Integration of Control and Dynamical Systems Perspectives to Machine Learning

Motoya Ohnishi

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2024

Reading Committee:
Sham Kakade, Chair
Emanuel Todorov, Chair
Fabio Ramos

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Integration of Control and Dynamical Systems Perspectives to Machine Learning

Motoya Ohnishi

Co-Chairs of the Supervisory Committee:
Sham Kakade
Computer Science & Engineering

Emanuel Todorov
Computer Science & Engineering

With the increasing demands on artificial intelligence technology operating over sequential data, represented by robotics and language processing, there has been a surge of interest in interdisciplinary research spanning machine learning – a data-driven approach based on statistics – and control or dynamical systems theory, which deals with dynamic environments. Because those streams of studies have evolved in a relatively separate manner under different settings and formulations, their integration becomes an intricate task, requiring a fresh look at the existing approach. This thesis primarily revolves around a discussion of research endeavors in the intersection of machine learning and dynamical systems to exploit the best of both worlds, and proposes some of the novel techniques and paradigms made possible by bringing the unique perspectives and concepts from these domains creatively. I initially provide a succinct overview of the state of the art in the related domains followed by my contributions to the fields. First of all, this thesis begins with the work that synthesizes control tools in a learning system to devise algorithms with control theoretic guarantees. In this process, a novel control concept, limited-duration safety, is proposed with discussions on its application within the context of transfer learning. Secondly, a novel model-based reinforcement learning (RL) algorithm is presented, leveraging a recent control theoretic tool as an oracle embedded in the algorithm to provably ensure learning efficiency. With a novel problem formulation with the Koopman operator, which is cast as a generalization of pole assignments to nonlinear decision making, a diverse array of dynamic behaviors are realized. Thirdly, as an additional highlight of exploration, I present the successful extension of RL beyond its conventional reliance on the Bellman equation, encompassing dynamic programming across entire paths. The new framework grounded in theoretical advancements of path signatures has proven beneficial in addressing challenges related to path following. On the other hand, merging machine learning, rooted in statistics, and dynamical systems raises several challenges. In particular, fourthly, this thesis discusses a specific challenge of loss of dynamic structure information that might be caused by concentrations of measures, which is overcome by carefully adopting the asymptotic results of exponential sums. Lastly, a machine learning algorithm itself can be seen as a dynamical system, and this perspective has theoretical and practical potential for handling complex machine learning domains. Especially for a deep RL algorithm, the constructive approach is taken to analyze, in a retroductive manner, the phenomena and performance separations observed in the systems of interest. This thesis is also intended to open a novel direction of further research emerging out of amalgamations of learning algorithms and dynamical systems perspectives, and is concluded with a remark for the potential future work.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

My Ph.D. study, started in 2019 with a newly built research center at the University of Washington (UW), has been full of events which have greatly led my personal growth. Just after around six months since joining the UW, COVID-19 pandemic hit the world, resulting in a long period of unusual experience which had sometimes brought loneliness and a feeling of incompetence. Nevertheless, I could finally reach to the completion of the study thanks to the warm supports, encouragements and tolerance from my family, the academic advisors, the program advisors, internship mentors and friends inside and outside the UW.

This thesis is the fruits of such five years of enjoyments, introspections, efforts, excitements and hopes for the future endeavour, and I would like to convey my sincerest gratitudes to those who have been toghether with me throughout my Ph.D. journey.

First of all, I am profoundly grateful to my parents, Kenji Ohnishi and Mika Ohnishi, for letting me come to the United States to pursue the Ph.D. study at the UW, and for their continual encouragements.

My study started with three advisors, Sham Kakade, Emanuel Todorov, and Siddhartha Srinivasa, and I can't thank them enough for accepting me as a Ph.D. student at the UW. In particular, after Sham and Emo officially became my permanent Ph.D. advisors, they have generously given me an academic freedom of pursuing my own research interests while also giving me a training of statistical machine learning, perspectives of control systems, and a precious network of the alumni, all of which have helped me grow as an independent researcher. I am very thankful for them.

Also, I am very grateful to the committee members Krithika Manohar, Armita Nourmohammad and Fabio Ramos for taking time out of their busy schedule for me. Krithika gave me valuable perspectives of the intersectional study of dynamical system and machine learning and kindly introduced me related venues and researchers, Armita kindly found relevance of my work in her research and gave me an advice on a potential future research within the intersection of biology and dynamical systems, and Fabio, who has also been my mentor at NVIDIA, has introduced me to an interesting research direction and to the network of researchers.

For the first years of studies, I oftentimes received academic and technical supports from Kendall Lowrey and Aravind Rajeswaran, and interacted with Svet Kolev, Vikash Kumar, Colin Summers and Ben Evans; Aravind's strong suggestion of choosing UW for my Ph.D. study made me determined, and Kendall has especially guided me to learn some of the software frameworks and "roboticist way" of thinking.

Along the way, my collaborators and friends in academia, including the Niels Bohr Institute I have stayed as a visitor, and at the companies I interned for, namely, Sony AI America, NVIDIA, and HYPRLABS, have greatly helped me; in particular, I would like to thank Magnus Egerstedt, Gennaro Notomista, Li Wang, Masashi Sugiyama, Yoshinobu Kawahara, Kenichi Bannai, Isao Ishikawa, Masahiro Ikeda, Yuko Kuroki, Kai Koike, Wen Sun, Akshay Krishnamurthy, Thomas Walsh, James MacGlashan, Kaushik Subramanian, Peter Wurman, Peter Stone, Takuma Seno, Kenta Kawamoto, Iretiayo Akinola, Jie Xu, Ajay Mandlekar, Dieter Fox, Tim Kentley Klay, Werner Duvaud, Maxime Deloche, Aditya Narayan, Mogens Jensen, and Kunihiko Kaneko. Other friends at the UW, including Aditya Kusupati, who joined the UW to be Sham's student in the same year, have encouraged me a lot along the pathway, and I have enjoyed daily conversations with them.

Importantly, I would like to express my huge thanks to Elise DeGoede Dorough, Elle Brown, Eric Eto, Willard Johnson, Jin Xu, Lisa Merlin, Joe Eckert, and Magdalena Balazinska. Their warm, kind and precise supports have always saved me throughout the Ph.D. study.

Finally, I would also like to acknowledge the monetary supports from Wissner-Slivka Endowed Fellowship, Funai Overseas Scholarship, and JST AIP Challenge (2021 and 2023). In particular, it was the pleasure meeting with Benjamin Slivka, Katsumi Emura (AIP Lab director), and the members and fellow collegues of Funai Foundations, including Akira Funai, Takashi Masuda, Keiko Saito, and Mikiko Kondo.

DEDICATION

to my family

NOMENCLATURE

Unless otherwise specified, the following notation is used throughout this thesis.

Symbol	**Meaning**
$\mathbb{R}$	The set of the real numbers
$\mathbb{R}_{\geq 0}$	The set of the nonnegative real numbers
$\mathbb{R}_{>0}$	The set of the positive real numbers
$\mathbb{N}$	The set of the natural numbers ($\{0,1,2,\ldots\}$)
$\mathbb{Z}_{>0}$	The set of the positive integers
$\mathbb{Q}$	The set of the rational numbers
$\mathbb{Q}_{>0}$	The set of the positive rational numbers
$\mathbb{C}$	The set of the complex numbers
$\mathcal{L}(\mathcal{A};\mathcal{B})$	The set of bounded linear operators from $\mathcal{A}$ to $\mathcal{B}$
$C^1(\mathcal{A})$	The class of continuously differentiable function defined over $\mathcal{A}$
$\mathrm{int}(\mathcal{A}),\ \partial\mathcal{A}$	The interior set and the boundary of a set $\mathcal{A}$, respectively
$\Delta(\mathcal{A})$	The set of probability distributions over $\mathcal{A}$
$(\cdot)^\top$	Transposition
$(\cdot)^\dagger$	Adjoint
$(\cdot)^+$	(The Moore-Penrose) pseudo inverse
$\langle\cdot,\cdot\rangle_{\mathcal{H}}$	The inner product in the Hilbert space $\mathcal{H}$; $\langle x,y\rangle_{\mathbb{R}^d} := x^\top y$
$\lVert x\rVert_{\mathbb{R}^d},\ \lVert x\rVert_1$	$\sqrt{\langle x,x\rangle}$, and $\sum_{i=1}^d \lvert x_i\rvert$ for $x := [x_1,\ldots,x_d]^\top \in \mathbb{R}^d$, respectively
$\lVert\mathcal{A}\rVert,\ \lVert\mathcal{A}\rVert_{\mathrm{HS}}$	The spectral and the Hilbert-Schmidt norms of $\mathcal{A}$, respectively
$\lVert f\rVert_\infty$	The infinity norm of f ((real-valued) bounded function or vector)

Symbol	**Meaning**
$[H]$	$\{0,1,\ldots,H-1\}$ for $H \in \mathbb{Z}_{>0}$
$\mathscr{I}(M),\ \mathscr{N}(M)$	The image and the null spaces of M, respectively
$\lfloor a \rfloor,\ \lceil a \rceil$	The floor and the ceiling of a real number a, respectively
$\mathrm{lcm}(\mathcal{L})$	The least common multiple of a set $\mathcal{L}$ of positive integers
$\gcd(\mathcal{L})$	The greatest common divisor of a set $\mathcal{L}$ of positive integers
L_f	The Lie derivative along a function f
$O(\cdot)$	Big O notation providing an upper bound on the growth rate
$\tilde{O}(\cdot)$	An upper bound on the growth rate dropping logarithmic factor
$\mathbb{E}[\cdot]$	Expectation
$\Pr[\mathcal{E}]$	Probability of an event $\mathcal{E}$
$\otimes$	Tensor product operation
$\mathbb{T}$	Either $\mathbb{N}$ (discrete-time) or $\mathbb{R}_{\geq 0}$ (continuous-time)
$\emptyset$	The empty set
$\mathfrak{Re}[\cdot]$	The real part of a complex number
$\mathfrak{Im}[\cdot]$	The imaginary part of a complex number
$\mathrm{tr}[\cdot]$	The trace of a linear operator (or square matrix)

- We use h or H as time indices for discrete-time settings in which case t and T may denote *episode*; while, for continuous-time settings, we often use t and T for time points. However, we also use t as a discrete-time index under non-episodic settings; this should be clear in the context.

Chapter 1
INTRODUCTION

Departing from the rule-based paradigm of artificial intelligence, the domain of machine learning (ML), deeply rooted in statistical foundations, has witnessed a remarkable trajectory of advancements over the past decades. This evolution has been propelled by studies of cost design, model selection, and optimization, coupled with the substantial acceleration made possible by the confluence of vast computational resources and expansive datasets.

1.1 Motivation

In [174], ML is perceived as a study of computer programs that improve on the *performance* with respect to some *tasks* through *experience*. On the other hand, [51] contrasts the study of statistics and ML based on what they serve for, and states that ML focuses more on finding generalizable predictive patterns. Therefore, it is naturally the case that optimization is an essential aspect of ML aside from statistic and *learning* or *improvement* happens only when there is a specific class of tasks in mind. This perspective also applies to controls or optimal control in particular. As such, encapsulating ML and controls through the lens of optimization could be a promising direction of research for advancing both domains. Nevertheless, the challenges of dealing with nonstationary environments and confronting problems that may not be easily dealt with by conventional cost optimization strategies without indefinitely expanding the computational resource pose intrinsic hurdles within the current paradigm. These hurdles echo the struggles encountered by the rule-based paradigm of yore [170], known as the frame problem.

In terms of practical applications, recent research activities of ML within the domains such as robotics, language processing, and AI-human collaborations highlight the needs of

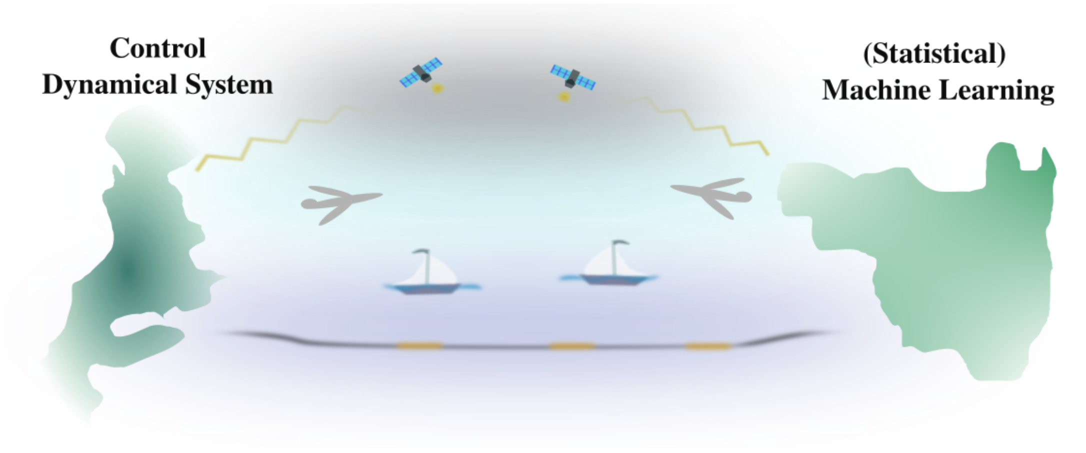


Figure 1.1: This thesis aims at bridging the fields of studies on dynamical systems/control and statistical machine learning from several angles and methodologies.

sophisticated ML systems that robustly operate in a dynamic environment and data. Especially, dealing with dynamically changing environments, catching up with the ever updating corpus, or ensuring smooth interactions of multiple agents including human users is an intricate task partially due to the nature of data-driven approach rooted in statistics and optimization (cf. [85, 107, 115, 87]). As biological phenomena are well-characterized as dynamical systems (e.g., universal biology [120] and applications of statistical physics [80]) such as attractor dynamics and dissipative systems, it is natural to imagine that incorporating the dynamical system viewpoint will enable constructions of simple yet robust AI systems.

In fact, an *improvement* or *update* of a system implicitly assumes some notion of *time*; and ML system itself is inseparable from dynamics. Not only the output of ML such as a control sequence, video stream, and a sequence of words, but the evolution of system parameters (e.g., neural network parameters) and the neural network itself can be regarded as dynamical systems. In lieu of this, it is insightful to recognize that the Turing machine, the very bedrock of computer science, lends itself to a mathematical characterization as a dynamical system (cf. [177]). This prompts us to contemplate the potential of reimagining ML models, algorithms, and analytical approaches through the lens of dynamical systems.

Because the studies of ML and dynamical systems (or control) have evolved in a relatively separate manner, there naturally exist conceptual discrepancies when attempting to adopt ideas from one to the other, requiring some creative leaps to alter the perspectives on the existing approach; and this is the fundamental motivation behind this thesis.

1.2 Thesis statement and summary

This thesis revolves around the research on bridging dynamical systems and (statistical) ML from several angles. In the course of my academic journey, particularly during my Ph.D. program, I have made tangible contributions aimed at bridging the gap between them; and this thesis presents the contributions from the following five angles (see fun picture in Figure 1.1):

- Control theoretic guarantees for ML algorithm
- Problem formulations through the lens of dynamical systems
- Trajectory-based optimization for control and learning tasks
- Algorithm design at the intersection of statistical ML and dynamical systems
- Constructive approach for analyzing complex ML algorithms as dynamical systems

Each of the above corresponds to a single chapter, constituting the main body of this thesis. Before delving into their summaries below, I hereby make the following statement penetrating the entire thesis.

1.2.1 Thesis Statement

Adoptions of concepts, unique problem formulations, control methods, methodologies, and mathematical tools and results of dynamical systems to (statistical) machine learning lead to

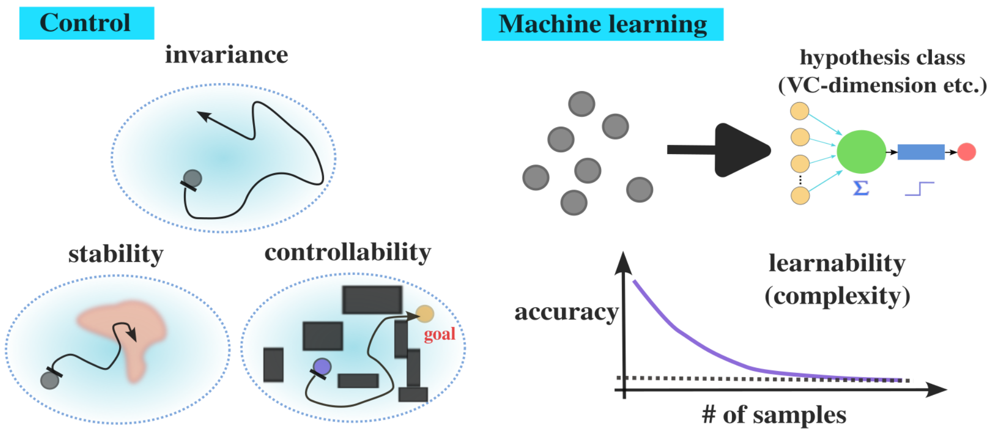


Figure 1.2: Some of the important concepts seen in control and ML. They exhibit attention to various properties and concepts inherent within the system from different perspectives.

- *creations of novel decision making and system identification paradigms with certain guarantees, which help efficiently solve important control and identification problems and produce novel concepts in both of the domains in a way, and further to*
- *a dynamical system perspective of learning algorithm itself that enables natural scientific analysis of the system, and that possibly creates a novel paradigm of artificial intelligence research in the future, including some revival of the aspects in the past.*

1.2.2 Control Theoretic Guarantees for Machine Learning Algorithm

The realm of theoretical exploration in ML algorithms predominantly relies on statistical analyses, with a primary emphasis on facets of learnability, including computational and sample complexity analyses [219]. On the contrary, control theory, traditionally studied from the perspectives of (asymptotic) stability, controllability, observability, invariance, and adaptation, finds its footing in the domain of differential (or difference) equations, linear algebra, and spectral theory [126] (see Figure 1.2). The recent surge of interest surrounding ML algorithms in the context of robotics and control systems underscores the pivotal importance of not only ensuring learnability but also providing control theoretic guarantees for the learning system that dynamically interacts with a controlled agent.

In Chapter 3, we especially consider the problem of ensuring state constraints, which we regard as *safety* (see Figure 1.3 (left)), for controlled agents. Recent advancements of theory and practical application of control barrier functions (cf. [16]) have enabled systematically ensuring (forward) invariance of a given set of states through a constraint on the control selection every time; which therefore reduces the long horizon state constraints to the constraint on instantaneous control selections. However, obtaining such a valid barrier function for a given set of states is not straightforward or even hopeless if the agent is unstabilizable.

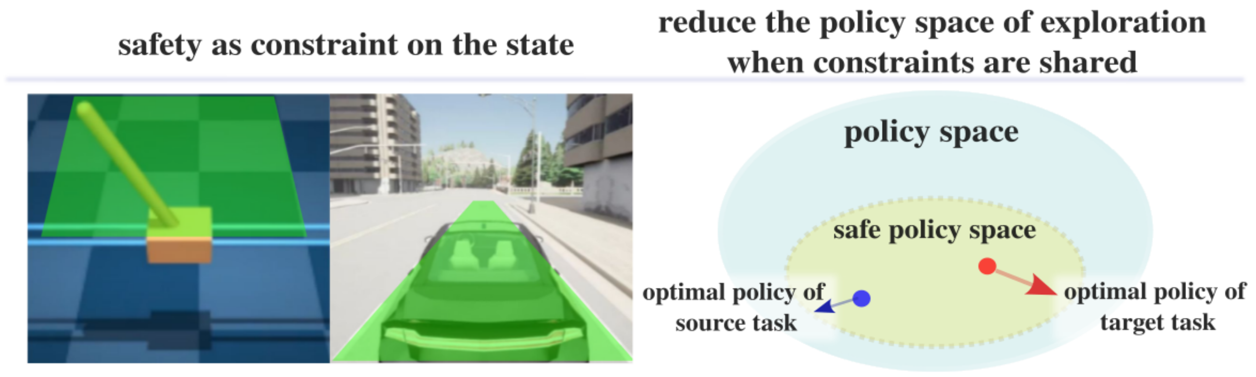

safety as constraint on the state
reduce the policy space of exploration when constraints are shared
policy space
safe policy space
optimal policy of source task
optimal policy of target task

Figure 1.3: We consider safety as constraints on the state in this thesis; e.g., keeping the pole staying straight up or ensuring that the car remains in a lane. A valid control barrier function, if exists, can be employed to ensure safety. Also, within the context of transfer learning, one could reduce the policy space of exploration when learning target task if some constraints are shared between the source and target tasks.

To overcome this challenge, we will present our work [191] that studied the value function induced by a cost adhering to particular conditions as a potential candidate for a control barrier function. Because in most cases obtaining a (bounded) value function necessitates the introduction of a discount factor, this work introduced a novel control theoretic concept, namely, limited-duration safety. We have demonstrated that such a value function serves as

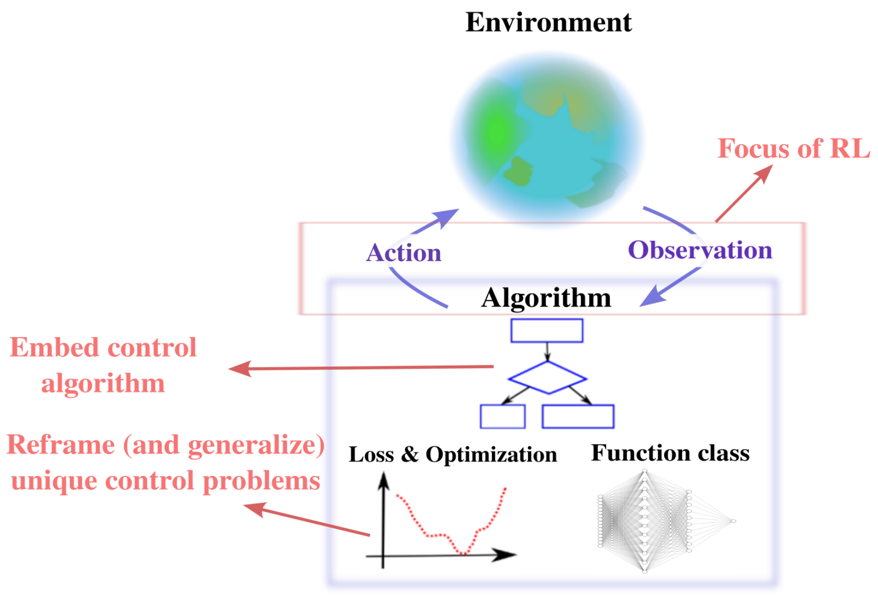


Figure 1.4: Reinforcement learning (RL), a domain of research in ML, focuses on the interaction of the agent and the environment. RL aims at finding a sequence of actions to influence the environment to obtain a desirable outcome based on the sequentially available observations sampled from the environment. We consider embedding a state-of-the-art control method into learning algorithms to design an efficient learner and reframing unique control problems as learning problems by properly designing the loss.

a *limited-duration control barrier function*, which is used to ensure limited-duration safety. This (relaxed) safety guarantee mechanism can be applied to transfer learning (cf. [194]) through reduction of the space to be explored for new tasks that share the same constraints as the source (see Figure 1.3 (right)).

1.2.3 Problem Formulations through the Lens of Dynamical Systems

When the problem of interest deals with dynamical systems – encompassing domains such as control, system identification, and future behavior prediction – ML algorithms often involve seamless interaction with a dynamically shifting environment. The objective is to ensure convergence toward a desired outcome under relatively benign assumptions. A prime illustration of this is found in reinforcement learning (RL) (cf. [7]), where the learning environment is commonly characterized as a Markov Decision Process (cf. [36]), and dynamic programming (cf. [35]) plays a key role (see Figure 1.4).

An RL algorithm is aimed at finding an optimal sequence of control actions to take to achieve the highest value (the cumulative rewards along the trajectory) by interacting with the environment; and *efficient* learning algorithm is typically demanded, where a sequence of actions attaining sufficiently high value can be obtained with a small number of interactions that generate data samples. This *sample complexity* of the algorithm should not depend on the number of states or control actions for those having continuous or extremely large size of state and action spaces, such as the ones in visual input based robot learning problems. In particular, for robotics problem we consider (see Figure 1.5 for some of the environments studied in this thesis), the algorithm is expected to work in the complex environment sufficiently well in practice as well.

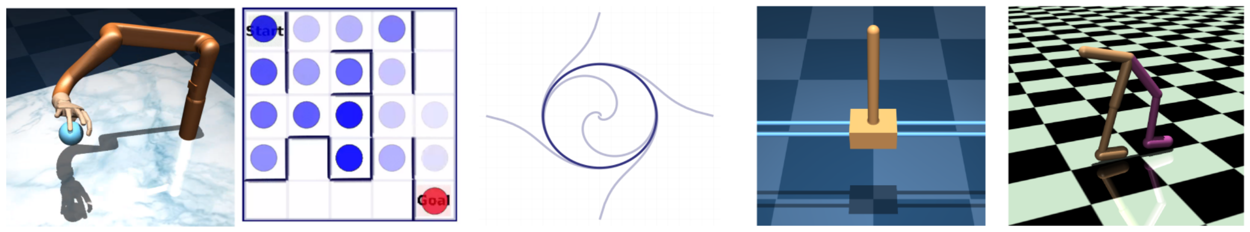


Figure 1.5: Some of the environments used for the control experiments. From the left, they are a robot arm picking up a spherical object, a discrete maze (red circle indicates the current agent's position and the thickness of blue circles corresponds to the times the agent has visited those states), limit cycle dynamics, a cart pole, and a walker.

In Chapter 4, we will present the work on a model-based RL [119] tailored for continuous controls, leveraging a recent control theoretic tool as an oracle to provably ensure learning efficiency. By assuming and exploiting *physical structure* represented as the transition dynamics spanned by a known set of features, and by embedding the control tools as an optimization oracle into the RL algorithm, the aforementioned requirements on sample

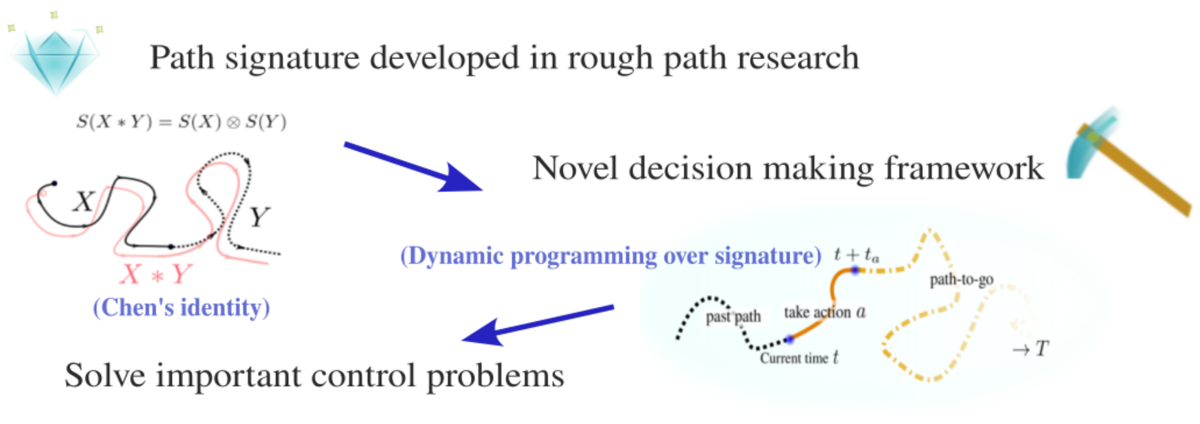

Path signature developed in rough path research
$S(X*Y) = S(X) \otimes S(Y)$
X
Y
$X*Y$
(Chen's identity)
Novel decision making framework
(Dynamic programming over signature)
$t+t_a$
path-to-go
past path
take action a
Current time t
$\to T$
Solve important control problems

Figure 1.6: Path signature is a mathematical concept developed in the rough path community studying controlled differential equations (cf. [163]). Using the algebraic property called Chen's identity [57], we developed novel decision making framework of ML which is then used to solve some of the important control problems.

complexity and on practicality are successfully met.

On the other hand, in the context of RL, the optimization objective is the cumulative rewards along the agent's trajectory; which resembles the principle of least action (cf. [84]) observed in the realm of physics. Our work [190] draws inspiration from classical eigenstructure (or pole) assignment techniques (cf. [19]), widely explored within the purview of linear control systems. It articulates the learning objective as the optimization not only of the cumulative rewards but of eigenstructures of the so-called Koopman operator [136], a linear operator that describes the evolution of observables. This approach enables dynamical systems realizations that remain infeasible within a cumulative cost (or reward) framework, embracing a broader spectrum of dynamic behaviors that evolve over stable manifolds, which encompass nonlinear oscillators, closed loops, and smooth movements.

1.2.4 Trajectory-based Optimization for Control and Learning Tasks

Dynamical systems generate trajectories or time sequences; it is hence natural to ask learning algorithms to operate on the space of trajectories, and in fact, the recent success on large language models (e.g., [47]) is highlighted by the capability of the Transformer model [252], which deals with sequential data.

The difficulty of dealing with problems such as trajectory following is particularly pronounced if an appropriate set of waypoints is unavailable. Within the current decision making paradigm relying on either an estimate of value (i.e., cumulative rewards) or single-step dynamics used to plan the optimal sequence of control actions through search, well-designed waypoints that are (almost) feasible to track along the target trajectory which could potentially be very long are oftentimes indispensable to direct the agent's behavior (cf. [198]).

Guided by the insights drawn from rough-path theory that considers controlled differential equations (cf. [163]), in Chapter 5 presenting the work [188], we extend the decision making framework based on the Bellman equation [35] to the domain of paths by leveraging the path signatures (cf. [63, 162]). Path signature is a powerful representation capturing the analytical and geometric traits of paths with remarkable efficiency, thanks to their algebraic properties including fast concatenation of paths achieved through tensor products (cf. [57]). This algebraic property is called Chen's identity [57], and is exploited as a basis of dynamic programming to build a novel decision making framework (see Figure 1.6).

Essentially, we establish connections between value functions and intriguing properties of path signatures. This generalized framework naturally deals with varying and adaptive time steps, propagating higher-level information more efficiently than traditional value function updates, all while retaining its resilience in the face of dynamical system misspecification over extended rollouts. The novel control method that stems from this framework, which particularly shines when applied to trajectory following problems, stands as a generalization of integral controls (cf. [126]), thereby exhibiting robustness against disturbance, and opens an innovative ML paradigm.

1.2.5 Algorithm Design at the Intersection of Statistical ML and Dynamical Systems

Recent investigations into Neural Ordinary Differential Equations (NODEs) [58], learning dynamics (e.g., [99]), and the associated analyses have reminded us that a composition of

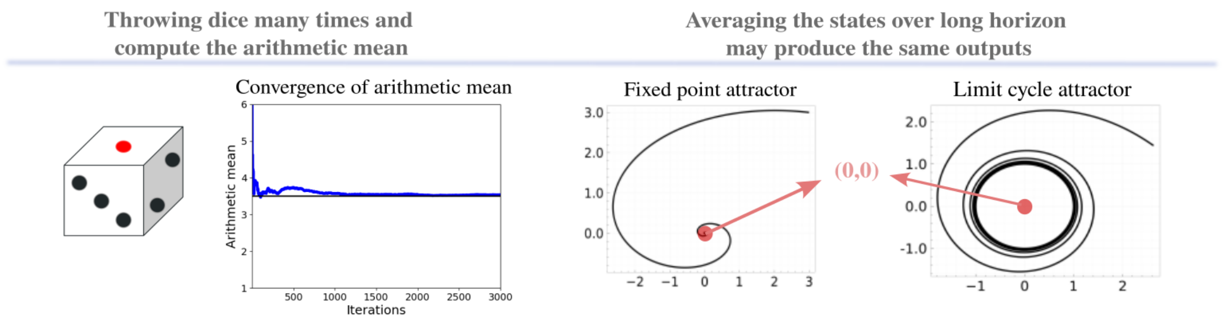


Figure 1.7: Illustration of loss of dynamic information through concentration of measure. The left picture shows an example of the law of large numbers showing the sample mean of the numbers given by throwing dice many times will converge towards the expected value, namely 3.5, as the number of throws increases. The right picture shows an intuitive understanding of how the concentration of measures or process of *averaging out* the observations may erase not only the noise effect but also the dynamic information; in this case, the dynamics on the left side is a fixed point attractor and that on the right side is a limit cycle attractor.

transformations, and even ML algorithms themselves, fundamentally manifest as dynamical systems, as they are inherently the Turing machines [248]. This fact naturally leads to an intriguing research question: How might we reframe the ML paradigm through the language of dynamical systems?

