \theta)$ dependence on θ is established. □

Remarks

- For the upper bound, it may be possible to compute or bound β in the scalar case as a function of a and α, but the analogous result will likely be harder to obtain in the matrix case.

- These results impose a lower bounds on α greater than 1. We believe this is a mild condition in control applications: if the application demands a suboptimality ratio very close to 1, then the size of the suboptimal cover is likely to become impractical for storage. However, further theoretical results building upon suboptimal coverings may require eliminating the bound.

6.5 Empirical results

For matrix DDF problems, we present empirical results as a first step towards covering number bounds. The proof technique of §6.4 is not easily extended to the $d > 1$ case. We discuss the difficulties and our intermediate results further in §6.7. In this section, we empirically validate a possible cover construction and use visualization better understand the topological and geometric properties of suboptimal neighborhoods when $d > 1$.

6.5.1 Geometric grid construction for upper bounds

We begin by testing a cover construction. If the construction fails to achieve a conjectured upper bound in a numerical experiment, then either the conjecture is false, or the construction is not efficient. A natural idea is to extend the geometrically spaced sequence of b values from Lemma 6.4.4 to multiple dimensions. We now make this notion, illustrated in Figure 6.2, precise.

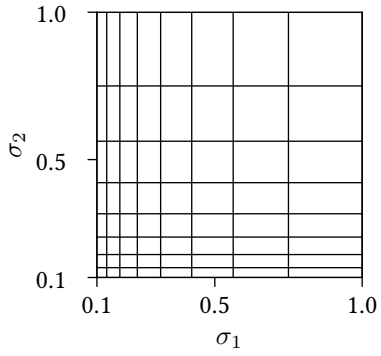


Figure 6.2: Illustration of geometric grid partition (Definition 6.5.1).

Definition 6.5.1 (Geometric grid partition)**.** Given a DDF problem with $\mathbf{\Sigma} = [\frac{1}{\theta}, 1]^d$, and a grid pitch $k \in \mathbb{N}_+$, select $s_1, \ldots, s_{k+1}$ such that $s_1 = \frac{1}{\theta}$, $s_{k+1} = 1$, and $\frac{s_{i+1}}{s_i} > 0$ is constant. For each $j \in \{1, \ldots, k\}^d$, define the grid cell $\mathbf{\Sigma}(j) = \prod_{i=1}^{d} [s_{j(i)}, s_{j(i)+1}]$, where $j(i)$ is the i^{th} component of j. The cells satisfy $\mathbf{\Sigma} = \bigcup_{j \in \{1,\ldots,k\}^d} \mathbf{\Sigma}(j)$, thus forming an partition (up to boundaries) of $\mathbf{\Sigma}$ into k^d cells.

6.5.1.1 Empirical upper bound on $N_\alpha^{\text{cov}}(\Phi)$.

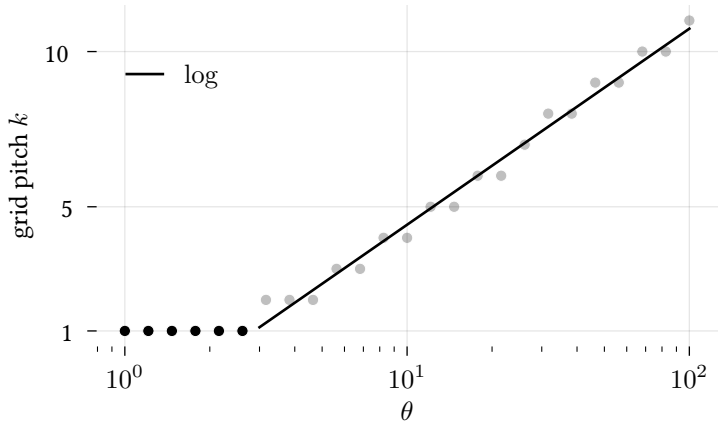


Figure 6.3: Empirical upper bound on grid pitch k needed to construct geometric grid covering of linearized quadrotor using GCC synthesis.

In this experiment, we construct 2-suboptimal covers of the linearized quadrotor for varying θ using geometric grids. We begin with the guess $k = 1$. For each grid cell $\mathbf{\Sigma}(j)$, we compute a controller $K(j)$ using GCC synthesis and check if $\mathbf{\Sigma}(j) \subseteq \mathscr{N}_2(K(j))$. (This requires evaluating only one Lyapunov equation due to Lemma 6.4.3.) If not, we increment k and try again. Termination is guaranteed by continuity. Results for this experiment with $\theta \in [1, 100]$ are shown in Figure 6.3. The required grid pitch k follows roughly logarithmic growth, as indicated by the linear least-squares best-fit curve in black. Small values of θ are excluded from the fit (indicated by black markers), as we do not expect the asymptotic behavior to appear yet.

These results do not rule out the $\log(\theta)^d$ growth suggested by the geometric grid construction. Testing larger values of θ is computationally difficult because the number of grid cells becomes huge and the GCC Riccati equation (6.4) becomes numerically unstable for very small Σ.

6.5.1.2 Efficiency of geometric grid partition.

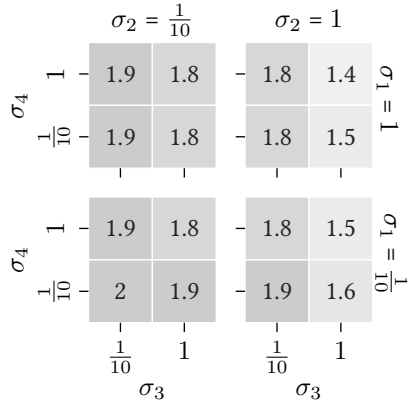


Figure 6.4: Suboptimality ratios for corner cells in geometric grid covering of linearized quadrotor.

Given an α-suboptimal geometric grid cover, we examine a measurable quantity that may reflect the "efficiency" of the cover. Intuitively, in a good cover we expect the worst-case suboptimality ratio of each controller $K(j)$ relative to its grid cell $\mathbf{\Sigma}(j)$ to be close to α. If it close to α for some cells but significantly less than α for others, then the grid pitch around the latter cells is finer than necessary. We visualize results for this computation on the linearized quadrotor with $\theta = 10,\ k = 4$ in Figure 6.4 — only the corners of the $4 \times 4 \times 4 \times 4$ grid are shown. The suboptimality ratio is close to $\alpha = 2$ for cells with low control authority (near $\Sigma = \frac{1}{\theta} I$), but drops to around 1.4 for cells with high control authority (near $\Sigma = I$). The difference suggests that the geometric grid cover could be more efficient in the high-authority regime.

6.5.1.3 Efficiency of GCC synthesis.

One possible source of the conservativeness of GCC in the high-authority regime is that Lemma 6.4.1 applies to the affine image of a $m \times n$-dimensional matrix norm ball, but we only require guaranteed

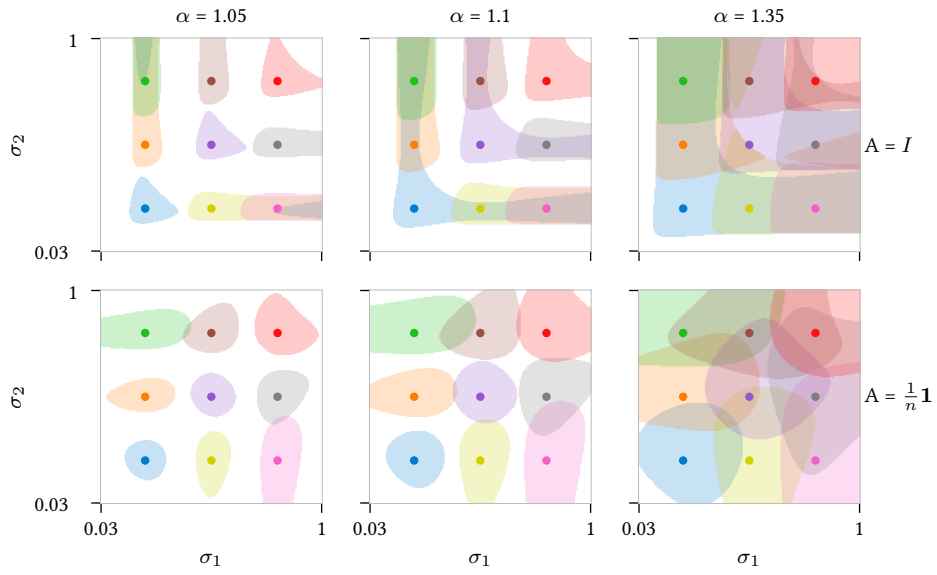


Figure 6.5: α-suboptimal neighborhoods for geometric grid partition in 2D system. *Top:* minimum coupling; $A = I$. *Bottom:* maximum coupling; $A = \frac{1}{n}\mathbf{1}$. *Columns:* varying suboptimality threshold α. All axes are logarithmic. Colors have no meaning. Discussion in §6.5.2.

cost on a d-dimensional affine subspace of matrices. In other words, we ask GCC synthesis to ensure α-suboptimality on systems that are not actually part of Φ. If this is negatively affecting the result, then we should observe that the worst-case cost of $K(j)$ on $\mathbf{\Sigma}(j)$ is less than the trace of the solution P for the GCC Riccati equation (6.4). The worst-case cost always occurs at the minimal $\Sigma \in \mathbf{\Sigma}(j)$ by Lemma 6.4.3; we evaluate it with (6.2). For the quadrotor, a mismatch sometimes occurs for smaller values of θ, but it does not occur for the large values of θ.

6.5.2 Suboptimal neighborhood visualizations

We now present intuition-building experiments towards a covering number lower bound for matrix DDF problems. A lower bound requires a class of DDF problem that can be instantiated for any dimensionality d. Two "extremal" systems come to mind: *minimum coupling*, where $A = I$, and *maximum coupling*, where $A = \frac{1}{n}\mathbf{1}$. Note that for minimum coupling, an α-suboptimal policy is not necessarily α-suboptimal on each scalar subsystem – if it were, the lower bound $\log(\theta)^d$ would trivially follow from the results in §6.4.

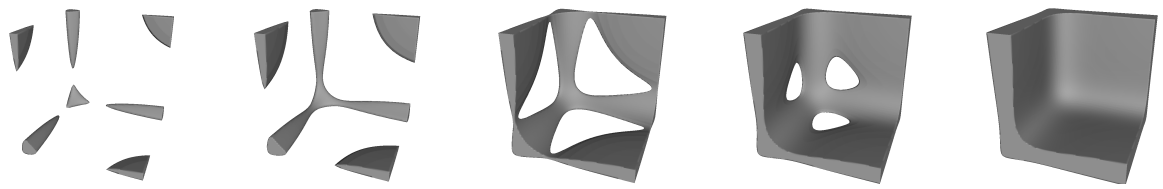


Figure 6.6: α-suboptimal neighborhoods for the three-dimensional decomposed dynamics system with minimal coupling ($A = U = V^\top = I_{3\times 3}$) and breadth $\theta = 100$. Neighborhoods shown for α ranging from 1.04 to 1.2 with a fixed controller.

We show approximate suboptimal neighborhoods for a two-dimensional system in Figure 6.5. We select a geometric grid of Σ values (indicated by the circular markers) and synthesize their LQR-optimal controllers. Then, we evaluate the suboptimality ratio of each controller on a finer grid of Σ values to get approximate neighborhoods, indicated by the semi-transparent regions. We repeat this experiment with three values of α for both choices of A.

Interestingly, the neighborhoods for $A = I$ are not always connected. In the plot for $\alpha = 1.05$ (far left), the neighborhood for the minimal Σ has another component that overlaps other neighborhoods to its top and right. If we increase to $\alpha = 1.1$, the components join into an "L"-shaped region. In contrast, the neighborhoods for $A = \frac{1}{n}\mathbf{1}$ seem more well-behaved. For both choices of A, the neighborhoods are of comparable size.

To verify that this behavior is not an artifact of the two-dimensional case only, we repeat the experiment in three dimensions. Figure 6.6 shows neighborhoods of one controller $K = K^\star_{(2/\theta)I}$ for α ranging from 1.04 to 1.2. As α grows, $\mathscr{N}_\alpha(K)$ shows similar topological phases as the 2D case. In the simply-connected phase (large α), the neighborhood appears to include any Σ where at least one σ_i is sufficiently small. If this property holds in higher dimensions, then it would be possible to construct a cover using only controllers of uniform gain in all dimensions for large α.

6.6 Proof of Lemma 6.4.8

We first present some supporting material. The following facts about scalar LQR problems can be derived from the LQR Riccati equation and some calculus (not shown).

Lemma 6.6.1. *For the scalar LQR problem with $a > 0$, $b > 0$ and $q = r = 1$, the optimal linear controller $k^\star_{a,b}$ is given by the closed-form expression*

$$k^\star_{a,b} = \min_{k \in \mathbb{R}} J_{a,b}(k) = \min_{k \in \mathbb{R}} \frac{1 + k^2}{-2(a + bk)} = -\frac{a + \sqrt{a^2 + b^2}}{b}.$$

For fixed a, the map from b to $k^\star_{a,b}$ is continuous and strictly increasing on the domain $b \in (0, \infty)$ and has the range $(-\infty, -1)$. For any $k \in (-\infty, -1)$, there exists a unique $b_k \in (-\infty, -1)$ for which $k = k^\star_{a,b_k}$, given by

$$b_k = \frac{2ak}{1 - k^2}.$$

We now recall the statement of the lemma.

Lemma 6.4.8. *For a scalar DDF problem, if $\alpha > 1$ and $k < -1$, then $\mathscr{N}_\alpha(k)$ is either empty or a closed interval $[b_1, b_2]$, with b_1 and b_2 positive and nondecreasing in k.*

Instead of a monolithic proof, we present supporting material in Lemmas 6.6.2 and 6.6.3. We then show the main result in Lemma 6.6.4, which considers α-suboptimal neighborhoods on all of $\mathbb{R}$ instead of restricted to $\mathbf{B}$. Lemma 6.4.8 will follow as a corollary.

We proceed with more setup. Recall that the scalar DDF problem is defined by $\mathbf{A} = \{a\}$ and $\mathbf{B} = [\frac{1}{\theta}, 1]$, where $a > 0$. For this section, let

$$\mathcal{D} = \{(b, k) \in (0, \infty) \times \mathbb{R} : a + bk < 0\}$$

(note that $J_b(k) < \infty \iff a + bk < 0$). Denote its projections by $\mathcal{D}^b(k) = \{b : (b,k) \in \mathcal{D}\}$ and $\mathcal{D}^k(b) = \{k : (b,k) \in \mathcal{D}\}$. We compute the suboptimality ratio $r : \mathcal{D} \mapsto \mathbb{R}$ by

$$r(b,k) = \frac{J_b(k)}{J_b^\star} = \frac{1+k^2}{-2(a+bk)} \Bigg/ \frac{a+\sqrt{a^2+b^2}}{b^2} = \frac{(1+k^2)b^2}{-2(a+bk)(a+\sqrt{a^2+b^2})}.$$

We denote its sublevel sets with respect to b for fixed k by

$$\mathcal{D}_\alpha^b(k) = \{b \in \mathcal{D}^b(k) : r(b,k) \leq \alpha\}.$$

Lemma 6.6.2. *For fixed $k < 0$, the ratio $r(b,k)$ is quasiconvex on $\mathcal{D}_k^b$, and there is at most one $b \in \mathcal{D}_k^b$ at which $\partial r / \partial b = 0$.*

Proof. By inspection, $r(b,k)$ is smooth on $\mathcal{D}^b$. We now show that the second-order condition of Lemma 2.4.4(b) holds. To solve $\partial r / \partial b = 0$ for b, we multiply $\partial r / \partial b$ (not shown due to length) by the strictly positive factor

$$\frac{2\left(a+bk\right)^2\left(a+\sqrt{a^2+b^2}\right)^2\sqrt{a^2+b^2}}{ab\left(k^2+1\right)}$$

and set the result equal to zero to get the equation

$$2a^2 + abk + b^2 = (-2a - bk)\sqrt{a^2+b^2}.$$

Squaring both sides (which may introduce spurious solutions) and collecting terms yields the equation $-2ak - bk^2 + b = 0$, with the solution $b = \frac{2ak}{1-k^2}$. This is the expression for b_k from Lemma 6.6.1. Note that it is only positive for $k < -1$. If $k \in [-1, 0)$, then there are no stationary points in $\mathcal{D}_k^b$. Otherwise, substitution into $\partial r / \partial b$ confirms that this solution is not spurious, so it is the only stationary point of r

with respect to b. We now must check the second-order condition for $k < -1$. Evaluating $\partial^2 r/\partial b^2$ (not shown due to length) and multiplying by the strictly positive factor

$$-\frac{\left(a+\sqrt{a^2+b^2}\right)(2a+2bk)}{k^2+1},$$

we have

$$\operatorname{sign}\left(\frac{\partial^2 r}{\partial b^2}\right) = \operatorname{sign}\left(\frac{b^4}{\left(a+\sqrt{a^2+b^2}\right)(a^2+b^2)^{\frac{3}{2}}} + \frac{2b^4}{\left(a+\sqrt{a^2+b^2}\right)^2(a^2+b^2)} + \right.$$
$$\left. \frac{2b^3k}{(a+bk)\left(a+\sqrt{a^2+b^2}\right)\sqrt{a^2+b^2}} + \frac{2b^2k^2}{(a+bk)^2} - \frac{5b^2}{\left(a+\sqrt{a^2+b^2}\right)\sqrt{a^2+b^2}} - \frac{4bk}{a+bk} + 2\right).$$

Evaluating at the stationary point b_k, this reduces to

$$\operatorname{sign}\left(\frac{\partial^2 r}{\partial b^2}\right)\Big|_{b_k,k} = \operatorname{sign}\left(\frac{2k^2(k-1)^2(k+1)^2}{(k^2+1)^3}\right). \tag{6.8}$$

Recalling that $k < -1$, the sign is positive. The conclusion follows from Lemma 2.4.4(b). □

Lemma 6.6.3. *For fixed b, the cost $J_b(k)$ is quasiconvex on $\mathcal{D}^k(b)$. Also, $J_b(k)$ is not monotonic, so case 3. of Lemma 2.4.4(a) applies.*

Proof. We have

$$J_b(k) = \frac{1+k^2}{-2(a+bk)}.$$

The numerator is nonnegative and convex on $k \in \mathbb{R}$. The denominator is linear (hence concave) and positive on $\mathcal{D}^k(b)$. Quasiconvexity follows from Lemma 2.4.4(c). Nonmonotonicity follows from the fact that $J_b(k)$ is smooth on $\mathcal{D}^k(b)$ and has a unique optimum at $k_b^\star$, which is not on the boundary of $\mathcal{D}^k(b)$. □

We now combine these into the main result.

Lemma 6.6.4. *For a scalar DDF problem, if $\alpha > 1$ and $k < -1$, then $\mathcal{D}_\alpha^b(k)$ is either: a bounded closed interval $[b_1, b_2]$, with b_1 and b_2 increasing in k, or a half-bounded closed interval $[b_1, \infty)$, with b_1 increasing in k.*

Proof. By Lemma 6.6.2, due to quasiconvexity $\mathcal{D}_\alpha^b$ is convex. The only convex sets on $\mathbb{R}$ are the empty set and all types of intervals: open, closed, and half-open. We know $\mathcal{D}_\alpha^b$ is not empty because it contains b_k. We can further assert that $\mathcal{D}_\alpha^b$ has a closed lower bound because $\lim_{b\to(-a/k)} r(b,k) = \infty$ (see Boyd and Vandenberghe (2004, §A.3.3) for details). However, the upper bound may be closed or infinite. We handle the two cases separately.

Bounded case. Fix $k_0 < -1$. Suppose $\mathcal{D}_\alpha^b(k_0) = [b_1, b_2]$ for $0 < b_1 < b_2 < \infty$. By the implicit function theorem (IFT), at any (b_0, k_0) satisfying $r(b_0, k_0) = \alpha$, if $\partial r/\partial b|_{b_0,k_0} \neq 0$ then there exists an open neighborhood around (b_0, k_0) for which the solution to $r(b,k) = \alpha$ can be expressed as $(g(k), k)$, where g is a continuous function of k and

$$\left.\frac{\partial g(k)}{\partial k}\right|_{k_0} = -\left(\frac{\partial r}{\partial b}\right)^{-1} \left.\frac{\partial r}{\partial k}\right|_{b_0,k_0}.$$

By the continuity and quasiconvexity of r, and the fact that $\partial r/\partial b = 0$ only at b_k (Lemma 6.6.2) we know that $r(b_1, k_0) = r(b_2, k_0) = \alpha$ and

$$\left.\frac{\partial r}{\partial b}\right|_{b_1,k_0} < 0 \quad \text{and} \quad \left.\frac{\partial r}{\partial b}\right|_{b_2,k_0} > 0.$$

By Lemma 6.6.1, since $k < -1$ there exists $b_k > 0$ satisfying $k = k^\star_{b_k}$. Since $r(b_k, k) = 1$ and $\alpha > 1$, we know $b_{k_0} \in (b_1, b_2)$. Again by Lemma 6.6.1, the map from b to $k_b^\star$ is increasing in b.

Therefore, $k^\star_{b_1} < k_0 < k^\star_{b_2}$. By the quasiconvexity and nonmonotonicity of $J_b(k)$ from Lemma 6.6.3, via Lemma 2.4.4(a) we have

$$\left.\frac{\partial r}{\partial k}\right|_{b_1,k_0} \geq 0 \quad \text{and} \quad \left.\frac{\partial r}{\partial k}\right|_{b_2,k_0} \leq 0.$$

Therefore, the functions g_1, g_2 satisfying the conclusion of the IFT in the neighborhoods around (b_1, k_0) and (b_2, k_0) respectively also satisfy

$$\left.\frac{\partial g_1(k)}{\partial k}\right|_{b_1,k_0} \geq 0 \quad \text{and} \quad \left.\frac{\partial g_2(k)}{\partial k}\right|_{b_2,k_0} \geq 0.$$

Therefore, b_1 and b_2 are locally nondecreasing in k.

Unbounded case. Suppose $\mathcal{D}^b_\alpha(k) = [b_1, \infty)$ for $b_1 < \infty$. By the same IFT argument as in the bounded case, b_1 is increasing in k. By the quasiconvexity of r in b, the value of r is increasing for $b > b_k$, but the definition of $\mathcal{D}^b_\alpha(k)$ implies that $r(b,k) \leq \alpha$ for all $b > b_k$. Therefore, $\lim_{b\to\infty} r(b,k)$ exists and is bounded by α. In particular,

$$\begin{aligned}
\lim_{b\to\infty} r(b,k) &= \lim_{b\to\infty} \frac{(1+k^2)b^2}{-2(a+bk)(a+\sqrt{a^2+b^2})} \\
&= \lim_{b\to\infty} \frac{(1+k^2)b^2/b^2}{-2(a+bk)(a+\sqrt{a^2+b^2})/b^2} \\
&= -\frac{1+k^2}{2k}.
\end{aligned}$$

Taking the derivative shows that this value is decreasing in k for $k < 0$. Therefore, if $k < k' < 0$ then

$$\lim_{b\to\infty} r(b,k') \leq \lim_{b\to\infty} r(b,k) \leq \alpha.$$

The property that $r(b,k')$ is increasing in b for $b > b_k$ further ensures that $r(b,k') \leq \alpha$ for all $b > b_k$. Therefore, $\mathcal{D}^b_\alpha(k')$ is also unbounded. □

For completeness, we prove Lemma 6.4.8.

Proof. (of Lemma 6.4.8). By Lemma 6.6.4, $\mathcal{D}_\alpha^b(k)$ is either a bounded closed interval $[b_1, b_2]$, with b_1 and b_2 increasing in k, or a half-bounded closed interval $[b_1, \infty)$, with b_1 increasing in k. Recall that $\mathscr{N}_\alpha(k) = \mathcal{D}_\alpha^b(k) \cap \mathbf{B}$ with $\mathbf{B} = [\frac{1}{\theta}, 1]$. Therefore, the half-bounded case can be reduced to the bounded case with $b_2 = 1$. The intersection can be expressed as

$$\mathscr{N}_\alpha(k) = [\max\{b_1, \tfrac{1}{\theta}\}, \min\{b_2, 1\}],$$

where the interval $[a, b]$ is defined as the empty set if $a > b$. Taking the maximum or minimum of a nonstrict monotonic function and a constant preserves the monotonicity, so we are done. □

6.7 Efforts towards matrix case

In this section, we present intermediate results in our effort to prove or disprove the following conjecture:

Conjecture 6.7.1. *For a general DDF problem as defined in Definition 6.2.2 with rank d and breadth θ, $N_\alpha^{\text{cov}}(\mathbf{B}) \in O(d^{\log \theta})$.*

This conjecture is not particularly strong – essentially, it posits that the covering number may suffer from a "curse of dimensionality" with respect to d, but the dependency on θ matches that of the scalar case. We also visualize suboptimal neighborhoods for multi-system LQR problems where the variations are in the A matrix instead of the B matrix.

To prove an upper bound for the matrix case using the geometric grid construction (Definition 6.5.1) with a similar proof technique to that of Theorem 6.4.6, the key obstacle appears to be Lemma 6.4.5. The proof of Lemma 6.4.5 relies on commutativity of scalar multiplication, which no longer holds in the matrix case.

6.7.1 Easy case: Scalar multiples of B

As an intermediate step towards Conjecture 6.7.1, we can address the case where A is arbitrary but $\mathbf{B}$ is a one-dimensional subspace as in the scalar case. To be precise, let $\mathbf{B} = \{\sigma B_0 : \sigma \in [\frac{1}{\theta}, 1]\}$. It turns out that this case is not much more difficult than the scalar case, even though the unforced dynamics $\dot{x} = Ax$ can exhibit oscillation due to complex eigenvalues, may contain a mixture of stable and unstable eigenvalues, and so on.

Whereas the proof for Theorem 6.4.6 relied on the closed-form solution for optimal LQR cost, here we will use a less constructive method. Our proof will require a more generic lower limit of α, of which the constraint $\alpha \geq \frac{2a+1}{2a}$ for the scalar setting was a special case. In what follows, we will use the notation

$$[s, t]B = \{\sigma B : \sigma \in [s, t]\}$$

and refer to such a set as an "interval". For legibility, when referring to an LQR problem (A, B, Q, R) with variations in matrices other than B, we will also use the notation $J^\star(A, B, Q, R)$ instead of the notation $J^\star_{A,B,Q,R}$ for the LQR problem's optimal cost. We begin with the following generalization of Lemma 6.4.5:

Lemma 6.7.2. *For (A, B) controllable and $Q \succeq 0$, if P_0 is the maximal solution of the ARE*

$$A^\top P + PA - PBB^\top P + Q = \mathbf{0}$$

associated with the LQR problem (A, B, Q, I), then for any $\beta > 0$, the matrix $\beta^{-2}P_0$ solves the ARE

$$A^\top P + PA - \beta^2 PBB^\top P + \beta^{-2}Q = \mathbf{0}, \tag{6.9}$$

and is the unique optimal cost matrix for the LQR problem defined by $(A, \beta B, \beta^{-2}Q, I)$.