My initial foray in this direction focused on establishing links between statistical methodologies and dynamics, with a particular emphasis on a novel problem of dynamic structure estimation from bandit feedback contaminated by sub-Gaussian noise (see [144] for a survey of bandit algorithms). The process of estimating structural information under conditions of noise contamination traditionally relies on leveraging the concentration of measures (cf. [219]) to mitigate the noise's impact. Yet, this approach bears the risk of erasing essential structural information that underlies the dynamics (see Figure 1.7 for an intuitive understanding).

To surmount this challenge, Chapter 6 presents our work [189] that harnesses asymptotic results for exponential sums, including those associated with the Weyl sum [260], studied within the number theory community. These findings are instrumental in averting the era-

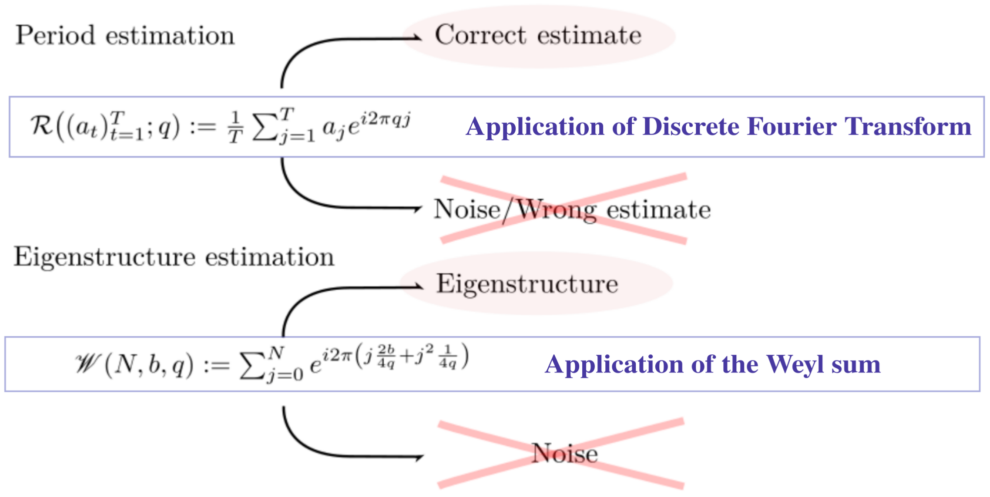

Period estimation
Correct estimate
$\mathcal{R}((a_t)_{t=1}^T; q) := \frac{1}{T}\sum_{j=1}^T a_j e^{i2\pi qj}$
Application of Discrete Fourier Transform
Noise/Wrong estimate
Eigenstructure estimation
Eigenstructure
$\mathscr{W}(N,b,q) := \sum_{j=0}^N e^{i2\pi\left(j\frac{2b}{4q}+j^2\frac{1}{4q}\right)}$
Application of the Weyl sum
Noise

Figure 1.8: Overview of how the exponential sum techniques are used in the work. For (nearly) period estimation problem, it is an application of discrete Fourier transform in our statistical settings, which ensures that the correct estimate remains while noise effect or wrong estimates are properly suppressed. When the system follows linear dynamics in addition to the (nearly) periodicity, our application of the Weyl sum, a variant of exponential sums, preserves some set of the eigenstructure information.

sure of crucial information. In particular, the observations are not just averaged but are processed by exponential sum techniques, which as a result produces an estimate that preserves an important set of correct dynamic structural information while effectively suppresses the noise's influence (see Figure 1.8). As a byproduct, the work also offers novel dynamical systems theoretic concepts, including *nearly periodicity*.

1.2.6 Constructive Approach for Analyzing Complex Machine Learning Algorithms as Dynamical Systems

A dynamical system perspective is not confined to theoretical studies alone but extends its reach into practical domains, particularly when deep neural networks are integrated into learning. Recent advancements of ML are highlighted by the use of large network models together with massive compute and data; and it is nearly impossible to thoroughly analyze

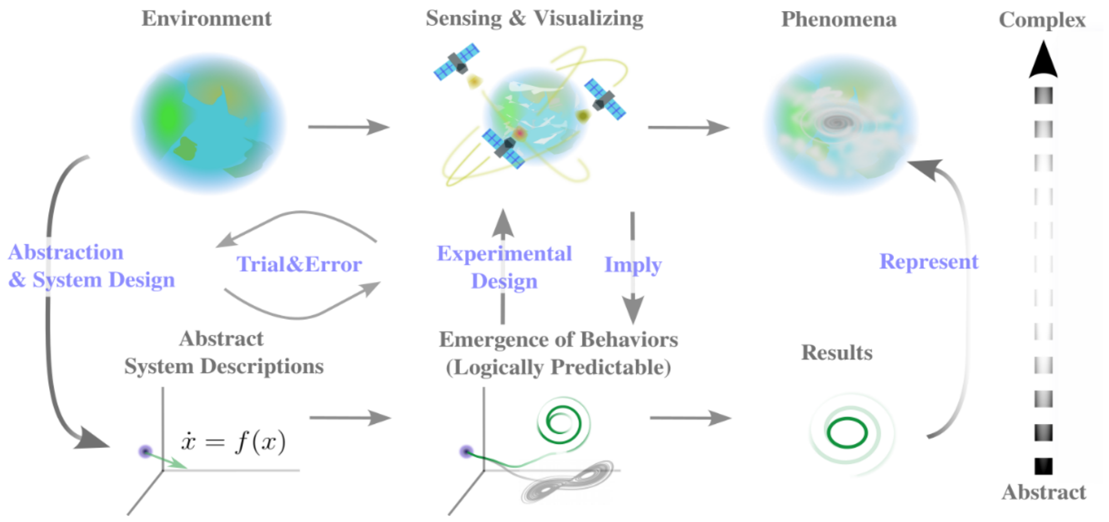


Figure 1.9: An illustration of the constructive approach. It is a bottom-up approach where one abstracts the target environment in a certain way so that the logically predictable and reproducible behaviors of the system and its resulting state help us infer what might be happening for the target complex system. The abstract system design and how one observes the target system in order to find a *reasonable* connection to the abstracted behaviors are determined through trial and error. The abstraction may have multiple layers.

the entire behaviors or phenomena observed in, for example, large language models or deep RL/imitation learning (IL) of large scale. In particular, with the existence of interactions of multiple agents including human users, the behavior of such systems emerges as exceedingly intricate, resisting facile reductionist approaches for precise description.

In Chapter 7, we aim at tackling this challenge by focusing on algorithmic behaviors generated by the soft-actor-critic (SAC) algorithm [94], a widely-used model-free deep RL algorithm, with three types of critic loss functions, namely, the mean-squared-error (MSE), Huber [104], and quantile-Huber [72] loss functions.

The novelty of this work lies in the methodology we employ, a constructive approach (see Figure 1.9) tailored to our problem. An illustrative example of constructive approach may be seen in the Lorenz attractor [153]; this simplified abstract system is not meant to fully capture the target dynamics of atmospheric convection and weather, but to show chaotic behavior (i.e., the sensitivity to the initial conditions) can even be observed in such a simple system.

Our approach uses some observations to categorize the learning behaviors, and designs a model that is sufficiently simple to analyze while involving *essential* components of the target systems, from which we can at least identify the sufficient conditions for some class of learning behaviors to emerge.

1.3 Summary of contributions

We list the contributions of thesis in this section which will be mentioned again in each chapter.

In Chapter 3, we mathematically conceptualize the novel control notion of limited-duration safety, present a way of learning valid control function that can be used to ensure limited-duration safety, and present practical applications including a constraint-driven control problem and a transfer learning problem where the developed control function is adopted to speed up the learning in the target tasks.

In Chapter 4, we design an algorithm (LC^3) that works in practice for complex (and continuous) robotics simulation environments under fairly general model assumptions, and provide regret bounds for the algorithm. Further, we propose the *Koopman spectrum cost* that complements the (cumulative) single-step cost for nonlinear control, which enables effective encoding/imitation of some desirable agent dynamics such as limit cycles, stable loops, and smooth movements, and propose theoretical online learning algorithm (KS-LC^3) to provide a regret bound under structural assumptions by using our novel operator theoretic arguments in addition to some of the results from the analysis of LC^3.

Chapter 5 presents a novel framework based on path signatures for control problems named signature control, and defines *Chen equation*, a dynamic programming based update rule of signatures, to show how it reduces to the Bellman equation as a special instance. Moreover, we propose its model predictive control algorithm that is shown to generalize the classical integral control, and present several control and robotics applications. Those applications showcase the benefits of our framework and demonstrate the concepts that inherit mathematical properties of the path signature. We also analyze some of the properties of our framework both numerically and theoretically.

In Chapter 6, a recoverable set of periodic/eigenvalues information of a (nearly) periodic dynamics when the observations are available in a form of bandit feedback is mathematically identified and defined (the feedback is contaminated by a sub-Gaussian noise, which is more general than those usually considered in system identification work). In particular, we present provably correct algorithms for efficiently estimating such information; this constitutes the first attempt of adopting asymptotic results on the Weyl sum, and the algorithms are implemented for toy examples to experimentally validate our claims.

Then, in Chapter 7, we take constructive approach to analyze the performance difference among RL algorithms with different critic loss functions; in particular, after presenting theoretical analyses that expose some of the basic logic behind the behavior difference caused by the choice of critic loss, we present *behavior class* to categorize the learning curves of algorithms with different critic loss functions. Subsequently, we propose abstracted systems whose learning behaviors fall into the proposed behavior classes and are logically predictable, and propose a few metrics that help quantify the behaviors observed in more complex environments. Finally, we also present an array of experimental results.

Chapter 2
BACKGROUND

In this chapter, we discuss background of the work presented in this thesis, including a succinct review of important mathematical concepts and related work.

2.1 Reviews of mathematical concepts

The work of this thesis rely on several mathematical frameworks; some of which will be detailed in each chapter. In this section, we therefore review selected ones below.

2.1.1 Existence and uniqueness of the solutions of differential equation

Consider the following initial-value problem:

$$\frac{dx}{dt} = f(t, x), \qquad x(t_0) = x_0 \in \mathbb{R}^{d_x}. \tag{2.1.1}$$

We discuss the solutions of (2.1.1) over a time interval $[t_0, t_1]$, i.e., a continuous function $x : [t_0, t_1] \to \mathbb{R}^{d_x}$ such that dx/dt is defined and $dx/dt = f(t, x(t))$ for all $t \in [t_0, t_1]$. In particular, we seek the conditions that ensure the solutions exist and that they are unique.

Theorem 2.1.1 (Local existence and uniqueness [126, Theorem 3.1])**.** *Suppose $f(t, x)$ is piecewise continuous in t and that the following Lipschitz condition holds:*

$$\forall x, y \in \left\{x \in \mathbb{R}^{d_x} : \|x - x_0\| \le r\right\}, \forall t \in [t_0, t_1], \exists L \ge 0 : \|f(t, x) - f(t, y)\|_{\mathbb{R}^{d_x}} \le L\|x - y\|_{\mathbb{R}^{d_x}},$$

where $r > 0$. Then, there exists some $\delta > 0$ such that (2.1.1) *admits a unique solution over the interval* $[t_0, t_0 + \delta]$.

The above theorem only ensures the *local* existence and uniqueness of the solutions under some conditions; below, we present the global existence and uniqueness result:

Theorem 2.1.2 (Global existence and uniqueness [126, Theorem 3.2])**.** *Suppose $f(t,x)$ is piecewise continuous in t and that the following Lipschitz condition holds:*

$$\forall x, y \in \mathbb{R}^{d_x}, \quad \forall t \in [t_0, t_1], \quad \exists L \geq 0 : \ \|f(t,x) - f(t,y)\|_{\mathbb{R}^{d_x}} \leq L\|x-y\|_{\mathbb{R}^{d_x}}.$$

Then, (2.1.1) *admits a unique solution over the interval* $[t_0, t_1]$.

2.1.2 Control barrier functions

We also briefly discuss the control barrier function for the system

$$\frac{dx}{dt} = f(x(t)) + g(x(t))u(t), \tag{2.1.2}$$

where $x(t) \in \mathcal{D} \subset \mathbb{R}^{d_x}$ and $u(t) \in \mathcal{U} \subset \mathbb{R}^{d_u}$ are the state and the instantaneous control input, $f : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, and $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x \times d_u}$. We define extended class-$\mathcal{K}$ functions and forward invariance:

Definition 2.1.1 (Extended class-$\mathcal{K}$ function [126, 270, 16])**.** A continuous function $\alpha : \mathbb{R} \to \mathbb{R}$ is said to belong to extended class-$\mathcal{K}_\infty$ if it is strictly increasing and $\alpha(0) = 0$.

Definition 2.1.2 (Forward invariance [126])**.** The set $\mathcal{C}$ is forward invariant with respect to the system (2.1.2) if

$$x(0) \in \mathcal{C} \implies x(t) \in \mathcal{C}, \ \forall t \geq 0.$$

Then, the control barrier functions are defined as below:

Definition 2.1.3 (Control barrier function [16, Definition 2])**.** Let $\mathcal{C} \subset \mathcal{D}$ be the superlevel set of a function $B : \mathcal{D} \to \mathbb{R}$ of class $C^1(\mathcal{D})$; then B is said to be a control barrier function if there exists a function α of extended class-$\mathcal{K}_\infty$ such that for the system (2.1.2), it holds that

$$\forall x \in \mathcal{D} : \ \sup_{u \in \mathcal{U}} \{L_f B(x) + L_g B(x) u\} \geq -\alpha(B(x)).$$

Now, the following theorem combined with the existence and uniqueness results in Section 2.1.1 guarantees the *safety*, i.e., the forward invariance of set $\mathcal{C}$:

Theorem 2.1.3 (Safety guarantee by control barrier functions [16, Theorem 2])**.** *Let $\mathcal{C} \subset \mathbb{R}^{d_x}$ be the superlevel set of a function $B : \mathcal{D} \to \mathbb{R}$ of class $C^1(\mathcal{D})$. If B is a control barrier function on $\mathcal{D}$ and $\frac{\partial B}{\partial x}(x) \neq 0$ for all $x \in \partial\mathcal{C}$, then any Lipschitz continuous controller $u(x)$ satisfying, for all $x \in \mathcal{D}$,*

$$L_f B(x) + L_g B(x) u(x) + \alpha(B(x)) \geq 0$$

for the system (2.1.2) *renders the set $\mathcal{C}$ forward invariant.*

2.1.3 Markov decision process

In this thesis, we sometimes rely on the convention of continuous control setups without explicitly defining Markov Decision Process (MDP); however, as otherwise stated, an MDP for discrete-time finite horizon case may be defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \times \Omega \to \mathbb{R}$ is the reward function (Ω is the sample space), $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, and $\gamma \in [0, 1)$ is the discount factor, respectively. For simplicity, by abuse of notation, we see $R(s, s', a)$ returns a random variable (reward). Also, we define $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ as the policy.

Let $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be the action-value (or Q) function w.r.t. the policy π, defined by

$$Q^\pi(s, a) := \mathbb{E} Z^\pi(s, a) := \mathbb{E}\left[\sum_{h=0}^{H'-1} \gamma^h r_h \,\middle|\, \pi, s_0 = s, a_0 = a\right], \quad r_h = R(s_h, s_{h+1}, a_h, \omega),$$

where $H' \in \mathbb{Z}_{>0}$ is given by

$$H' := \min\left\{H, \min_{h \in \mathbb{Z}_{>0}} \left\{h \middle| s_h \in \mathcal{S}_e, s_0 = s, a_0 = a, \pi\right\}\right\}.$$

Here, $H \in \mathbb{Z}_{>0}$ is the maximum time steps, and $\mathcal{S}_e \subset \mathcal{S}$ denotes the set of exit states.

Distributional reinforcement learning

Although we consider the Q function described above in the classical reinforcement learning, it may be sometimes beneficial to study the distribution of the *return* $Z^\pi(s, a)$ through *distributional Bellman equation* [34]. For such cases, one keeps the distribution of return at each pair of state and action while the policy may be updated by using its expected value (i.e., the Q value). A reasonable approach is to use quantile regression; recall that, for a (real-valued) random variable X, the value $x \in \mathbb{R}$ falls in the i/N-quantile for $i \in \{1, 2, \dots, N\}$ if

$$\Pr[X \leq x] \leq \frac{i}{N}.$$

For quantile regression, we use the *quantile midpoints* $\tau_i = \frac{2i-1}{2N}$. Quantile regression minimizes the following loss function for quantile τ:

$$\rho_\tau(u) = u\left(\tau - \delta_{\{u<0\}}\right),$$

where $\delta_{\{\texttt{cond}\}}$ is the indicator function returning 1 when `cond` is satisfied and 0 otherwise; and we know that this optimization gives us x that falls in the quantile τ (see [134] for more details).

2.1.4 Random dynamical systems

Let $\mathcal{X} \subset \mathbb{R}^{d_x}$ be the state space, and Π a set of parameters each of which corresponds to one random dynamical system (RDS) as described below. Given a parameter $\pi \in \Pi$, let (Ω_π, P_π) be a probability space, where Ω_π is a measurable space and P_π is a probability measure. Let $\{\mu_\pi(r)\}_{r\in\mathbb{T}}$ be a semi-group of measure preserving measurable maps, where $\mu_\pi(r) : \Omega_\pi \to \Omega_\pi$, $(\mu_\pi(r))_* P_\pi = P_\pi$, $\mu_\pi(0) = \mathrm{id}_{\Omega_\pi}$, and $\mu_\pi(r) \circ \mu_\pi(s) = \mu_\pi(r + s)$ for all $s, r \in \mathbb{T}$. For each parameter $\pi \in \Pi$, the corresponding nonlinear RDS is given by

$$\mathcal{F}^\pi : \mathbb{T} \times \Omega_\pi \times \mathcal{X} \to \mathcal{X},$$

that satisfies

$$\mathcal{F}^\pi(0, \omega, x) = x, \quad \mathcal{F}^\pi(r + s, \omega, x) = \mathcal{F}^\pi(r, \mu_\pi(s)\omega, \mathcal{F}^\pi(s, \omega, x)), \quad \forall r, s \in \mathbb{T},\ \omega \in \Omega_\pi,\ x \in \mathcal{X}. \tag{2.1.3}$$

The above definition of RDS is standard in the community studying dynamical systems (refer to [24], for example). Roughly speaking, an RDS consists of the following two models:

- A model of the *noise*;
- A function representing the *physical* dynamics of the system.

RDSs subsume many practical systems including Markov chains, solutions to stochastic differential equations, and additive-noise systems, i.e.,

$$x_{h+1} = f(x_h) + \eta_h, \quad x_0 \in \mathbb{R}^d, \quad h \in [H],$$

where $f : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ represents the dynamics, and $\eta_h \in \mathbb{R}^{d_x}$ is the zero-mean i.i.d. additive noise vector. Intuitively, dynamical systems with an *invariant noise-generating mechanism* could be described as RDSs by an appropriate translation.

2.1.5 Koopman operator

For the RDSs being defined above, we define the operator-valued map $\mathscr{K}$ below, using the dynamical system model.

Definition 2.1.4 (Koopman operator)**.** Let $\mathcal{H}$ be a function space on $\mathcal{X}$ over $\mathbb{C}$ and let $\{\mathcal{F}^\pi\}_{\pi\in\Pi}$ be a dynamical system model. We define an operator-valued map $\mathscr{K}$ by $\mathscr{K} : \Pi \to \mathcal{L}(\mathcal{H}, \mathcal{H})$ such that for any $\pi \in \Pi$ and $g \in \mathcal{H}$,

$$[\mathscr{K}(\pi)g](x) := \mathbb{E}_{\Omega_\pi}[g \circ \mathcal{F}^\pi(1, \omega, x)], \quad x \in \mathcal{X}.$$

We will choose a suitable $\mathcal{H}$ to define the map $\mathscr{K}$, and $\mathscr{K}(\pi)$ is the Koopman operator for $\mathcal{F}^\pi$. Essentially, the Koopman operator represents a nonlinear dynamics as a linear (infinite dimensional) operator that describes the evolution of *observables* in a lifted space (see Figure 2.1 (left) for an illustration).

2.1.6 Reproducing kernel Hilbert spaces

Here, we review the reproducing kernel Hilbert spaces (RKHSs).

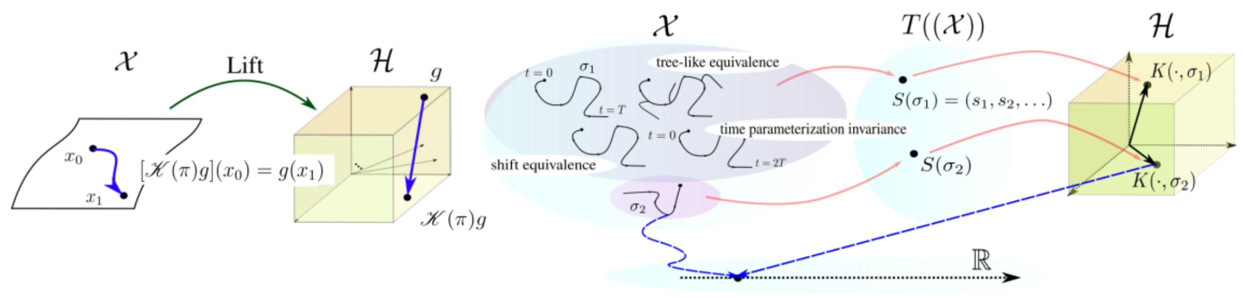


Figure 2.1: Illustrations of the Koopman operator (left) and the path signatures (right).

Definition of the reproducing kernel

We give the definition of the reproducing kernel:

Definition 2.1.5 (Reproducing kernel [39, 25]). Let E be an nonempty set. A function

$$K : E \times E \to \mathbb{C}$$
$$(s, t) \mapsto K(s, t)$$

is a reproducing kernel of the Hilbert space $\mathcal{H}$ if and only if

1. $\forall t \in E : \; K(\cdot, t) \in \mathcal{H}$,

2. $\forall t \in E, \quad \forall \varphi \in \mathcal{H} : \; \langle \varphi, K(\cdot, t) \rangle_{\mathcal{H}} = \varphi(t)$ (a.k.a. reproducing property).

The following theorem gives the first characterization of RKHS.

Theorem 2.1.4 (Characterization of RKHS [39, Theorem 1]). *A Hilbert space $\mathcal{H}$ of functions on E over $\mathbb{C}$ has a reproducing kernel if and only if all the evaluation functionals $e_t : \mathcal{H} \to \mathbb{C}$ $(\varphi \mapsto e_t(\varphi) = \varphi(t))$, for all $t \in E$, are continuous on $\mathcal{H}$.*

RKHSs have several remarkable properties: (1) uniqueness of the reproducing kernel if it exists, (2) any reproducing kernel is a positive type function (converse is also true), (3) all closed linear subspaces of RKHSs are again RKHSs, (4) restriction of an RKHS of functions on E to $E_1 \subset E$ admits another RKHS with an adequate norm, (5) sum and product of reproducing kernels become the reproducing kernels of properly defined Hilbert spaces, just to name a few (see [39, 25]).

Function-valued RKHS

Function-valued RKHSs are defined below.

Definition 2.1.6 ([116]). A Hilbert space $(\mathcal{H}_K, \langle \cdot, \cdot \rangle_{\mathcal{H}_K})$ of functions from Π to a Hilbert space $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is called a reproducing kernel Hilbert space if there is a nonnegative $\mathcal{L}(\mathcal{H}; \mathcal{H})$-valued kernel K on $\Pi \times \Pi$ such that:

1. $\pi \mapsto K(\pi', \pi)\phi$ belongs to $\mathcal{H}_K$ for all $\pi' \in \Pi$ and $\phi \in \mathcal{H}$,

2. for every $\mathscr{G} \in \mathcal{H}_K,\ \pi \in \Pi$ and $\phi \in \mathcal{H}$, $\langle \mathscr{G}, K(\pi, \cdot)\phi \rangle_{\mathcal{H}_K} = \langle \mathscr{G}(\pi), \phi \rangle_{\mathcal{H}}$.

For function-valued RKHSs, the following proposition holds.

Proposition 2.1.5 (Feature map [45]). *Let $\mathcal{H}'$ be a Hilbert space and $\Psi : \Pi \to \mathcal{L}(\mathcal{H}; \mathcal{H}')$. Then the operator $W : \mathcal{H}' \to \mathcal{H}^{\Pi}$ defined by $[W\psi](\pi) := \Psi(\pi)^{\dagger}\psi,\ \forall \psi \in \mathcal{H}'$ and $\forall \pi \in \Pi$, is a partial isometry from $\mathcal{H}'$ onto the reproducing kernel Hilbert space $\mathcal{H}_K$ with a reproducing kernel $K(\pi_2, \pi_1) = \Psi(\pi_2)^{\dagger}\Psi(\pi_1),\ \ \forall \pi_1, \pi_2 \in \Pi$.*

2.1.7 Path signature

Let $\mathcal{X} \subset \mathbb{R}^{d_x}$ be a state space and suppose a path (a continuous stream of states) is defined over a compact time interval $[s, t]$ for $0 \leq s < t$ as an element of $\mathcal{X}^{[s,t]}$. The path signature is a collection of infinitely many features (scalar coefficients) of a path with *depth* one to infinite. Coefficients of each depth roughly correspond to the geometric characteristics of paths, e.g., displacement and area surrounded by the path can be expressed by coefficients of depths one and two.

Tensor algebra

We present the definition of tensor algebra here. In the main text, we use some of the notations, including $T((\mathcal{X}))$, defined below.

Definition 2.1.7 (Tensor algebra)**.** Let $\mathcal{X}$ be a Banach space. The space of formal power series over $\mathcal{X}$ is defined by

$$T((\mathcal{X})) := \prod_{k=0}^{\infty} \mathcal{X}^{\otimes k},$$

where $\mathcal{X}^{\otimes k}$ is the tensor product of k vector spaces ($\mathcal{X}$s). For $A = (a_0, a_1, \ldots)$, $B = (b_0, b_1, \ldots) \in T((\mathcal{X}))$, the addition $+$ and multiplication $\otimes$ are defined by

$$A + B = (a_0 + b_0, a_1 + b_1, \ldots),\ A \otimes B = (c_0, c_1, \ldots)\ ,\ c_k = \sum_{\ell=0}^{k} a_\ell \otimes b_{k-\ell}.$$

Also, $\lambda A = (\lambda a_0, \lambda a_1, \ldots)$ for any $\lambda \in \mathbb{R}$. The truncated tensor algebra for a positive integer m is defined by the quotient $T^m(\mathcal{X})$

$$T^m(\mathcal{X}) := T((\mathcal{X}))/T_m,$$

where

$$T_m = \{A = (a_0, a_1, \ldots) \in T((\mathcal{X})) : a_0 = a_1 = \ldots = a_m = 0\}.$$

Now, we give the definition of path signatures.

Definition of signature

The formal definition of path signatures is given below.

Definition 2.1.8 (Path signatures [163])**.** Let $\Sigma \subset \mathcal{X}^{[0,T]}$ be a certain space of paths. Define a map on Σ over $T((\mathcal{X}))$ by $S(\sigma) \in T((\mathcal{X}))$ for a path $\sigma \in \Sigma$ where its coefficient corresponding to the basis $e_i \otimes e_j \otimes \ldots \otimes e_k$ is given by

$$S(\sigma)_{i,j,\ldots,k} := \int_{0<\tau_k<T} \cdots \int_{0<\tau_j<\tau_k} \int_{0<\tau_i<\tau_j} dx_{\tau_i,i} dx_{\tau_j,j} \ldots dx_{\tau_k,k}.$$

Here, $x_{t,i}$ is the ith coordinate of σ at time t.

The space Σ is chosen so that the path signature $S(\sigma)$ of $\sigma \in \Sigma$ is well-defined. Given a positive integer m, the truncated signature $S_m(\sigma)$ is defined accordingly by a truncation of $S(\sigma)$ (as an element of the quotient $T^m(\mathcal{X})$).

We summarize the basic properties of path signatures that are exploited in this thesis below (see [63, 163] as well):

- The signature of a path is invariant under a constant shift and time reparameterizations. Straightforward applications of signatures thus represent *shape* information of a path irrespective of waypoints and/or absolute initial positions.

- A path is uniquely recovered from its signature up to tree-like equivalence (e.g., path with *detours*) and the magnitudes of coefficients decay as depth increases. As such, (truncated) path signatures contain sufficiently rich information about the state trajectory, providing a valuable and compact representation of a path in several control problems.

- Any real-valued continuous map on the certain space of paths can be approximated to arbitrary accuracy by a linear map on the space of signatures. This universality property enables us to construct a kernel operating over the space of trajectories, which will be critical to derive our control framework in Chapter 5.

- The path signature has a useful algebraic property known as Chen's identity [57], stating that the signature of the concatenation of paths can be computed by the tensor product of the signatures of the paths. Let $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$ be two paths. Then,

$$S(X * Y)_{a,c} = S(X)_{a,b} \otimes S(Y)_{b,c},$$

  where $*$ denotes the concatenation operation (we shift the starting time of path when necessary).

Signature kernel

We will use signature kernels for computing the metric (or cost) for control problems in Chapter 5. A path signature is a collection of infinitely many real values, and in general, the computations of inner product of a pair of signatures in the space of formal polynomials are intractable. Although it is still not the exact computation in practice, the work [214] showed that the *signature kernel* can be obtained as the solution of a Goursat PDE.

Definition 2.1.9 (Signature kernel [214])**.** Let $\mathcal{X}$ be a d-dimensional space with canonical basis $\{e_1, \dots, e_d\}$ equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{X}}$. Let $T(\mathcal{X}) := \bigoplus_{k=0}^{\infty} \mathcal{X}^{\otimes k}$ be the space of formal polynomials endowed with the same operators $+$ and $\otimes$ as $T((\mathcal{X}))$, and with the inner product

$$\langle A, B \rangle := \sum_{k=0}^{\infty} \langle a_k, b_k \rangle_{\mathcal{X}^{\otimes k}} ,$$

where $\langle \cdot, \cdot \rangle_{\mathcal{X}^{\otimes k}}$ is defined on basis elements $\{e_{i_1} \otimes \dots \otimes e_{i_k} : (i_1, \dots, i_k) \in \{1, \dots, d\}^k\}$ as

$$\langle e_{i_1} \otimes \dots \otimes e_{i_k}, e_{j_1} \otimes \dots \otimes e_{j_k} \rangle_{\mathcal{X}^{\otimes k}} = \langle e_{i_1}, e_{j_1} \rangle_{\mathcal{X}} \cdots \langle e_{i_k}, e_{j_k} \rangle_{\mathcal{X}} .$$

Let $\overline{T(\mathcal{X})}$ be the completion of $T(\mathcal{X})$, and $\mathcal{H} := (\overline{T(\mathcal{X})}, \langle \cdot, \cdot \rangle)$ is a Hilbert space. The signature kernel $K : \Sigma \times \Sigma \to \mathbb{R}$ is defined by

$$K(X, Y) := \langle S(X), S(Y) \rangle ,$$

for X and Y such that $S(X), S(Y) \in \overline{T(\mathcal{X})}$.