Proof. It is easy to see that $\beta^{-2}P_0$ solves the ARE (6.9). In general, AREs may have more than one positive semidefinite solution, and we would need to show that $\beta^{-2}P_0$ is maximal to ensure that it actually represents the LQR optimal cost matrix. However, for AREs defined by well-posed LQR problems, it can be shown that the LQR cost matrix is the *unique* positive semidefinite solution of the associated ARE (Lan95, Theorem 16.3.3). Therefore, we are assured that $\beta^{-2}P_0$ is also the LQR cost matrix. □

Theorem 6.7.3. *For the multi-system LQR problem defined by $\mathbf{A} = \{A\}$ and $\mathbf{B} = [\frac{1}{\theta}, 1]B_0$, where (A, B_0) is controllable, with fixed cost matrices $Q \succ 0$ and $R = I$, if there exists a constant c such that*

$$J^\star(A, B, sQ, sI) \leq cJ^\star(A, B, Q, sI)$$

for all $s > 0$, then if $\alpha > c$, we have $N^{\text{cov}}_\alpha(\mathbf{B}) = O(\log \theta)$.

Proof. Let P_0 denote the solution to the ARE

$$A^T P + PA - PB_0 B_0^T P + Q = \mathbf{0}, \tag{6.10}$$

so that $J^\star_{B_0} = \operatorname{tr}(P_0)$. By Lemmas 6.4.1 and 6.4.2, there exists $\beta \in (0,1)$ and P_{GCC}, τ, K such that

$$P_{GCC}, \tau, K = \texttt{GCC}\left(A, \frac{1-\beta}{2} B_0, \frac{1+\beta}{2} B_0, Q\right) \tag{6.11}$$

and $\operatorname{tr}(P_{GCC}) \leq \frac{\alpha}{c} J^\star_{B_0}$. Then for all $B \in [\beta, 1] B_0$, we have

$$J_B(K) \leq \operatorname{tr}(P_{GCC}) \leq \frac{\alpha}{c} J^\star_{B_0} \leq \frac{\alpha}{c} J^\star_B,$$

where the first inequality follows from the definition of GCC synthesis and the last inequality follows from the ARE comparison lemma (Lemma 6.4.3).

Now let us consider covering $[\beta^N, 1] B_0$ by the intervals $\{\mathbf{B}_n\}_{n=0}^{N-1}$, where $\mathbf{B}_n = [\beta^{n+1}, \beta^n] B_0$. By applying Lemma 6.7.2 to the Riccati equation (6.10) we find that

$$J^\star(A, \beta^n B_0, \beta^{-2n} Q, I) = \beta^{-2n} \operatorname{tr}(P_0) = \beta^{-2n} J^\star_{B_0}.$$

Invoking Lemma 6.7.2 on the GCC Riccati equation (6.4) with the base case of (6.11) yields a controller K_n for each $\mathbf{B}_n$ such that

$$\beta^{-2n} P_{GCC}, \tau, K_n = \texttt{GCC}\left(A, \frac{\beta^n - \beta^{n+1}}{2} B_0, \frac{\beta^n + \beta^{n+1}}{2} B_0, \beta^{-2n} Q\right).$$

The ARE comparison lemma with respect to the state cost matrix then implies that, for all $B \in \mathbf{B}_n$,

$$J_B(K_n) \leq \beta^{-2n} \operatorname{tr}(P_{GCC}).$$

Putting it all together, for each $\mathbf{B}_n$ and K_n, we have

$$J_B(K_n) \leq \beta^{-2n} \operatorname{tr}(P_{GCC}) \leq \beta^{-2n} \frac{\alpha}{c} J^\star_{B_0} = \frac{\alpha}{c} J^\star(A, \beta^n B_0, \beta^{-2n} Q, I) \leq \alpha J^\star(A, \beta^n B_0, Q, I) \leq \alpha J^\star_B$$

for all $B \in \mathbf{B}_n$, where the second-to-last inequality is due to the hypothesis and the last is due to the ARE comparison lemma. Therefore, $\mathbf{B}_n \subseteq \mathscr{N}_\alpha(K_n)$. We cover the full $\mathbf{B}$ when $\beta^N \leq \frac{1}{\theta}$, which is satisfied by $N \geq -\log\theta / \log\beta$. Recalling that the original choice of β did not depend on θ, we are done. □

In the next section, we elaborate on the cost ratio limit c in the hypothesis of Theorem 6.7.3 and how it differs between the scalar and matrix cases.

6.7.2 Role of α's lower bound

We have been focusing on what happens as B is scaled down towards zero. From the LQR Riccati equation (2.31), we can see that scaling down B gives the same ARE solution as scaling up the control cost R. To be precise,

$$J^\star(A, \tfrac{1}{s}B, Q, R) = J^\star(A, B, Q, s^2 R)$$

for any $s \neq 0$. However, in our proof for the scalar case, we do something equivalent to

$$J^\star(A, \tfrac{1}{s}B, Q, R) = J^\star(A, B, Q, s^2 R) \leq J^\star(A, B, s^2 Q, s^2 R). \tag{6.12}$$

The role of α is to accommodate for the looseness of that inequality. In the scalar case we (implicitly) proved that the ratio

$$\frac{J^\star(A, B, sQ, sR)}{J^\star(A, B, Q, sR)} \tag{6.13}$$

never gets bigger than $(2a + 1)/2a$. In Figure 6.7, we plot the ratio from Equation (6.13) for the scalar system when $a = 1$. It converges to a value slightly larger than 1.2.

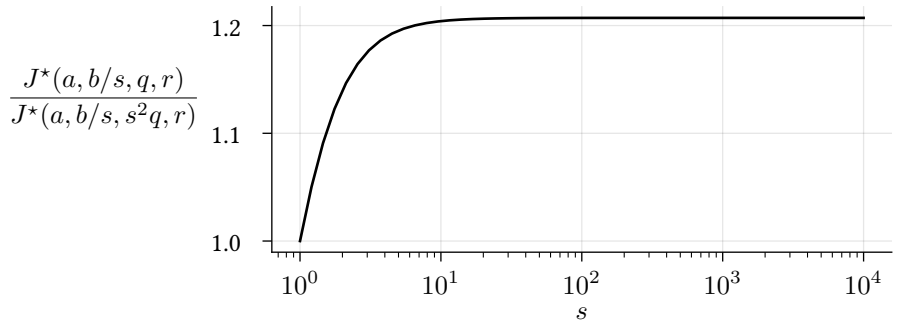


Figure 6.7: "Approximation error" accounted for by $\alpha > \frac{2a+1}{2a}$ assumption in scalar upper bound proof.

We can also plot the ratio from Equation (6.13) for non-scalar LQR problems. We sample random A and B matrices with dimensions $2 \leq n, m \leq 10$ and entries i.i.d. normally distributed. We reject samples where $\rho^+(A) \leq 0$. Otherwise, we scale A such that $\rho^+(A) = 1$, and scale B such that $\|B\|_{2,2} = 1$. These plots are shown in Figure 6.8.

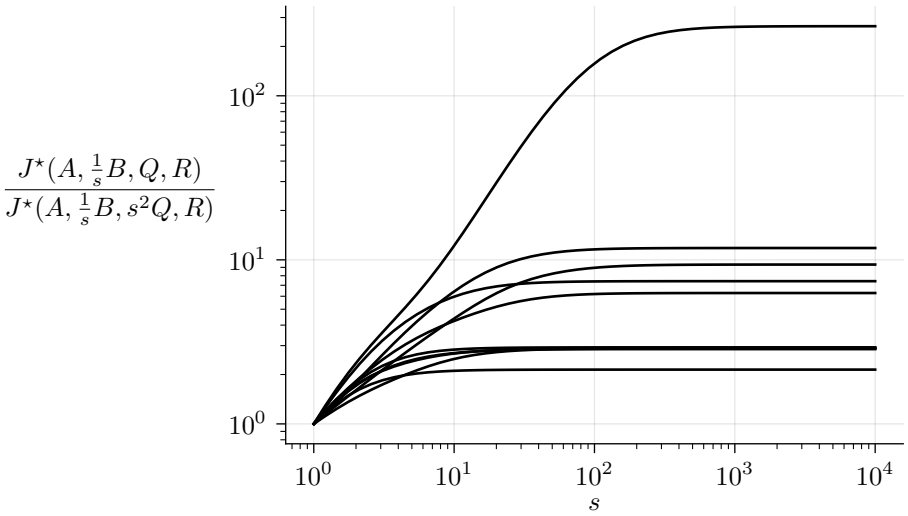


Figure 6.8: Looseness introduced by the inequality (6.12) for random LQR problems.

In Figure 6.8, we see that the ratio of Equation (6.13) still appears to converge to a finite value for these examples, but it can be much bigger than 1.2. This explains the constant c in the hypothesis of Theorem 6.7.3. The large values seen in Figure 6.8 make this constraint unsatisfying, because a guarantee that only holds for a suboptimality ratio $\alpha \gg 1$ is unlikely to be useful in practice. On the other hand, it seems possible that the asymptotic dependence on θ obtained from adding this constraint would not be any different from the true dependence for arbitrary α.

The existence of such a constant c is of course not guaranteed. Based on the empirical behavior in Figure 6.8, we conjecture that it does exist:

Conjecture 6.7.4. *For any well-posed LQR problem (A, B, Q, R), there exists a constant c such that*

$$J^\star(A, B, sQ, sI) \leq cJ^\star(A, B, Q, sI)$$

for all $s > 0$.

6.7.3 Form of Riccati perturbation for geometric grid recursion

If we are to use a grid-based construction to prove an upper bound on the suboptimal covering numbers for DDF problems, it will likely involve some kind of induction or dynamic programming-style recursion. If we again start from the $\Sigma = I$ case and move "down" to less control authority, then we will be recursing from $B = U\Sigma V^\top$ to $B' = U\Sigma' V^\top$ with $\Sigma \succ \Sigma'$. If we consider stepping from one grid cell to its face-sharing neighbors, then only one element of Σ' will change. Without loss of generality, we may assume that element $\Sigma_{1,1}$ is the one changing.

Since only one element of Σ' changes, the change to B is a rank-one update. It is not immediately clear if this is useful.

In the scalar case of § 6.4 and the one-dimensional subspace case of § 6.7.1, we performed a series of algebraic manipulations on the Riccati equation for the optimal cost of the system with B' to arrive at a new value of the cost matrix P' expressed in terms of the solution P for the system with B. Unfortunately, we were not able to use such a simple approach for the matrix case. We attempted representing the changes in $B,\ P,\ Q$ as both additive and multiplicative perturbations. Ultimately they both failed due to the structural difference between the $A^\top P + PA$ term and the $PBB^\top P$ term in the Riccati equation. We present details in the next two subsubsections.

6.7.3.1 Multiplicative change in P

Recall we are trying to recurse from a solution for the Riccati equation

$$A^\top P + PA - PBB^\top P + Q = \mathbf{0} \tag{6.14}$$

to a solution for the Riccati equation

$$A^\top P + PA - PB'B'^\top P + Q = \mathbf{0},$$

with B' as described in § 6.7.3. Let us first consider the simple case where $d = n$ and $U = V = I_{n\times n}$. We use the notation $\Sigma' = M\Sigma$, where $M \preceq I$ is diagonal and positive definite, to express the change in Σ as a multiplicative perturbation. The new Riccati equation then becomes

$$A^\top P' + P'A - P'M\Sigma\Sigma MP' + Q' = \mathbf{0}.$$

Our initial hope might be to follow the scalar case and test if something like

$$P' = M^{-1}PM^{-1}, \quad Q' = M^{-1}QM^{-1}$$

works. This proposed solution converts the Riccati equation above to

$$A^\top M^{-1}PM^{-1} + M^{-1}PM^{-1}A - M^{-1}P\Sigma\Sigma PM^{-1} + M^{-1}Q'M^{-1} \stackrel{?}{=} \mathbf{0}. \tag{6.15}$$

But now, to follow the same pattern as the scalar case, we want to perform some algebraic manipulation to get rid of all instances of M and arrive back at the hypothesis (6.14), thus showing that eq. (6.15) holds. This does not seem possible.

6.7.3.2 Additive change in P

Now we will write the perturbation to B as additive instead of multiplicative. Suppose we have solved the Riccati equation

$$A^\top P + PA - PBB^\top P + Q = 0$$

for P. Now we recurse from B to $B' = B - \Delta B$ such that $BB^\top \succeq B'B'^\top$ (and therefore $\Delta B \succeq 0$). We are interested in the solution P' for the Riccati equation

$$A^\top P' + P'A - P'B'B'^\top P' + Q = 0.$$

Due to the ARE comparison lemma (Lemma 6.4.3), we know that $P' \succeq P$. If we write $P' = P + S$ with $S \succeq 0$, we get the Riccati equation

$$A^\top (P+S) + (P+S)A - (P+S)B'B'^\top (P+S) + Q = 0.$$

Expanding certain terms gives us

$$A^\top P + PA + A^\top S + SA - (PBB^\top P - PB\Delta B^\top P - P\Delta BB^\top P + P\Delta B\Delta B^\top P)$$
$$-PB'B'^\top S - SB'B'^\top P - SB'B'^\top S + Q = 0.$$

Subtracting the original Riccati equation yields

$$A^\top S + SA - (-PB\Delta B^\top P - P\Delta BB^\top P + P\Delta B\Delta B^\top P)$$
$$-PB'B'^\top S - SB'B'^\top P - SB'B'^\top S = 0.$$

Now if we define $A' = A - B'B'^\top P$ and group like terms with respect to S, we get

$$A'^\top S + SA' - SB'B'^\top S + P(BB^\top - B'B'^\top)P = 0.$$

So we have an expression for the change in cost as the solution to another Riccati equation. Regarding the expression

$$A' = A - B'B'^\top P,$$

we note that the optimal controller for the original problem was $K = -B^\top P$, so this term is closely related to $A + BK$, that is, the closed loop dynamics of the original system with its optimal controller. Therefore, by some continuity or gain margin argument, we may be able to argue that A' is Hurwitz. This could be useful.