2.1.8 Concentration inequality

Lastly, we briefly discuss concentration inequalities. As a basic statistical fact, we introduce the strong law of large numbers. To this end, recall that the sequence of (real-valued) random variables $\{X_n\}$ is called *pairwise independent* if the joint cumulative distribution function F_{X_i,X_j} satisfies $F_{X_i,X_j}(x, y) = F_{X_i}(x)F_{X_j}(y)$ for any pair (i, j) such that $i \neq j$ and for all x, y. Also, it is called *identically distributed* if $F_{X_i}(x) = F_{X_j}(x)$ for all i, j, x.

Theorem 2.1.6 (Strong law of large numbers (cf. [81])). *Let $\{X_n\}$ be a sequence of pairwise independent, identically distributed random variables with $\mathbb{E}[|X_i|] < \infty$. Let $S_n = \sum_{i=1}^n X_n$. Then,*

$$\lim_{n\to\infty} \frac{S_n}{n} = \mathbb{E}[X_i] \quad \text{a.s.}$$

Since the law of large numbers is an asymptotic result, it does not provide a bound of the empirically estimated value for given samples of finite size. Concentration inequalities are used to quantify this error bound; we only provide the most basic inequality called Markov's inequality here: Let X be a nonnegative random variable; then we have

$$(\text{Markov's inequality}) \ \ \forall a \geq 0: \ \Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

Refer to [219] for concentration inequalities and for how they are exploited to provide sample complexity guarantees in ML problems.

2.2 Related work

Studies in the intersection of machine learning (ML) and dynamical systems span a variety of fields including robotics, control, and reinforcement learning (RL), just to name a few.

This section reviews relevant work to this thesis in such fields of studies, namely, RL, control theoretic guarantees for learning, operator theoretical and trajectory based study on learning dynamics, bandit for nonstationary environments, exponential sums, and dynamical system perspectives on ML.

2.2.1 Reinforcement learning

RL is a learning paradigm where the learner acts on an environment to receive observations and rewards (or costs) over a horizon of certain length, which is aimed at eventually finding an optimal sequence of actions that maximizes (or minimizes) the cumulative rewards (or costs). Historically, RL has served as a bridge between ML and controls (see [233] for the history of RL). RL research has traditionally been conducted within optimal control (based on dynamic programming [35]) and psychology of animal learning (e.g., [224]), and their interdisciplinary study has produced more quantitative and algorithmic research represented by the invention of temporal-difference (TD) learning (cf. [235]). Recently, we have seen success of (deep) RL methods categorized as *model-based* or *model-free* on a variety of domains in addition to the extensive studies on the complexities of RL algorithms.

Model-free and model-based algorithms: Model-free approach typically relies on dynamic programming over the *value* which is to be learned through interactions, and value function based methods are widely adopted in RL for control problems. However model-free RL algorithms [112, 94, 175, 266] treat the state transition model as a black box and access the state information exclusively through the scalar value, typically resulting in low sample efficiency. Model-based RL methods alleviate this problem of value function based approaches by using or learning the environment model of transition across the states to plan an optimal sequence of actions (cf. [231, 79, 258, 64]). In practical algorithms, however, even a small amount of error on one-step dynamics could diverge along multiple time steps, which hinders the performance (cf. [176]). To improve sample complexity and generalizability of value function based methods, on the other hand, successor features (e.g., [31]) have been developed for encoding representations of values for a diverse set of reward signals spanned by the features. We mention the work [231] shows one way of defining what qualifies for the model-based algorithm.

Sample complexity studies: Statistical complexity such as VC-theory plays a key role in the sample complexity analysis of ML algorithms (cf. [219]) and it is oftentimes studied within the framework of probably approximately correct (PAC) learnability [251]. Provably efficient algorithms for RL have been studied in, for example, [193, 29, 155, 113, 8]. Efficient learner for the special case of linear quadratic regulators (LQRs) has been studied as well [166, 65, 223]. As general frameworks permitting sample efficient RL under *structural conditions*, some low rank structures have been proposed (e.g., the Bellman rank [112] and the Witness rank [231]) which are shown to be the special cases of Bilinear class [79]. Although it is not directly related to RL, there exist researches on sample complexity of system identification problems as well. As those related to our work, partially observed linear system identification methods include [247, 246, 142, 145, 97, 222, 146], under either controlled or autonomous settings. Most of them consider additive Gaussian noise and make controllability and/or observability assumptions (for autonomous case, transition with Gaussian noise with positive definite covariance is required). While [173] considers nonlinear observation, it assumes Gaussian noise and controllability.

Distributional RL: Distribution of returns within the context of RL has been studied in [75, 180, 30, 143] for example. Followed by the work [34] showing improvements made by a distributional RL over an expected RL for several environments, a variant of distributional RL called quantile regression soft actor-critic (QRSAC) algorithm was proposed for training of the superhuman racing AI agent, Gran Turismo Sophy [266]. Theoretical analyses have also been made (e.g., [237, 210, 158]); in particular, [158] proves that, with the squared Cramér distance loss, distributional RL behaves exactly the same as expected RL for the tabular and linear approximation settings, while the difference is pronounced in the nonlinear approximation settings.

Analysis of deep RL algorithms: Analyzing the performance of deep RL algorithms is an intricate task due to its complexity. There exist some attempts on analyzing the effects of critic loss choice and of smoothness of the critic surface. The use of the Huber loss [104] for critic is often preferred in practice, and its equivalence, namely, magnitude clipping on TD errors has shown better learning stability on some problem instances [175]; on the other hand, [55] shows that the Huber loss combined with RMSProp optimizer leads to inferior performance than the MSE loss with Adam optimizer [129] for a wide variety of environments. Recently, the authors of the work [197] analyzed the convergence and performance of algorithms with the Huber loss and the MSE loss w.r.t. the Bellman errors. They claim that capacity constraints on approximators lead to distinct critic surfaces. Also, [78] proposes the use of loss on the *gradient* of critic surface to facilitate its smoothness. Importance of well-behaving critic surface has been studied in generative modeling, representation learning and RL as discussed in [209]; which mentions that imposing smoothness constraints on critic plays some important roles in generative adversarial networks (GANs) ([272, 92, 82, 21]), for example.

2.2.2 Control theoretic guarantees for learning

Giving control theoretic guarantees to learning systems has been recently studied as an attempt of exploiting the best of both worlds. While statistical ML places importance on learnability, including computational and sample efficiency, control theory possesses an accumulation of studies on (asymptotic) stability, controllability, observability, invariance, and adaptation.

Control Lyapunov and barrier functions: In particular, Lyapunov stability is an essential control theoretic concept (cf. [126]). For controlled (affine) systems, constraining the control inputs using control Lyapunov functions (CLFs; cf. [86]) ensures stability of the system efficiently. Stability naturally implies (forward) invariance of certain set of states; however, it is typically too conservative just to ensure invariance. Recently, control barrier functions (CBFs; cf. [16, 270, 261, 90, 255, 17, 10]) have been proposed to enforce the (forward) invariance of a given set of states.

Applications to safe learning: Safety is particularly pronounced in the domains such as robotics, and the learning agent should be tasked to learn a desirable action strategy without violating safety. There have been numerous solutions proposed [216, 12, 220, 18, 184, 4], and Lyapunov stability is applied to safe learning of robotics (see one of the early work [38, 37]), and later, barrier functions are employed in, for example, [256, 192, 157]. Lyapunov stability may be utilized for goal-reaching behaviors while barrier functions guarantee safe maneuver as well (e.g., [114]). Barrier functions may be learned from expert demonstrations (e.g., [206]). See, for example [74] for a survey of safe control with learned certificates.

Applications to neural network training: These control tools have recently been applied to ML problems as well; for example, Lyapunov stability is enforced when learning neural network [135, 207, 121]. Also the work [267] proposes the use of CBFs in learning NODEs; they propose the notion of invariant propagation and present an approach of forcing NODEs to satisfy output specifications.

2.2.3 Operator theoretical and trajectory based study on learning dynamics

The evolution of the system is sometimes more effectively described by operators (e.g., a matrix for linear systems), and learning from sequential data such as system trajectories may require optimizing directly over the space of trajectories.

Eigenstructure and pole assignment Classically, for linear systems where the system evolution can be described by a matrix, the system behaviors such as stability or oscillation have been extensively studied from linear algebra and spectral theoretical perspectives [126]. In particular, eigenstructure or pole assignment (cf. [19]) is aimed at designing the feedback controller so that the resulting system dynamics has desirable eigenstructure. The purpose of this control technique is distinct from that of optimal controls and hence is one of the unique problems considered in the control community.

Linearly solvable MDPs: Linearization largely simplifies the problem; the work [242, 243] introduced a class of MDPs that enables closed-form computation of optimal policy with respect to the optimal value function and that makes Bellman equation linear and describable by a linear operator. As such, the class of optimal control problems is reduced to a linear eigenvalue problem for this operator.

Koopman operator: Moreover, by lifting the state space into a space of *observables*, a nonlinear system over the state space is represented by the *linear* operator in the lifted space. This idea of Koopman operator was first introduced in [136]; and, during the last two decades, it has gained traction, leading to the developments of theory and algorithm (e.g., [70, 124, 167, 108, 109, 50, 132]) partially due to the surge of interest in data-driven approaches. The analysis of nonlinear dynamical system with Koopman operator has been applied to control (e.g., [137, 169, 118, 149, 138]) to design model predictive control (MPC) and LQR although nonlinear controlled systems in general cannot be transformed to LQR problems even by lifting to a feature space. For unknown systems, active learning of Koopman operator has been proposed [3].

Transformer: On the other hand, it is sometimes more efficient to solve a learning task with sequential data over the space of trajectories as it enables learning the relations among each point in a trajectory through *attention* and through compact representations. As an representative example of neural network architectures dealing with time sequence data, the Transformer model was proposed in [252] where the sequential data are processed in *nonsequential* manner by *self-attention* and by *positional embeddings*. This is conceptually different from the classical recurrent neural network (RNN) models (e.g., [101]) that process data one by one in a sequential manner.

Path signature: Having an efficient (and compact) representation of trajectories or paths helps learning; in particular, path signatures are mathematical tools developed in rough path research [162, 57, 43]. For efficient computations of metrics over signatures, kernel methods [102, 25] are employed [130, 214, 53, 214]. Signatures have been applied to various applications, such as sound compression [164], time series data analysis and regression [93, 161, 147], action and text recognition [271, 268], and neural rough differential equations [181]. Also, deep signature transform is proposed in [127] with applications to RL but is still within the Bellman equation based paradigm. Theory and practice of path signatures in ML are summarized in [63, 83].

2.2.4 Bandit for nonstationary environments

Another related example of research that falls into the intersection of ML and dynamics is the bandit problems for nonstationary environments, represented by the adversarial bandit problems (e.g., [48, 96]). Recently, some studies on nonstationary rewards have been made (cf. [28, 40, 60, 62, 156, 211, 245, 265]) although they do not deal with periodically behaved dynamical system properly (see discussions in [52] as well). For discrete action settings, [186] proposed the periodic bandit, which aims at minimizing the total regret. Also, if the period is known, Gaussian process bandit for periodic reward functions was proposed [52] under Gaussian noise assumption. Refer to [144] for bandit algorithms that are not covered here, and see, for example, [2, 27, 73, 1], for the work on stochastic linear bandit.

2.2.5 Exponential sums

For reconstruction of dynamical system information, we adopt exponential sums in this thesis. Exponential sums, also known as trigonometric sums, have developed as a significant area of study in number theory, employing various methods from analytical and algebraic number theory (see [23] for an overview). An exponential sum consists of a finite sum of complex numbers, each with an absolute value of one, and its absolute value can trivially be bounded by the number of terms in the sum. However, due to the cancellation among terms, nontrivial upper and lower bounds can sometimes be established. Several classes of exponential sums with such nontrivial bounds are known. In the mathematical community, these bounds are valuable not only in themselves but also for applications in fields like analysis within mathematics [123].

2.2.6 Dynamical system perspectives on machine learning

As ML algorithms or in general Turing machines [248] fundamentally manifest as dynamical systems, where the *symbols* are updated over iterations or time, it is insightful to view learning and prediction as dynamics.

Neural differential equations: One of the most notable inventions that has brought a dynamical system view point to the modern ML is the work of neural differential equations [58]. While dynamical system perspectives of neural network have been studied in the past (e.g., RNN and Hopfield network [103]), the fresh perspective of NODEs combined with an efficient use of ODE solver has provoked extensive research efforts in this field. There are variants of NODEs, namely, neural stochastic differential equations (e.g., [151, 250]) and neural controlled differential equations (e.g., [181, 128]), and a hybrid system extension has also been proposed (e.g., [111]). On the other hand, the expressivity of neural network has been studied through the lens of dynamics and has shown to be maximized when the dynamics of network is on the *edge of chaos* (e.g., [200]).

Besides, while we do not delve into the details, the work [225] introduced diffusion model inspired by statistical physics as a generative model; and the work [100] further progressed the research in this direction.

Learning dynamics: Since the training process described by an evolution of parameters can be viewed as a dynamical system, analyzing the *learning dynamics* oftentimes clarifies the mechanism behind training (e.g., [215, 5]). The work [201] provides a nonasymptotic analysis of stochastic gradient Langevin dynamics (which can be viewed as a discretization of Langevin diffusion process) for non-convex learning problems. Also, predicting the learning dynamics could potentially speed up the later stage of learning (e.g., [77]).

In the context of RL, the temporal-difference algorithm is analyzed as a flow of learning dynamics which gives a new insight into the behaviors of value function evolution [160, 159].

Chapter 3
CONTROL THEORETIC GUARANTEES FOR MACHINE LEARNING ALGORITHM

Chapter's key takeaways

With a policy that is not stabilizing the controlled agent on the zero cost (or reward) state may not produce a bounded value function, and we need a discount factor to remedy this. This introduction of discount factor makes the value function invalid as a CBF, necessitating a novel control concepts of *limited-duration safety* that can be ensured by *limited-duration CBFs*. For some tasks, it is shown that the existence of limited-duration CBFs is sufficient to ensure successful long-duration autonomy. Also, by reducing the policy space effectively through the use of limited-duration CBFs obtained by learning the source tasks, it becomes faster to learn the target tasks that share the same state constraints as the source.

3.1 Introduction

When deploying autonomous agents in unstructured environments over sustained periods of time, adaptability and robustness oftentimes outweigh optimality as a primary consideration. In other words, safety and survivability constraints play a key role. In this chapter, *safety* stands for the state constraint satisfaction over infinite time horizon.

To avoid the computational burden of fully solving constrained optimization problems for control sequences over a sufficiently long horizon, which potentially results in a suboptimal

This chapter largely consists of the excerpts from the article "Constraint learning for control tasks with limited duration barrier functions, Automatica, Volume 127, 2021, Ohnishi et al., Copyright Elsevier. DOI: https://doi.org/10.1016/j.automatica.2021.109504" [191] with permissions from Elsevier.

solution violating safety, we consider employing the recently developed control tool called control barrier function (CBF; see Chapter 2). CBFs are used to ensure safety by only constraining the instantaneous control at every time step (see Figure 3.1).

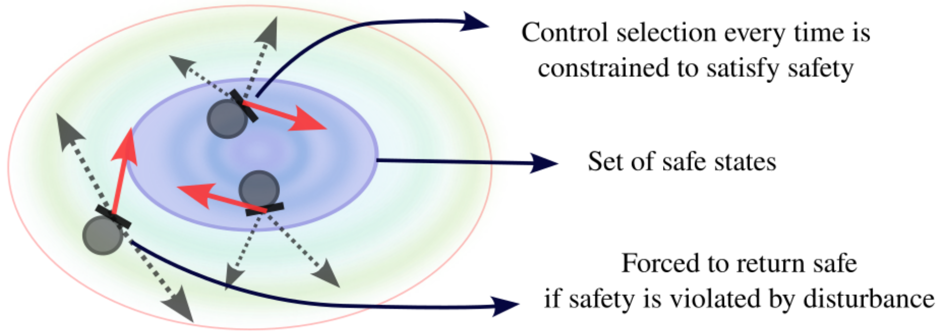

Control selection every time is constrained to satisfy safety
Set of safe states
Forced to return safe if safety is violated by disturbance

Figure 3.1: Safety is ensured by constraining the control selection at every time step using a control barrier function for the given set of safe states.

However, since control policies that keep a dynamical agent within state constraints over an infinite horizon are not always available, we relax safety to state constraint satisfaction over some finite time horizon $T > 0$, which we refer to as limited-duration safety (see Figure 3.2). Consequently, value function learning employed in RL can be used to find limited-duration safe policies, and this chapter presents a novel constraint-learning framework for control tasks and discusses a control theoretic guarantee for ML algorithm for robotics problems from different angles than the existing approaches, within a context of transfer learning.

Contributions: The contributions of this chapter are three folds: (1) conceptualize the novel control notion of limited-duration safety mathematically, (2) present a way of learning valid control function that can be used to ensure limited-duration safety, (3) and present applications to a constraint-driven control problem and to a transfer learning problem where the developed control function is adopted to speed up the learning in the target tasks.

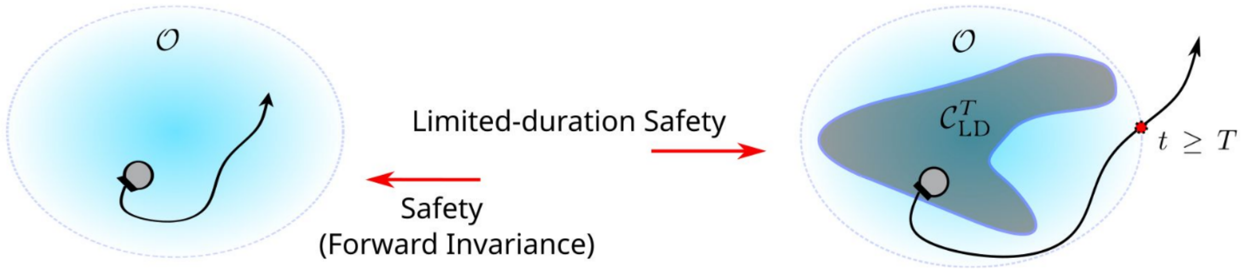


Figure 3.2: An illustration of relaxation of safety (forward invariance) to limited-duration safety. Limited-duration safety refers to the property where an agent stays in $\mathcal{O}$ for all $0 \leq t < T$ whenever starting from inside the set $\mathcal{C}_{\mathrm{LD}}^{T} \subset \mathcal{O}$.

3.2 *Problem setups*

In this chapter, we consider an agent with system dynamics described by an ordinary differential equation:

$$\frac{dx}{dt} = f(x(t)) + g(x(t))u(t), \tag{3.2.1}$$

where $x(t) \in \mathbb{R}^{d_x}$ and $u(t) \in \mathcal{U} \subset \mathbb{R}^{d_u}$ are the state and the instantaneous control input of dimensions $d_x, d_u \in \mathbb{Z}_{>0}$, $f : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, and $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x \times d_u}$. Let $\mathcal{D}$ be the state space which is an open connected subset of $\mathbb{R}^{d_x}$, and let $\mathcal{X} \subset \mathcal{D}$ be its compact subset. We make the following assumptions.

Assumption 3.2.1. *For any locally Lipschitz continuous policy $\pi : \mathcal{D} \to \mathcal{U}$, $f + g\pi$ is locally Lipschitz over $\mathcal{D}$.*

With this setup, we present the main contribution.

3.3 *Constraint learning for control tasks*

In this section, we propose *limited duration control barrier functions* (LDCBFs), and present their properties and a practical way to find an LDCBF.

3.3.1 Limited duration control barrier functions

We start this section by the following definition.

Definition 3.3.1 (Limited-duration safety)**.** Given an open set of safe states $\mathcal{O} \subset \mathcal{D}$, let $\mathcal{C}_{\text{LD}}^T$ be a closed nonempty subset of $\mathcal{O}$. The dynamical system (3.2.1) is said to be *safe up to time* T, if there exists a policy π that ensures $x(t) \in \mathcal{O}$ for all $0 \leq t < T$ whenever $x(0) \in \mathcal{C}_{\text{LD}}^T$.

Given $B_{\text{LD}} : \mathcal{D} \to \mathbb{R}_{\geq 0}$ of class $C^1(\mathcal{D})$, let

$$\mathcal{O} := \left\{ x \in \mathcal{X} : B_{\text{LD}}(x) < \frac{L}{\beta} \right\}, \ L > 0, \ \beta > 0, \tag{3.3.1}$$

$$\mathcal{C}_{\text{LD}}^T = \left\{ x \in \mathcal{X} : B_{\text{LD}}(x) \leq \frac{Le^{-\beta T}}{\beta} \right\} \subset \mathcal{O}, \tag{3.3.2}$$

for some $T > 0$. Now, LDCBFs are defined by below.

Definition 3.3.2 (Limited duration control barrier function)**.** A function $B_{\text{LD}} : \mathcal{D} \to \mathbb{R}_{\geq 0}$ of class $C^1(\mathcal{D})$ is called a limited duration control barrier function (LDCBF) for $\mathcal{O}$ defined by (3.3.1) and for T if the following conditions are met:

1. $\mathcal{O} \subset \text{int}(\mathcal{X})$.

2. $\mathcal{C}_{\text{LD}}^T$ defined by (3.3.2) is nonempty and there exists a monotonically increasing locally Lipschitz continuous function[1] $\alpha : \mathbb{R} \to \mathbb{R}$ such that $\alpha(0) = 0$ and

$$\inf_{u \in \mathcal{U}} \{L_f B_{\text{LD}}(x) + L_g B_{\text{LD}}(x) u\} \leq \alpha\left(\frac{Le^{-\beta T}}{\beta} - B_{\text{LD}}(x)\right) + \beta B_{\text{LD}}(x), \ \forall x \in \mathcal{O}.$$

Given an LDCBF, the admissible control space $\mathcal{S}_{\text{LD}}^T(x), \ x \in \mathcal{O}$, is defined by

$$\mathcal{S}_{\text{LD}}^T(x) := \left\{ u \in \mathcal{U} : L_f B_{\text{LD}}(x) + L_g B_{\text{LD}}(x) u \leq \alpha\left(\frac{Le^{-\beta T}}{\beta} - B_{\text{LD}}(x)\right) + \beta B_{\text{LD}}(x) \right\}.$$

If the initial state is taken in $\mathcal{C}_{\text{LD}}^T$ and an admissible control is employed, safety up to time T is guaranteed.

[1]Note α is not necessarily an extended class-$\mathcal{K}$ function [126].

Theorem 3.3.1. *Suppose that a set of safe states $\mathcal{O}$ defined by* (3.3.1) *and an LDCBF $B_{\rm LD}$ defined on $\mathcal{D}$ are given. Suppose also that $x(0) \in \mathcal{C}_{\rm LD}^T$, where $\mathcal{C}_{\rm LD}^T$ is defined by* (3.3.2)*. Then, under Assumption 3.2.1, any locally Lipschitz continuous policy $\pi : \mathcal{D} \to \mathcal{U}$ that satisfies $\pi(x) \in \mathcal{S}_{\rm LD}^T(x)$, $\forall x \in \mathcal{O}$, renders the dynamical system* (3.2.1) *safe up to time T.*

Proof. Under Assumption 3.2.1, the trajectories $x(t)$ with an initial condition $x(0) \in \mathcal{C}_{\rm LD}^T \subset \mathcal{D}$ exist and are unique over $0 \leq t \leq \delta$ for some $\delta > 0$. Let $[0, T^*)$, $T^* > 0$, be its maximum interval of existence (T^* can be ∞), and let T_e be the first time at which the trajectory $x(t)$ exits $\mathcal{O}$, i.e.,

$$T_e := \inf\{t \in [0, T^*) : x(t) \notin \mathcal{O}\}. \tag{3.3.3}$$

Because $B_{\rm LD} \in C^1(\mathcal{D})$ and $\mathcal{O} \subset \mathrm{int}(\mathcal{X})$ imply $\mathcal{O}$ is open, it follows that $T_e > 0$. If T^* is finite, it must be the case that $x(t^*) \notin \mathcal{X}$ for some $t^* \in [0, T^*)$ which implies $0 \leq t \leq T_{\mathcal{X}} < T^*$, where

$$T_{\mathcal{X}} := \inf\{t \in [0, T^*) : x(t) \in \partial\mathcal{X}\}.$$

In this case, because $\mathcal{O} \subset \mathrm{int}(\mathcal{X})$, it follows that $0 < T_e \leq T_{\mathcal{X}} < T^*$. If, on the other hand, $T^* = \infty$, then T_e can still be defined by (3.3.3), and either $T_e = \infty$ or $T_e < T^*$ hold. When $T_e = \infty$, it is straightforward to prove the claim; therefore, we focus on the case where T_e is finite and $T_e < T^*$. Let T_p denote the last time at which the trajectory $x(t)$ passes through the boundary of $\mathcal{C}_{\rm LD}^T$ from inside before first exiting $\mathcal{O}$, i.e.,

$$T_p := \sup\left\{t \in [0, T_e) : x(t) \in \partial\mathcal{C}_{\rm LD}^T\right\}.$$

Because $\mathcal{C}_{\rm LD}^T$ is closed subset of the open set $\mathcal{O}$ and because $x(0) \in \mathcal{C}_{\rm LD}^T$, by continuity of the solution, it follows that $0 \leq T_p < T_e$. Now, the solution to $\dot{s}(t) = \beta s(t)$, where the initial condition is given by $s(T_p) = B_{\rm LD}(x(T_p)) = \frac{Le^{-\beta T}}{\beta}$, is

$$s(t) = B_{\rm LD}(x(T_p))e^{\beta(t-T_p)}, \ \forall t \geq T_p.$$

It thus follows that

$$s(T_p + T) = \frac{L}{\beta} e^{-\beta T} e^{\beta T} = \frac{L}{\beta},$$

and $T_p + T$ is the first time at which the trajectory $s(t),\ t \geq T_p$, reaches $\frac{L}{\beta}$.

Because $\alpha(\frac{Le^{-\beta T}}{\beta} - B_{\mathrm{LD}}(t)) \leq 0,\ \forall t \in [T_p, T_e)$, and $\pi(x) \in \mathcal{S}^T_{\mathrm{LD}}(x),\ \forall x \in \mathcal{O}$, we obtain, by the Comparison Lemma [126], [141, Theorem 1.10.2] and by continuity of the solutions over $t \in [0, T_e]$, that $B_{\mathrm{LD}}(x(t)) \leq s(t),\ \forall t \in [T_p, T_e]$. If we assume $T_e < T_p + T$, it follows that $B_{\mathrm{LD}}(x(T_e)) \leq s(T_e) < s(T_p + T) = \frac{L}{\beta}$ which is a contradiction because $\mathcal{O} \in \mathrm{int}(\mathcal{X})$ and $B_{\mathrm{LD}} \in C^1(\mathcal{D})$ imply $B_{\mathrm{LD}}(x(T_e)) = \frac{L}{\beta}$. Hence, $T_e \geq T_p + T$, which proves the Theorem. □

In practice, one can constrain the control input within the admissible control space $\mathcal{S}^T_{\mathrm{LD}}(x),\ x \in \mathcal{O}$, via QPs in the same manner as CBFs and CLFs.

Remark 3.3.2. The conditions when the solution to such QPs satisfies local Lipschitz continuity have been investigated in the literature including the work [182].

Theorem 3.3.1 implies that, through LDCBFs, global property (i.e., limited-duration safety) is ensured by constraining instantaneous control inputs. A benefit of relaxing the safety to limited-duration safety and of considering LDCBFs is highlighted by the ease of finding valid LDCBFs described below.

3.3.2 Finding a Limited Duration Control Barrier Function

We present one way of finding an LDCBF B_{LD} for the set of safe states through value function learning. Here, we mention that, to exploit value function learning in practice, one may assume a nominal model or a simulator is available during training time, or otherwise assume that getting outside of safe regions during training is not "fatal".

Let $\ell : \mathcal{D} \to \mathbb{R}_{\geq 0}$, be the immediate cost[2], and suppose $\mathcal{O} \subset \mathrm{int}(\mathcal{X})$ where the set of safe states $\mathcal{O}$ is given by

$$\mathcal{O} := \{x \in \mathcal{D} : \ell(x) < L\},\ L > 0.$$

[2]In this chapter, we consider the costs that do not depend on control inputs.

Given a policy $\pi : \mathcal{D} \to \mathcal{U}$, suppose that the system (3.2.1) is locally Lipschitz and that the initial condition $x(0) = x$ is in $\mathcal{O}$. Then, following the first argument in the proof of Theorem 3.3.1, $x(t)$ can be uniquely defined by extending the solution until reaching $\partial\mathcal{X}$. Let $T_e(x)$ be the first time at which the trajectory $x(t)$ exits $\mathcal{O}$ when $x(0) = x \in \mathcal{O}$. Now, we define the value function $V^{\pi,\beta} : \mathcal{D} \to \mathbb{R}_{\geq 0}$ by

$$V^{\pi,\beta}(x) := \begin{cases} \int_0^{T_e(x)} e^{-\beta t}\ell(x(t))dt + \frac{Le^{-\beta T_e(x)}}{\beta} & (x \in \mathcal{O}) \\ \frac{\ell(x)}{\beta} & (x \in \mathcal{D} \setminus \mathcal{O}) \end{cases}$$

where $\beta > 0$ is the discount factor. When the restriction of $V^{\pi,\beta}$ to $\mathcal{O}$, denoted by $V^{\pi,\beta}|_{\mathcal{O}}$, is of class $C^1(\mathcal{O})$, we obtain the continuous-time Bellman equation [148]:

$$\beta V^{\pi,\beta}(x) = L_f V^{\pi,\beta}(x) + L_g V^{\pi,\beta}(x)\pi(x) + \ell(x), \quad \forall x \in \mathcal{O}. \tag{3.3.4}$$

Now, for $V^{\pi,0}(x) := \int_0^{\infty} \ell(x(t))dt,\ x \in \mathcal{O}$, to exist and to be a CLF that ensures controlled invariance of its sublevel sets, one may at least assume that the policy π stabilizes the agent in a state $x^* \in \mathcal{O}$ where $\ell(x^*) = 0$ and $\ell(x) > 0,\ \forall x \in \mathcal{O} \setminus \{x^*\}$, which is restrictive. Instead, one can use $V^{\pi,\beta}$ as an LDCBF when $\beta > 0$.

Let $\hat{V}^{\pi,\beta} : \mathcal{D} \to \mathbb{R}_{\geq 0}$ of class $C^1(\mathcal{D})$ denote an approximation of $V^{\pi,\beta}$. Since $V^{\pi,\beta}(x) \geq \frac{L}{\beta}$ for all $x \in \mathcal{D} \setminus \mathcal{O}$ by definition, it follows that

$$\left\{x \in \mathcal{D} : V^{\pi,\beta}(x) < \frac{L}{\beta}\right\} \subset \mathcal{O} \subset \mathrm{int}(\mathcal{X}).$$

Therefore, we wish to use the approximation $\hat{V}^{\pi,\beta}$ as an LDCBF for the set $\mathcal{O}$; however, because it has an approximation error, we take the following steps to guarantee limited-duration safety. Using (3.3.4), define the estimated immediate cost function $\hat{\ell}$ by

$$\hat{\ell}(x) = \beta\hat{V}^{\pi,\beta}(x) - L_f\hat{V}^{\pi,\beta}(x) - L_g\hat{V}^{\pi,\beta}(x)\pi(x), \quad \forall x \in \mathcal{O}.$$

Select $c \geq 0$ so that $\hat{\ell}_c(x) := \hat{\ell}(x) + c \geq 0$ for all $x \in \mathcal{O}$, and define the function $\hat{V}_c^{\pi,\beta}(x) := \hat{V}^{\pi,\beta}(x) + \frac{c}{\beta}$.

Theorem 3.3.3. *Given $T > 0$, consider the set*

$$\hat{\mathcal{C}}_{\mathrm{LD}}^{T} = \left\{ x \in \mathcal{X} : \hat{V}_c^{\pi,\beta}(x) \leq \frac{\hat{L}e^{-\beta T}}{\beta} \right\}, \tag{3.3.5}$$

where $\hat{L} := \inf_{y \in \mathcal{X} \setminus \mathcal{O}} \beta \hat{V}_c^{\pi,\beta}(y)$. If $\hat{\mathcal{C}}_{\mathrm{LD}}^{T}$ is nonempty, then $\hat{V}_c^{\pi,\beta}(x)$ is an LDCBF for T and for the set

$$\hat{\mathcal{O}} := \left\{ x \in \mathcal{X} : \hat{V}_c^{\pi,\beta}(x) < \frac{\hat{L}}{\beta} \right\} \subset \mathcal{O}.$$

Proof. Because, by definition,

$$\hat{V}_c^{\pi,\beta}(x) \geq \frac{\hat{L}}{\beta}, \ \forall x \in \mathcal{X} \setminus \mathcal{O},$$

it follows that

$$\hat{\mathcal{O}} = \left\{ x \in \mathcal{X} : \hat{V}_c^{\pi,\beta}(x) < \frac{\hat{L}}{\beta} \right\} \subset \mathcal{O}.$$

Because $\hat{V}_c^{\pi,\beta} \in C^1(\mathcal{D})$ satisfies

$$L_f \hat{V}_c^{\pi,\beta}(x) + L_g \hat{V}_c^{\pi,\beta}(x)\pi(x) = \beta \hat{V}_c^{\pi,\beta}(x) - \hat{\ell}_c(x), \ \forall x \in \mathcal{O},$$

and $\hat{\ell}_c(x) \geq 0, \ \forall x \in \mathcal{O}$, it follows that

$$L_f \hat{V}_c^{\pi,\beta}(x) + L_g \hat{V}_c^{\pi,\beta}(x)\pi(x) \leq \alpha \left(\frac{\hat{L}e^{-\beta T}}{\beta} - \hat{V}_c^{\pi,\beta}(x) \right) + \beta \hat{V}_c^{\pi,\beta}(x),$$

for all $x \in \mathcal{O}$ and for a monotonically increasing locally Lipschitz continuous function α such that $\alpha(q) = 0, \ \forall q \leq 0$. Therefore, $\pi(x) \in \mathcal{U}, \ \forall x \in \mathcal{O} \subset \mathrm{int}(\mathcal{X})$ and $\hat{\mathcal{C}}_{\mathrm{LD}}^{T} \neq \emptyset$ imply that $\hat{V}_c^{\pi,\beta}$ is an LDCBF for the set $\hat{\mathcal{O}}$ and for T. □

Remark 3.3.4. The procedures above basically considers more conservative sets $\hat{\mathcal{C}}_{\mathrm{LD}}^{T}$ and $\hat{\mathcal{O}}$ so that the approximation error incurred by using $\hat{V}_c^{\pi,\beta}$ is taken into account. One can select sufficiently large c and sufficiently small $\hat{L}$ in practice to make the set of safe states more conservative. To enlarge the set $\mathcal{C}_{\mathrm{LD}}^{T}$, the immediate cost $\ell(x)$ is preferred to be close to zero for $x \in \mathcal{O}$, and L needs to be sufficiently large. Also, to make T as large as possible, the given policy π should keep the system safe up to sufficiently long time (see also Definition 3.4.1 for *good enough policy*); given a policy, the larger T one selects the smaller the set $\mathcal{C}_{\text{LD}}^T$ becomes. In addition, when $\ell(x)$ is almost zero inside $\mathcal{O}$ and when $L \gg 1$, the choice of β does not matter significantly to conservativeness of $\mathcal{C}_{\text{LD}}^T$.

As our approach is set-theoretic rather than specifying a single optimal policy, it is also compatible with the constraints-driven control and transfer learning.

3.4 Applications

In this section, we present two practical applications of LDCBFs, namely, long-duration autonomy and transfer learning.

3.4.1 Applications to long-duration autonomy

In many applications, guaranteeing particular properties (e.g., forward invariance) over an infinite-time horizon is difficult. Nevertheless, it is often sufficient to guarantee safety up to certain finite time, and our proposed LDCBFs act as useful relaxations of CBFs. To see that one can still achieve long-duration autonomy by using LDCBFs, we consider the settings of work in [185].

Problem formulation

Suppose that the state $x := [E, p^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^3$ has the information of energy level $E \in \mathbb{R}_{\geq 0}$ and the position $p \in \mathbb{R}^2$ of an agent, and the dynamics is given by (3.2.1) for

$$f(x) = [\hat{F}(x), F(x)]^\mathsf{T}, \quad g(x) = [\hat{G}(x), G(x)]^\mathsf{T},$$

where $\hat{F} : \mathbb{R}^3 \to \mathbb{R}$, $F : \mathbb{R}^3 \to \mathbb{R}^2$, $\hat{G} : \mathbb{R}^3 \to \mathbb{R}^2$, and $G : \mathbb{R}^3 \to \mathbb{R}^{2\times 2}$. Suppose also that the minimum necessary energy level E_{min} and the maximum energy level E_{max} satisfy $0 < E_{\text{min}} < E_{\text{max}}$, and that $\rho(p) \geq 0$ (equality holds only when the agent is at a charging station) is the energy required to bring the agent to a charging station from $p \in \mathbb{R}^2$, where $\rho \in C^1(\mathbb{R}^2)$. Define $\Delta E := E_{\max} - E_{\min}$ and

$$\mathcal{X} := \{x \in \mathbb{R}^3 : E_{\min} \le E \le E_{\max} \wedge 0 \le \rho(p) \le \Delta E\}.$$

Also, let $\mathcal{U} \subset \mathbb{R}^2$ be a control space. The open connected set $\mathcal{D} \supset \mathcal{X}$ is assumed to be properly chosen. Then, for $L := \beta \cdot \Delta E,\ \beta > 0$, we define $B_{\mathrm{LD}} : \mathcal{D} \to \mathbb{R}_{\ge 0}$ by

$$B_{\mathrm{LD}}(x) := \tilde{H}_\epsilon(E) + \rho(p),$$

where $\frac{\Delta E}{4} \gg \epsilon > 0$ and $\tilde{H}_\epsilon(E)$ is defined to make $\mathcal{O} \subset \mathrm{int}(\mathcal{X})$ as below. For brevity, let $\bar{E} := E_{\max} - E$ and $\bar{E}_{2\epsilon} := E - E_{\max} + 2\epsilon$. Then $\tilde{H}_\epsilon(E)$ is defined by

$$\tilde{H}_\epsilon(E) := \begin{cases} \bar{E} & (\bar{E}_{2\epsilon} < -\epsilon) \\ \frac{\Delta E}{8\epsilon^2}(\bar{E}_{2\epsilon}^2 + 2\epsilon \bar{E}_{2\epsilon} + \epsilon^2) + \bar{E} & (|\bar{E}_{2\epsilon}| \le \epsilon) \\ \frac{\Delta E}{2\epsilon} \bar{E}_{2\epsilon} + \bar{E} & (\epsilon < \bar{E}_{2\epsilon}) \end{cases}$$

Note $\tilde{H}_\epsilon(E)$ is continuously differentiable. It is straightforward to see that $E = E_{\max} \implies x \notin \mathcal{O}$, which is necessary to make $\mathcal{O} \subset \mathrm{int}(\mathcal{X})$.

Given $T > 0$, we can define $\mathcal{O}$ and $\mathcal{C}_{\mathrm{LD}}^T$ by (3.3.1) and (3.3.2). Note $x \in \mathcal{O}$ implies $E > E_{\min} + \rho(p)$.

Assumption 3.4.1. *The energy dynamics satisfies*

$$\exists K_d > 0, \quad \frac{dE}{dt} = \hat{F}(x) + \hat{G}(x)u \ge -K_d, \qquad \forall x \in \mathcal{D},\ \forall u \in \mathcal{U},$$

and is upper bounded by $\frac{dE}{dt} \le 0,\ \forall x \in \mathcal{D} \setminus \mathcal{A}_{\rho(p)=0},\ \forall u \in \mathcal{U}$*, where*

$$\mathcal{A}_{\rho(p)=0} := \{x \in \mathcal{O} : \rho(p) = 0\}.$$

In addition, the set

$$\tilde{\mathcal{S}}_{\mathrm{LD}}^T(x) := \left\{ u \in \mathcal{U} : L_{\tilde{f}} B_{\mathrm{LD}}(x) + L_{\tilde{g}} B_{\mathrm{LD}}(x) u \le \alpha \left(\frac{L e^{-\beta T}}{\beta} - B_{\mathrm{LD}}(x) \right) + \beta B_{\mathrm{LD}}(x) \right\} \quad (3.4.1)$$

is nonempty for all $x \in \mathcal{O} \setminus (\mathcal{A}_{\rho(p)=0} \cup \mathcal{A}_E)$*, where* $\tilde{f}(x) = [-K_d, F(x)]^\mathsf{T},\ \tilde{g} = [\mathbf{0}, G(x)]^\mathsf{T}$*, and*

$$\mathcal{A}_E := \{x \in \mathcal{O} : E \ge E_{\max} - 4\epsilon\}.$$

Remark 3.4.2. Suppose $\mathcal{S}_{\mathrm{LD}}^{T}(x)$ is nonempty for all $x \in \mathcal{A}_{\rho(p)=0} \cup \mathcal{A}_{E}$. Suppose also that Assumption 3.4.1 holds, then B_{LD} is an LDCBF for $\mathcal{O}$ and T s.t. $\mathcal{C}_{\mathrm{LD}}^{T}$ is nonempty, because $\tilde{\mathcal{S}}_{\mathrm{LD}}^{T}(x) \subset \mathcal{S}_{\mathrm{LD}}^{T}(x)$ for all $x \in \mathcal{O} \setminus (\mathcal{A}_{\rho(p)=0} \cup \mathcal{A}_{E})$.

Assumption 3.4.1 implies that the least possible exit time $\hat{T}_{\mathrm{energy}}(E)$ of $E > E_{\min}$ being below $E_{\min}$ is

$$\hat{T}_{\mathrm{energy}}(E) = \frac{(E - E_{\min})}{K_d}.$$

Under these settings, the following proposition holds.

Proposition 3.4.3. *Suppose Assumption 3.2.1 and Assumption 3.4.1 hold, and that* $T > \hat{T}_{\mathrm{energy}}(E_0)$ *for the initial energy level* $E_{\max} - 4\epsilon \geq E_0 > E_{\min}$*. Suppose also that* $x(0) \in \mathcal{C}_{\mathrm{LD}}^{T}$*, and that a locally Lipschitz continuous policy* $\pi : \mathcal{D} \to \mathcal{U}$ *satisfies* $\pi(x) \in \tilde{\mathcal{S}}_{\mathrm{LD}}^{T}(x)$ *for all* $x \in \mathcal{O} \setminus (\mathcal{A}_{\rho(p)=0} \cup \mathcal{A}_{E})$*. Further, assume the maximum interval of existence of unique solutions* E_t *and* ρ_t*, namely, the trajectories of* E *and* $\rho(p)$*, is given by* $[0, T^*)$ *for some* $T^* > 0$*. Then,*

$$T_{\rho_t=0} := \inf\{t \in [0, T^*) : \rho_t = 0\} \leq T_{\mathrm{energy}} := \inf\{t \in [0, T^*) : E_t - E_{\min} \leq 0\}.$$

Proof. Under Assumption 3.2.1, the trajectories $x(t)$ with an initial condition $x(0) \in \mathcal{C}_{\mathrm{LD}}^{T}$, and hence E_t and ρ_t, exist and are unique over $0 \leq t \leq \delta$ for some $\delta > 0$. Let $[0, T^*)$ be its maximum interval of existence (T^* can be ∞), which indeed exists. Following the same argument as in the proof of Theorem 3.3.1, we only focus on the case where T_e defined by (3.3.3) is finite and $0 < T_e < T^*$ (note $T_e = \infty \implies T_{\mathrm{energy}} = \infty$ and, in this case, the claim is trivially validated.) Also, if $E_t - E_{\min} > 0$ for all $t \in [0, T^*)$, then $T_{\mathrm{energy}} = \inf \emptyset = \infty$, and the claim is trivially validated again. Therefore, we assume that $T_{\mathrm{energy}} < T^*$. Further, define

$$\begin{aligned}
\hat{T} &:= \inf\{t \in [0, T^*) : \rho_t = 0 \wedge x(t) \notin \mathcal{C}_{\mathrm{LD}}^{T}\},\\
\hat{T}_e &:= \min\left\{T_e, \hat{T}\right\} \leq T_{\mathrm{energy}},\\
T_p &:= \sup\left\{t \in [0, T_e) : x(t) \in \partial\mathcal{C}_{\mathrm{LD}}^{T}\right\}.
\end{aligned}$$

Following the same argument as in the proof of Theorem 3.3.1, we have $0 \leq T_p < T_e$. Because it must be the case that $E_t > E_{\min},\ \forall t \in [0, T_p]$, we should only consider the case where $\rho_t > 0,\ \forall t \in [0, T_p]$. If we assume $\hat{T}_e = \hat{T}$, we obtain $T_{\rho t=0} \leq \hat{T} \leq T_e \leq T_{\text{energy}}$ which proves the claim. Therefore, we assume $\hat{T}_e = T_e$.

Let $\hat{E}_t$ be the trajectory following the virtual battery dynamics $d\hat{E}/dt = -K_d$ with the initial condition $\hat{E}_{T_p} = E_{T_p}$, and let $s(t)$ be the unique solution to

$$\dot{s}(t) = \beta s(t), \quad t \geq T_p,$$

where

$$s(T_p) = B_{\text{LD}}(x(T_p)) = \Delta E e^{-\beta T}.$$

Also, let

$$\varrho(t) = s(t) + \hat{E}_t - E_{\max},\ t \geq T_p.$$

Then, the time at which $s(t)$ reaches ΔE is $T_p + T$ because

$$s(T + T_p) = B_{\text{LD}}(x(T_p))e^{\beta(T+T_p-T_p)} = \Delta E e^{-\beta T} e^{\beta(T+T_p-T_p)} = \Delta E.$$

Since we assumed $x_t \notin \mathcal{A}_{\rho(p)=0},\ \forall t \in [0, T_p]$, under Assumption 3.4.1, we have

$$\hat{T}_{\text{energy}}(E_{T_p}) \leq \hat{T}_{\text{energy}}(E_0) < T.$$

Further, we have

$$\varrho(t) = B_{\text{LD}}(x(T_p))e^{\beta(t-T_p)} + \hat{E}_{T_p} - K_d(t - T_p) - E_{\max}.$$

Hence, we obtain

$$\hat{T}_0 := \inf\{t \geq T_p : \varrho(t) = 0\} \leq T_p + \hat{T}_{\text{energy}}(E_{T_p}).$$

On the other hand, under Assumption 3.2.1, the actual battery dynamics can be written as

$$dE/dt = -K_d + \Delta(x),$$

where $\Delta(x) \geq 0$. Also, because $E_0 \leq E_{\max} - 4\epsilon$, it follows that

$$E_t \leq E_{\max} - 4\epsilon$$

for all $t \in [0, T^*)$ under Assumption 3.4.1, implying

$$B_{\mathrm{LD}}(x(t)) = E_{\max} - E_t + \rho_t, \ \forall t \in [0, T^*).$$

Therefore,

$$\pi(x) \in \tilde{\mathcal{S}}^T_{\mathrm{LD}}(x), \ \forall x \in \mathcal{O} \setminus (\mathcal{A}_{\rho(p)=0} \cup \mathcal{A}_E),$$

indicates

$$\frac{dB_{\mathrm{LD}}(x(t))}{dt} \leq \beta B_{\mathrm{LD}}(x(t)) - \Delta(x(t)), \ \ \forall t \in [T_p, T_e).$$

Then, because

$$\frac{d\left(B_{\mathrm{LD}}(x(t)) - s(t)\right)}{dt} \leq \beta\left(B_{\mathrm{LD}}(x(t)) - s(t)\right) - \Delta(x(t)) \leq \beta\left(B_{\mathrm{LD}}(x(t)) - s(t)\right), \ \forall t \in [T_p, T_e),$$

and $\beta\left(B_{\mathrm{LD}}(x(T_p)) - s(T_p)\right) = 0$, we obtain

$$B_{\mathrm{LD}}(x(t)) - s(t) \leq -\int_0^t \Delta(x(t))dt, \ \forall t \in [T_p, T_e).$$

Here, following the same arguments as the proof of Theorem 3.3.1, we have that $T_e \geq T_p + T > T_p + \hat{T}_{\mathrm{energy}}(E_{T_p})$. Further, it follows that

$$\rho_t - \varrho(t) = B_{\mathrm{LD}}(x(T_p)) - s(t) + E_t - \hat{E}_t \leq -\int_0^t \Delta(x(t))dt + \int_0^t \Delta(x(t))dt = 0, \ \forall t \in [T_p, T_e),$$

which, by continuity of the solutions, leads to the inequality $\rho_t \leq \varrho(t), \ \forall t \in [T_p, T_e]$. Hence, we conclude that

$$\hat{T} \leq \hat{T}_0 \leq T_p + \hat{T}_{\mathrm{energy}}(E_{T_p}) < T_e.$$

This is a contradiction to the assumption $\hat{T}_e = T_e$, from which the proposition is proved. $\square$

Remark 3.4.4. When a function B_{LD} satisfying Assumption 3.4.1 and the set $\mathcal{O}$ are given, one may consider more restrictive set than $\tilde{\mathcal{S}}_{\mathrm{LD}}^T$; and if it is still nonempty, the agent stays in $\mathcal{O}$ longer than T if taking the control input in $\tilde{\mathcal{S}}_{\mathrm{LD}}^T$ starting from inside $\mathcal{C}_{\mathrm{LD}}^T$, which is defined for β.

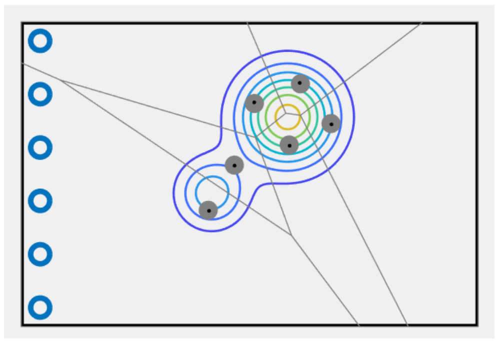
Remark 3.4.5. Instead of assuming the set (3.4.1) is nonempty, one may learn an LDCBF following the arguments in Section 3.3.2. In such a case, the immediate cost function $\ell(x)$ may be defined so that $0 \leq \ell(x) \ll 1$ for $E \in (E_{\min}, E_{\max})$ and that $\ell(x) \geq 1$ otherwise. Then, one may learn the value function of some policy for the system where $\hat{F}(x) = -K_d,\ \forall x \in \mathcal{O} \setminus \mathcal{A}_{\rho(p)=0},\ \hat{G}(x) = \mathbf{0},\ \forall x \in \mathcal{O}$, and $\exists y \in \mathcal{A}_{\rho(p)=0},\ \exists K_i > 0,\ \hat{F}(y) \geq K_i$. If $\hat{\mathcal{C}}_{\mathrm{LD}}^T$ defined by (3.3.5) is nonempty for $T > \hat{T}_{\mathrm{energy}}(E_0)$, then similar claims to Proposition 3.4.3 hold. Note the policy does not have to stabilize the system around $\mathcal{A}_{\rho(p)=0}$ but can be anything as long as $\hat{\mathcal{C}}_{\mathrm{LD}}^T$ becomes nonempty.

Simulated Experiment

Let the parameters be $E_{\max} = 1.0$, $E_{\min} = 0.55$, $K_d = 0.01$, $\beta = 0.005$ and $T = 50.0 > 45.0 = \Delta E / K_d$. We consider six agents (robots) with single integrator dynamics. An agent of the position $p_i := [\mathrm{x}_i, \mathrm{y}_i]^{\mathsf{T}}$ is assigned a charging station of the position $p_{\mathrm{charge},i}$, where x and y are the X position and the Y position, respectively. When the agent is close to the station (i.e., $\|p_i - p_{\mathrm{charge},i}\|_{\mathbb{R}^2} \leq 0.05$), it remains there until the battery is charged to $E_{\mathrm{ch}} = 0.92$. Actual battery dynamics is given by $dE/dt = -0.01E$. The coverage control task is encoded as Lloyd's algorithm [67] aiming at converging to the Centroidal Voronoi Tesselation, but with a soft margin so that the agent prioritizes the safety constraint. The locational cost used for the coverage control task is given by the following [68]:

$$\sum_{i=1}^{6} \int_{V_i(p)} \|p_i - \hat{p}\|^2 \varphi(\hat{p}) d\hat{p},$$

where $V_i(p) = \{\hat{p} \in \mathbb{R}^2 : \|p_i - \hat{p}\| \leq \|p_j - \hat{p}\|, \forall j \neq i\}$ is the Voronoi cell for the agent i. In particular, we used $\varphi([\hat{\mathrm{x}}, \hat{\mathrm{y}}]^{\mathsf{T}}) = e^{-\left\{(\hat{\mathrm{x}}-0.2)^2+(\hat{\mathrm{y}}-0.3)^2\right\}/0.06} + 0.5e^{-\left\{(\hat{\mathrm{x}}+0.2)^2+(\hat{\mathrm{y}}+0.1)^2\right\}/0.03}$. In MATLAB simulation (the simulator is provided on the Robotarium [199] website: www.robotarium.org),


(a)

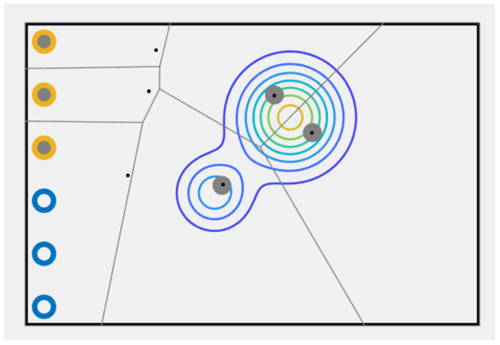

(b)

Figure 3.3: (a) Screenshot of agents executing coverage controls. (b) Screenshot of agents three of which are charging their batteries.

we used the random seed rng(5) for determining the initial states. Note, for every agent, the energy level and the position are set so that it starts from inside the set $\mathcal{C}_{\mathrm{LD}}^{T}$. Note also that battery information is local to each agent who has its own LDCBFs[3]; limited-duration safety is thus enforced in a decentralized manner.

Figure 3.3 shows (a) an image of six agents executing coverage tasks and (b) an image of the agents three of which are charging their batteries. Figure 3.4 shows the simulated battery voltage data of the six agents, in which we can observe that LDCBFs worked effectively for the swarm of agents to avoid depleting their batteries.

3.4.2 Applications to transfer learning

Another benefit of using LDCBFs is that, once a set of *good enough* policies that guarantee limited-duration safety for sufficiently large T and for sufficiently large $\mathcal{C}_{\mathrm{LD}}^{T}$ is obtained, one can reuse them for different tasks.

Definition 3.4.1 (Good enough policy)**.** Suppose a policy π guarantees safety up to time T if the initial state is in $\mathcal{C}_{\mathrm{LD}}^{T} \subset \mathcal{D}$. Suppose also that an initial state of a task $\mathcal{T}$ is always

[3]We are implicitly assuming that the conditions in Remark 3.4.2 are satisfied to make B_{LD} an LDCBF.

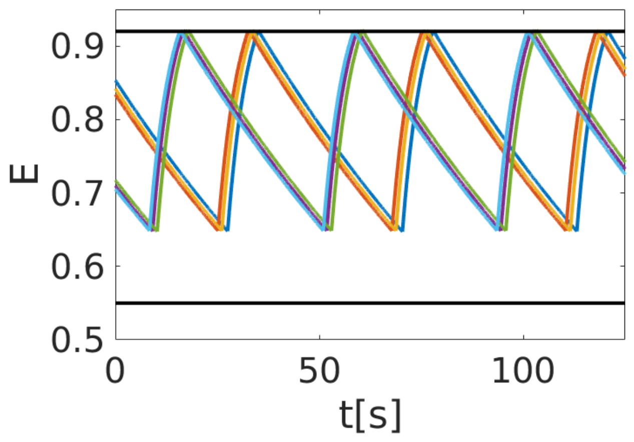


Figure 3.4: Battery levels of the six agents over time. Two black lines indicate the energy level when charged ($E_{\text{ch}} = 0.92$) and the minimum energy level ($E_{\min} = 0.55$). All agents successfully executed tasks without depleting their batteries.

taken in $\mathcal{C}_{\text{LD}}^{T}$ and that the task can be achieved within the time horizon T. Then, the policy π is said to be *good enough* with respect to the task $\mathcal{T}$.

We introduce the definition of transfer learning below.

Definition 3.4.2 (Transfer learning, [194, modified version of Definition 1])**.** Given a set of training data D_S for one task (i.e., source task) denoted by $\mathcal{T}_S$ (e.g., an MDP) and a set of training data D_T for another task (i.e., target task) denoted by $\mathcal{T}_T$, transfer learning aims to improve the learning of the target predictive function f_T (i.e., a policy in our example) in D_T using the knowledge in D_S and $\mathcal{T}_S$, where $D_S \neq D_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

In our example, we assume we know that the target task $\mathcal{T}_T$ shares some of the known state constraints with a source task $\mathcal{T}_S$. If a set of training data D_S is used to obtain a good enough policy for the source task, one can learn an LDCBF with this policy. When this policy is also good enough for the target task, then the learned LDCBF can be used to speed up the learning of the target task.

Illustrative example

For example, when learning a good enough policy for the balance task of the cart-pole problem, one can simultaneously learn a set of limited-duration safe policies that keep the pole from falling down up to certain time $T > 0$. The set of these limited-duration safe policies is obviously useful for other tasks such as moving the cart to one direction without letting the pole fall down.

We study some practical implementations. Given an LDCBF B_{LD}, define the set Π^T of admissible policies as

$$\Pi^T := \{\pi \in \Pi : \pi(x) \in \mathcal{S}_{\mathrm{LD}}^T(x),\ \forall x \in \mathcal{O}\},$$

where $\Pi := \{\pi : \pi(x) \in \mathcal{U},\ \forall x \in \mathcal{D}\}$, and $\mathcal{S}_{\mathrm{LD}}^T(x)$ is the set of admissible control inputs at x. If an optimal policy $\pi^{\mathcal{T}_T}$ for the target task $\mathcal{T}_T$ is included in Π^T, one can conduct learning for the target task within the policy space Π^T. If not, one can still consider Π^T as a soft constraint and explore the policy space $\Pi \setminus \Pi^T$ with a given probability or one may just select the initial policy from Π^T.

In practice, a parameterized policy is usually considered; a policy π_θ expressed by a parameter $\theta \in \mathbb{R}^{d_\theta}$ for $d_\theta \in \mathbb{Z}_{>0}$ is updated by policy gradient methods [234]. If the policy is in the linear form with a fixed feature vector, the projected policy gradient method [239] can be used. Suppose the policy $\pi(x)$ is linear with respect to θ at each $x \in D$ and that LDCBF constraints are affine with respect to $\pi(x)$ at each $x \in D$; in such a case, given a set of finite data points $D \subset \mathcal{X}$, $\tilde{\Pi}_\theta^T := \{\theta \in \mathbb{R}^{d_\theta} : \pi_\theta(x) \in \mathcal{S}_{\mathrm{LD}}^T(x),\ \forall x \in D\}$ is an intersection of finite affine constraints, which is a polyhedron. Hence, the projected policy gradient method looks like $\theta \leftarrow P_{\tilde{\Pi}_\theta^T}[\theta + \lambda \nabla_\theta F^{\mathcal{T}_T}(\theta)]$. Here, $P_{\tilde{\Pi}_\theta^T} : \mathbb{R}^{d_\theta} \to \tilde{\Pi}_\theta^T$ projects a policy onto $\tilde{\Pi}_\theta^T$ and $F^{\mathcal{T}_T}(\theta)$ is the objective function for the target task which is to be maximized. For the policy not in the linear form, one may update policies based on LDCBFs by modifying the deep deterministic policy gradient (DDPG) method [150]: because through LDCBFs, the global property (i.e., limited-duration safety) is ensured by constraining local control inputs, it suffices to add penalty terms to the cost when updating a policy over a batch of samples. For example, one may employ the log-barrier extension proposed in [125], which is a smooth approximation of the hard indicator function for inequality constraints but is not restricted to feasible inputs.

Simulated experiment

The simulation environment and the deep learning framework used in this simulated experiment are Cartpole in DeepMind Control Suite [238] and PyTorch [195], respectively. We take the following steps:

1. Learn a policy that balances the pole by using DDPG [150] over sufficiently long time horizon.

2. Learn an LDCBF by using the obtained actor network.

3. Try a random policy with the learned LDCBF and a (locally) accurate model to see if LDCBF works reasonably.

4. With and without the learned LDCBF, learn a policy that moves the cart to left without letting the pole fall down, which we refer to as move-the-pole task.

The parameters used for this simulated experiment are summarized in Table 3.1. Here, angle threshold stands for the threshold of $\cos\psi$ where ψ is the angle of the pole from the standing position, and position threshold is the threshold of the cart position p. The angle threshold and the position threshold are used to terminate an episode. Note that the cart-pole environment of MuJoCo [244] xml data in DeepMind Control Suite is modified so that the cart can move between -3.8 and 3.8. We use prioritized experience replay when learning an LDCBF. Specifically, we store the positive and the negative data, and sample 4 data points from the positive one and the remaining 60 data points from the negative one. In this simulated experiment, actor, critic and LDCBF networks use ReLU nonlinearities. The actor network and the LDCBF network have two hidden layers of 300, 200 units, and the critic network has two hidden layers of 400, 300 units. The control input vector is concatenated to the state vector from the second critic layer.

Step1: The average duration (i.e., the first exit time, namely, the time when the pole first falls down) out of 10 seconds (corresponding to 1000 time steps), over 10 trials for the policy learned through the balance task by DDPG was 10 seconds.

Step2: Then, by using this successfully learned policy, an LDCBF is learned by assigning the cost $\ell(x) = 1.0$ for $\cos\psi < 0.2$ and $\ell(x) = 0.1$ elsewhere. Also, because the LDCBF is learned in a discrete-time form, we transform it to a continuous-time form via multiplying it by $\Delta_t = 0.01$. When learning an LDCBF, we initialize each episode as follows: the angle ψ is uniformly sampled within $-1.5 \leq \psi \leq 1.5$, the cart velocity $\dot{p}$ is multiplied by 100 and the angular velocity $\dot{\psi}$ is multiplied by 200 after the initialization by DeepMind Control Suite. The LDCBF learned by using this policy is illustrated in Figure 3.5, which agrees with our intuitions. Note that $\frac{L}{\beta}$ in this case is $\frac{1.0}{-\log(0.999)/0.01} \approx 10.0$.

Step3: To test this LDCBF, we use a uniformly random policy ($\pi(x)$ takes the value between -1 and 1) constrained by the LDCBF with the function $\alpha(q) = \max\{0.1q, 0\}$ and with the time constant $T = 5.0$. When imposing constraints, we use the (locally accurate) control-affine model of the cart-pole in the work [32], where we replace the friction parameters by zeros for simplicity. The average duration out of 10 seconds over 10 trials for this random policy was 10 seconds, which indicates that the LDCBF worked sufficiently well. We also tried this LDCBF with the function $\alpha(q) = \max\{3.0q, 0\}$ and $T = 5.0$, which resulted in the average duration of 5.58 seconds. Moreover, we tried the fixed policy $\pi(x) = 1.0$, with the function $\alpha(q) = \max\{0.1q, 0\}$ and $T = 5.0$, and the average duration was 4.73 seconds, which was sufficiently close to $T = 5.0$.