6.7.4 How we would use bounds on cost change due to B perturbations

Although we have not yet derived any useful results about the cost change resulting from a change to B as described in §6.7.3, let us look ahead and think about how such a result might be used by the full proof if we follow the geometric grid construction. Suppose we have a result like:

> If P solves the ARE
>
> $$A^\top P + PA - PU\Sigma V^\top V \Sigma U^\top P + Q = 0$$
>
> and $\Sigma' = \operatorname{diag}(1, \ldots, 1, \beta_i, 1, \ldots, 1)\Sigma$ for $\beta_i < 1$, then the solution P' to the ARE
>
> $$A^\top P + PA - PU\Sigma' V^\top V \Sigma' U^\top P + Q = 0$$
>
> satisfies
>
> $$\operatorname{tr}[P'] \leq \beta_i^{-2} c_i \operatorname{tr}[P],$$
>
> where P' is the solution to the ARE for the new Riccati equation and $c_i > 0$ and $\beta_i \in (0, 1)$ are some constants specific to the i^{th} dimension of Σ.

The constants $c_1, \ldots, c_d$ and $\beta_1, \ldots, \beta_d$ should be allowed to depend on A, U, V in arbitrarily complex ways, but they need to be constant.

Following the template for the proof of Theorem 6.7.3, we need to obtain two quantities for each cell of the geometric grid:

- An upper bound on the numerator of the suboptimality ratio, which comes from a GCC Riccati equation, and
- A lower bound on the denominator of the suboptimality ratio, which comes from a standard LQR Riccati equation applied to the "worst-case" B in the grid cell, which can be selected using Lemma 6.4.3.

In the scalar case, the numerator upper bound came from induction but the denominator lower bound came from the closed-form solution of the scalar Riccati equation. Since there is no closed-form solution for the matrix case, we will likely need some kind of inductive argument for the lower bound as well.

6.7.5 Existing Riccati solution and perturbation bounds

There are many published bounds on spectral properties of either 1) solutions to the ARE, or 2) changes in the solution to an ARE caused by perturbations to its matrix coefficients. Most of these results appear to be very general bounds phrased in terms of properties like matrix norms and minimum/maximum eigenvalues of the matrix coefficients (or their perturbations). It is not clear if any of these bounds are fine-grained enough to be useful with the highly structured setup of rank-one perturbations to B, that we would need to handle in the geometric grid construction for an upper bound on α-suboptimal covering numbers for DDF problems.

Wang et al. (1986) give some fairly easy-to-derive and user-friendly bounds under strong assumptions on A:

Lemma 6.7.5 (Theorems 1 and 2, Wang et al. (1986))**.** *For the algebraic Riccati equation*

$$A^\top P + PA - PRP + Q = 0$$

*where A **is Hurwitz**, the positive semidefinite solution P satisfies*

$$\operatorname{tr}(P) \leq \frac{\lambda_{\max}(A_s) + \sqrt{\lambda_{\max}(A_s)^2 + (\lambda_{\min}(R)/n)\operatorname{tr}(Q)}}{\lambda_{\min}(R)/n} \tag{6.16}$$

and

$$\operatorname{tr}(P) \geq \frac{\lambda_{\min}(A_s) + \sqrt{\lambda_{\min}(A_s)^2 + \lambda_{\max}(R)\operatorname{tr}(Q)}}{\lambda_{\max}(R)}, \tag{6.17}$$

where $A_s = (A^\top + A)/2$.

Note that in Lemma 6.7.5, the $\lambda_{\max}$ and $\lambda_{\min}$ functions are only applied to symmetric matrices, so they are only comparing real eigenvalues. Regarding the requirement that A is Hurwitz, we note that the A' in the Riccati equation of § 6.7.3.2, whose solution tells us the how much the cost has changed additively, is Hurwitz.

Many other Riccati bounds have been published. Bounds on various properties of the solution are collected in the survey of Kwon et al. (1996). Perturbation bounds are given by Konstantinov et al. (2003) and Sun (1998), among many others.

6.7.6 Lower bound candidates

For a lower bound on the covering number, we may need as a lemma some overestimate of suboptimal neighborhoods, containing them into sets that are easy to work with like boxes, balls, etc. Overestimating suboptimal neighborhoods requires underestimating suboptimality ratio. Underestimating suboptimality ratios requires overestimating $J_B^\star$ and underestimating $J_B(K)$ for arbitrary K.

6.7.6.1 Lower bound for $A = I$

Although the lower bound candidate $A = I$ leads to complicated topology of suboptimal neighborhoods, as discussed in § 6.5.2, it is easier to work with algebraically than the candidate $A = \frac{1}{n}\mathbf{1}$. We therefore begin our attempt to find a covering number lower bound using $A = I$, even though it may lead to a loose bound, so that we can get any bound at all (beyond the trivial dimension-independent lower bound we get from the scalar case).

Overestimating $J_B^\star$ Let P denote the unique positive definite solution for the ARE

$$AP + PA - PBB^T P + I = 0,$$

in which $A = I$ and $B = \operatorname{diag}(\sigma_1, \ldots, \sigma_n) \succ 0$. Then P is a diagonal matrix with entries

$$P_{ii} = \frac{1 + \sqrt{1 + \sigma_i^2}}{\sigma_i^2},$$

where the entry P_{ii} corresponds to the optimal scalar cost for the scalar LQR system with $a = 1$ and $b = \sigma_i$, as discussed in § 6.4. An easy calculation verifies that the proposed P solves the given ARE. We see that $P \succ 0$ by construction, so P is the unique positive definite solution (Lan95, Theorem 16.3.3). The resulting optimal controller $K_B^\star = -B^T P$ is diagonal with

$$(K_B^\star)_{ii} = \frac{1 + \sqrt{1 + \sigma_i^2}}{\sigma_i}.$$

We can simplify the result for P by applying Lemma 6.4.4, concluding that

$$\operatorname{tr}(P) \leq 3 \sum_{i=1}^{n} \frac{1}{\sigma_i^2}. \tag{6.18}$$

As in the scalar case, as the σ_i approach zero asymptotically this "user-friendly" upper bound becomes tight up to the constant factor.

Underestimating $J_B(K)$. We have to account for the possibility that a suboptimal covering contains non-diagonal K's. However, for now we make the following conjecture:

Conjecture 6.7.6. *For the DDF problem defined by Φ with $A = I$ and $U = V = I$, if the controller $K \in \mathbb{R}^{n \times n}$ is not diagonal, then there exists a diagonal K_d such that $\mathscr{N}_\alpha(K) \subseteq \mathscr{N}_\alpha(K_d)$.*

If Conjecture 6.7.6 is true, then we can assume any suboptimal covering $\mathcal{C}$ for this problem is composed entirely of diagonal controllers. We can then again build upon the scalar results by invoking Lemma 6.4.7. This gives us the following expression for diagonal K:

$$J_B(K) = -\sum_{i=1}^{n} \frac{1 + K_{ii}^2}{2(1 + \sigma_i K_{ii})} \geq -\sum_{i=1}^{n} \frac{K_{ii}^2}{2(1 + \sigma_i K_{ii})}.$$

Suboptimality ratio. These two bounds leave us with an optimistic suboptimality ratio estimate which we denote as $\hat{r}$:

$$\frac{J_B(K)}{J_B^\star} \geq \hat{r} = \frac{\displaystyle\sum_{i=1}^{n} \frac{K_{ii}^2}{1 + \sigma_i K_{ii}}}{\displaystyle -6 \sum_{i=1}^{n} \frac{1}{\sigma_i^2}}.$$

This ratio-of-sums-of-ratios structure is challenging to work with. Considering that the shapes of the α-suboptimal neighborhoods for this system in Figure 6.5 do not appear easy to approximate, we did not devote further to working with this estimate.

6.7.6.2 Lower bound for $A = \frac{1}{n}\mathbf{1}$

For the case $A = \frac{1}{n}\mathbf{1}$, we can no longer easily build upon the results from the scalar case. To upper-bound $J_B^\star$, we would like to start by applying the upper bound of Lemma 6.7.5, but we cannot do it immediately because $\frac{1}{n}\mathbf{1}$ is not Hurwitz. However, Wang et al. (1986) do not clearly state where the stability of A is used in their proof of Lemma 6.7.5. Further investigation is required.

6.7.7 Packing-based strategies for lower bounds

In more well-known (often geometric) covering problems based on metrics/norms, the notion of coverings is closely related to the notion of packings. We can also consider packings for our problem.

Definition 6.7.7. Consider a multi-system optimal control problem $(\mathcal{X}, \mathcal{U}, \Phi, \Pi_{\text{ref}})$ and a suboptimality ratio $\alpha > 1$. Given a system $\phi \in \Phi$, we define its *α-suboptimal policy neighborhood* as

$$\mathscr{P}_\alpha(\phi) = \left\{ \pi \in \Pi_{\text{ref}} : \frac{J_\phi(\pi)}{J_\phi^\star} \leq \alpha \right\}.$$

A set of systems $\mathcal{P} \subseteq \Phi$ is an *α-suboptimal packing of* Φ if its corresponding family of α-suboptimal policy neighborhoods $\{\mathscr{P}_\alpha(\phi)\}_{\phi \in \mathcal{P}}$ is pairwise disjoint.

The *α-suboptimal packing number* of Φ, denoted $N_\alpha^{\text{pack}}(\Phi)$, is the size of the largest α-suboptimal packing of Φ.

It is clear that $N_\alpha^{\text{pack}}(\Phi) \leq N_\alpha^{\text{cov}}(\Phi)$. Constructing a α-suboptimal packing could be a useful strategy for covering number lower bounds. With packings we have a "for all" condition with respect to the suboptimality ratio of arbitrary controllers for a finite set of Bs. As discussed in §2.9.1.6, the cost of an arbitrary suboptimal controller K for a particular B depends on the solution to the (linear) Lyapunov equation

$$(A + BK)X + X(A + BK)^\top + I = \mathbf{0}$$

and is given by

$$J(K) = \operatorname{tr}\left[(Q + K^\top R K)X\right]. \tag{6.19}$$

On the other hand, the optimal cost for a particular B depends on the solution to the (quadratic) algebraic Riccati equation

$$A^\top P + PA - PBR^{-1}B^\top P + Q = \mathbf{0},$$

but the optimal cost is simply

$$J_B^\star = \operatorname{tr}[P].$$

So for suboptimal costs we have a simpler matrix equation, but K appears both in the equation coefficients and in a product with the solution to get the final cost. It is not clear which is easier to work with.

6.7.8 Reparameterization

The LQR cost is not a convex function of the controller matrix K, but it can be rendered convex by a reparameterization. We follow the presentation of Mohammadi et al. (2019). Let $Y = KX$ where $X \succ 0$. Substituting $K = YX^{-1}$ into (6.19) yields

$$J(X, Y) = \operatorname{tr}\left[QX + Y^\top R Y X^{-1}\right],$$

where X solves the same Lyapunov equation as before, which under our reparameterization becomes

$$AX + XA^\top - (BY + Y^\top B^\top) + I = 0.$$

Subsequently we name the linear operators (overloading notation with the sets of matrices in the general multi-system LQR problem)

$$\mathcal{A}(X) = AX + XA^\top, \quad \mathcal{B}(Y) = BY + Y^\top B^\top,$$

and rewrite the Lyapunov equation as $\mathcal{A}(X) - \mathcal{B}(Y) + I = 0$. Then, under the assumption that $\mathcal{A}$ is invertible, X becomes an affine function of Y:

$$X(Y) = \mathcal{A}^{-1}(\mathcal{B}(Y) - I).$$

We now denote the set of stabilizing solutions by

$$\mathcal{S}_Y = \{Y \in \mathbb{R}^{m\times n} : X(Y) \succ 0\},$$

which is equivalent to the set of stabilizing controllers. We can then define

$$J(Y) = \begin{cases} J(X(Y), Y) & : Y \in \mathcal{S}_Y \\ \infty & : \text{otherwise } . \end{cases}$$

Mohammadi et al. (2019) show that, over the a-sublevel set $\mathcal{S}_Y(a) = \{Y : J(Y) \leq a\}$, the cost $J(Y)$ is μ-strongly convex (see §2.4.2) with strong convexity constant

$$\mu = \frac{2\lambda_{\min}(R)\lambda_{\min}(Q)}{a(1 + a^2\eta)^2},$$

where

$$\eta = \frac{\|\mathcal{B}\|_2}{\lambda_{\min}(Q)\lambda_{\min}(I)\sqrt{\nu\lambda_{\min}(R)}},$$

where

$$\nu = \frac{\lambda^2_{\min}(I)}{4}\left(\frac{\|A\|_2}{\sqrt{\lambda_{\min}(Q)}} + \frac{\|B\|_2}{\sqrt{\lambda_{\min}(R)}}\right)^{-2}.$$

Mohammadi et al. (2019) do not explicitly state the norm to which the strong convexity constant μ applies, but they also give a Lipschitz smoothness constant with respect to the Frobenius norm $\|Y\|_F = \sqrt{\operatorname{tr} Y^T Y}$, so we we will assume that μ is also w.r.t. the Frobenius norm.