Step4: For the move-the-pole task, we define the success by the situation where the cart position p, $-3.8 \leq p \leq 3.8$, ends up in the region of $p \leq -1.8$ without letting the pole fall down. The angle ψ is uniformly sampled within $-0.5 \leq \psi \leq 0.5$ and the rest follow the initialization of DeepMind Control Suite. The reward is given by $(1+\cos\psi)/2\times$(utils.rewards.tolerance($\dot{p}+$ 1.0, bounds $= (-2.0,\ 0.0)$, margin $= 0.5$)), where utils.rewards.tolerance is the function de-

Table 3.1: Summary of the parameter settings for the cart-pole problem. These parameters are chosen so that a policy for the balance task can be obtained, an LDCBF can be learned, and the move-the-pole task can be accomplished within 15 episodes when using an LDCBF. We could not find parameters that make the move-the-pole task work without LDCBFs within 15 episodes.

Parameters	Balance task	For Learning an LDCBF	Move-the-pole task with LDCBF	Move-the-pole task without LDCBF
Discount β	$-\log(0.99)/0.01$	$-\log(0.999)/0.01$	$-\log(0.999)/0.01$	$-\log(0.999)/0.01$
Angle threshold $\cos\psi_{\text{thre}}$	0.75	0.2	0.75	0.75
Position threshold p_{thre}	±1.8	±3.8	±3.8	±3.8
Soft-update μ	10^{-3}	10^{-2}	10^{-3}	10^{-3}
Step size for target NNs	10^{-4}	10^{-2}	10^{-4}	10^{-4}
Time steps per episode	300	50	300	300
Number of episodes	80	200	Up to 15	Up to 15
Minibatch size	64	64	64	64
Random seed	10	10	10	10
States x	$\sin\psi,\ 0.1\dot{p},\ 0.1\dot{\psi}$	$\sin\psi,\ \dot{p},\ \dot{\psi}$	$\sin\psi,\ \dot{p},\ \dot{\psi}$	$\sin\psi,\ \dot{p},\ \dot{\psi}$

fined in [238]. In other words, we give high rewards when the cart velocity is negative and the pole is standing up. To use the learned LDCBF for DDPG, we store matrices and vectors used in linear constraints along with other variables such as control inputs and states, which we use for experience replay. Then, the log-barrier extension cost proposed in [125] is added when updating policies. Also, we try DDPG without using the LDCBF for the move-the-pole task. Both approaches initialize the policy by the one obtained after the balance task. The average success rates of the policies obtained after the numbers of episodes up to 15 over 10 trials are given in Table 3.2 for DDPG with the learned LDCBF and for DDPG without LDCBFs. This result implies that our proposed approach successfully transferred information from the source task to the target task.

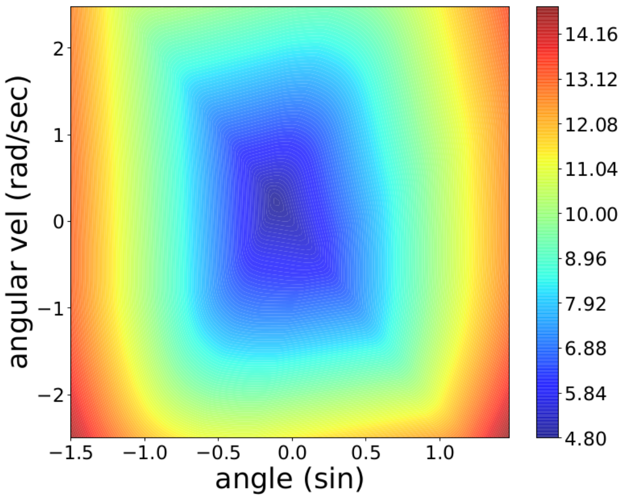


Figure 3.5: Illustration of the LDCBF for $\sin\psi$ and $\dot{\psi}$ at zero cart velocity. The center has lower value. Also, unsafe regions have the values over $\frac{L}{\beta} = \frac{1.0}{-\log(0.999)/0.01} \approx 10.0$.

Table 3.2: Summary of the results for the move-the-pole task. Over 15 episodes, the success rates over 10 trials are shown. DDPG with LDCBF uses learned LDCBF to constrain control inputs when updating policies by DDPG, and DDPG without LDCBF does not use LDCBFs. By using LDCBFs the move-the-pole task is shown to be easily learned.

Algorithm\Episode	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DDPG with LDCBF	0.0	0.0	0.0	0.4	0.7	0.8	1.0	1.0	1.0	0.7	0.7	1.0	1.0	1.0	1.0
DDPG without LDCBF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3.5 Chapter summary and discussion

In this chapter, a novel control theoretic notion named limited-duration safety is presented as a relaxation of forward invariance of a set of states. Then, limited-duration control barrier functions are introduced to guarantee limited-duration safety. LDCBFs can be obtained through value function learning. The properties of LDCBFs are analyzed and their applications to a persistent coverage control task and to a transfer learning problem by shrinking the space of desirable policies are presented.

Efficient enforcement of limited-duration safety by LDCBFs is made possible when we have access to a dynamical system model that is affine in control; although it is possible to transform nonlinear control systems to this control-affine form by regarding the time derivative of control as a new *control input*, we anyway need access to some knowledge on the model. An advantage of our approach is highlighted by the fact that we only need local information on the dynamics to enforce (limited-duration) safety, and combining our technique with learning is hence efficient for long horizon decision making. As a future work, incorporating model learning and discussing the role of model misspecification in detail will be beneficial for some practical applications.

On the other hand, we did not explore sample complexities of our transfer learning application. As we know how the control theoretic techniques can give some guarantees to learning, it should also be beneficial to consider how it improves sample complexities from the statistical point of view.

Chapter 4

PROBLEM FORMULATIONS THROUGH THE LENS OF DYNAMICAL SYSTEMS

Chapter's key takeaways

By exploiting the continuity property of continuous control problems and by assuming the transition dynamics is in a known reproducing kernel Hilbert space (RKHS), we design a provably sample efficient RL algorithm that works for continuous control problems. The key is to assume that a state-of-the-art optimal control technique produces an optimal sequence of control actions under a known system model, and we embed it to our RL algorithm as an oracle. By employing the simulators with different physics parameters as the featurizers for the unknown transition dynamics, our algorithm works successfully under a complex robotics simulation environment.

On the other hand, leaping from the classical MDP formulation, this chapter considers the *spectrum cost* that is not subsumed by a classical single-step cost or an episodic cost, which could be viewed as a generalization of eigenstructure / pole assignments to non-linear decision making problems. The framework systematically deals with the *shaping* of behavior. For online learning settings under unknown dynamics, the spectrum cost is unobservable, which makes it differ from other types of costs such as a cumulative single-step cost and a policy cost. As such, we need some specific structural assumptions to obtain sample efficient (if not computationally efficient) algorithm. Proof techniques include some operator theoretic arguments that are novel in this context.

4.1 Introduction

RL is a prime example of machine learning for dynamical systems, and recent years have seen a number of successes in demanding sequential decision making tasks ranging from robotic hand manipulation [244, 14, 139, 241, 154, 13] to game playing [221, 33, 196, 49, 266].

Historically, provably correct methods have been actively studied within the RL literature [212, 112, 231, 9]. However, tackling the continuous (nonlinear) control problems in practice typically requires a specific problem setups which have not been researched in detail in such studies; in fact, continuity properties inherent in continuous control problems with respect to the underlying “disturbance” (often modeled as statistical additive noise) can, if properly exploited, actually be useful for fast path planning algorithms [110, 262] for example. Therefore, by leveraging such continuity properties, we propose a sample efficient RL algorithm tailored for continuous control problems. Importantly, our algorithm requires an optimal control *oracle* to theoretically ensure sample efficiency; which is reasonable given the recent advancement of control methods that can produce (near) optimal sequence of control actions for robotics problems in practice.

Subsequently, we reconsider the optimal control objective, which is oftentimes the long-term costs resembling the principle of least action. The optimized motions are often ‘unnatural’, representing, for example, behaviors with sudden accelerations that waste energy and lack predictability. Intuitively, the motion specified by the task-oriented cumulative costs formulation may ignore “how to” achieve the task unless careful design of the cumulative cost is in place, necessitating a systematic approach that effectively regularizes or *constrains* the dynamics to guarantee predictable global property such as stability. In this chapter, we present a novel paradigm of controlling nonlinear systems via the minimization of the *Koopman spectrum cost*: a cost over the Koopman operator of the controlled dynamics. This induces a broader class of dynamical behaviors that evolve over stable manifolds such as nonlinear oscillators, closed loops, and smooth movements. We demonstrate that some dynamics characterizations that are not possible with a cumulative cost optimization are feasible in this paradigm. Moreover, we present a sample efficient (theoretical) online learning algorithm for our problem that enjoys a sub-linear regret bound under some structural assumptions.

Contributions: The contributions of this chapter are four folds: (1) design an algorithm (LC^3) that works in practice for complex robotics simulation environments under fairly general model assumptions, (2) provide regret bounds for the algorithm, (3) propose the Koopman spectrum cost that complements the (cumulative) single-step cost for nonlinear control, which enables to effectively encode/imitate some desirable agent dynamics such as limit cycles, stable loops, and smooth movements, (4) and propose theoretical online algorithm (KS-LC^3) to provide regret bounds under structural assumptions by using some of the results from the analysis of LC^3 with novel operator theoretic arguments.

4.2 Model-based RL with an optimal control oracle embedded

In the first half of this chapter, we present our model-based RL algorithm for continuous (nonlinear) control problems with sample complexity analysis.

4.2.1 Problem setups

In particular, we consider the following nonlinear control problem, where the transition dynamics are described, for $h \in [H]$, by

$$x_{h+1} = f(x_h, u_h) + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

where the state $x_h \in \mathbb{R}^{d_x}$; the control $u_h \in \mathcal{U}$ where $\mathcal{U}$ may be an arbitrary set (not necessarily a vector space); $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is assumed to live within a known RKHS; the additive noise is assumed to be independent across time steps. Specifically, the model considered in this work was introduced in [165], which we refer to as the *kernelized nonlinear regulator* (KNR) for the infinite dimensional extension. Equivalently, the primal version of this assumption is that:

$$f(x,u) = W^\star \phi(x,u)$$

for some known function $\phi : \mathcal{X} \times \mathcal{U} \to \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space (either finite or countably infinite dimensional) and where $W^\star$ is a linear mapping. Given an instantaneous cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$, the KNR problem can be described by the following optimization problem:

$$\min_{\pi \in \Pi} J^\pi(x_0; c) \text{ where } J^\pi(x_0; c) = \mathbb{E}\left[\sum_{h=0}^{H-1} c(x_h, u_h) \middle| \pi, x_0\right]$$

where x_0 is a given starting state; Π is some set of feasible controllers; and where a controller (or a policy) is a mapping $\pi : \mathcal{X} \times [H] \to \mathcal{U}$. We denote the best-in-class cumulative cost as $J^\star(x_0; c) = \min_{\pi \in \Pi} J^\pi(x_0; c)$. Given any model parameterization W, we denote $J^\pi(x_0; c, W)$ as the expected total cost of π under the dynamics $W\phi(x,u) + \epsilon$.

We consider an online version of this KNR problem: in each episode t, we observe an instantaneous cost function c^t; we choose a policy π^t; we execute π^t and observe a sampled trajectory $x_0, u_0, \ldots, x_{H-1}, u_{H-1}$; we incur the cumulative cost under c^t. Our goal is to minimize the sum of our costs over T episodes, and we employ *cumulative regret* as our performance metric defined by:

$$\textsc{Regret}_T = \sum_{t=0}^{T-1} \sum_{h=0}^{H-1} c^t(x_h^t, u_h^t) - \sum_{t=0}^{T-1} \min_{\pi \in \Pi} J^\pi(x_0; c^t)$$

where $\{x_h^t\}$ is the observed states and $\{u_h^t\}$ is the observed sequence of controls. A desirable asymptotic property of an algorithm is to be no-regret, i.e. the time averaged version of the regret goes to 0 as T tends to infinity.

4.2.2 The Lower Confidence-based Continuous Control algorithm

The Lower Confidence-based Continuous Control algorithm (LC^3) is based on "optimism in the face of uncertainty", which is described in Algorithm 1. At episode t, we use all previous experience to define an uncertainty region (an ellipse). The center of this region, $\overline{W}^t$, is the

Algorithm 1 Lower Confidence-based Continuous Control (LC^3)

```
Require: Policy class $\Pi$; regularizer $\lambda$; confidence parameter $C_1$ (see (4.2.3)).
1: Initialize BALL$^0$ to be any set containing $W^\star$.
2: for $t = 0 \dots T$ do
3:     $\pi^t = \arg\min_{\pi \in \Pi} \min_{W \in \textsc{Ball}^t} J^\pi(x_0; c^t, W)$
4:     Execute $\pi^t$ to sample a trajectory $\tau^t := \{x_h^t, u_h^t, c_h^t, x_{h+1}^t\}_{h=0}^{H-1}$
5:     Update BALL$^{t+1}$ (as specified in (4.2.2)).
6: end for
```

solution of the following regularized least squares problem:

$$\overline{W}^t = \arg\min_W \sum_{\tau=0}^{t-1} \sum_{h=0}^{H-1} \|W\phi(x_h^\tau, u_h^\tau) - x_{h+1}^\tau\|_{\mathbb{R}^{d_x}}^2 + \lambda \|W\|_{\mathrm{HS}}^2, \tag{4.2.1}$$

where λ is a parameter, and the shape of the region is defined through the feature covariance:

$$\Sigma^t = \lambda I + \sum_{\tau=0}^{t-1} \sum_{h=0}^{H-1} \phi(x_h^\tau, u_h^\tau)(\phi(x_h^\tau, u_h^\tau))^\top, \text{ with } \quad \Sigma^0 = \lambda I.$$

The uncertainty region, or confidence ball, is defined as:

$$\textsc{Ball}^t = \left\{ W : \left\| \left(W - \overline{W}^t \right) \left(\Sigma^t \right)^{1/2} \right\|^2 \le \beta^t \right\}, \tag{4.2.2}$$

where

$$\beta^t := C_1 \Big(\lambda \sigma^2 + \sigma^2 \Big(d_x + \log \left(t \det(\Sigma^t) / \det(\Sigma^0) \right) \Big) \Big), \tag{4.2.3}$$

with C_1 being a parameter of the algorithm.

At episode t, the LC^3 algorithm will choose an optimistic policy in Line 3 of Algorithm 1. Solving this optimistic planning problem in general is NP-hard [73]. Given this computational hardness, we focus on the statistical complexity and explicitly assume access to the following computational oracle:

Assumption 4.2.1 (Black-box computation oracle)**.** *We assume access to an oracle that implements Line 3 of Algorithm 1.*

Recent advancement of control heuristics through gradient based methods such as DDP [110], iLQG [236] and CIO [179], or sampling based methods, such as MPPI [262] and DMD-MPC [254], makes it reasonable to assume the oracle in practice. In particular, these planning algorithms are natural to use in conjunction with Thompson sampling [240, 193] in practice, i.e. we sample W^t from $\mathcal{N}(\overline{W}^t, (\Sigma^t)^{-1})$ and then compute and execute the corresponding optimal policy $\pi^t = \arg\min_{\pi\in\Pi} J^\pi(x_0; c^t, W^t)$ using a planning oracle.

4.2.3 Information theoretic regret bounds

We analyze the regret of Algorithm 1. Following [228], let us define the (expected) Maximum Information Gain by:

$$\begin{aligned}\gamma_T(\lambda) &:= \max_{\mathcal{A}} \mathbb{E}_{\mathcal{A}} \left[\log\left(\det\left(\Sigma^T\right)/\det\left(\Sigma^0\right)\right)\right] \\ &= \max_{\mathcal{A}} \mathbb{E}_{\mathcal{A}} \left[\log\det\left(I + \frac{1}{\lambda}\sum_{t=0}^{T-1}\sum_{h=0}^{H-1} \phi(x_h^t, u_h^\tau)(\phi(x_h^t, u_h^\tau))^\top\right)\right],\end{aligned}$$

where the max is over algorithms $\mathcal{A}$, (an algorithm is a mapping from the history before episode t to the next policy $\pi_t \in \Pi$).

Remark 4.2.2. (Finite dimensional RKHS) If $\phi \in \mathbb{R}^{d_\phi}$, with $\|\phi(x,u)\| \le B \in \mathbb{R}_{\ge 0}$ for all (x,u), then $\gamma_T(\lambda)$ will be $O(d_\phi \log(1 + THB^2/\lambda)$ (see Lemma C.3.4). Furthermore, it may be the case that $\gamma_T(\lambda) \ll d_\phi$ if the eigenspectrum of the covariance matrices tends to concentrate in a lower dimensional subspace. See [228] for details and for how $\gamma_T(\lambda)$ scales for a number of kernels.

Now, we make the following assumption which avoids requiring bounded costs.

Assumption 4.2.3. *(Bounded second moments at x_0) Assume that c^t is a nonnegative function for all t and that the realized cumulative cost, when starting from x_0, has uniformly bounded second moments, over all policies and cost functions c^t. Precisely, suppose for every c^t,*

$$\sup_{\pi\in\Pi} \mathbb{E}\left[\left(\sum_{h=0}^{H-1} c^t(x_h, u_h)\right)^2 \,\middle|\, x_0, \pi\right] \le V_{\max}.$$

With these preparations in place, we give the main regret bound theorem.

Theorem 4.2.4 (LC^3 regret bound)**.** *Suppose Assumptions 4.2.1 and 4.2.3 hold. Set* $\lambda = \frac{\sigma^2}{\|W^\star\|^2}$ *and define*

$$\widetilde{d}_T^{\,2} := \gamma_T(\lambda) \cdot \big(\gamma_T(\lambda) + d_x + \log(T) + H\big).$$

There exist constants $C_1, C_2 \leq 20$ *such that if LC^3 (Algorithm 1) is run with input parameters* λ *and* C_1 *(in* (4.2.3)*), then following regret bound holds for all* T*,*

$$\mathbb{E}_{\mathrm{LC}^3}\left[\mathrm{Regret}_T\right] \leq C_2 \ \widetilde{d}_T \sqrt{V_{\max} H T}.$$

While the above regret bound is applicable to the infinite dimensional RKHS setting and does not require uniformly bounded features ϕ, it is informative to specialize the regret bound to the finite dimensional case with bounded features.

Corollary 4.2.5 (LC^3 Regret for finite dimensional, bounded features)**.** *Suppose that Assumptions 4.2.1 and 4.2.3 hold;* d_ϕ *is finite; and that* ϕ *is uniformly bounded, with* $\|\phi(x,u)\| \leq B$*. Under the same parameter choices as in Theorem 4.2.4, we have, for all* T*,*

$$\mathbb{E}_{\mathrm{LC}^3}\left[\mathrm{Regret}_T\right] \leq C_2 \sqrt{d_\phi\Big(d_\phi + d_x + \log(T) + H\Big) V_{\max} H T} \cdot \log\left(1 + \frac{B^2\|W^\star\|^2}{\sigma^2}\frac{TH}{d}\right).$$

The above immediately follows from a bound on the finite dimensional information gain (see Lemma C.3.4).

Remark 4.2.6 (Logarithmic parameter dependencies)**.** It is worthwhile noting that our regret bound has only logarithmic dependencies in $\|W^\star\|$ and σ^2.

Remark 4.2.7 (Linear Quadratic Regulators (LQRs) as special cases)**.** Our model generalizes the Linear Quadratic Regulator (LQR). Specifically, we can set $\phi(x,u) = [x^\top, u^\top]^\top$, $c(x,u) = x^\top Q x + u^\top R u$ with Q and R being some positive semi-definite matrices. We can consider a policy class to be a (subset) of all linear controls, i.e., $\Pi = \{\pi : \pi(x) = Kx, K \in \mathbb{R}^{d_u \times d_x}\}$.

Proof Techniques

While our proof techniques utilize methods developed for the analysis of linear bandits [73, 1] and Gaussian process bandits [228], there are a number of new technical challenges to be addressed with regards to the multi-step extension to RL. A key technical, "self-bounding" lemma bounds the difference in cost under two different models, i.e. $J^{\pi}(x; c, W^{\star}) - J^{\pi}(x; c, W)$, in terms of the second moment of the cumulative cost itself. The proof involves the construction of a certain stopping time martingale along with a novel way to handle Gaussian smoothing through the chi-squared distance function between two distributions.

4.2.4 Experiments

We evaluate LC^3 on three domains: a set of continuous control tasks, a maze environment that requires exploration, and a dexterous manipulation task. Throughout these experiments, we use model predictive path integral control (MPPI) [262] for planning, and posterior reshaping [56] (i.e., scaling of posterior covariance) for Thompson sampling – we are not implementing LC^3 as analyzed, but rather a Thompson sampling variation. The algorithms are implemented in the Lyceum framework under the Julia programming language [230, 41]. Comparison algorithms are provided by [257, 258]. Note these experiments use reward (negative cost) for evaluations. Further details of the experiments in this section can be found in Appendix A.1.

Benchmark tasks with random features

We use some common benchmark tasks, including MuJoCo [244] environments from OpenAI Gym [46]. We use Random Fourier Features (RFF) [202] to represent ϕ. Figure 4.1 plots the learning curves against Ground-Truth-MPPI and the best model-based RL (MBRL) algorithm reported in [258]. It is observed that LC^3 with RFFs quickly increased reward in early stages, indicating low sample complexities empirically. Table 4.1 shows the final performances (at 200k time steps) of LC^3 with RFFs for six environments, and includes

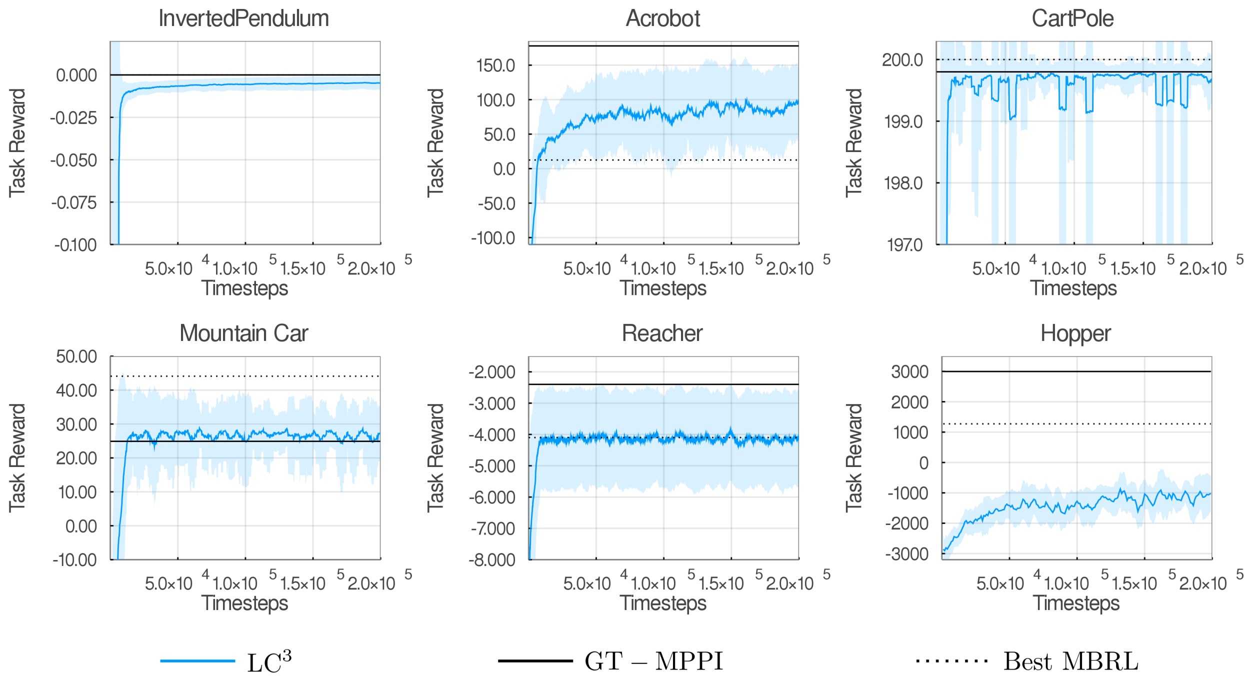


Figure 4.1: Performance curves of LC^3 with RFFs for different Gym environments. Note the reward (negative cost) ranges of those plots are made different. The final mean performances of GT-MPPI and the best MBRL algorithm reported in [258] are also shown for reference. The algorithm is run for 200,000 time steps and with four random seeds. The curves are averaged over the four random seeds and over a window size of 5,000 time steps.

its ranking compared to the benchmarks results from [258]. We find that LC^3 consistently performs well on simple continuous control tasks, and it works well even without posterior sampling. However, when the dynamical complexity increases, such as with the contact-rich Hopper model, our method's performance suffers due to insufficient approximation capabilities of a finite set of RFFs. This suggests that more interesting scenarios require different feature representation.

	InvertedPendulum	Acrobot	CartPole	Mountain Car	Reacher	Hopper
LC^3	-0.0 ± 0.0	95.4 ± 52.5	199.7 ± 0.4	27.3 ± 8.1	-4.1 ± 1.6	-1016.5 ± 607.4
(Ranking)	1/11	1/11	2/11	2/11	1/11	7/11
GT-MPPI	-0.0 ± 0.0	177.8 ± 25.0	199.8 ± 0.1	24.9 ± 2.9	-2.4 ± 0.1	2995.7 ± 215.3
PETS-CEM	-20.5 ± 28.9	12.5 ± 29.0	199.5 ± 3.0	-57.9 ± 3.6	-12.3 ± 5.2	1125.0 ± 679.6
PILCO	-194.5 ± 0.8	-394.4 ± 1.4	-1.9 ± 155.9	-59.0 ± 4.6	-13.2 ± 5.9	-1729.9 ± 1611.1

Table 4.1: Final performances for six Gym environments. Algorithms are run under the same conditions of [258]. The performances of PETS-CEM and PILCO are copied for reference, and the performance of ground-truth MPPI (GT-MPPI) that has access to the true model is also shown. The results are averaged over four random seeds and a window size of 5,000 time steps.

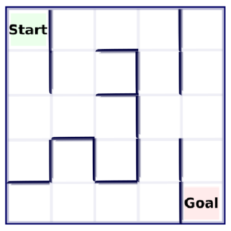
Exploring the maze

We construct a maze environment to study the exploration capability of LC^3 (see Fig. 4.2 (left)). State and control take values in $[-1,1]^2 \subset \mathbb{R}^2$ and in $[-1,1] \subset \mathbb{R}$, respectively. The task is to bring an agent to the goal state being guided by the negative cost (reward) $-c(x_h, u_h) = 8 - \|x_h - [1,1]^\top\|^2_{\mathbb{R}^2}$. We use a one-hot vector of states and actions as features.

We compare the performances, over 50 episodes with task trajectory length 30, of LC^3 (with different scale parameters for posterior reshaping) to random walk and PETS-CEM [64]. Fig. 4.2 (right) plots the means and standard deviations, across four random seeds, of the number of state-action pairs visited over episodes. We observe that LC^3's strategic exploration better modeled the setting for higher rate of success.

Practical application

As we might consider learning model dynamics for the real world in applications such as robotics, we need sufficiently complex features – without resorting to large scale data collection for feature learning. One solution to this problem is creating an ensemble of parametric models, such as found in [241, 178] (see Figure 4.3). We take the perspective that most model parameters of a robotic system will be known, such as kinematic lengths, actuator



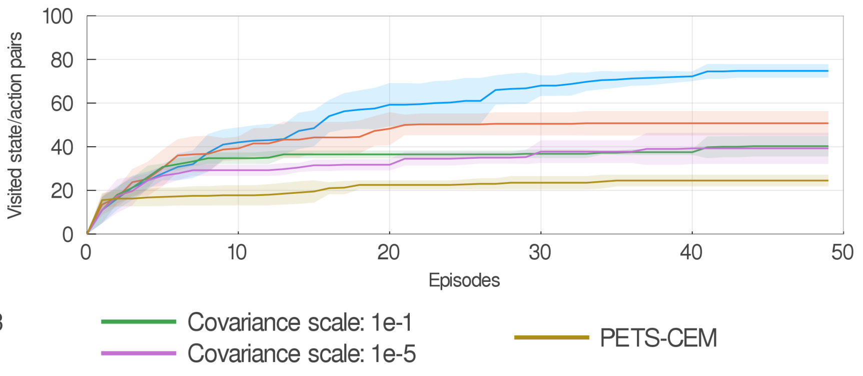


Figure 4.2: Left: An illustration of the maze environment. Start and End states are $[-1,-1]^\top$ and $[1,1]^\top$, respectively. Dark lines are "walls". Right: The means and standard deviations, across four random seeds, of the number of state-action pairs already explored over episodes. Covariance scale is the posterior reshaping constant of Thompson sampling. Random walk takes actions uniformly sampled within $[-1,1]$. PETS-CEM is a representative model-based RL which uses uncertainty of dynamics but without exploration. The agent always reaches the goal within 50 episodes under the best setting of LC^3 and the average number of episodes required for the first success is 25.0, while random walk and PETS-CEM never bring the agent to the goal within 50 episodes.

specifications, and inertial configurations. Since we would like robots to operate in the wild, some dynamical properties may be unknown: in our case, it is going to be the manipulated object's dynamical properties. Said another way, the robot knows about itself, but only a little about the object.

In this experiment, we demonstrate our model learning algorithm on a robotics inspired, dexterous manipulation task. An arm and hand system (see Fig. 4.4) must pick up a spherical object with unknown dynamical properties, and hold it at a target position. The entire system has 33 degrees of freedom, and an optimal trajectory would involve numerous discontinuous contacts; the system dynamics are not well captured by random features and such features are not easily learned. We instead use the predictive output of an ensemble of six MuJoCo models as our features ϕ, each with randomized parameters for the object. Using

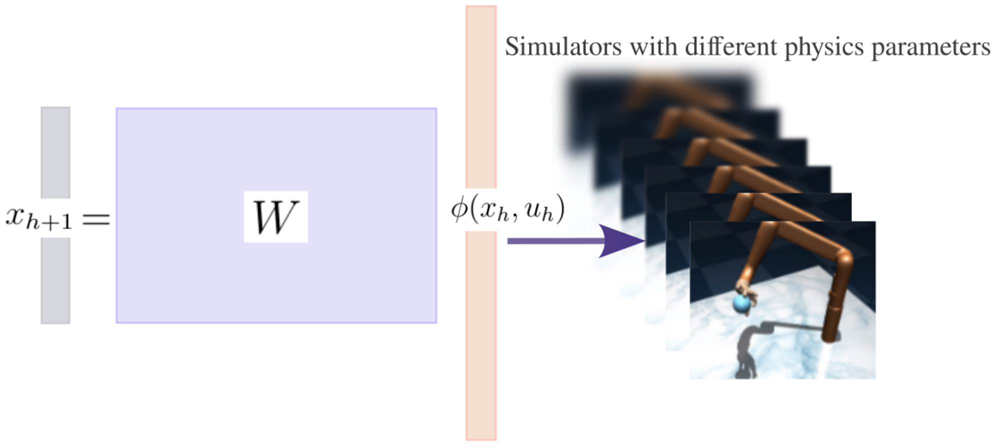


Figure 4.3: Using simulators with different physics parameters as featurizers for practical applications to complex domains.

a single model from the ensemble with the planner is unable to perform the task.

Fig. 4.4 plots the learning curves of LC^3 with different features. We observe that, within 10 attempts at the task, LC^3 with ensemble features is successful, while the same method with RFF features makes no progress. Additionally, we use LC^3 with the top layers of a neural network – trained on a data set of 30 optimized trajectories with the correct model – as our features. It also makes little progress.

We clarify the setting in which this approach may be relevant as follows. Complex dynamics, such as that in the real world, are difficult to represent with function approximation like neural networks or random features. Rather than collect inordinate amounts of data to mimic the combination of features and model, we instead use structured representations of the real world, such as dynamics simulators, to produce features, and use the method in this work to learn the model. Since dynamics simulators represent the current distillation of physics into a computational form and accurate measurement of engineered systems is paramount for the modern world, this instantiation of this method is reasonable.

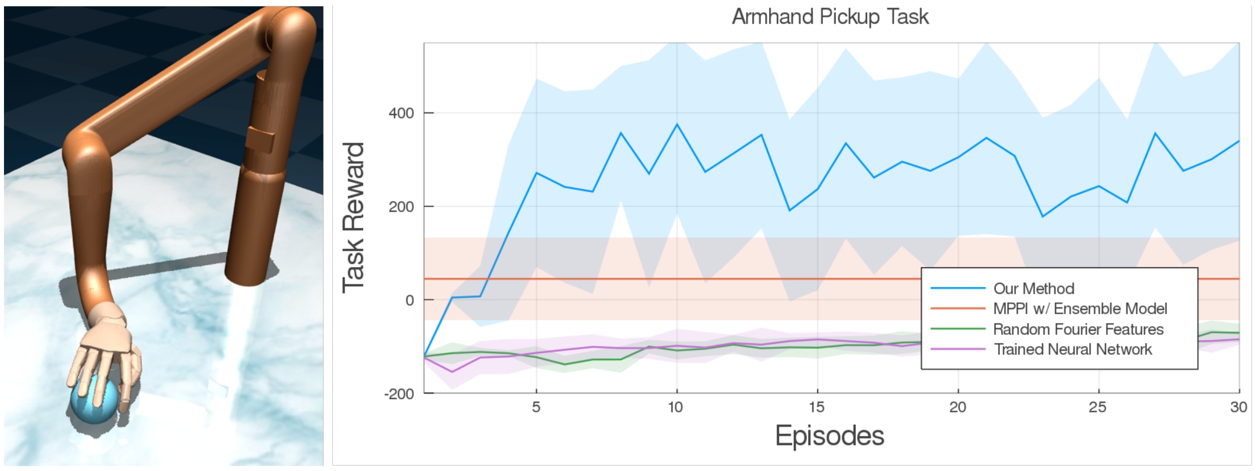


Figure 4.4: Left: An illustration of Armhand environment. Right: Performance curves averaged across 12 random seeds. For reference, we include the average reward of MPPI using a random model from the ensemble. Its score represents the system moving the hand to the object, but unable to grasp and lift it: exactly what we would expect for randomized object dynamics parameters.

4.3 Reframing and generalizing a unique control problem as ML

So far, we have considered an instantaneous (single-step) cost (reward) that accumulates over a certain time horizon as the objective for decision making. In fact, such objectives characterize the most modern RL problems modeled as Markov decision processes to encode tasks of interest. Meanwhile, many dynamic phenomena found in nature are known to be represented as simple trajectories, such as nonlinear oscillators, on low-dimensional manifolds embedded in high-dimensional spaces that we observe [229]. Its mathematical concept is known as *phase reduction* [263, 183], and recently its connection to the Koopman operator has been attracted much attention in response to the growing abundance of measurement data and the lack of known governing equations for many systems of interest [136, 172, 140].

In the latter half of this chapter, a novel paradigm of controlling nonlinear systems based on the spectrum of the Koopman operator is presented. The Koopman operator is used to extract global properties of the dynamics such as its dominating modes and eigenspectrum through spectral decomposition; as such, we propose the *Koopman spectrum cost* as the cost over the Koopman operator of controlled dynamics, defining a preference of the dynamical system in the reduced phase space. Working in the spectrum (or frequency) domain has been standard in the control community (e.g., [19, 98]), and this chapter reframes and generalizes the problem through the adoption of the Koopman operator to create a novel decision making framework within the context of ML as well.

Below, we present our regulator (control) framework, KSNR, with several illustrative numerical examples based on population based policy search, followed by an introduction of its example online learning algorithm (Section 4.3.2) with its theoretical insights on sample complexity and on reduction of the model to that of eigenstructure assignments problem as a special case.

4.3.1 Koopman Spectrum Nonlinear Regulator

In this section, we propose our decision making framework based on the Koopman spectrum regularization followed by simulation examples.

Problem setups

We assume the random dynamical system (RDS) model described in Section 2.1.4, where we focus on the case $\mathbb{T} = \mathbb{N}$, i.e., the discrete-time scenario.

Fix a set $X_0 := \{(x_{0,0}, H_0), (x_{0,1}, H_1), \ldots, (x_{0,N-1}, H_{N-1})\} \subset \mathcal{X} \times \mathbb{Z}_{>0}$, for $N \in \mathbb{Z}_{>0}$, and define $c : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a cost function. The Koopman Spectrum Nonlinear Regulator (KSNR), which we propose, is the following optimization problem:

$$\text{Find} \quad \pi^\star \in \operatorname*{arg\,min}_{\pi \in \Pi} \left\{ \Lambda[\mathscr{K}(\pi)] + J^\pi(X_0; c) \right\}, \tag{4.3.1}$$

where $\Lambda : \mathcal{L}(\mathcal{H}; \mathcal{H}) \to \mathbb{R}_{\geq 0}$ is a mapping that takes a Koopman operator as an input and returns its cost; and

$$J^\pi(X_0; c) := \sum_{n=0}^{N-1} \mathbb{E}_{\Omega_\pi} \left[\sum_{h=0}^{H_n - 1} c(x_{h,n}) \middle| \pi, x_{0,n} \right],$$

where $x_{h,n}(\omega) := \mathcal{F}^{\pi}(h, \omega, x_{0,n})$.

Example 4.3.1 (Examples of Λ)**.** Some of the examples of Λ are:

1. $\Lambda[\mathscr{A}] = \max\{1, \rho(\mathscr{A})\}$, where $\rho(\mathscr{A})$ is the spectral radius of $\mathscr{A}$, prefers stable dynamics.

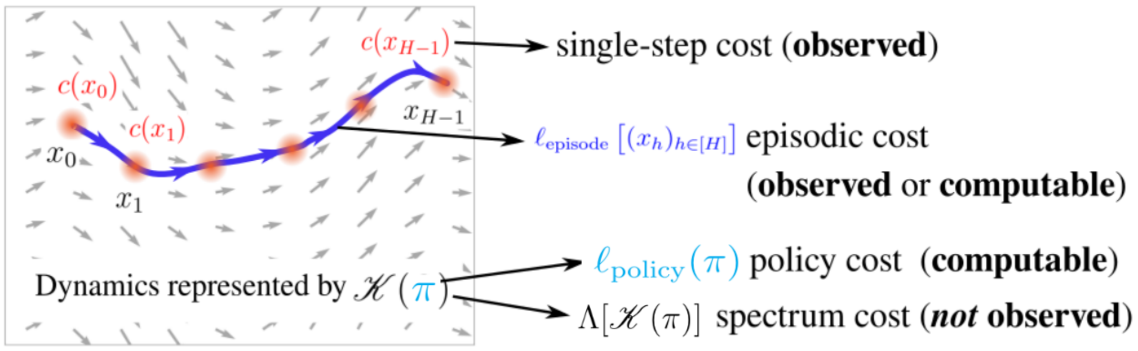
2. $\Lambda[\mathscr{A}] = \ell_{\mathscr{A}^\star}(\mathscr{A})$ can be used for imitation learning, where $\ell_{\mathscr{A}^\star}(\mathscr{A}) : \mathcal{L}(\mathcal{H}; \mathcal{H}) \to \mathbb{R}_{\geq 0}$ is a loss function measuring the gap between $\mathscr{A}$ and the given $\mathscr{A}^\star \in \mathcal{L}(\mathcal{H}; \mathcal{H})$.

3. $\Lambda[\mathscr{A}] = \sum_i |\lambda_i(\mathscr{A})|$, prefers agent behaviors described by fewer dominating modes. Here, $\{\lambda_i\}_{i \in \mathbb{Z}_{>0}}$ is the set of eigenvalues of the operator $\mathscr{A}$ (assuming that the operator has discrete spectrum).

Remark 4.3.1 (Remarks on the Koopman spectrum cost)**.** If the Koopman operator over $\mathcal{H}$ *fully represents* the dynamics in the sense that it can reproduce the dynamical system over the state space, the spectrum cost is viewed as a cost that directly takes the dynamical system itself as an input. On the other hand, depending on the choice of $\mathcal{H}$, it is often the case that the Koopman operator does not uniquely reproduce the dynamics (e.g., the extreme case is a one-dimension space spanned by a constant function). For such cases, the cost only acts as a regularizer. Compared to the MDP formulation, which aims at representing the dynamics by the cumulative cost incurred on local trajectories, which resembles the principle of least action, our KSNR considers global property of the dynamics (i.e., spectrum). We will see more details below.

Useful properties of the Koopman spectrum cost

Since the domain of the Koopman spectrum cost is a set of linear operators that represents global properties of corresponding dynamical system, it is advantageous to employ this cost Λ over sums of single-step costs $c(x)$ when encoding the properties such as stability of the system. To illustrate this perspective, we remark the following.



$c(x_{H-1})$
single-step cost (**observed**)
$c(x_0)$
$c(x_1)$
x_0
x_1
x_{H-1}
$\ell_{\text{episode}}\left[(x_h)_{h\in[H]}\right]$ episodic cost (**observed** or **computable**)
Dynamics represented by $\mathscr{K}(\pi)$
$\ell_{\text{policy}}(\pi)$ policy cost (**computable**)
$\Lambda[\mathscr{K}(\pi)]$ spectrum cost (***not* observed**)

Figure 4.5: Comparisons of several costs for decision making problems. The Koopman spectrum cost is the cost over the global properties of the dynamical system itself which is typically unknown for learning problems, and is unobservable.

Let $\mathcal{X} = \mathbb{R}$, $\upsilon \in (0,1]$, and let $c : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be nonconstant cost function. We consider the following loss function for an RDS $\mathcal{F} : \mathbb{N} \times \Omega \times \mathcal{X} \to \mathcal{X}$:

$$\ell(\mathcal{F}, x) := \mathbb{E}_{\Omega} \sum_{h=0}^{\infty} \upsilon^h c\left(\mathcal{F}(h, \omega, x_0)\right).$$

Now consider the dynamical system $\mathcal{F}(1, \omega, x) = -x$, $\forall x \in \mathcal{X}$, $\forall \omega \in \Omega$, for example. Then, it holds that, for any choice of υ and c, there exists another RDS $\mathcal{G}$ satisfying that for all $x \in \mathcal{X}$,

$$\ell(\mathcal{G}, x) < \ell(\mathcal{F}, x).$$

This fact indicates that there exists a dynamical system that cannot be described by a solution to the above optimization. However, note that, given any deterministic map $f^\star : \mathcal{X} \to \mathcal{X}$, if one employs a (different form of) cost $\mathbf{c} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, where $\mathbf{c}(x, y)$ evaluates to 0 only if $y = f^\star(x)$ and otherwise evaluates to 1, it is straightforward to see that the dynamics $f^\star$ is the one that simultaneously optimizes $\mathbf{c}(x, y)$ for any $x \in \mathcal{X}$. In other words, this form of single-step cost uniquely identifies the (deterministic) dynamics by defining its evaluation at each state.

In contrast, when one wishes to constrain or regularize the dynamics in the (spatio-temporal) spectrum domain, to obtain the set of stable dynamics for example, single-step costs become powerless. To see this, we give the following proposition.

Proposition 4.3.2. *Let $\mathcal{X} = \mathbb{R}$. Consider the set $\mathcal{S}_f$ of dynamics converging to some point in $\mathcal{X}$. Then, for any choice of $\upsilon \in (0,1]$, set-valued map $\mathscr{S} : \mathcal{X} \to 2^{\mathcal{X}} \setminus \emptyset$, and cost $\mathbf{c}$ of the form*

$$\mathbf{c}(x,y) = \begin{cases} 0 & (y \in \mathscr{S}(x)), \\ 1 & (\text{otherwise}), \end{cases}$$

we have $\{\mathcal{F}\} \neq \mathcal{S}_f$. Here, $\{\mathcal{F}\}$ are given by

$$\bigcap_{x_0 \in \mathcal{X}} \left\{ \operatorname*{arg\,min}_{\mathcal{F}:\text{r.d.s.}} \mathbb{E}_{\Omega} \sum_{h=0}^{\infty} \upsilon^h \mathbf{c}\left(\mathcal{F}(h,\omega,x_0), \mathcal{F}(h+1,\omega,x_0)\right) \right\}.$$

Proof. Assume there exist υ, a set-valued map $\mathscr{S}$, and $\mathbf{c}$ such that the set $\{\mathcal{F}^{\pi}\} = \mathcal{S}_f$. Then, because $\mathcal{S}_f$ is strictly smaller than the set of arbitrary dynamics, it must be the case that $\exists x_0 \in \mathcal{X},\ \exists y_0 \in \mathcal{X},\ \ y_0 \notin \mathscr{S}(x_0)$. However, the dynamics $f^{\star}$, where $f^{\star}(x_0) = y_0$ and $f^{\star}(x) = x,\ \forall x \in \mathcal{X} \setminus \{x_0\}$, is an element of $\mathcal{S}_f$. Therefore, the proposition is proved. $\square$

Remark 4.3.3 (Interpretation of Proposition 4.3.2)**.** Proposition 4.3.2 intuitively says that the set of "stable dynamics" cannot be determined by specifying every single transition. This is because, the dynamics that does not follow any pre-specified transition can always be made "stable".

Although aforementioned facts are simple, they give us some insights on the limitation of the use of (cumulative) single-step cost for characterizing dynamical systems. We also illustrate how the spectrum cost differs from other types of costs used in decision making problems in Figure 4.5. The cost is not incurred on a single-step or on one trajectory (episode), but is over a (part of) the global properties of the dynamical system model. It typically corresponds to a policy, but a policy regularization cost used in, for example [94], is computable for the current policy while the spectrum cost is unobservable if the dynamics is unknown (and hence is not directly computable).

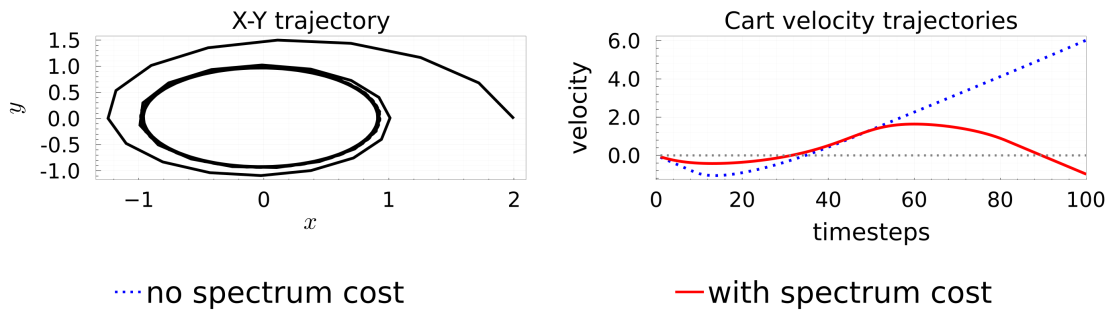


Figure 4.6: Left: We minimize solely for Koopman spectrum cost $\Lambda(\mathscr{A}) = \|\mathbf{m} - \mathbf{m}^\star\|_1$ to imitate the top mode of a reference spectrum to recover a desired limit cycle behavior for the single-integrator system. Right: By regularizing the spectral radius of Cartpole with a cumulative cost that favors high velocity, the cartpole performs a stable oscillation rather than moving off to infinity.

Simulated experiments

We illustrate how one can use the costs in Example 4.3.1. See Appendix A.2 for detailed descriptions and results of the experiments. Throughout, we used Julia language [41] based robotics control package, Lyceum [230], for simulations and visualizations. Also, we use Cross-entropy method (CEM) based policy search ([133]).

Imitating target behaviors through the Koopman operators: We consider the limit cycle dynamics

$$\dot{r} = r(1 - r^2),\ \dot{\theta} = 1,$$

described by the polar coordinates, and find the Koopman operator for this dynamics by sampling transitions, assuming $\mathcal{H}$ is the span of Random Fourier Features (RFFs) [202]. With Π, a space of RFF policies that determine $\dot{r}$ and $\dot{\theta}$ as a single-integrator model, we solve KSNR for the spectrum cost $\Lambda(\mathscr{A}) = \|\mathbf{m} - \mathbf{m}^\star\|_1$, where $\mathbf{m} \in \mathbb{C}^{d_\phi}$ is the top mode (i.e., eigenvector of $\mathscr{A}$ corresponding to the largest absolute eigenvalue) and $\mathbf{m}^\star$ is the top mode of the target Koopman operator found previously. Figure 4.6 (left) plots the trajectory (of

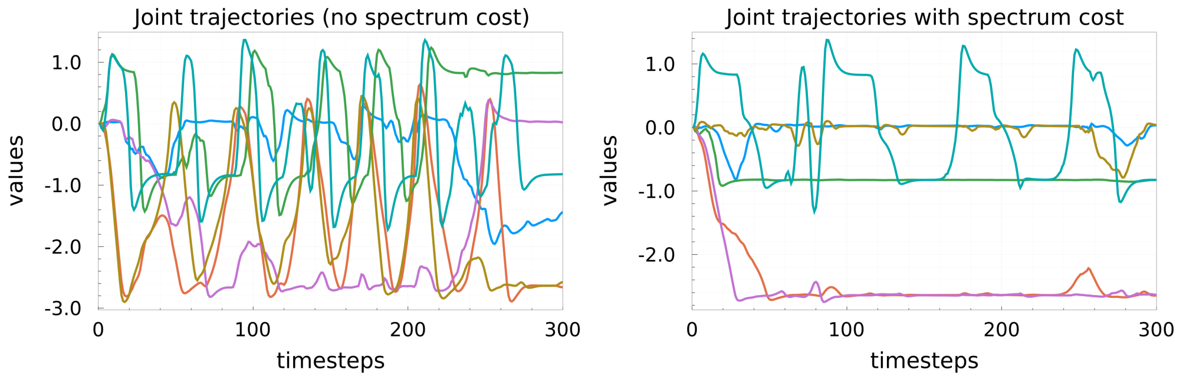


Figure 4.7: The joint angle trajectories generated by a combination of linear and RFF policies. Left: When only cumulative reward is maximized. Right: When both the cumulative cost and the spectrum cost $\Lambda(\mathscr{A}) = 5\sum_{i=1}^{d_\phi} |\lambda_i(\mathscr{A})|$ are used, where the factor 5 is multiplied to balance between the spectrum cost and the cumulative cost.

the Cartesian coordinates) generated by RFF policies that minimize this cost; it is observed that the agent successfully converged to the desired limit cycle of radius one by imitating the dominating mode of the target spectrum.

Generating stable loops (Cartpole): We consider Cartpole environment (where the rail length is extended from the original model). The single-step reward (negative cost) is $10^{-3}|v|$ where v is the velocity of the cart, plus the penalty -1 when the pole falls down (i.e., directing downward). The additional spectrum cost considered in this experiment is $\Lambda(\mathscr{A}) = 10^4 \max(1, \rho(\mathscr{A}))$, which highly penalizes spectral radius larger than one. Figure 4.6 (right) plots the cart velocity trajectories generated by RFF policies that (approximately) solve KSNR with/without the spectrum cost. It is observed that spectral regularization led to a back-and-forth motion while the non-regularized policy preferred accelerating to one direction to solely maximize velocity. When the spectrum cost was used, the cumulative rewards were 0.072 and the spectral radius was 0.990, while they were 0.212 and 1.003 when the spectrum cost was not used; limiting the spectral radius prevents the ever increasing change in position.

Generating smooth movements (Walker): We use the Walker2d environment and compare movements with/without the spectrum cost. The single-step reward (negative cost) is given by $v - 0.001\|a\|^2_{\mathbb{R}^6}$, where v is the velocity and a is the action vector of dimension 6. The spectrum cost is given by $\Lambda(\mathscr{A}) = 5\sum_{i=1}^{d_\phi} |\lambda_i(\mathscr{A})|$, where the factor 5 is multiplied to balance between the spectrum and cumulative cost. Figure 4.7 plots typical joint angles along a trajectory generated by a combination of linear and RFF policies that (approximately) solves KSNR with/without the spectrum cost. It is observed that the spectrum cost led to simpler (i.e., some joint positions converge) and smoother dynamics while doing the task sufficiently well. With the spectrum cost, the cumulative rewards and the spectrum costs averaged across four random seeds (plus-minus standard deviation) were 584 ± 112 and 196 ± 8.13. Without the spectrum cost, they were 698 ± 231 and 310 ± 38.6. We observe that, as expected, the spectrum cost is lower for KSNR while the classical cumulative reward is higher for the behavior generated by the optimization without spectrum cost. We emphasize that we are *not* competing against the MDP counterparts in terms of the cumulative reward, but rather showing an effect of additional spectrum cost. Please also refer to Appendix A.2 for detailed results.

Computation: Throughout, we used the following version of Julia; for each experiment, the running time was less than around 10 minutes.

```
Julia Version 1.5.3
Platform Info:
OS: Linux (x86_64-pc-linux-gnu)
CPU: AMD Ryzen Threadripper 3990X 64-Core Processor
WORD_SIZE: 64
LIBM: libopenlibm
LLVM: libLLVM-9.0.1 (ORCJIT, znver2)
Environment:
JULIA_NUM_THREADS = 12
```

4.3.2 Theoretical algorithm of online KSNR and its insights on the complexity

Next, we present a (theoretical) learning algorithm for online KSNR. Although the KSNR itself is a general regulator framework, we need some structural assumptions to the problem in order to devise a sample efficient (if not computation efficient) algorithm. Despite the unobservability of the spectrum cost, KSNR admits a sample efficient model-based algorithm through operator theoretic arguments under those assumptions. The learning goal is to find a parameter $\pi^{\star t}$ that satisfies (4.3.1) at each episode $t \in [T]$. We employ episodic learning, and let π^t be a parameter employed at the t-th episode. Adversary chooses $X_0^t := \{(x_{0,0}^t, H_0^t), (x_{0,1}^t, H_1^t), \dots, (x_{0,N^t-1}^t, H_{N^t-1}^t)\} \subset \mathcal{X} \times \mathbb{Z}_{>0}$, where $N^t \in \mathbb{Z}_{>0}$, and the cost function c^t at the beginning of each episode. Let $c_{h,n}^t := c^t(x_{h,n}^t)$. $\omega \in \Omega_{\pi^t}$ is chosen according to P_{π^t}.

Algorithm evaluation. We employ the following cumulative regret as the performance metric:

$$\textsc{Regret}_T := \sum_{t=0}^{T-1} \left(\Lambda[\mathscr{K}(\pi^t)] + \sum_{n=0}^{N^t-1} \sum_{h=0}^{H_n^t-1} c_{h,n}^t \right) - \sum_{t=0}^{T-1} \min_{\pi \in \Pi} \left(\Lambda[\mathscr{K}(\pi)] + J^\pi(X_0^t; c^t) \right).$$

Below, we present model assumptions and an algorithm with a regret bound.

Models and algorithms

We make the following modeling assumptions.

Assumption 4.3.4. *Let $\mathscr{K}(\pi)$ be the Koopman operator corresponding to a parameter π. Then, assume that there exists a finite dimensional subspace $\mathcal{H}_0$ on $\mathcal{X}$ over $\mathbb{R}$ and its basis $\phi_1, \dots, \phi_{d_\phi}$ such that the RDS* (2.1.3) *satisfies the following:*

$$\forall \pi \in \Pi,\ \forall x \in \mathcal{X}:\ \phi_i(\mathcal{F}^\pi(1, \omega, x)) = [\mathscr{K}(\pi)\phi_i](x) + \epsilon_i(\omega),$$

where the additive noise $\epsilon_i(\omega) \sim \mathcal{N}(0, \sigma^2)$ is assumed to be independent across time steps, parameters π, and indices i.

Remark 4.3.5 (On Assumption 4.3.4)**.** Although the added noise term is expected to deal with stochasticity and misspecification to some extent in practice, Assumption 4.3.4 is strong to ask for; in fact, claiming existence of the Koopman operator over a useful RKHS (e.g., with Gaussian kernel) is not trivial for most of the practical problems. Studying *misspecified case* with small error margin is an important future direction of research; however, as our regulator problem is novel, we believe this work guides the future attempts of further algorithmic research.

Assumption 4.3.6 (Function-valued RKHS (see Section 2.1.6 for details))**.** *$\mathscr{K}(\cdot)\phi$ for $\phi \in \mathcal{H}$ is assumed to live in a known function valued RKHS with the operator-valued reproducing kernel $K(\cdot,\cdot) : \Pi \times \Pi \to \mathcal{L}(\mathcal{H};\mathcal{H})$, or equivalently, there exists a known map $\Psi : \Pi \to \mathcal{L}(\mathcal{H};\mathcal{H}')$ for a specific Hilbert space $\mathcal{H}'$ satisfying for any $\phi \in \mathcal{H}$, there exists $\psi \in \mathcal{H}'$ such that*

$$\mathscr{K}(\cdot)\phi = \Psi(\cdot)^{\dagger}\psi. \tag{4.3.2}$$

Remark 4.3.7 (On Assumption 4.3.6)**.** In practice, one can use *decomposable kernel* [45]; when the kernel K is given by $K(\pi_1,\pi_2) = k(\pi_1,\pi_2)A$ for some scalar-valued kernel $k(\pi_1,\pi_2)$ and for positive semi-definite operator $A \in \mathcal{L}(\mathcal{H},\mathcal{H})$, the kernel K is called decomposable kernel. For an RKHS of a decomposable kernel K, (4.3.2) becomes

$$\mathscr{K}(\pi)\phi = (\zeta(\pi)^{\dagger} \otimes B)\psi,$$

where $\zeta : \Pi \to \mathcal{H}''$ is known ($\mathcal{H}''$ is some Hilbert space), and $A = BB^{\dagger}$. Further, to use RFFs, one considers a shift-invariant kernel $k(\pi_1,\pi_2)$.

When Assumption 4.3.6 holds, we obtain the following claim which is critical for our learning framework.

Lemma 4.3.8. *Suppose Assumption 4.3.6 holds. Then, there exists a linear operator $M^{\star} : \mathcal{H} \to \mathcal{H}'$ such that*

$$\mathscr{K}(\pi) = \Psi(\pi)^{\dagger} \circ M^{\star}.$$

Proof. It is easy to see that one can define a map M_0 so that it satisfies $M_0(\phi) = \psi$ such that Assumption 4.3.6 holds. Define

$$M^\star := P_{\mathrm{proj}} M_0,$$

where P_{proj} is the orthogonal projection operator onto the sum space of the orthogonal complement of the null space of $\Psi(\pi)^\dagger$ for all $\pi \in \Pi$, namely,

$$\overline{\sum_{\pi\in\Pi} \ker(\Psi(\pi)^\dagger)^\perp}.$$

Also, define $P_{\mathrm{proj},\pi}$ by the orthogonal projection onto

$$\ker(\Psi(\pi)^\dagger)^\perp.$$

Then, for any $\pi \in \Pi$, we obtain

$$P_{\mathrm{proj},\pi} M_0 = {\Psi(\pi)^\dagger}^+ \mathscr{K}(\pi),$$

where $\mathscr{A}^+$ is the pseudoinverse of the operator $\mathscr{A}$, and $P_{\mathrm{proj},\pi} M_0$ is linear. Let $a, b \in \mathbb{R}$ and $\phi_1, \phi_2 \in \mathcal{H}$, and define $\tilde{\psi}$ by

$$\tilde{\psi} := P_{\mathrm{proj}} M_0(a\phi_1 + b\phi_2) - a P_{\mathrm{proj}} M_0(\phi_1) - b P_{\mathrm{proj}} M_0(\phi_2).$$

Because $P_{\mathrm{proj},\pi} P_{\mathrm{proj}} M_0 = P_{\mathrm{proj},\pi} M_0$, it follows that $P_{\mathrm{proj},\pi} \tilde{\psi} = 0$ for all $\pi \in \Pi$, which implies

$$\tilde{\psi} \in \bigcap_{\pi\in\Pi} \ker P_{\mathrm{proj},\pi} = \bigcap_{\pi\in\Pi} \ker(\Psi(\pi)^\dagger).$$

On the other hand, we have

$$\tilde{\psi} \in \overline{\sum_{\pi\in\Pi} \ker(\Psi(\pi)^\dagger)^\perp}.$$

Therefore, it follows that $\tilde{\psi} = 0$, which proves that $M^\star$ is linear. $\square$

In the reminder of this chapter, we work on the invariant subspace $\mathcal{H}_0$ in Assumption 4.3.4 and thus we regard $\mathcal{H} = \mathcal{H}_0 \cong \mathbb{R}^{d_\phi}$, $\mathcal{L}(\mathcal{H};\mathcal{H}) \cong \mathbb{R}^{d_\phi \times d_\phi}$, and, by abuse of notations, we view $\mathscr{K}(\pi)$ as the realization of the operator over $\mathbb{R}^{d_\phi}$, i.e.,

$$\boldsymbol{\phi}_{\mathcal{F}^\pi(1,\omega,x)} = \mathscr{K}(\pi)\boldsymbol{\phi}_x + \epsilon(\omega) = [\Psi(\pi)^\dagger \circ M^\star]\boldsymbol{\phi}_x + \epsilon(\omega),$$

where $\boldsymbol{\phi}_x := [\phi_1(x), \phi_2(x), \ldots, \phi_{d_\phi}(x)]^\top \in \mathbb{R}^{d_\phi}$, and $\epsilon(\omega) := [\epsilon_1(\omega), \epsilon_2(\omega), \ldots, \epsilon_{d_\phi}(\omega)]^\top \in \mathbb{R}^{d_\phi}$.

Finally, we assume the following.

Assumption 4.3.9 (Realizability of costs). *For all t, the single-step cost c^t is known and satisfies $c^t(x) = w^t(\boldsymbol{\phi}_x)$ for some known map $w^t : \mathbb{R}^{d_\phi} \to \mathbb{R}_{\geq 0}$.*

For later use, we define, for all $t \in [T]$, $n \in [N^t]$, and $h \in [H_n^t]$; $\mathscr{A}_{h,n}^t \in \mathcal{L}(\mathcal{L}(\mathcal{H}; \mathcal{H}'); \mathcal{H})$ and $\mathscr{B}^t \in \mathcal{L}(\mathcal{L}(\mathcal{H}; \mathcal{H}'); \mathcal{L}(\mathcal{H}; \mathcal{H}))$ by

$$\mathscr{A}_{h,n}^t(M) = \left[\Psi(\pi^t)^\dagger \circ M\right] \left(\boldsymbol{\phi}_{x_{h,n}^t}\right), \qquad \mathscr{B}^t(M) = \Psi(\pi^t)^\dagger \circ M.$$

Remark 4.3.10 (Hilbert-Schmidt operators). Both $\mathscr{A}_{h,n}^t$ and $\mathscr{B}^t$ are Hilbert-Schmidt operators because the ranges $\mathcal{H}$ and $\mathcal{L}(\mathcal{H}; \mathcal{H})$ are of finite dimension. Note, in case $\mathcal{H}'$ is finite, we obtain

$$\boldsymbol{\phi}_{\mathcal{F}^\pi(1,\omega,x)} = \Psi(\pi)^\dagger M^\star \boldsymbol{\phi}_x + \epsilon(\omega) = (\boldsymbol{\phi}_x^\dagger \otimes \Psi(\pi)^\dagger)\mathrm{vec}(M^\star) + \epsilon(\omega), \tag{4.3.3}$$

where vec is the vectorization of matrix.