Note that the silly expression $\lambda_{\min}(I)$ appears because Mohammadi et al. (2019) give the result for an arbitrary initial state covariance, whereas we have already fixed it to I. Further applying our simplifying assumptions $Q = I$ and $R = I$, the latter becomes

$$\nu = \frac{1}{4}\left(\|A\|_2 + \|B\|_2\right)^{-2},$$

so the former becomes

$$\eta = \frac{\|\mathcal{B}\|_2}{\sqrt{\frac{1}{4}\left(\|A\|_2 + \|B\|_2\right)^{-2}}} = 2\|\mathcal{B}\|_2(\|A\|_2 + \|B\|_2),$$

and thus the strong convexity constant becomes

$$\mu = \frac{2}{a\Big(1 + 2a^2\|\mathcal{B}\|_2(\|A\|_2 + \|B\|_2)\Big)^2}. \tag{6.20}$$

We are interested in the growth of the cost as we move away from $Y_B^\star \triangleq \operatorname{argmin}_{Y \in \mathcal{S}_Y} J(Y)$. The smoothness and convexity of the cost J imply that $\nabla J(Y^\star) = \mathbf{0}$. Therefore, the linear term in the strong convexity definition (2.4) becomes zero when centered on $Y^\star$, leaving the lower bound

$$J(Y) \geq J_B^\star + \frac{\mu}{2}\|Y - Y_B^\star\|_F^2. \tag{6.21}$$

Unfortunately this bound seems to be vacuous in numerical experiments. We instantiate it for a scalar LQR problem with $A = 1,\ B = 1$ and plot a comparison between the actual LQR cost and the lower bound implied by (6.20) and (6.21) in Figure 6.9. The sublevel set constant a is chosen as $a = \alpha J_B^\star$ for two different values: $\alpha \in \{1.001, 1.2\}$. Only those values of Y for which $J_B(YX(Y)^{-1}) \leq a$ are shown, that is, we restrict each plot to the domain $\mathcal{S}_Y(a)$ where the lower bound is valid.

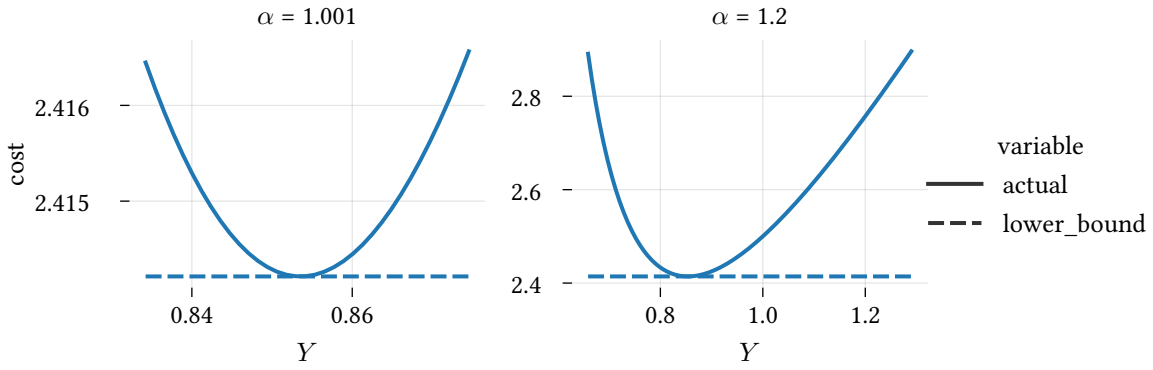


Figure 6.9: Scalar LQR problem: Actual cost (solid) and lower bound (dashed) based on the strong convexity constant derived by Mohammadi et al. (2019). The horizontal axis is the value Y in the convex reparameterization $Y = KX$ as described in §6.7.8.

The curvature of the lower bound is barely visible when plotted alongside the true cost. This is disappointing. It seems unlikely that this would be useful for bounding covering numbers. Note that this happens even for the small sublevel set induced by $\alpha = 1.001$, where we can see in Figure 6.9 that the true cost is close to a quadratic. Therefore, the lower bound appears loose relative to the second-order Taylor expansion, not just when far from the minimum.

6.7.9 Suboptimal neighborhoods for variations in A

Since most of our theoretical efforts do not appear fruitful, we return to empirical work. One major area of interest not explored in our original paper was variations in the A matrix instead of the B matrix. The key question is: what kind of sets of A matrices should we consider? While the decomposed dynamics form had an appealing interpretation in terms of actuator strength, applying the same construction to the A matrix does not seem to be as interpretable.

Instead of proposing a particular structure for a set of A matrices, we can simply plot suboptimal neighborhoods for well-known control systems without requiring that they all share some structure. We show two-dimensional suboptimal neighborhoods plots analogous to Figure 6.5 for various systems. In all systems, the dynamics are parameterized by two real values, represented by the horizontal and vertical

graph axes. We sample a 3×3 grid of parameter pairs, indicated by points on the plots, and synthesize their LQR-optimal controllers. To visualize approximate suboptimal neighborhoods, we evaluate the suboptimality ratio of each controller on a finer grid of parameter pairs, indicated by the semi-transparent regions. We repeat each experiment with three values of α.

6.7.9.1 Cart-pole system

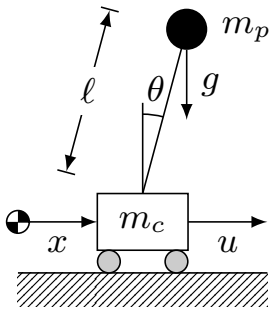


Figure 6.10: Cart-pole system.

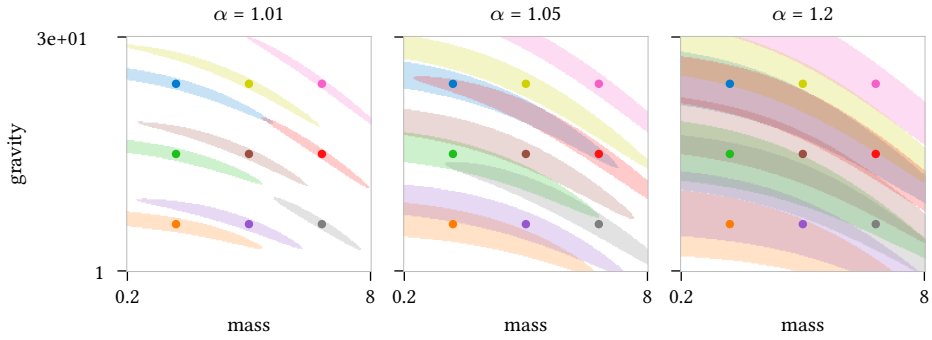


Figure 6.11: α-suboptimal neighborhoods for cart-pole system. Each dot indicates an LQR-optimal controller for a particular (mass, gravity) pair; the surrounding transparent region indicates its α-suboptimal neighborhood. Axes are logarithmic. Colors have no meaning. Discussion in §6.7.9.1.

As our first example for variations in A, we consider the cart-pole system illustrated in Figure 6.10. The cart rolls on a frictionless surface. A rigid massless rod is attached to a frictionless hinge joint on the cart. At the other end of the rod is a point mass. The state space is the position of the cart x and the angle of the rod θ, and their derivatives. The input u is a force upon the cart in the positive-x direction. The system has four physical parameters: cart mass m_c, pole mass m_p, pole length ℓ, and gravitational constant g. All

are strictly positive. The cart-pole system has an unstable equilibrium at $\theta = 0$ and a stable equilibrium at $\theta = \pi$. We consider stabilization at the $\theta = 0$ (pole-up) position. Linearizing the nonlinear dynamics (not shown) about the pole-up state yields the system matrices

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{gm_p}{m_c} & 0 & 0 \\ 0 & \frac{g(m_c + m_p)}{m_c \ell} & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_c} \\ \frac{1}{m_c \ell} \end{bmatrix}.$$

We see that the parameters m_c and ℓ influence both A and B, but the parameters g and m_p only influence A. If we fix $m_c = 1,\ \ell = 1$, then we are left with two parameters that affect only A. Although variations in gravity are perhaps not of concern in most applications, we proceed with this experiment and show the results in Figure 6.11.

The parameter m_p (horizontal axis) varies from $1/4$ to 8 and the parameter g (vertical axis) varies from 1 to 32. We observe that the suboptimal neighborhoods are oblong and oriented diagonally to the grid.

6.7.9.2 Two real eigenvalues

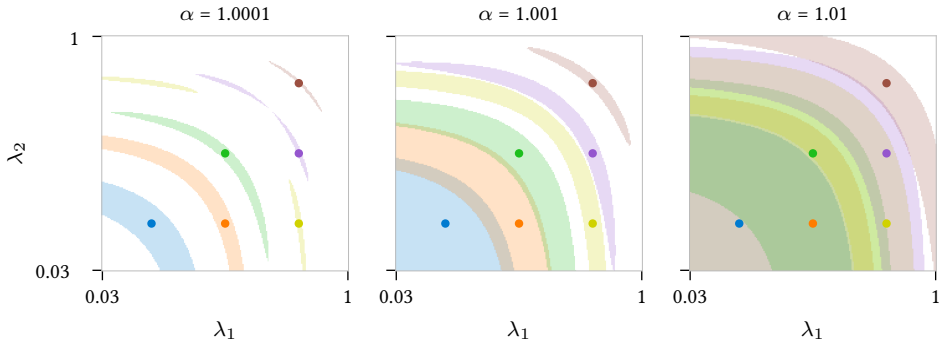


Figure 6.12: α-suboptimal neighborhoods for system in controllable canonical form (CCF) with A having two positive real eigenvalues. Each dot indicates an LQR-optimal controller for a particular pair of eigenvalues; the surrounding transparent region indicates its α-suboptimal neighborhood. Axes are logarithmic. Colors have no meaning. Discussion in §6.7.9.2.

Next we plot suboptimal neighborhoods for systems in controllable canonical form (defined in §2.9.3) when A has two real eigenvalues λ_1, λ_2. Because we are mainly interested in stabilizing unstable systems, we select both eigenvalues to be positive: $\lambda_1, \lambda_2 \in [0.03, 1]$. Results are shown in Figure 6.12.

Recall from §2.9.3 that in controllable canonical form, the A matrix is a permutation-invariant function of the eigenvalues. This is apparent in the plot, where we see that all suboptimal neighborhoods are mirror-symmetric across the $\lambda_1 = \lambda_2$ line. Therefore, we only show suboptimal neighborhoods for the optimal controllers of systems with $\lambda_1 \geq \lambda_2$. We observe that the suboptimal neighborhoods are elongated and highly curved.

6.7.9.3 Pair of conjugate eigenvalues

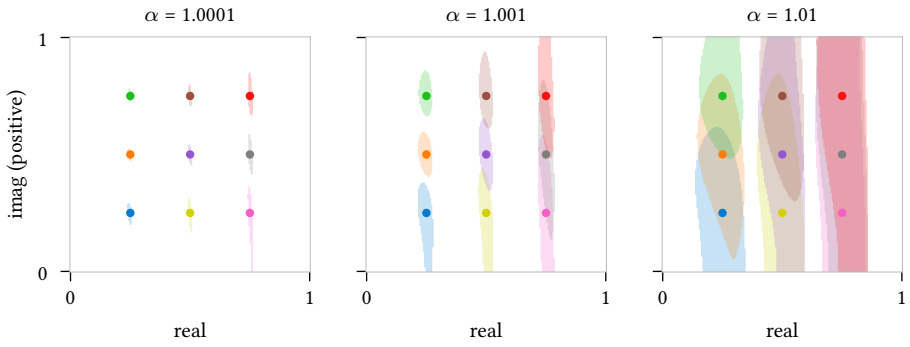


Figure 6.13: α-suboptimal neighborhoods for system in controllable canonical form (CCF) with A having two complex conjugate eigenvalues. Plot corresponds to upper right quadrant of complex plane. Each dot indicates an LQR-optimal controller for the corresponding eigenvalue and its conjugate. The surrounding transparent region indicates its α-suboptimal neighborhood. Axes are linear. Colors have no meaning. Discussion in §6.7.9.3.

Next we plot suboptimal neighborhoods for systems in controllable canonical form when A has a conjugate pair of complex eigenvalues $\lambda, \overline{\lambda}$. In this case, A is parameterized by the two real values $\operatorname{Re}\lambda,\ \operatorname{Im}\lambda$. Again, because we are interested in stabilizing unstable systems, we select $\operatorname{Re}\lambda,\ \operatorname{Im}\lambda \in (0, 1]$. Results are shown in Figure 6.13.

Here we observe that the suboptimal neighborhoods are more aligned with the grid, but they are much narrower with respect to $\operatorname{Re}\lambda$. In other words, the optimal controllers tend to be robust with respect to the LQR cost against variations in $\operatorname{Im}\lambda$, but not in $\operatorname{Re}\lambda$.

6.7.9.4 Spring-mass-damper

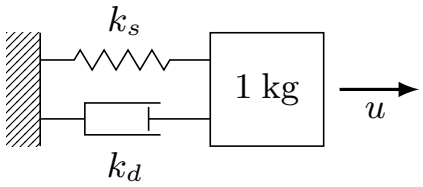


Figure 6.14: Spring-mass-damper system. There is no gravity.

Finally, we plot suboptimal neighborhoods for a spring-mass-damper system under external forces, as illustrated in Figure 6.14. This is the canonical second-order linear ordinary differential equation given by

$$\ddot{z} = -kz - dz + u,$$

where z is the displacement from the spring's resting length, k is the spring constant, d is the damping constant, and the mass is fixed at 1. Raised into first-order state-space form, the dynamics are

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & 1 \\ -k & -d \end{bmatrix}}_{A} x + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B} u,$$

where $x = \begin{bmatrix} z & \dot{z} \end{bmatrix}^\top$. We observe that A is already in controllable canonical form. Also, unlike our other example systems, the unforced dynamics of this system are either passively stable (when $d > 0$) or marginally stable (when $d = 0$).