With these preparations in mind, our proposed information theoretic algorithm, which is an extension of LC^3 to KSNR problem (estimating the true operator $M^\star$) is summarized in Algorithm 2.[1] This algorithm assumes the following oracle.

Definition 4.3.1 (Optimal parameter oracle). Define the oracle, `OptDynamics`, that computes the minimization problem (4.3.1) for any $\mathscr{K}$, X_0, Λ and c^t satisfying Assumption 4.3.9.

Information theoretic regret bounds

Here, we present the regret bound analysis. To this end, we make the following assumptions.

[1]See Appendix B.2.1 for the definitions of the values in this algorithm.

Algorithm 2 Koopman-Spectrum LC^3 (KS-LC^3)

```
Require: Parameter set $\Pi$; regularizer $\lambda$
1: Initialize $\textsc{Ball}_M^0$ to be a set containing $M^\star$.
2: for $t = 0 \dots T-1$ do
3:     Adversary chooses $X_0^t$.
4:     $\pi^t, \hat{M}_t = \arg\min_{\pi\in\Pi,\ M\in\textsc{Ball}_M^t} \Lambda[\Psi(\pi)^\dagger \circ M] + J^\pi(X_0^t; M; c^t)$
5:     Under the dynamics $\mathcal{F}^{\pi^t}$, sample transition data $\tau^t := \{\tau_n^t\}_{n=0}^{N^t-1}$, where $\tau_n^t := \{x_{h,n}^t\}_{h=0}^{H_n^t}$
6:     Update $\textsc{Ball}_M^{t+1}$.
7: end for
```

Assumption 4.3.11. *The operator Λ satisfies the following (modified) Hölder condition:*

$$\begin{aligned}&\exists L \in \mathbb{R}_{\geq 0},\ \exists \alpha \in (0,1],\ \forall \mathscr{A} \in \mathcal{L}(\mathcal{H},\mathcal{H}),\ \forall \pi \in \Pi,\\ &|\Lambda[\mathscr{A}] - \Lambda[\mathscr{K}(\pi)]| \leq L \cdot \max\left\{\|\mathscr{A} - \mathscr{K}(\pi)\|^2, \|\mathscr{A} - \mathscr{K}(\pi)\|^\alpha\right\}.\end{aligned}$$

Further, we assume there exists a constant $\Lambda_{\max} \geq 0$ such that, for any $\pi \in \Pi$ and for any $M \in \textsc{Ball}_M^0$,

$$\left|\Lambda[\Psi(\pi)^\dagger \circ M]\right| \leq \Lambda_{\max}.$$

For example, for spectral radius ρ, the following proposition holds using the result from [227, Corollary 2.3].

Proposition 4.3.12. *Assume there exists a constant $\Lambda_{\max} \geq 0$ such that, for any $\pi \in \Pi$ and for any $M \in \textsc{Ball}_M^0$, $\rho(\Psi(\pi)^\dagger \circ M) \leq \Lambda_{\max}$. Let the Jordan condition number of $\mathscr{K}(\pi)$ be the following:*

$$\kappa := \sup_{\pi\in\Pi} \inf_{\mathcal{Q}(\pi)} \left\{\|\mathcal{Q}(\pi)\|\|\mathcal{Q}(\pi)^{-1}\| : \mathcal{Q}(\pi)^{-1}\mathscr{K}(\pi)\mathcal{Q}(\pi) = \mathscr{J}(\pi)\right\},$$

where $\mathscr{J}(\pi)$ is a Jordan canonical form of $\mathscr{K}(\pi)$ transformed by a nonsingular matrix $\mathcal{Q}(\pi)$. Also, let m be the order of the largest Jordan block. Then, if $\kappa < \infty$, the cost $\Lambda(\mathscr{A}) = \rho(\mathscr{A})$

satisfies the Assumption 4.3.11 for

$$L := (1+\kappa)d_\phi^2(1+\sqrt{d_\phi - 1}), \qquad \alpha = \frac{1}{m}.$$

Proof. Fix $\pi \in \Pi$ and suppose that the eigenvalues λ_i ($i \in \mathcal{I} := \{1, 2, \ldots, d_\phi\}$) of $\mathscr{K}(\pi)$ are in descending order according to their absolute values, i.e., $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_{d_\phi}|$. Given $\mathscr{A} \in \mathcal{L}(\mathcal{H};\mathcal{H})$, suppose also that the eigenvalues ν_i ($i \in \mathcal{I}$) of $\mathscr{A}$ are in descending order according to their absolute values. Let

$$\kappa_\pi := \inf_{\mathcal{Q}(\pi)} \left\{ \|\mathcal{Q}(\pi)\| \|\mathcal{Q}(\pi)^{-1}\| : \mathcal{Q}(\pi)^{-1}\mathscr{K}(\pi)\mathcal{Q}(\pi) = \mathscr{J}(\pi) \right\}.$$

Also, let

$$\overline{\kappa}_\pi := \max_{m \in \{1,\ldots,d_\phi\}} \left\{ \kappa_\pi^{\frac{1}{m}} \right\}.$$

Then, by [227, Corollary 2.3], we have

$$\begin{aligned} ||\nu_i| - |\lambda_i|| &\leq |\nu_i - \lambda_i| \leq \sum_i |\nu_i - \lambda_i| \leq \sum_i |\nu_{\varsigma(i)} - \lambda_i| \\ &\leq d_\phi \sqrt{d_\phi}(1+\sqrt{d_\phi - p}) \max\left\{ \kappa_\pi \sqrt{d_\phi} \|\mathscr{A} - \mathscr{K}(\pi)\|, (\kappa_\pi \sqrt{d_\phi})^{\frac{1}{m}} \|\mathscr{A} - \mathscr{K}(\pi)\|^{\frac{1}{m}} \right\}, \end{aligned}$$

for any $i \in \mathcal{I}$ and for some permutation ς, where $p \in \{1, 2, \ldots, d_\phi\}$ is the number of the Jordan block of $\mathcal{Q}(\pi)^{-1}\mathscr{K}(\pi)\mathcal{Q}(\pi)$ and $m \in \{1, 2, \ldots, d_\phi\}$ is the order of the largest Jordan block. Because $\sqrt{d_\phi - p} \leq \sqrt{d_\phi - 1}$ for any $p \in \{1, 2, \ldots, d_\phi\}$, and because

$$\max_{m \in \{1,\ldots,d_\phi\}} \left[\sqrt{d_\phi} \cdot \max\{\sqrt{d_\phi}, (\sqrt{d_\phi})^{\frac{1}{m}}\} \cdot \max\left\{ \kappa_\pi, \kappa_\pi^{\frac{1}{m}} \right\} \right] \leq d_\phi \overline{\kappa}_\pi,$$

it follows that

$$\begin{aligned} &|\rho(\mathscr{A}) - \rho(\mathscr{K}(\pi))| = ||\nu_1| - |\lambda_1|| \\ &\leq \overline{\kappa}_\pi d_\phi^2 (1+\sqrt{d_\phi - 1}) \max\left\{ \|\mathscr{A} - \mathscr{K}(\pi)\|, \|\mathscr{A} - \mathscr{K}(\pi)\|^{\frac{1}{m}} \right\}. \end{aligned}$$

Since $\overline{\kappa}_\pi < 1 + \kappa$, simple computations complete the proof. $\square$

Note when $\mathscr{K}(\pi)$ is diagonalizable for all π, $\alpha = 1$.

Assumption 4.3.13. *For every t, adversary chooses N^t trajectories such that $\{\boldsymbol{\phi}_{x^t_{0,n}}\}$ satisfies that the smallest eigenvalue of $\sum_{n=0}^{N^t-1} \boldsymbol{\phi}_{x^t_{0,n}} \boldsymbol{\phi}^\dagger_{x^t_{0,n}}$ is bounded below by some constant $C > 0$. Also, there exists a constant $H \in \mathbb{Z}_{>0}$ such that for every t, $\sum_{n=0}^{N^t-1} H^t_n \leq H$.*

Remark 4.3.14 (On Assumption 4.3.13)**.** In practice, user may wait to end an episode until a sufficient number of trajectories is collected in order for the assumption to hold. Note, this assumption does not preclude the necessity of exploration because the smallest eigenvalue of $\sum_{n=0}^{N^t-1} \sum_{h=0}^{H^t_n-1} \mathscr{A}^{t\ \dagger}_{h,n} \mathscr{A}^t_{h,n}$ is in general not bounded below by a positive constant. We hope that this assumption can be relaxed by assuming the bounds on the norms of $\boldsymbol{\phi}_{x_{0,n}}$ and $\mathscr{K}(\pi)$ and by using matrix Bernstein inequality under additive Gaussian noise assumption.

Lastly, the following assumption is the modified version of Assumption 4.2.3.

Assumption 4.3.15. *[Modified version of Assumption 4.2.3] Assume there exists a constant $V_{\max} > 0$ such that, for every t,*

$$\sup_{\pi \in \Pi} \sum_{n=0}^{N^t-1} \mathbb{E}_{\Omega_\pi} \left[\left(\sum_{h=0}^{H^t_n-1} c^t(x^t_{h,n}) \right)^2 \middle| \, \pi, x^t_{0,n} \right] \leq V_{\max}.$$

Theorem 4.3.16. *Suppose Assumption 4.3.4 to 4.3.15 hold. Set $\lambda = \frac{\sigma^2}{\|M^\star\|^2}$. Then, there exists an absolute constant C_1 such that, for all T, KS-LC^3 (Algorithm 2) using* `OptDynamics` *enjoys the following regret bound:*

$$\mathbb{E}_{\text{KS-LC}^3} [\text{REGRET}_T] \leq C_1 (\tilde{d}_{T,1} + \tilde{d}_{T,2}) T^{1-\frac{\alpha}{2}},$$

where

$$\tilde{d}^2_{T,1} := (1 + \gamma_{T,\mathscr{B}}) \left[L^2 (1 + C^{-1})^2 \tilde{\beta}_{2,T} + (L^2 + \Lambda_{\max} L)(1 + C^{-1}) \tilde{\beta}_{1,T} + \Lambda^2_{\max} + L^2 \right],$$

$$\tilde{d}^2_{T,2} := \gamma_{T,\mathscr{A}} H V_{\max} \left(\gamma_{T,\mathscr{A}} + d_\phi + \log(T) + H \right),$$

$$\tilde{\beta}_{1,T} := \sigma^2 (d_\phi + \log(T) + \gamma_{T,\mathscr{A}}),$$

$$\tilde{\beta}_{2,T} := \sigma^4 ((d_\phi + \log(T) + \gamma_{T,\mathscr{A}})^2 + \gamma_{2,T,\mathscr{A}}).$$

Here, $\gamma_{T,\mathscr{A}}$, $\gamma_{2,T,\mathscr{A}}$, and $\gamma_{T,\mathscr{B}}$ are the expected maximum information gains that scale (poly-)logarithmically with T under practical settings (see Appendix B.2.1 for details).

Remark 4.3.17. We note that, when $\alpha = 1$, we obtain the order of $\tilde{O}(\sqrt{T})$.

The proof techniques include our *positive operator norm bounding lemma* (see Lemma B.2.2 in Appendix B.2), which is another crucial operator theoretic lemma in addition to Lemma 4.3.8.

4.3.3 Relations to the kernelized nonlinear regulator and to eigenstructure assignments

As mentioned in Remark 4.3.5, for a given dynamics described by the system model studied in the KNR (i.e., the transition map from the current state and control to the next state is in a known RKHS), the existence of a *useful* (finite dimensional) space $\mathcal{H}_0$ in Assumption 4.3.4 is in general not guaranteed. As such, the system model considered in this section is no more general than that of the KNR (although our spectrum cost formulation is indeed a generalization of that of the KNR). Now, we consider the finite dimensional description (see (4.3.3)) for simplicity. The equation (4.3.3) can be rewritten by

$$\boldsymbol{\phi}_{\mathcal{F}^{\pi}(1,\omega,x)} = \left(I \otimes \mathrm{vec}(M^{\star})^{\top}\right) \mathrm{vec}\left[\left(\boldsymbol{\phi}_x^{\dagger} \otimes \Psi(\pi)^{\dagger}\right)^{\top}\right] + \epsilon(\omega),$$

and is a special case of the system model of the KNR.

We see that the model associated with our spectrum cost formulation reduces to the eigenstructure assignment problem for the linear systems described by

$$x_{h+1} = Ax_h + Bu_h + \epsilon, \quad A \in \mathbb{R}^{d_x \times d_x}, \ B \in \mathbb{R}^{d_x \times d_u}, \ \epsilon \sim \mathcal{N}(0, \sigma^2 I),$$

where $u \in R^{d_u}$ is a control input. In particular, considering the feedback policy in the form of $u_h = Kx_h$ where $K \in \mathbb{R}^{d_u \times d_x}$ (or $\pi = K$ in this case), the system becomes $x_{h+1} = (A + BK)x_h + \epsilon$. To see how our system model studied in (4.3.3) reduces to linear system case, take $\mathbb{R}^{d_x}$ as $\mathcal{H}_0$ with the canonical basis (i.e., $\boldsymbol{\phi}_x = x$) and let

$$\Phi(\pi) = [I_{d_x} \otimes k_1^{\top}, I_{d_x} \otimes k_2^{\top}, \ldots, I_{d_x} \otimes k_{d_x}^{\top}, I_{d_x}]^{\top},$$

where the feedback matrix is given by $K := [k_1, k_2, \dots, k_{d_x}]$, and let

$$M^* = \left[[\boldsymbol{b}_1, a_1]^\top, \dots, [\boldsymbol{b}_{d_x}, a_{d_x}]^\top\right],$$

where the entries of the row vector $\boldsymbol{b}_i$ are all zero except for the entries from the index $(i-1)d_x d_u + 1$ to the index $i d_x d_u$ given by vec $\left(B^\top\right)$, and $A = [a_1^\top, a_2^\top, \dots, a_{d_x}^\top]$.

The spectrum cost may be designed not only to constrain the eigenstructure but also to balance with the cumulative single-step cost in our framework as well.

4.4 Chapter summary and discussion

This chapter first introduced our MBRL algorithm LC^3 tailored for continuous control problems. LC^3 attains $O(\sqrt{T})$ regret bounds, where we utilize a continuity property of control problems, an optimal control oracle, and number of analysis concepts from RL.

We did not discuss the role of model misspecification; in fact, for complicated (discontinuous) systems such as contact-rich dynamics, our model assumptions may fail. Especially, Gaussian smoothing was one of the key technical novelties in the regret analysis of LC^3; and it is unclear if more general distributions of noise still admit sublinear regret with some conditions and algorithms.

Moreover, in this chapter we assumed the RKHS is known; however, it is usually the case that we have no access to reasonable feature representations in practice and incorporating representation learning is practically important.

The latter half of this chapter proposed a novel paradigm of regulating dynamical systems, which we refer to as Koopman Spectrum Nonlinear Regulator, and presented an information theoretic algorithm that achieves a sublinear regret bound under model assumptions. We showed that behaviors such as limit cycles of interest, stable motion, and smooth movements are effectively realized within our framework, which is an effective generalization of classical eigenstructure (pole) assignments.

When the dynamical systems of interest do not meet certain conditions, Koopman operators might not be well defined over functional spaces that can be practically managed (e.g., RKHSs). Studying such conditions is an on-going research topic. KS-LC^3 also requires several assumptions for tractable regret analysis. It is again somewhat strong to assume that one can find a Koopman invariant space $\mathcal{H}$. Further, noise effects should in practice be exaggerated by lifting to a space of observables, which implies the necessity of studying robustness. To solve KSNR on the other hand, one needs heuristic approximations when the (policy) space or the state space is continuous; therefore, a better heuristic algorithm that is suited to our problem in order to scale the methodology to more complicated domains should be studied.

Chapter 5

TRAJECTORY-BASED OPTIMIZATION FOR CONTROL AND LEARNING TASKS

Chapter's key takeaways

This chapter proposes a dynamic programming based novel decision making framework that exploits the algebraic property of the path signatures, which are the powerful representation (infinite collection of coefficients) of paths that efficiently capture the paths' analytic and geometric characteristics. The resulting update rule, which we refer to as Chen equation, maintains *S-function* that returns the (truncated) signature of the future path (*path-to-go*) w.r.t. the current policy and action input. Chen equation subsumes the update of value (i.e., the classical Bellman update), and its application to control problems through minimization of the cost over signatures effectively generalizes integral controls and hence is robust against an unknown disturbance. Our signature control is applied to trajectory following problems in robotics simulation showing better tracking accuracy and robustness even when waypoint information is unavailable.

5.1 Introduction

Path tracking has been a central problem for autonomous vehicles (e.g., [217, 11]), imitation learning, learning from demonstrations (cf. [106, 22]), character animations (with mocap systems; e.g., [198]), robot manipulation for executing plans returned by a solver (cf. [117, 89]), and for flying drones (e.g., [273]), just to name a few.

Typically, path tracking is dealt with by using reference dynamics or is formulated as a control problem with a sequence of goals to follow using optimal controls based on dynamic programming (DP) over cumulative costs (rewards) or *value* [35]. However, for path tracking where reference dynamics or appropriate waypoints are unavailable, both model-based and value-based (model-free) approaches become inapplicable without introducing heuristic or ad hoc techniques to the problems, hindering systematic procedures.

In this chapter, we adopt a rough-path theoretical approach in DP; specifically, we exploit path signatures (cf. [63, 162]), which have recently attracted the attention of the ML community (e.g., [63, 127, 214, 181, 147, 83]). Our decision making framework predicated on signatures, named signature control, describes an evolution of signatures over an agent's trajectory through DP (see Figure 5.1). By demonstrating how it reduces to the Bellman equation as a special case, we show that the S*-function* representing the signatures of the future path (we call it *path-to-go* in this thesis) is cast as an effective generalization of value function. In addition, since an S-function naturally encodes information of a long trajectory, it is robust against misspecification of dynamics. Our signature control inherits some of the properties of signatures (cf. [163, 43]); as such, when applied to tracking problems, there is no need to specify waypoints.

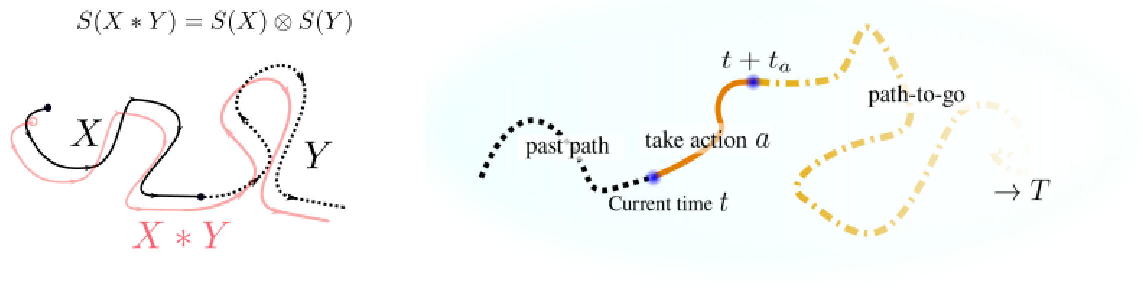


Figure 5.1: Left: Chen's identity, i.e., the signature of the entire path is expressed by the tensor product of the signature of its subpath and that of the rest. Right: Illustration of path-to-go formulation as an analogy to value-to-go in the classical settings.

Contributions: The contributions of this chapter are five folds: (1) devise a novel framework based on path signatures for control problems named signature control, (2) define *Chen* *equation*, a DP based update rule of signatures, and show how it reduces to the Bellman equation as a special instance, (3) propose its model predictive control algorithm and discuss the relation to classical integral control, (4) present several control and robotics applications showcasing the benefits of our framework and demonstrating the concepts that inherit mathematical properties of the path signature, (5) and analyze some of the properties of our framework.

5.2 Problem setups

Since we are interested in cost definitions over the entire path and not limited to the form of the cumulative cost over a trajectory with fixed time interval (or integral along continuous time), MDPs are no longer the most suitable representation. Instead, we assume that the system dynamics of an agent is described by an RDS $\mathcal{F}^{\pi}$ defined by a policy parameter $\pi \in \Pi$ (it does not have to be a map from a state to an action). For simplicity, we assume in this chapter that the RDS generated by a policy $\pi \in \Pi$ shares the same sample space Ω ($\Omega_{\pi} = \Omega$ for all π). Roughly speaking, this means that the *noise mechanism* of RDSs is the same for all policies (not necessarily the same probability distribution). Also, the action $a \in \mathcal{A}$ is for constraining the event of downstream trajectories of RDS to be of some subset of Ω, which we define $\Omega_a \subset \Omega$. Further, we suppose that $\bigsqcup_{a \in \mathcal{A}} \Omega_a = \Omega$.

Our main problem of interest is path tracking which we formally define below:

Definition 5.2.1 (Path tracking)**.** Let $\Gamma : \Sigma \times \Sigma \to \mathbb{R}_{\geq 0}$ be a cost function on the product of the spaces of paths over the nonnegative real number, satisfying:

$$\forall \sigma \in \Sigma : \ \Gamma(\sigma, \sigma) = 0; \qquad \forall \sigma^{*} \in \Sigma, \ \forall \sigma \in \Sigma \text{ s.t. } \sigma \not\equiv_{\sigma} \sigma^{*} : \ \Gamma(\sigma, \sigma^{*}) > 0,$$

where $\equiv_{\sigma}$ is any equivalence relation of path in Σ. Given a reference path $\sigma^{*} \in \Sigma$ and a cost, the goal of path tracking is to find a policy $\pi^{*} \in \Pi$ such that a path $\sigma_{\pi^{*}}$ generated by the RDS $\mathcal{F}^{\pi^{*}}$ satisfies

$$\Gamma(\sigma_{\pi^{*}}, \sigma^{*}) = \min_{\pi \in \Pi} \Gamma(\sigma_{\pi}, \sigma^{*}).$$

In particular, we define σ_π carefully below. Let $T \in \mathbb{T}$ be a time horizon, and define the map $\sigma_{\pi,F} : \mathcal{X} \times [0,T] \times \Omega \to \mathcal{X}^{[0,T]}$ by

$$\begin{aligned}&[\sigma_{\pi,F}(x_0,T,\omega)](t)\\ &= \begin{cases} \mathcal{F}^\pi(t,\omega,x_0) & (\forall t \in \mathbb{T} \cap [0,T]) \\ [F(\mathcal{F}^\pi(\lfloor t \rfloor,\omega,x_0), \mathcal{F}^\pi(\lfloor t \rfloor + 1,\omega,x_0))](t - \lfloor t \rfloor) & (\forall t \in [0,T] \setminus \mathbb{T}) \end{cases},\end{aligned}$$

where $F : \mathcal{X} \times \mathcal{X} \to \mathcal{X}^{[0,1]}$ is an interpolation between two given points.

Also, let *practical* partition $\mathcal{D} = \{0 = t_0 < t_1 < \ldots < t_{k-1} = T\}$ of the time interval $[0,T]$ be such that there exists a sequence of actions $\{a_1, a_2, \ldots\}$ over that partition, i.e., t_0, t_1, t_2... represent 0, t_{a_1}, $t_{a_1} + t_{a_2}$,....

Now, let $\mathcal{T} : \mathcal{X}^{[0,T]} \to \Sigma$ be some (possibly nonlinear) transformation such that, for any practical partition $\mathcal{D}$ of the time interval $[0,T]$, for any $j \in [k-2]$, a pair of feasible paths $\sigma_1 \in \mathcal{X}^{[t_j,t_{j+1}]}, \sigma_2 \in \mathcal{X}^{[t_{j+1},t_{j+2}]}$ satisfies

$$\sigma \equiv_\sigma \sigma_1 * \sigma_2 \Longrightarrow \mathcal{T}(\sigma) \equiv_\sigma \mathcal{T}(\sigma_1) * \mathcal{T}(\sigma_2),$$

where $*$ denotes the concatenation of paths (defined by shifting the starting time and the state) and $\equiv_\sigma$ is the tree-like equivalence relation. The interpolation F and the transformation $\mathcal{T}$ are illustrated in Figure 5.2.

With these definitions in place, we propose our control framework, signature control.

5.3 Signature control

The signature-based optimal control problem reads

Problem 5.3.1.

$$\text{Find } \pi^* \text{ s.t. } \pi^* \in \operatorname*{arg\,min}_{\pi \in \Pi} c\left(\mathbb{E}_\Omega\left[S\left(\sigma^{\mathcal{T}}_{\pi,F}(x_0,T,\omega)\right)\right]\right),$$

where $c : T((\mathcal{X})) \to \mathbb{R}_{\geq 0}$ is a cost function on the space of formal power series, and $\sigma^{\mathcal{T}}_{\pi,F} := \mathcal{T} \circ \sigma_{\pi,F}$; we use, for simplicity, σ_π instead of $\sigma^{\mathcal{T}}_{\pi,F}$ when F and $\mathcal{T}$ are clear in the contexts.

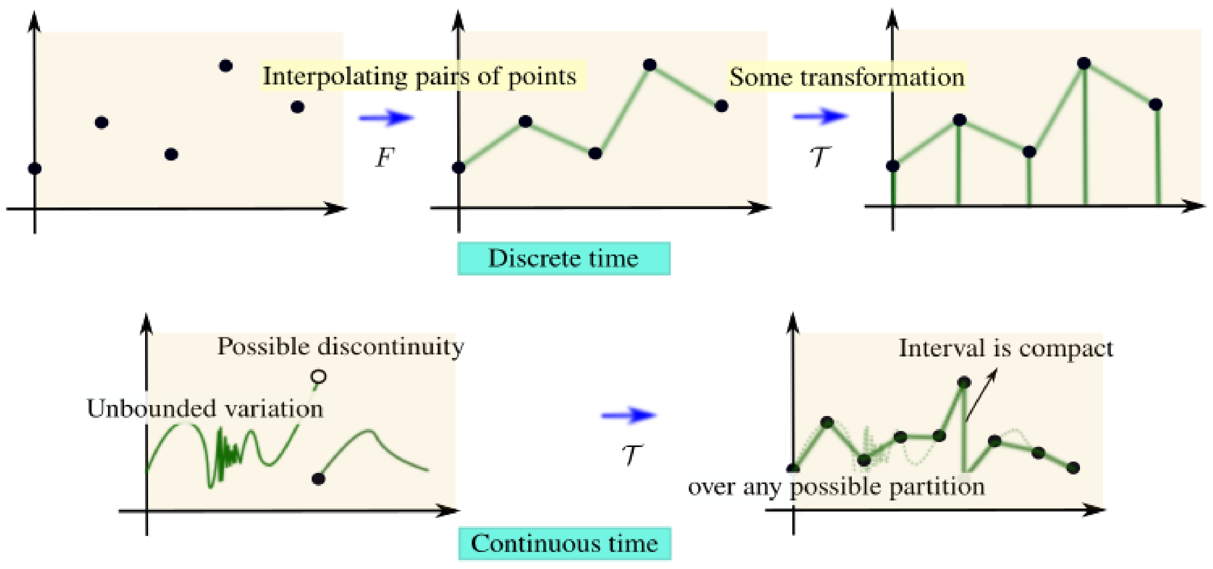


Figure 5.2: Top (discrete-time): Interpolation for pairs of points will produce a path and one may possibly transform them. Down (continuous-time): A path generated by an RDS may include discontinuity or unbounded variation. Transformation makes it to be a path for which the signatures are defined. Discontinuous points may be interpolated (a path is defined over a compact interval), and unbounded variations may be overcome by down-sampling over points of any practical partition.

Remark 5.3.2 (Remark on Problem 5.3.1)**.** This problem subsumes the MDP as we shall see in Section 5.3.1. The given formulation covers both discrete-time and continuous-time cases through interpolation over time. Given a reference σ^*, when $c(S(\sigma)) = \Gamma(\sigma, \sigma^*)$ and $\equiv_\sigma$ denotes tree-like equivalence, signature control becomes the path tracking Problem 5.2.1.

To effectively solve it we exploit DP.

5.3.1 Dynamic programming over signatures

Path-to-go

Let $a \in \mathcal{A}$ be an *action* which basically constrains the realizations of path up to time t_a (actions studied in MDPs constrain the one-step dynamics from a given state). Let the projection on $\mathcal{A} \times \mathcal{B}$ over $\mathcal{A}$ is denoted by $P_{\mathcal{A}} : (a, b) \mapsto a$. Without loss of generality, we assume that $\mathcal{X} = \mathcal{Y} \times \mathcal{O}$, and that $P_{\mathcal{O}}[\mathcal{F}^\pi(t, \omega, x)]$ is known when $P_{\mathcal{Y}}[x]$, t, and ω are given (i.e., $\mathcal{O}$ is the space of observations). Given $T \in \mathbb{T}$, define the *path-to-go* function $\mathcal{P}^\pi$ on $\mathcal{Y} \times \mathbb{T}$ over Σ^Ω by

$$\mathcal{P}^\pi(y, t)(\omega) = \sigma_\pi(x, T - t, \omega), \ \forall t \in [0, T].$$

We use the following Markov property.

Assumption 5.3.3 (Markov property [24])**.** *For each action $a \in \mathcal{A}$, there exists $t_a \geq 0$ such that the RDS $\mathcal{F}^\pi$ satisfies the Markov property, i.e., for each $B \in 2^{\mathcal{X}}$, $a \in \mathcal{A}$, and $s \geq 0$,*

$$\Pr\left[\mathcal{F}^\pi(t_a + s, \omega, z) \in B | \mathcal{F}^\pi(t_a, \omega, z) = x, \omega \in \Omega_a\right] = \Pr\left[\omega | \mathcal{F}^\pi(s, \omega, x) \in B\right]. \tag{5.3.1}$$

Remark 5.3.4. When $\mu(\Omega_a) = 0$, we can still assign the probability of the right hand side of (5.3.1) to its left hand side; however, one can define arbitrarily the probability of a future path conditioned on $\omega \in \Omega_a$ and it does not harm the current arguments for now.

Now, the path-to-go formulation is expressed by

$$\mathcal{P}^\pi(y, t)(\omega) = \mathcal{P}^\pi_a(y, t)(\omega) * \mathcal{P}^\pi(y^+, t + t_a)(\theta_{t_a}\omega),$$

where

$$\mathcal{P}^\pi_a(y, t)(\omega) := \sigma_\pi(x, \min\{T - t, t_a\}, \omega), \quad y^+ = P_{\mathcal{Y}}\mathcal{F}^\pi(t_a, \omega, x).$$

To express this in the signature form, we exploit the Chen's identity, and define the *signature-to-go* function (or in short S-function):

$$\mathcal{S}^\pi(a, y, t) := \mathbb{E}\left[S(\mathcal{P}^\pi(y, t)(\omega)) | a\right]. \tag{5.3.2}$$

Then, we obtain the following update rule:

Theorem 5.3.5 (Signature Dynamic Programming for Decision Making)**.** *Let the function $\mathcal{S}$ be defined by* (5.3.2)*. Under the Markov assumption 5.3.3, it follows that*

$$\mathcal{S}^{\pi}(a, y, t) = \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y, t)(\omega)) \otimes \mathcal{ES}^{\pi}(y^{+}, t + t_a)|a\right]$$

where the expected S-function $\mathcal{ES}^{\pi} : \mathcal{Y} \times \mathbb{T} \to T((\mathcal{X}))$ *is defined by*

$$\mathcal{ES}^{\pi}(y, t) := \mathbb{E}_{\Omega}\left[\mathcal{S}^{\pi}(b(\omega), y, t)\right],$$

and $b : \Omega \to \mathcal{A}$ is defined by $b(\omega) = a$ for $\omega \in \Omega_a$.