Results are shown in Figure 6.15. Here the suboptimal neighborhoods are the closest to balls of all the example systems. Further experiments are warranted to investigate potential links between stability of

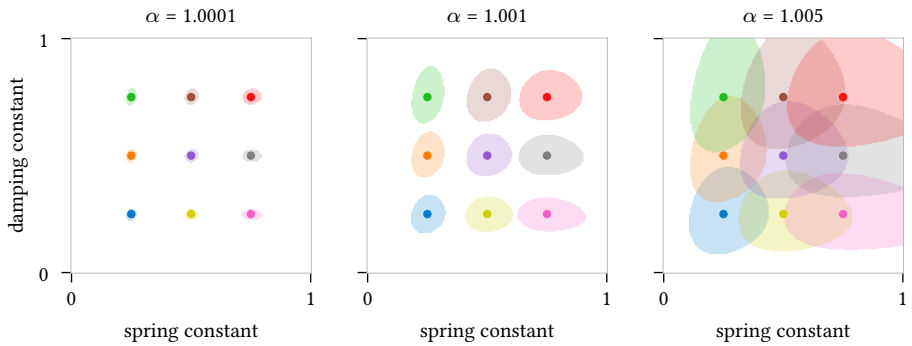


Figure 6.15: α-suboptimal neighborhoods for spring-mass-damper system. Each dot indicates an LQR-optimal controller for a particular (spring constant, damping constant) pair; the surrounding transparent region indicates its α-suboptimal neighborhood. Axes are linear. Colors have no meaning. Discussion in §6.7.9.4.

A and the behavior of suboptimal neighborhoods. It is also possible that systems with stable A could be easier to work with analytically, but this requires more theoretical work. For example, we need to think about the conditions under which a suboptimal controller could destabilize an inherently stable system.

6.7.9.5 Discussion

In summary, we visualized the suboptimal neighborhoods of LQR-optimal controllers for several families of simple linear control systems for which the dynamics parameters Φ can be parameterized by $\mathbb{R}^2_{>0}$. We saw that in general, the suboptimal neighborhoods are substantially nonconvex, not isotropic, and not aligned with the grid of parameters.

If we were to use a grid-based constructive covering of the Φ parameterization space, then we would need to find α-suboptimal policies for each grid cell. These experiments show that there exist systems with the following property: if the policy π is α-suboptimal for a particular grid cell, then it is also α-suboptimal for substantial area outside the cell. We illustrate this in Figure 6.16 for the cart-pole system. Any axis-aligned box of systems for which the brown controller is α-suboptimal will be far smaller than its true α-suboptimal neighborhood.

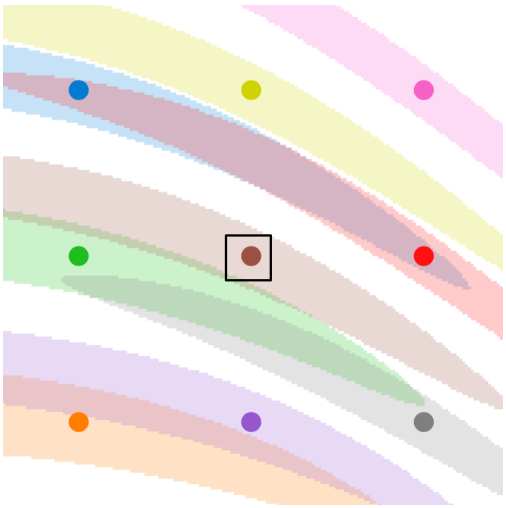


Figure 6.16: Example of a poor match between a grid partition of Φ and true suboptimal neighborhoods of LQR-optimal controllers for Φ in the cart-pole system.

This example suggests that a grid-based covering would be an inefficient construction for the cart-pole system. Still, it is possible that this inefficiency is not significant from an asymptotic perspective.

6.8 Conclusion and future work

In this chapter, we introduced and motivated the α-suboptimal covering number to quantify infinite system spaces for multi-system control problems. We defined a particular class of multi-system linear-quadratic regulator problems amenable to analysis of the α-suboptimal covering number, and showed logarithmic dependency on the problem "breadth" parameter θ in the scalar case. Towards analogous results for the matrix case, we presented empirical studies intended to shed light on possible proof techniques. For the upper bound, we considered a natural covering construction that would preserve logarithmic dependence on θ but give exponential dependence on dimensionality. Experiments did not rule out its validity. For the lower bound, we visualized suboptimal neighborhoods for two possible system classes and observed interesting topological behavior for the minimal-coupling class. We also presented some intermediate theoretical tools, reported on lines of inquiry that did not yield results, and visualized more suboptimal neighborhoods for a generalization of our original setting.

After extending our current results to the matrix case, in future work the analysis can be applied to other classes of multi-system LQR problems including variations in A, Q, R, discrete time, and stochastic dynamics. It will be interesting to see if there are major differences between LQR variants. We also hope that suboptimal covers and covering numbers will be a useful tool for analyzing how the size of the system space affects the required expressiveness of function classes used in practice as multi-system policies, such as neural networks.

Chapter 7

Conclusions

In this dissertation we presented four projects in the intersection of learning and control.

7.1 Summary of contributions

Our empirical work on deep reinforcement learning for domain adaptation shares an overall structure with self-tuning regulators in classic adaptive control, but is generic enough to apply to any parameterized family of MDPs. Introducing deep neural networks enables learning an embedding space representation of the dynamics parameters that is both useful for adaptation and easier to identify from state-action trajectories. By representing the full multi-system policy as a pre-trained neural network, we decouple the adapting process from the control synthesis process, allowing us to use arbitrarily large amounts of computation in the preparation stage. Although our experiments demonstrated the favorable properties of the learned embedding and showed a modest improvement in an ablation study, they also provoked our interest in more fundamental questions about algorithms and problem structure.

In our work on deformable manipulation, we demonstrated techniques to use a recurrent neural network dynamics model for control in a partially observable system. We compensated for the inevitable modeling error by using nonlinear state estimation to identify a value of the RNN internal state that is

consistent with past inputs and outputs. For control, we implemented nonlinear model-predictive control using gradient descent, allowing us to take advantage of the performance optimizations for RNNs in deep learning libraries. Our technique is suitable for systems with complex dynamics but known reward functions, exemplified by the task of tracking a trajectory with a point on a highly deformable object.

The challenges we faced in empirical reinforcement learning projects motivated interest in RL theory. Our work on bounding the variance of the REINFORCE policy gradient estimator for LQR systems enabled us to see how the variance is influenced by various problem parameters, including the dynamics stability, reward function coefficients, environment stochasticity, and policy stochasticity. Our upper bound was tight with respect to most parameters, and matched the behavior of the empirical variance qualitatively. On the other hand, we also demonstrated with experiments that the lowest gradient variance does not necessarily translate into the fastest convergence of REINFORCE.

Our work on suboptimal covering numbers attempted to answer a question raised by the first project: How can we quantify "how much" a good adaptive policy must change its behavior with respect to the problem parameters? We proposed the suboptimal covering number as a highly general, parameter-independent way to do this. As an example, we showed matching logarithmic covering number bounds for scalar and one-dimensional fully observable LQR problem families modeling adaptivity with respect to variations in actuator strength. Our results showed a mild dependency, indicating that for these systems we can handle a large range of actuator strengths, approaching uncontrollability, with only a few distinct policies. We also conjectured an bound exponential in the dimension for multi-input problems, but have not yet resolved it. Our intermediate mathematical results and intuition-building experiments suggest that answering this question require nontrivial insight into the behavior of algebraic Riccati equations. Our choice to focus on suboptimality ratio instead of cost difference was deliberate, but appears to create extra challenges.

7.2 Future work

Our control framework for deformable manipulation involved standard choices for recurrent modeling, estimation, and control. For modeling, we are interested in replacing LSTM networks with more specialized classes of learnable models, such as models that place expressive learnable components in a computational graph alongside fixed or low-parameter nodes derived from physics principles (Heiden et al., 2021b). For estimation and control, there are many other standard methods. More broadly, we note that our framework can be seen as the inner loop of a model-based RL algorithm. By closing the loop and updating the model continuously, we can relax the assumption that our initial data gathering sufficiently covers the state space. Our encouraging experimental results, along with the improved debugging and interpretability of our framework relative to model-free RL, make us enthusiastic about further research into model-based RL.

Our work on RL theory suggests two possible follow-ups. As we discussed in § 5.2, the RL theory community has already made great strides on the LQR problem in the past few years. These results include sample complexity bounds that tie the problem parameters more directly to the RL algorithm performance, instead of indirectly through the gradient variance. To our knowledge, the sample complexity upper bound for REINFORCE in LQR problems remains open. Such a bound might help us gain insight into the REINFORCE algorithm. However, due to the strict suboptimality of REINFORCE for cheap-control LQR problems discussed by Tu and Recht (2019), it would not necessarily improve our understanding of the LQR problem itself.

Beyond LQR, our experiments also suggest there is an interesting and complex relationship between action-space noise, environmental noise, and algorithm performance in policy gradient methods. We find our results somewhat surprising, because action-space noise is historically thought to be related to exploration, but exploration is not hard in linear dynamical systems. We are intrigued by the possibility of

designing an RL algorithm that controls action noise specifically for its effect on policy gradient optimization. If sufficient state-space exploration is already guaranteed, can additional action noise still be helpful for policy gradient methods?

We view our work on suboptimal coverings as the first step in a potentially large line of inquiry. There is still much work to do on LQR problems. Beyond LQR, control theory offers many other classes of "almost linear" systems, such as systems with delays, actuator limits, dead zones, sector-bounded (Lur'e) nonlinearities, linear-in-features dynamics, and more. Bounding suboptimal covering numbers for these system classes could also lead to insights. Looking further, we hope to find applications where suboptimal covering numbers can be used to derive other useful properties about sets of dynamical systems. Connecting back to our original motivation, we are especially interested in the possibility of converting covering number bounds into insights about representing multi-system policies with neural networks. We also wish to explore designing RL or adaptive control algorithms based on switching between policies in a known suboptimal cover.

Outside these specific projects, we hope that our theoretical and empirical work can overlap more. In the theoretical work in this dissertation, we have mainly used settings and mathematical tools from control theory to analyze the behavior of existing algorithms and to derive general insights into problem structure. However, the ultimate goal is to use insights and analysis to guide algorithm design. There is a large gap between the empirical success of deep RL and the systems for which provably efficient RL algorithms exist. We hope to contribute to bridging that gap, by building upon the current thorough understanding of simple cases and gradually relaxing assumptions without losing all theoretical guarantees.

Bibliography

Abernethy, Jacob D. and Hazan, Elad (2016). Faster convex optimization: Simulated annealing with an efficient universal barrier. In *International Conference on Machine Learning (ICML)*, pages 2520–2528.

Agarwal, Alekh, Henaff, Mikael, Kakade, Sham, and Sun, Wen (2020). PC-PG: Policy cover directed exploration for provable policy gradient learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13399–13412.

Agarwal, Alekh, Jiang, Nan, Wen, Sham M. Kakade, and Sun (2022). Reinforcement learning: Theory and algorithms. Working draft.

Agarwal, Alekh, Kakade, Sham M., Lee, Jason D., and Mahajan, Gaurav (2019). Optimality and approximation with policy gradient methods in Markov decision processes. *CoRR*, abs/1908.00261.

Agarwal, Rishabh, Schwarzer, Max, Castro, Pablo Samuel, Courville, Aaron C, and Bellemare, Marc (2021). Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 29304–29320.

Agazzi, Andrea and Lu, Jianfeng (2021). Global optimality of softmax policy gradient with single hidden layer neural networks in the mean-field regime. In *International Conference on Learning Representations (ICLR)*.

Allgöwer, Frank and Zheng, Alex (2012). *Nonlinear model predictive control*, volume 26. Birkhäuser.

Amos, Brandon, Rodriguez, Ivan Dario Jimenez, Sacks, Jacob, Boots, Byron, and Kolter, J. Zico (2018). Differentiable MPC for end-to-end planning and control. *CoRR*, abs/1810.13400.

Anderson, Brian D. O., Brinsmead, Thomas S., De Bruyne, Franky, Hespanha, Joao, Liberzon, Daniel, and Morse, A. Stephen (2000). Multiple model adaptive control. Part 1: Finite controller coverings. *International Journal of Robust and Nonlinear Control*, 10(11-12):909–929.

Andrychowicz, Marcin, Crow, Dwight, Ray, Alex, Schneider, Jonas, Fong, Rachel, Welinder, Peter, McGrew, Bob, Tobin, Josh, Abbeel, Pieter, and Zaremba, Wojciech (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5048–5058.

Andrychowicz, Marcin, Raichuk, Anton, Stanczyk, Piotr, Orsini, Manu, Girgin, Sertan, Marinier, Raphaël, Hussenot, Léonard, Geist, Matthieu, Pietquin, Olivier, Michalski, Marcin, Gelly, Sylvain, and Bachem, Olivier (2020). What matters in on-policy reinforcement learning? A large-scale empirical study. *CoRR*, abs/2006.05990.

Antonova, Rika, Cruciani, Silvia, Smith, Christian, and Kragic, Danica (2017). Reinforcement learning for pivoting task. *CoRR*, abs/1703.00472.

Arriola-Rios, Veronica E., Guler, Puren, Ficuciello, Fanny, Kragic, Danica, Siciliano, Bruno, and Wyatt, Jeremy L. (2020). Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, 7:82.

Åström, Karl J and Wittenmark, Björn (2013). *Adaptive Control*. Courier Corporation.

Åström, Karl Johan and Murray, Richard M (2010). *Feedback systems: An introduction for scientists and engineers*. Princeton University Press.

Barbič, Jernej and Popović, Jovan (2008). Real-time control of physically based simulations using gentle forces. *ACM Transactions on Graphics*, 27(5):1–10.

Bern, James M, Banzet, Pol, Poranne, Roi, and Coros, Stelian (2019). Trajectory optimization for cable-driven soft robot locomotion. In *Robotics: Science and Systems (RSS)*.

Bern, James M., Schnider, Yannick, Banzet, Pol, Kumar, Nitish, and Coros, Stelian (2020). Soft robot control with a learned differentiable model. In *International Conference on Soft Robotics (RoboSoft)*, pages 417–423.

Bertsekas, Dimitri P. and Shreve, Steven E. (1978). *Stochastic Optimal Control: The Discrete Time Case*. Mathematics in Science and Engineering. Academic Press.

Bhandari, Jalaj and Russo, Daniel (2019). Global optimality guarantees for policy gradient methods. *CoRR*, abs/1906.01786.

Bhojanapalli, Srinadh, Wilber, Kimberly, Veit, Andreas, Rawat, Ankit Singh, Kim, Seungyeon, Menon, Aditya Krishna, and Kumar, Sanjiv (2021). On the reproducibility of neural network predictions. *CoRR*, abs/2102.03349.

Boffi, Nicholas M., Tu, Stephen, and Slotine, Jean-Jacques E. (2021). Regret bounds for adaptive nonlinear control. In *Conference on Learning for Dynamics and Control (L4DC)*, pages 471–483.

Bousmalis, Konstantinos, Trigeorgis, George, Silberman, Nathan, Krishnan, Dilip, and Erhan, Dumitru (2016). Domain separation networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 343–351.

Boyd, Stephen and Vandenberghe, Lieven (2004). *Convex Optimization*. Cambridge University Press.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech (2016). OpenAI Gym. *CoRR*, abs/1606.01540.

Bruder, Daniel, Gillespie, Brent, Remy, C. David, and Vasudevan, Ram (2019). Modeling and control of soft robots using the Koopman operator and model predictive control. In *Robotics: Science and Systems (RSS)*.

Brunton, Steven L and Kutz, J Nathan (2019). *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.

Bu, Jingjing, Mesbahi, Afshin, Fazel, Maryam, and Mesbahi, Mehran (2019a). LQR through the lens of first order methods: Discrete-time case. *CoRR*, abs/1907.08921.

Bu, Jingjing, Mesbahi, Afshin, and Mesbahi, Mehran (2019b). On topological and metrical properties of stabilizing feedback gains: The MIMO case. *CoRR*, abs/1904.02737.

Bubeck, Sébastien (2015). Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357.

Cai, Qi, Yang, Zhuoran, Lee, Jason D., and Wang, Zhaoran (2019). Neural temporal-difference learning converges to global optima. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11312–11322.

Cassel, Asaf B and Koren, Tomer (2021). Online policy gradient for model free learning of linear quadratic regulators with $\sqrt{T}$ regret. In *International Conference on Machine Learning (ICML)*, pages 1304–1313.

Chebotar, Yevgen, Hausman, Karol, Zhang, Marvin, Sukhatme, Gaurav, Schaal, Stefan, and Levine, Sergey (2017). Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 703–711.

Chen, Tao, Murali, Adithyavairavan, and Gupta, Abhinav (2018). Hardware conditioned policies for multi-robot transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9355–9366.

Deisenroth, Marc Peter and Rasmussen, Carl Edward (2011). PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, and Fei-Fei, Li (2009). ImageNet: A large-scale hierarchical image database. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

Devin, Coline, Gupta, Abhishek, Darrell, Trevor, Abbeel, Pieter, and Levine, Sergey (2017). Learning modular neural network policies for multi-task and multi-robot transfer. In *International Conference on Robotics and Automation (ICRA)*, pages 2169–2176.

Doyle, John C (1978). Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757.

Du, Jingjing, Song, Chunyue, and Li, Ping (2012). Multimodel control of nonlinear systems: An integrated design procedure based on gap metric and H_∞ loop shaping. *Industrial & Engineering Chemistry Research*, 51(9):3722–3731.

Du, Simon, Kakade, Sham, Lee, Jason, Lovett, Shachar, Mahajan, Gaurav, Sun, Wen, and Wang, Ruosong (2021). Bilinear classes: A structural framework for provable generalization in RL. In *International Conference on Machine Learning (ICML)*, pages 2826–2836.

Duan, Yan, Chen, Xi, Houthooft, Rein, Schulman, John, and Abbeel, Pieter (2016a). Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778.

Duan, Yan, Schulman, John, Chen, Xi, Bartlett, Peter L., Sutskever, Ilya, and Abbeel, Pieter (2016b). RL^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779.

Duenser, Simon, Bern, James M, Poranne, Roi, and Coros, Stelian (2018). Interactive robotic manipulation of elastic objects. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3476–3481.

Dullerud, Geir E. and Paganini, Fernando (2000). *A Course in Robust Control Theory: A Convex Approach*. Springer-Verlag New York.

Engstrom, Logan, Ilyas, Andrew, Santurkar, Shibani, Tsipras, Dimitris, Janoos, Firdaus, Rudolph, Larry, and Madry, Aleksander (2020). Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations (ICLR)*.

Eysenbach, Benjamin and Levine, Sergey (2021). Maximum entropy RL (provably) solves some robust RL problems. *CoRR*, abs/2103.06257.

Fan, Jianqing, Wang, Zhaoran, Xie, Yuchen, and Yang, Zhuoran (2020). A theoretical analysis of deep q-learning. In *Conference on Learning for Dynamics and Control (L4DC)*, pages 486–489.

Fazel, Maryam, Ge, Rong, Kakade, Sham, and Mesbahi, Mehran (2018). Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning (ICML)*, pages 1467–1476.

Fekri, Sajjad, Athans, Michael, and Pascoal, Antonio (2006). Issues, progress and new results in robust adaptive control. *International Journal of Adaptive Control and Signal Processing*, 20(10):519–579.

Feng, Fei, Yin, Wotao, Agarwal, Alekh, and Yang, Lin (2021). Provably correct optimization and exploration with non-linear policies. In *International Conference on Machine Learning (ICML)*, pages 3263–3273.

Finn, Chelsea, Abbeel, Pieter, and Levine, Sergey (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135.

Fu, Minyue (1996). Minimum switching control for adaptive tracking. In *Conference on Decision and Control (CDC)*, pages 3749–3754.

Fu, Minyue and Barmish, B. Ross (1986). Adaptive stabilization of linear systems via switching control. *IEEE Transactions on Automatic Control*, 31(12):1097–1103.

Gillespie, Morgan T., Best, Charles M., Townsend, Eric C., Wingate, David, and Killpack, Marc D. (2018). Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. In *International Conference on Soft Robotics (RoboSoft)*, pages 39–45.

Greensmith, Evan, Bartlett, Peter L, and Baxter, Jonathan (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530.

Ha, David and Schmidhuber, Jürgen (2018). Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2455–2467.

Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1856–1865.

Hahn, David, Banzet, Pol, Bern, James M, and Coros, Stelian (2019). Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)*, 38(6):1–13.

He, Fengxiang and Tao, Dacheng (2020). Recent advances in deep learning theory. *CoRR*, abs/2012.10931.

Heiden, Eric, Macklin, Miles, Narang, Yashraj S., Fox, Dieter, Garg, Animesh, and Ramos, Fabio (2021a). DiSECt: A differentiable simulation engine for autonomous robotic cutting. In *Robotics: Science and Systems (RSS)*.

Heiden, Eric, Millard, David, Coumans, Erwin, Sheng, Yizhou, and Sukhatme, Gaurav S. (2021b). NeuralSim: Augmenting differentiable simulators with neural networks. In *International Conference on Robotics and Automation (ICRA)*, pages 9474–9481.

Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina, and Meger, David (2018). Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, pages 3207–3214.

Hespanha, João P, Liberzon, Daniel, and Morse, A Stephen (2000). Bounds on the number of switchings with scale-independent hysteresis: Applications to supervisory control. In *Conference on Decision and Control (CDC)*, pages 3622–3627.

Higgins, Irina, Pal, Arka, Rusu, Andrei A., Matthey, Loïc, Burgess, Christopher, Pritzel, Alexander, Botvinick, Matthew, Blundell, Charles, and Lerchner, Alexander (2017). DARLA: improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1480–1490.

Hinrichsen, Diederich and Pritchard, Anthony J. (2005). *Mathematical Systems Theory I: Modelling, State Space Analysis, Stability and Robustness*. Texts in Applied Mathematics. Springer Berlin Heidelberg.

Hochreiter, Sepp and Schmidhuber, Jürgen (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Holzapfel, Gerhard A. (2002). Nonlinear solid mechanics: A continuum approach for engineering science. *Meccanica*, 37(4):489–490.

Hu, Yuanming, Liu, Jiancheng, Spielberg, Andrew, Tenenbaum, Joshua B., Freeman, William T., Wu, Jiajun, Rus, Daniela, and Matusik, Wojciech (2019). ChainQueen: A real-time differentiable physical simulator for soft robotics. In *International Conference on Robotics and Automation (ICRA)*, pages 6265–6271.

Huang, Sandy H., Papernot, Nicolas, Goodfellow, Ian J., Duan, Yan, and Abbeel, Pieter (2017). Adversarial attacks on neural network policies. *CoRR*, abs/1702.02284.

Hwangbo, Jemin, Sa, Inkyu, Siegwart, Roland, and Hutter, Marco (2017). Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103.

Jalali, Ali Akbar and Golmohammad, Hassan (2012). An optimal multiple-model strategy to design a controller for nonlinear processes: A boiler-turbine unit. *Computers & Chemical Engineering*, 46:48–58.

Jin, Chi, Allen-Zhu, Zeyuan, Bubeck, Sébastien, and Jordan, Michael I. (2018). Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4868–4878.

Jin, Chi, Liu, Qinghua, and Miryoosefi, Sobhan (2021). Bellman eluder dimension: New rich classes of RL problems, and sample-efficient algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13406–13418.

Jin, Chi, Yang, Zhuoran, Wang, Zhaoran, and Jordan, Michael I (2020). Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory (COLT)*, pages 2137–2143.

Kakade, Sham M., Krishnamurthy, Akshay, Lowrey, Kendall, Ohnishi, Motoya, and Sun, Wen (2020). Information theoretic regret bounds for online nonlinear control. *CoRR*, abs/2006.12466.

Kalashnikov, Dmitry, Irpan, Alex, Pastor, Peter, Ibarz, Julian, Herzog, Alexander, Jang, Eric, Quillen, Deirdre, Holly, Ethan, Kalakrishnan, Mrinal, Vanhoucke, Vincent, and Levine, Sergey (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning (CoRL)*, pages 651–673.

Kalashnikov, Dmitry, Varley, Jake, Chebotar, Yevgen, Swanson, Benjamin, Jonschkowski, Rico, Finn, Chelsea, Levine, Sergey, and Hausman, Karol (2021). Scaling up multi-task robotic reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 557–575.

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Konstantinov, Mihail, Gu, D Wei, Mehrmann, Volker, and Petkov, Petko (2003). *Perturbation Theory for Matrix Equations*. Elsevier Science.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114.

Kwon, Wook Hyun, Moon, Young Soo, and Ahn, Sang Chul (1996). Bounds in algebraic Riccati and Lyapunov equations: A survey and some new results. *International Journal of Control*, 64(3):377–389.

Lancaster, Peter and Rodman, Leiba (1995). *Algebraic Riccati Equations*. Clarendon Press.

Lattimore, Tor and Szepesvári, Csaba (2020). *Bandit Algorithms*. Cambridge University Press.

L'ecuyer, Pierre (1990). A unified view of the IPA, SF, and LR gradient estimation techniques. *Management Science*, 36(11):1364–1383.

Levine, Sergey (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909.

Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373.

Li, Yifei, Du, Tao, Wu, Kui, Xu, Jie, and Matusik, Wojciech (2021). DiffCloth: Differentiable cloth simulation with dry frictional contact. *CoRR*, abs/2106.05306.

Lipton, Zachary Chase (2015). A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019.

Liu, Boyi, Cai, Qi, Yang, Zhuoran, and Wang, Zhaoran (2019). Neural trust region/proximal policy optimization attains globally optimal policy. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10564–10575.

Macklin, Miles, Müller, Matthias, and Chentanez, Nuttapong (2016). XPBD: Position-based simulation of compliant constrained dynamics. In *International Conference on Motion in Games*, pages 49–54.

Malik, Dhruv, Pananjady, Ashwin, Bhatia, Kush, Khamaru, Koulik, Bartlett, Peter L., and Wainwright, Martin J. (2018). Derivative-free methods for policy optimization: Guarantees for linear quadratic systems. *CoRR*, abs/1812.08305.

Mania, Horia, Jordan, Michael I., and Recht, Benjamin (2022). Active learning for nonlinear system identification with guarantees. *Journal of Machine Learning Research*, 23:32:1–32:30.

Mania, Horia, Tu, Stephen, and Recht, Benjamin (2019). Certainty equivalence is efficient for linear quadratic control. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10154–10164.

Mcconachie, Dale and Berenson, Dmitry (2018). Estimating model utility for deformable object manipulation using multiarmed bandit methods. *IEEE Transactions on Automation Science and Engineering*, 15(3):967–979.

McConachie, Dale, Dobson, Andrew, Ruan, Mengyao, and Berenson, Dmitry (2020). Manipulating deformable objects by interleaving prediction, planning, and control. *The International Journal of Robotics Research*, 39(8):957–982.

McNichols, Kenneth H. and Fadali, M. Sami (2003). Selecting operating points for discrete-time gain scheduling. *Computers & Electrical Engineering*, 29(2):289–301.

Mirza, Nada Masood (2020). Machine learning and soft robotics. In *International Arab Conference on Information Technology (ACIT)*, pages 1–5.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin (2013). Playing Atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Mohammadi, Hesameddin, Zare, Armin, Soltanolkotabi, Mahdi, and Jovanovic, Mihailo R. (2019). Global exponential convergence of gradient methods over the nonconvex landscape of the linear quadratic regulator. In *Conference on Decision and Control (CDC)*, pages 7474–7479.

Molchanov, Artem, Chen, Tao, Hönig, Wolfgang, Preiss, James A, Ayanian, Nora, and Sukhatme, Gaurav S (2019). Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 59–66.

Müller, Matthias, Heidelberger, Bruno, Teschner, Matthias, and Gross, Markus (2005). Meshless deformations based on shape matching. *ACM Transactions on Graphics (TOG)*, 24(3):471–478.

Murray-Smith, Roderick and Johansen, Tor Arne (1997). *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, London.

Nelles, Oliver (2001). *Nonlinear System Identification*. Springer, Berlin.

Nesterov, Yurii (1983). A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. In *Proceedings of the USSR Academy of Sciences*, pages 543–547.

Nesterov, Yurii (2003). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Nocedal, Jorge and Wright, Stephen J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, second edition.

OpenAI, Akkaya, Ilge, Andrychowicz, Marcin, Chociej, Maciek, Litwin, Mateusz, McGrew, Bob, Petron, Arthur, Paino, Alex, Plappert, Matthias, Powell, Glenn, Ribas, Raphael, Schneider, Jonas, Tezak, Nikolas, Tworek, Jerry, Welinder, Peter, Weng, Lilian, Yuan, Qiming, Zaremba, Wojciech, and Zhang, Lei (2019). Solving Rubik's cube with a robot hand. *CoRR*, abs/1910.07113.

Papadimitriou, Christos H and Tsitsiklis, John N (1987). The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450.

Parisotto, Emilio, Ba, Lei Jimmy, and Salakhutdinov, Ruslan (2016). Actor-mimic: Deep multitask and transfer reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Köpf, Andreas, Yang, Edward Z., DeVito, Zach, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie, and Chintala, Soumith (2019). PyTorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703.

Peng, Xue Bin, Andrychowicz, Marcin, Zaremba, Wojciech, and Abbeel, Pieter (2017). Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537.

Peshkin, Leonid, Meuleau, Nicolas, and Kaelbling, Leslie Pack (1999). Learning policies with external memory. In *International Conference on Machine Learning (ICML)*, pages 307–314.

Petersen, Ian R. and McFarlane, Duncan C. (1994). Optimal guaranteed cost control and filtering for uncertain linear systems. *IEEE Transactions on Automatic Control*, 39(9):1971–1977.

Pinto, Lerrel, Davidson, James, Sukthankar, Rahul, and Gupta, Abhinav (2017). Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702.

Preiss, James A., Arnold, Sébastien M. R., Wei, Chen-Yu, and Kloft, Marius (2019). Analyzing the variance of policy gradient estimators for the linear-quadratic regulator. In *NeurIPS Workshop on Optimization Foundations for Reinforcement Learning*.

Preiss, James A., Hausman, Karol, and Sukhatme, Gaurav S. (2018). Learning a system-ID embedding space for domain specialization with deep reinforcement learning. In *NeurIPS Workshop on Reinforcement Learning under Partial Observability*.

Preiss, James A., Millard, David, Yao, Tao, and Sukhatme, Gaurav S. (2022). Tracking fast trajectories with a deformable object using a learned model. In *International Conference on Robotics and Automation (ICRA)*.

Preiss, James A. and Sukhatme, Gaurav S. (2021). Suboptimal coverings for continuous spaces of control tasks. In *Conference on Learning for Dynamics and Control (L4DC)*, pages 547–558.

Qiao, Yi-Ling, Liang, Junbang, Koltun, Vladlen, and Lin, Ming C. (2020). Scalable differentiable physics for learning and control. In *International Conference on Machine Learning (ICML)*, pages 7847–7856.

Raffin, Antonin, Hill, Ashley, Gleave, Adam, Kanervisto, Anssi, Ernestus, Maximilian, and Dormann, Noah (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Rosenblatt, Frank (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rusu, Andrei A., Vecerík, Matej, Rothörl, Thomas, Heess, Nicolas, Pascanu, Razvan, and Hadsell, Raia (2017). Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning (CoRL)*, pages 262–270.

Sabelhaus, Andrew P. and Majidi, Carmel (2021). Gaussian process dynamics models for soft robots with shape memory actuators. In *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, pages 191–198.

Sadeghi, Fereshteh and Levine, Sergey (2017). CAD2RL: real single-image flight without a single real image. In *Robotics: Science and Systems (RSS)*.

Safonov, Michael and Athans, Michael (1977). Gain and phase margin for multiloop LQG regulators. *IEEE Transactions on Automatic Control*, 22(2):173–179.

Schaul, Tom, Horgan, Daniel, Gregor, Karol, and Silver, David (2015). Universal value function approximators. In *International Conference on Machine Learning (ICML)*, pages 1312–1320.

Schrittwieser, Julian, Antonoglou, Ioannis, Hubert, Thomas, Simonyan, Karen, Sifre, Laurent, Schmitt, Simon, Guez, Arthur, Lockhart, Edward, Hassabis, Demis, Graepel, Thore, et al. (2020). Mastering Atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael I., and Moritz, Philipp (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pages 1889–1897.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Shalev-Shwartz, Shai, Shamir, Ohad, Srebro, Nathan, and Sridharan, Karthik (2010). Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11:2635–2670.

Shi, Guanya, Hönig, Wolfgang, Shi, Xichen, Yue, Yisong, and Chung, Soon-Jo (2021). Neural-Swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. *IEEE Transactions on Robotics*, pages 1–17.

Shi, Guanya, Shi, Xichen, O'Connell, Michael, Yu, Rose, Azizzadenesheli, Kamyar, Anandkumar, Animashree, Yue, Yisong, and Chung, Soon-Jo (2019). Neural lander: Stable drone landing control using learned dynamics. In *International Conference on Robotics and Automation (ICRA)*, pages 9784–9790.

Siciliano, Bruno, Sciavicco, Lorenzo, Villani, Luigi, and Oriolo, Giuseppe (2009). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer-Verlag, London.

Silver, David, Hubert, Thomas, Schrittwieser, Julian, Antonoglou, Ioannis, Lai, Matthew, Guez, Arthur, Lanctot, Marc, Sifre, Laurent, Kumaran, Dharshan, Graepel, Thore, et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Simchowitz, Max and Foster, Dylan (2020). Naive exploration is optimal for online LQR. In *International Conference on Machine Learning (ICML)*, pages 8937–8948.

Singh, Sumeet, Richards, Spencer M, Sindhwani, Vikas, Slotine, Jean-Jacques E, and Pavone, Marco (2021). Learning stabilizable nonlinear dynamics with contraction-based regularization. *The International Journal of Robotics Research*, 40(10-11):1123–1150.

Song, Yuda and Sun, Wen (2021). PC-MLP: Model-based reinforcement learning with policy cover guided exploration. In *International Conference on Machine Learning (ICML)*, pages 9801–9811.

Sontag, Eduardo D. (2013). *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer Science & Business Media.

Sperry, Elmer A (1921). Aeroplane stabilizer. US Patent US1368226A.

Stilwell, Daniel J. and Rugh, Wilson J. (1999). Interpolation of observer state feedback controllers for gain scheduling. *IEEE Transactions on Automatic Control*, 44(6):1225–1229.

Sulsky, D., Chen, Z., and Schreyer, H. L. (1994). A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 118(1):179–196.

Sun, Ji-Guang (1998). Perturbation theory for algebraic Riccati equations. *SIAM Journal on Matrix Analysis and Applications*, 19(1):39–65.

Sun, Yue and Fazel, Maryam (2021). Learning optimal controllers by policy gradient: Global optimality via convex parameterization. In *Conference on Decision and Control (CDC)*, pages 4576–4581.

Sutton, Richard S. and Barto, Andrew G. (2018). *Reinforcement learning: An introduction*. Adaptive computation and machine learning. MIT Press, second edition.

Tan, Wen, Marquez, Horacio J., and Chen, Tongwen (2004). Operating point selection in multimodel controller design. In *American Control Conference*, pages 3652–3657.

Tao, Terence (2012). *Topics in random matrix theory*. Graduate studies in mathematics. American Mathematical Society, Providence, RI.

Teh, Yee Whye, Bapst, Victor, Czarnecki, Wojciech M., Quan, John, Kirkpatrick, James, Hadsell, Raia, Heess, Nicolas, and Pascanu, Razvan (2017). Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4499–4509.

Terzi, Enrico, Bonassi, Fabio, Farina, Marcello, and Scattolini, Riccardo (2021). Learning model predictive control with long short-term memory networks. *International Journal of Robust and Nonlinear Control*.

Thieffry, Maxime, Kruszewski, Alexandre, Duriez, Christian, and Guerra, Thierry-Marie (2018). Control design for soft robots based on reduced-order model. *IEEE Robotics and Automation Letters*, 4(1):25–32.

Thuruthel, Thomas George, Falotico, Egidio, Renda, Federico, and Laschi, Cecilia (2019). Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. *IEEE Transactions on Robotics*, 35(1):124–134.

Tits, André L and Yang, Yaguang (1996). Globally convergent algorithms for robust pole assignment by state feedback. *IEEE Transactions on Automatic Control*, 41(10):1432–1452.

Tobin, Josh, Fong, Rachel, Ray, Alex, Schneider, Jonas, Zaremba, Wojciech, and Abbeel, Pieter (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

Todorov, Emanuel, Erez, Tom, and Tassa, Yuval (2012). MuJoCo: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033.

Tonkens, Sander, Lorenzetti, Joseph, and Pavone, Marco (2021). Soft robot optimal control via reduced order finite element models. In *International Conference on Robotics and Automation (ICRA)*, pages 12010–12016.

Trefethen, Lloyd N and Embree, Mark (2005). *Spectra and pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press.

Tropp, Joel A. (2015). An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230.

Tu, Stephen and Recht, Benjamin (2019). The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. In *Conference on Learning Theory (COLT)*, pages 3036–3083.

van Baar, Jeroen, Sullivan, Alan, Cordorel, Radu, Jha, Devesh K., Romeres, Diego, and Nikovski, Daniel (2019). Sim-to-real transfer learning using robustified controllers in robotic tasks involving complex dynamics. In *International Conference on Robotics and Automation (ICRA)*, pages 6001–6007.

van Hasselt, Hado P, Guez, Arthur, Hessel, Matteo, Mnih, Volodymyr, and Silver, David (2016). Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems*, pages 4287–4295.

Wang, Lingxiao, Cai, Qi, Yang, Zhuoran, and Wang, Zhaoran (2020). Neural policy gradient methods: Global optimality and rates of convergence. In *International Conference on Learning Representations (ICLR)*.

Wang, Sheng-De, Kuo, Te-Son, and Hsu, Chen-Fa (1986). Trace bounds on the solution of the algebraic matrix Riccati and Lyapunov equation. *IEEE Transactions on Automatic Control*, 31(7):654–656.

Wei, Chen-Yu, Jahromi, Mehdi Jafarnia, Luo, Haipeng, Sharma, Hiteshi, and Jain, Rahul (2020). Model-free reinforcement learning in infinite-horizon average-reward Markov decision processes. In *International Conference on Machine Learning (ICML)*, pages 10170–10180.

Wierstra, Daan, Foerster, Alexander, Peters, Jan, and Schmidhuber, Juergen (2007). Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, pages 697–706.

Williams, Ronald J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Wriggers, Peter (2008). *Nonlinear Finite Element Methods*. Springer-Verlag, Berlin Heidelberg.

Xu, Pan and Gu, Quanquan (2020). A finite-time analysis of Q-learning with neural network function approximation. In *International Conference on Machine Learning (ICML)*, pages 10555–10565.

Yang, Zhaoyang, Merrick, Kathryn E, Abbass, Hussein A, and Jin, Lianwen (2017). Multi-task deep reinforcement learning for continuous action control. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3301–3307.

Yang, Zhuoran, Jin, Chi, Wang, Zhaoran, Wang, Mengdi, and Jordan, Michael I. (2020). Provably efficient reinforcement learning with kernel and neural function approximations. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yoon, Myung-Gon, Ugrinovskii, Valery A., and Pszczel, Marek (2007). Gain-scheduling of minimax optimal state-feedback controllers for uncertain LPV systems. *IEEE Transactions on Automatic Control*, 52(2):311–317.

Yu, Tianhe, Kumar, Saurabh, Gupta, Abhishek, Levine, Sergey, Hausman, Karol, and Finn, Chelsea (2020). Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.

Yu, Tianhe, Quillen, Deirdre, He, Zhanpeng, Julian, Ryan, Hausman, Karol, Finn, Chelsea, and Levine, Sergey (2019). Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 1094–1100.

Yu, Wenhao, Tan, Jie, Liu, C. Karen, and Turk, Greg (2017). Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems (RSS)*.

Zhang, Chiyuan, Vinyals, Oriol, Munos, Rémi, and Bengio, Samy (2018). A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893.

Zhu, Jihong, Cherubini, Andrea, Dune, Claire, Navarro-Alarcon, David, Alambeigi, Farshid, Berenson, Dmitry, Ficuciello, Fanny, Harada, Kensuke, Li, Xiang, Pan, Jia, and Yuan, Wenzhen (2021). Challenges and outlook in robotic manipulation of deformable objects. *CoRR*, abs/2105.01767.

Zhu, Yuke, Wang, Ziyu, Merel, Josh, Rusu, Andrei A., Erez, Tom, Cabi, Serkan, Tunyasuvunakool, Saran, Kramár, János, Hadsell, Raia, de Freitas, Nando, and Heess, Nicolas (2018). Reinforcement and imitation learning for diverse visuomotor skills. In *Robotics: Science and Systems (RSS)*.

Zimmermann, Simon, Poranne, Roi, and Coros, Stelian (2021). Dynamic manipulation of deformable objects with implicit integration. *IEEE Robotics and Automation Letters*, 6(2):4209–4216.