Proof. Using the Chen's identity (first equality), tower rule (second equality), Assumption 5.3.3 (third and forth equalities), and the properties of tensor product and the transformation (first and third equalities) we obtain

$$\begin{aligned}
&\mathcal{S}^{\pi}(a, y, t) \\
&= \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y, t)(\omega)) \otimes A|\omega \in \Omega_a\right] = \mathbb{E}\left[\mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y, t)(\omega)) \otimes A|\mathcal{P}_a^{\pi}(y, t)(\omega), \omega \in \Omega_a\right]|\omega \in \Omega_a\right] \\
&= \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y, t)(\omega)) \otimes \mathbb{E}\left[A|\mathcal{P}_a^{\pi}(y, t)(\omega)\right]|\omega \in \Omega_a\right] \\
&= \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y, t)(\omega)) \otimes \mathcal{ES}^{\pi}(y^{+}, t + t_a)|\omega \in \Omega_a\right]
\end{aligned}$$

where

$$A := S(\mathcal{P}^{\pi}(y^{+}, t + t_a)(\theta_{t_a}\omega)).$$

□

Throughout, we call the above update rule as *Chen (expectation) equation.*

Optimality: Next, we briefly cover optimality; i.e., we present Chen optimality equation. Optimality is tricky for Chen formulation because some relation between policy and action is required in addition to the Markov assumption. To obtain our Chen optimality, we make the following assumption.

Assumption 5.3.6 (Relations between policy and action)**.** *For any policy $\pi \in \Pi$, state $x \in \mathcal{X}$, time $t \in \mathbb{T} \cap (0, T]$, and an action $a \in \mathcal{A}$, there exists a policy $\pi' \in \Pi$ such that*

$$\begin{aligned} &\mathbb{E}_\Omega \left[\mathcal{S}^{\pi'}(b(\omega), P_\mathcal{Y}(x_0), 0) \right] \\ &= \mathbb{E}_\Omega \left[\mathcal{S}^{\pi}(b(\omega), P_\mathcal{Y}(x_0), 0) \middle| (\mathcal{F}^\pi(t, \omega, x_0) = x) \Longrightarrow (\theta_t \omega \in \Omega_a) \right]. \end{aligned}$$

Also, there exists $a \in \mathcal{A}$ such that $\pi' = \pi$.

Given a cost function $c : T((\mathcal{X})) \to \mathbb{R}_{\geq 0}$, suppose π^* satisfies

$$c\left(\mathcal{E}\mathcal{S}^{\pi^*}(P_\mathcal{Y}(x_0), 0)\right) = \inf_{\pi \in \Pi} c\left(\mathcal{E}\mathcal{S}^{\pi}(P_\mathcal{Y}(x_0), 0)\right).$$

Then, under Assumption 5.3.6, Chen optimality reads

$$\begin{aligned} &c\left(\mathbb{E}_\Omega\left[\mathcal{S}^{\pi^*}(b(\omega), P_\mathcal{Y}(x_0), 0)\right]\right) \\ &= \min_{a \in \mathcal{A}} c\left(\mathbb{E}_\Omega\left[\mathcal{S}^{\pi^*}(b(\omega), P_\mathcal{Y}(x_0), 0) \middle| \left(\mathcal{F}^{\pi^*}(t, \omega, x_0) = x\right) \Longrightarrow (\theta_t \omega \in \Omega_a)\right]\right), \end{aligned} \tag{5.3.3}$$

when the right hand side is defined.

Infinite time horizon extension of Chen formulation: Extending Chen equation to infinite time interval requires an argument of the extended real line. Let $[0, \infty] \subset \overline{\mathbb{R}}$ be the subset of the extended real line $\overline{\mathbb{R}}$. Now, we make the following assumption:

Assumption 5.3.7. *For any policy $\pi \in \Pi$, initial state $x_0 \in \mathcal{X}$, and realization $\omega \in \Omega$, the limit*

$$\lim_{\tau \to \infty} \sigma^{\mathcal{T}}_{\pi,F}(x_0, \tau, \omega)(\tau)$$

exists and the path $\sigma^{\mathcal{T}}_{\pi,F}$ can be continuously extended to $[0, \infty]$. In addition, there exists a homeomorphism $\psi : [0, \infty] \to [0, T]$ for some $T > 0$ such the path $\sigma_\pi : \mathcal{X} \times \Omega \to \Sigma$ is properly defined by

$$\sigma_\pi(x_0, \omega)(t) = \left[\lim_{\tau \to \infty} \sigma^{\mathcal{T}}_{\pi,F}(x_0, \tau, \omega)\right](\psi^{-1}(t)), \quad \forall x_0 \in \mathcal{X},\ \omega \in \Omega,\ t \in [0, T],$$

to be an element of Σ.

Now, we redefine (for avoiding introducing more notations)

$$\mathcal{P}^{\pi}(y)(\omega) = \sigma_{\pi}(x, \omega),$$

and $\mathcal{P}_a^{\pi}(y, 0)$ by taking $T \to \infty$; and we effectively drop off the dependence on t for $\mathcal{S}^{\pi}$ and $\mathcal{ES}^{\pi}$. Chen equation becomes

$$\begin{aligned}
&\mathcal{S}^{\pi}(a, y) \\
&= \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y,0)(\omega)) \otimes A | \omega \in \Omega_a\right] = \mathbb{E}\left[\mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y,0)(\omega)) \otimes A | \mathcal{P}_a^{\pi}(y,0)(\omega), \omega \in \Omega_a\right] | \omega \in \Omega_a\right] \\
&= \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y,0)(\omega)) \otimes \mathbb{E}\left[A | \mathcal{P}_a^{\pi}(y,0)(\omega)\right] | \omega \in \Omega_a\right] \\
&= \mathbb{E}\left[S(\mathcal{P}_a^{\pi}(y,0)(\omega)) \otimes \mathcal{ES}^{\pi}(y^+) | \omega \in \Omega_a\right]
\end{aligned}$$

where A is redefined by

$$A := S(\mathcal{P}^{\pi}(y^+)(\theta_{t_a}\omega)).$$

Truncated signature formulation

For the mth-depth truncated signature (note that $m = \infty$ for signature with no truncation), we obtain,

$$(S(X) \otimes S(Y))_m = (S(X)_m \otimes S(Y)_m)_m =: S(X)_m \otimes_m S(Y)_m. \tag{5.3.4}$$

This is immediate from the definition of the multiplication $\otimes$ of the formal power series (see Section 2.1.7), implying that $S(X) \otimes S(Y)$ up to depth m only depends on the signatures of X and Y up to depth m. Therefore, when the cost only depends on the first mth-depth signatures, keeping track of the first mth-depth S-function $\mathcal{S}_m^{\pi}(a, y, t)$ suffices, and the cost function c can be efficiently computed as

$$c(\mathcal{S}_m^{\pi}(a, y, t)) = c\left(\mathbb{E}\left[S_m(\mathcal{P}_a^{\pi}(y,t)(\omega)) \otimes_m \mathcal{ES}_m^{\pi}(y^+, t + t_a) | a\right]\right).$$

Reduction to the Bellman equation

Recall that the Bellman expectation equation w.r.t. action-value function or Q-function is given by

$$Q^\pi(a,z,t) = \mathbb{E}_\Omega\left[r(a,z,\omega) + \gamma V^\pi(z^+, t+1)\middle|\omega \in \Omega_a\right], \tag{5.3.5}$$

where $V^\pi(z,t) = \mathbb{E}[Q^\pi(a,z,t)]$ for all $t \in \mathbb{N}$, where $\gamma \in (0,1]$ is a discount factor. We carefully show how Chen equation reduces to the Bellman expectation equation (see Figure 5.3 (left)). We suppose $\mathcal{X} := \mathcal{Z} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \subset \mathbb{R}^d$, for $d > 2$, is the state space augmented by the instantaneous reward and time, and suppose $t_a = 1$ for all $a \in \mathcal{A}$. And note $\mathcal{Z} \times \mathbb{R}_{\geq 0}$ which is the original state and time is now the space $\mathcal{Y}$. Let $m = 2$. We define the interpolation F, the transformation $\mathcal{T}$, and the cost function c so that

$$\begin{aligned}
&\forall x, w \in \mathcal{X} \text{ s.t. } x_{d-1:d} = [r_x, t_x], w_{d-1:d} = [r_w, t_x + 1]: \\
&\quad F(x,w)(\tau)_{d-1:d} = \begin{cases} [r_x + 2\tau \cdot (r_w - r_x), t_x] & (\tau \in [0, 0.5]) \\ [r_w, t_x + 2(\tau - 0.5)] & (\tau \in (0.5, 1]) \end{cases}, \\
&\forall \sigma : [0,t] \to \mathcal{X} \text{ s.t. } t \in \mathbb{Z}_{>0}: \\
&\quad \mathcal{T}(\sigma)(\tau + s) = \begin{cases} \left[2\tau\gamma^{[\sigma(\tau)]_d}[\sigma(\tau)]_{d-1}, [\sigma(\tau)]_d\right], & (\tau \in [0, 0.5]) \\ \left[\gamma^{\xi_1(\tau)}[\sigma(\tau)]_{d-1}, [\sigma(\tau)]_d\right], & (\tau \in [0.5, t-0.5)) \\ \left[\gamma^{\lfloor[\sigma(\tau)]_d\rfloor}[\sigma(\tau)]_{d-1}, \xi_2(\tau)\right], & (\tau \in [t-0.5, t-0.25)) \\ \left[4(t-\tau)\gamma^{[\sigma(t)]_d - 1}[\sigma(t)]_{d-1}, [\sigma(t)]_d\right], & (\tau \in [t-0.25, t]) \end{cases} \\
&\quad c(s) = -s_{1,2},
\end{aligned}$$

where

$$\xi_1(\tau) = \begin{cases} 2(\tau - \lfloor\tau\rfloor) + \lfloor[\sigma(\tau)]_d\rfloor - 1, & (\tau - \lfloor\tau\rfloor \leq 0.5) \\ \lfloor[\sigma(\tau)]_d\rfloor, & (\tau - \lfloor\tau\rfloor > 0.5), \end{cases}$$

and $\xi_2(\tau) = 2[\sigma(\tau)]_d - \lfloor[\sigma(\tau)]_d\rfloor$.

Then Chen equation reduces to the Bellman equation (5.3.5) by

$$\begin{aligned}c(\mathcal{S}_2^\pi(a,y,t)) &= c\left(\mathbb{E}\left[S_2(\mathcal{P}_a^\pi(y,t)(\omega))\otimes_2 \mathcal{E}\mathcal{S}_2^\pi(y^+,t+1)|\omega\in\Omega_a\right]\right)\\ &= \mathbb{E}\left[-S_{1,2}(\mathcal{P}_a^\pi(y,t)(\omega)) + c\left(\mathcal{E}\mathcal{S}_2^\pi(y^+,t+1)\right)|\omega\in\Omega_a\right]\end{aligned}$$

and

$$c\left(\mathcal{S}_2^\pi(a,y,t)\right) = -\gamma^t Q^\pi(a,z,t),\ c\left(\mathcal{E}\mathcal{S}_2^\pi(y,t)\right) = -\gamma^t V^\pi(z,t),\ S_{1,2}(\mathcal{P}_a^\pi(y,t)(\omega)) = \gamma^t r(a,z,\omega),$$

where $y=[z;t]$, and it reduces to the Bellman expectation equation.

Infinite horizon case: To see how infinite horizon Chen equation reduces to infintie horizon Bellman expectation equation, note that one cannot consider time axis now because it diverges and signatures are no longer defined. Therefore, instead we consider $\mathcal{Y}$ involving the space of discount factor and $\mathcal{O}$ to be the space of discounted cumulative reward, i.e., $\mathcal{Y}$ involves $1,\gamma,\gamma^2,\ldots$ and $\mathcal{O}$ is given by $r_0, r_0+\gamma r_1, r_0+\gamma r_1+\gamma^2 r_2,\ldots$. Extracting the path over discounted cumulative reward, and transforming it so that it starts from 0, a first depth signature (displacement) corresponds to the value-to-go. Here, because the extracted signature term only depends on $1,\gamma,\gamma^2,\ldots$ in a simple way, we could safely store the $\mathcal{S}$ function that has no dependence on this discount factor variable. We omit the details but we mention that the value function is again captured by signatures.

5.4 Signature MPC

We discuss an effective cost formulation over signatures for flexible and robust MPC, followed by additional numerical properties of signatures that benefit signature control.

Signature model predictive control: We present an application of Chen equation to MPC control– an iterative, finite-horizon optimization framework for control. In our signature MPC formulation, the optimization cost is defined for the signature of the full path being tracked, i.e., the previous path seen so far and the future path generated by the optimized control inputs (e.g., distance from the reference path signature for path tracking

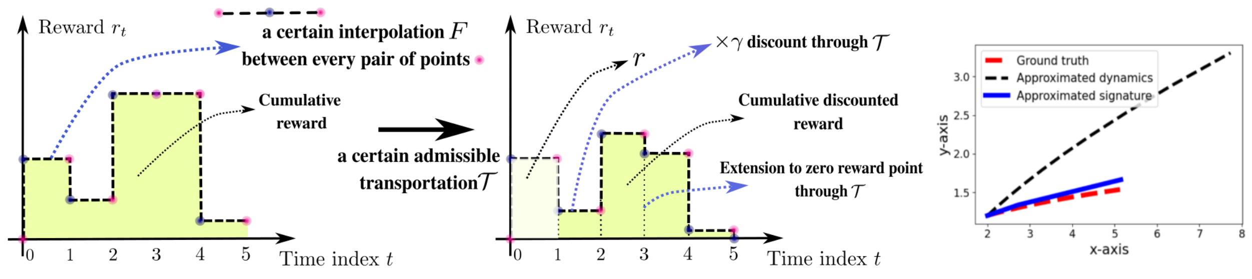

Reward r_t
a certain interpolation F
between every pair of points
Cumulative reward
Time index t
0 1 2 3 4 5
a certain admissible transportation $\mathcal{T}$
Reward r_t
r
$\times\gamma$ discount through $\mathcal{T}$
Cumulative discounted reward
Extension to zero reward point through $\mathcal{T}$
Time index t
0 1 2 3 4 5
Ground truth
Approximated dynamics
Approximated signature
y-axis
x-axis

Figure 5.3: Left: Illustrations of how a cumulative (discounted) reward is represented by our path formulation by an interpolation for representing the value as surface and by a transportation for discounting and concatenations of paths. Right: An error of approximated one-step dynamics propagates through time steps; while an error on signature has less effect over long horizon.

Algorithm 3 Signature MPC

```
    Input: initial state $x_0$; signature depth $m$; initial signature of past path $s_0 = \mathbf{1}$; # actions for rollout $N$; surrogate cost $\ell$ and regularizer $\ell_{\mathrm{reg}}$; terminal $S$-function $\mathcal{TS}_m$; simulation model $\hat{\mathcal{F}}$
1: while not task finished do
2:     Observe the current state $x_t$
3:     Update the signature of past path: $s_t = s_{t-1} \otimes_m S_m(\sigma(x_{t-1}, x_t))$, where $S_m(\sigma(x_{t-1}, x_t))$ is the signature transform of the subpath traversed since the last update from $t-1$ to $t$
4:     Compute the $N$ optimal future actions $\mathbf{a}^* := (a_0, a_1, \ldots, a_{N-1})$ using a simulation model $\hat{\mathcal{F}}$ that minimize the cost of the signature of the entire path (See Equation (5.4.1)).
5:     Run the first action $a_0$ for the associated duration $t_{a_0}$
6: end while
```

problem). Our algorithm, given in Algorithm 3, works in the receding-horizon manner and computes a fixed number of actions (the execution time for the full path can vary as each action may have a different time scale; i.e., each action is taken effect up to optimized (or fixed) time $t_a \in \mathbb{T}$).

Given the signature s_t of transformed past path (depth m) and the current state x_t at time t, the actions are selected by minimizing a two-part objective which is the sum of the surrogate cost ℓ and some regularizer ℓ_{reg}:

$$J = \begin{cases} \ell\Bigg(s_t \quad \otimes_m \quad \mathbb{E}\left[S_m(\sigma_{\mathbf{a}}(x_t)) \otimes_m \mathcal{TS}_m(x_0, s_t, \sigma_{\mathbf{a}}(x_t))\right]\Bigg) & \text{surrogate cost} \\ + \quad \ell_{\mathrm{reg}}\Bigg(\mathbb{E}\left[\mathcal{TS}_m(x_0, s_t, \sigma_{\mathbf{a}}(x_t))\right]\Bigg) & \text{regularizer} \end{cases} \tag{5.4.1}$$

where the path $\sigma_{\mathbf{a}}(x_t)$ is traced by the optimization variable $\mathbf{a} := (a_0, a_1, \ldots a_{N-1})$, and $\mathcal{TS}_m : \mathcal{X} \times T^m(\mathcal{X}) \times \Sigma \to T^m(\mathcal{X})$ is the *terminal S-function* that may be defined arbitrarily (as an analogy to terminal value in Bellman equation based MPC; see Appendix A.3.1).

Terminal S-function returns the signature of the terminal path-to-go. For the tracking problems studied in this work, we define the terminal subpath (path-to-go) as the final portion of the reference path starting from the closest point to the end-point of roll-out. This choice optimizes for actions up until the horizon anticipating that the reference path can be tracked afterward. We observed that this choice worked the best for simple classical examples analyzed in this work.

Error explosion along time steps: We consider robustness against misspecification of dynamics. Figure 5.3 (right) shows an example where the dashed red line is the ground truth trajectory with the true dynamics. When there is an approximation error on the one-step dynamics being modeled, the trajectory deviates significantly (black dashed line). On the other hand, when the same amount of error is added to each term of signature, the recovered path (blue solid line) is less erroneous. This is because signatures capture the entire (future) trajectory globally.

Computations of signatures: We compute the signatures through the kernel computations using an approach in [214]. We emphasize that the discrete points we use for computing the signatures of (past/future) trajectories are not regarded as waypoints, and their placement has negligible effects on the signatures as long as they sufficiently maintain the "shape" of the trajectories.

5.5 *Experimental results*

We conduct experiments on both simple classical examples and simulated robotic tasks (see Figure 5.6). We also present the relation of a specific instance of signature control to the classical integral control to show its robustness against disturbance. For more experiment details, see Appendix A.3.2.

Simple point-mass: We use double-integrator point-mass as a simple example to demonstrate our approach. In this task, a point-mass is controlled to reach a goal position while avoiding the obstacle in the scene. We first generate a collision-free path via RRT* [122] which is suboptimal in terms of tracking speed (**taking** 72 **seconds**). We then employ our signature MPC to follow this reference by producing the actions (*i.e.* accelerations).

Specifically, define $\mathcal{X} := [0,100] \times [0,100] \times [0,5] \times [0,5]$. The dynamics is approximated by the Euler approximation:

$$[p_{t+1}, v_{t+1}] = P_{\mathcal{X}} \left[p_t + v_t \Delta t, v_t + a_t \Delta t\right],$$

where p is the 2D position and v is the 2D velocity and $P_{\mathcal{X}} : \mathbb{R}^4 \to \mathcal{X}$ is the orthogonal projection.

A feasible reference path for the obstacle avoidance goal reaching task is generated by RRT* with local CEM planner (*wiring* of nodes is done through CEM planning with some margin). The generated reference path is shown in Figure 5.4 (left).

The reference path is then splined by using natural cubic spline (illustrated in Figure 5.5; using package (`https://github.com/patrick-kidger/torchcubicspline`)); using only the path over positions (2D path), we run signature MPC. The time duration of each action is also optimized at the same time. For comparison we also run an MPC with zero terminal S-function case. Note we are using RBF kernel for signature kernels, which makes this terminal S-function choice less unfavorable.

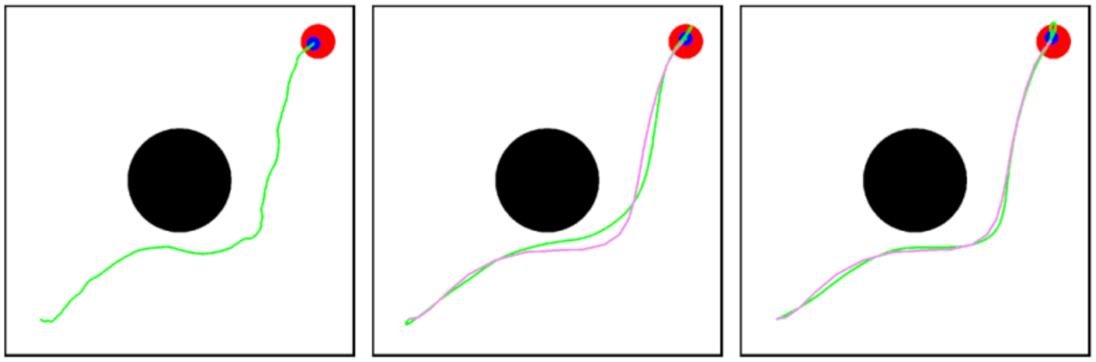


Figure 5.4: Left: Suboptimal feasible path generated by RRT* with local CEM planner. Middle: Signature MPC with zero terminal S-function. Purple one is the splined reference path and the green one is the executed one. Right: Signature MPC with the best choice of terminal S-function.

Figure 5.4 (middle) shows the zero terminal S-function case; which tracks well but with slight deviation. Figure (right) shows that of the best choice of terminal S-function.

We see that the adoption of signature control results in a better tracking speed (**taking around** 30 **seconds**) while matching the trajectory shape.

Two-mass, spring, damper system: To view the integral control [126] within the scope of our proposed signature control formulation, recall a second depth signature term corresponding to the surface surrounded by the time axis, each of the state dimension, and the path, represents each dimension of the integrated error. In addition, a first depth signature term with the initial state x_0 represents the immediate error, and the cost c may be a weighted sum of these two. To test this, we consider two-mass, spring, damper system; the disturbance is assumed zero for planning, but is 0.03 for executions.

Our continuous-time system of two-mass, spring, damper system is given by

$$\dot{v_1} = -\frac{(k_1+k_2)p_1}{m_1} - \frac{(b_1+b_2)v_1}{m_1} + \frac{k_2p_2}{m_1} + \frac{b_2v_2}{m_1} + \frac{a_1}{m_1} + w_1,$$
$$\dot{v_2} = \frac{k_2p_1}{m_2} + \frac{b_2v_1}{m_2} - \frac{k_2p_2}{m_2} - \frac{b_2v_2}{m_2} + \frac{a_2}{m_1} + w_2,$$

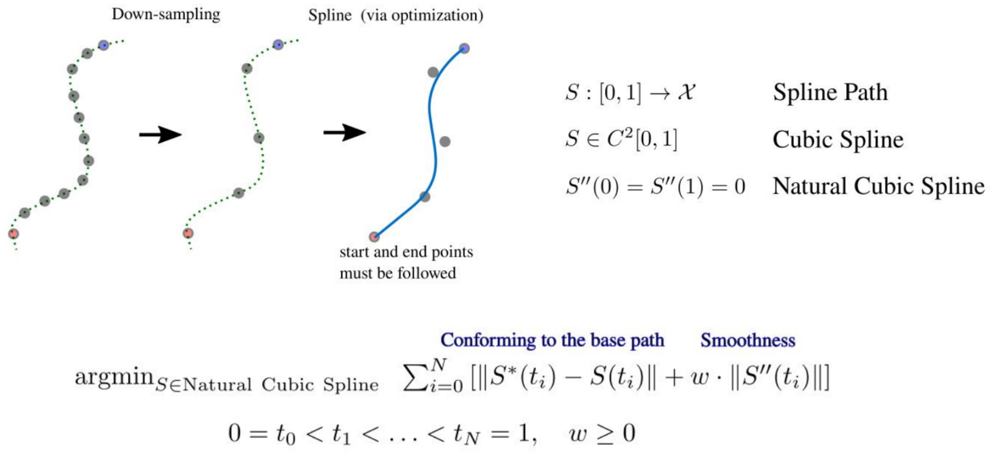

Down-sampling
Spline (via optimization)
start and end points must be followed
$S : [0,1] \to \mathcal{X}$ Spline Path
$S \in C^2[0,1]$ Cubic Spline
$S''(0) = S''(1) = 0$ Natural Cubic Spline
Conforming to the base path
Smoothness
$\mathrm{argmin}_{S \in \text{Natural Cubic Spline}} \sum_{i=0}^{N} [\|S^*(t_i) - S(t_i)\| + w \cdot \|S''(t_i)\|]$
$0 = t_0 < t_1 < \ldots < t_N = 1, \quad w \geq 0$

Figure 5.5: Illustrations of natural cubic spline. We down-sampled the reference path with skip number 10. Then, with weight $w = 0.5$, step size 0.01, using Adam optimizer, we optimized the path to obtain a spline with iteration number 150. Note S here indicates spline, not signatures.

Table 5.1: Parameters for two-mass, spring, damper system.

$k_1 = 2.0$	$b_1 = 0.05$	$m_1 = 1.0$
$k_2 = 1.0$	$b_2 = 0.05$	$m_2 = 2.0$

where $u_1, u_2 \in [-1, 1]$ are control inputs, and ws are disturbances. The actual parameters are listed in Table 5.1. We augment the state with time that obviously follows $\dot{t} = 1$.

We compare signature MPC where the cost is the squared Euclidean distance between the signatures of the reference and the generated paths with truncation up to the first and the second depth.

On the other hand, to test traditional P and PI controls, we obtain an approximation of the time derivative of signatures by

$$\frac{\partial S_2(\sigma_t)}{\partial t} \approx \frac{S_2(\sigma_t * \sigma_{t,t+\Delta t}) - S_2(\sigma_t)}{\Delta t},$$

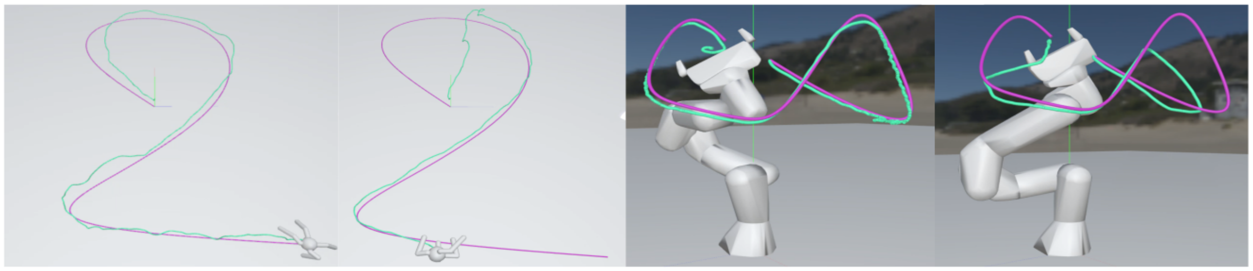


Figure 5.6: (Left two; Ant) Tracking behaviors of signature control (left) and baseline MPC (right) for the same reaching time, where green lines are the executed trajectories. (Right two; Robotic arm): Tracking behaviors of signature control (left) and baseline MPC (right) under disturbance -30.

where σ_t is the path from time 0 to t, and $\sigma_{t,t+\Delta t}$ is the linear path between the current state and the next state (after Δt), and $\Delta t = 0.1$ is the discrete time interval. By further augmenting the state with signature $S_2(\sigma_t)$, we compute the linearized dynamics around the point $p = v = \mathbf{0}_2$ and the unit signature.

P control (the state includes velocities, so it might be viewed as PD control) is obtained by computing the optimal gain for the system over p and v by using the cost $p^\top p + v^\top v + 0.01u^\top u$. For PI control, we use the linearized system over p, v and $s_{2,5}$, $s_{2,15}$ (corresponding to integrated errors for p_1 and p_2), and the cost is $p^\top p + v^\top v + s_{2,5}^2 + s_{2,15}^2 + 0.01u^\top u$. We added $-0.0001I$ to the linearized system to ensure that the python control package (`https://github.com/python-control/python-control`) returns a stabilizing solution.

The unknown constant disturbance is assumed zero for planning, but is 0.03 (on both acceleration terms) for executions; the plots are given in Figure 5.7. Our signature MPC reflects the well-known behaviors of P control and PI control. As expected, the black line (first depth) does not converge to zero error state while the blue line (second depth) does. If we further include other signature terms, signature control effectively becomes a generalization of integral controls, which we will see for robotic arm experiments later.

Ant path tracking: In this task, an Ant robot is controlled to follow a "2"-shape reference path. The action being optimized is the torque applied to each joint actuator. We test the tracking performances of signature control and baseline standard MPC on this problem. Also, we run soft actor-critic (SAC) [94] RL algorithm where the state is augmented with time index to manage waypoints and the reward (negative cost) is the same as that of the baseline MPC. For the baseline MPC and SAC, we equally distribute 880 waypoints to be tracked along the path; here time stamp is determined by equally dividing the total tracking time achieved by signature control. Table 5.2 compares the mean/variance of deviation (measured by distance in meter) from the closest of 2000 points over the reference, and Figure 5.6 (left) shows the resulting behaviors of MPCs, showing significant advantages of our method in terms of tracking accuracy. The performance of SAC RL is insufficient as we have no access to sophisticated waypoints over joints (see [198] for the discussion). When more time steps are used, baseline MPC becomes a bit better. Note our method can tune the trade-off between accuracy and progress through regularizer.

Robotic manipulator path tracking: In this task, a robotic manipulator is controlled to track an end-effector reference path. Similar to the Ant task, 270 waypoints are equally sampled along the reference path for the baseline MPC method and SAC RL to track. To show robustness of signature control against unknown disturbance (torque: $\mathrm{N} \cdot \mathrm{m}$), we test different scales of disturbance force applied to each joint of the arm. The means/variances of the tracking deviation of the three approaches are reported in Table 5.3 and the tracking paths are visualized in Figure 5.6 (right). For all cases, our signature control outperforms the baseline MPC and SAC RL, especially the difference becomes much clearer when the disturbance becomes larger. This is because the signature MPC is insensitive to waypoint designs but rather depends on the "distance" between the target and the rollout paths in the signature space, making the tracking speed adaptive.

Computation: For all of the experiments, we used the computer with

Table 5.2: Results on path following with an ant robot model. Comparing signature control, and baseline MPC and SAC RL with equally assigned waypoints. "Slow" means it uses more time steps than our signature control for reaching the goal.

	Deviation (distance) from reference			
	Mean (10^{-2}m)	Variance (10^{-2}m)	# waypoints	goal
signature control	**21.266**	**6.568**	N/A	success
baseline MPC	102.877	182.988	880	fail
SAC RL	446.242	281.412	880	fail
baseline MPC (slow)	10.718	5.616	1500	success
baseline MPC (slower)	1.866	0.026	2500	success

- Ubuntu 20.04.3 LTS
- Intel(R) Core(TM) i7-6850K CPU 3.60GHz (max core 12)
- RAM 64 GB/ 1 TB SSD
- GTX 1080 Ti (max 4; we used the same GPU for all of the experiments)
- GPU RAM 11 GB / CUDA 10.1

The computation time for each (seed) run for any of the numerical experiments was around 30 mins to $1-2$ hours (MPC problems). For small analysis experiments, it took less than a few minutes for each one.

5.6 Chapter summary and discussion

This work presented signature control, a novel framework that generalizes value-based dynamic programming to reason over entire paths through Chen equation. There are many promising avenues for future work (which are relevant to the current limitations of our work), such as developing a more complete theoretical understanding of guarantees provided by the signature control framework, and developing additional RL algorithms that inherit the benefits of our framework.

In fact, we did not provide an analogue of policy/value iteration algorithms in our signature control framework; and a set of conditions required to apply similar arguments (i.e., contraction mapping and fixed point arguments) used in value iteration to our generalized framework or otherwise a novel technical approach to show convergence to certain suboptimal solutions with error bounds has not yet been addressed.

Furthermore, when studying Itô diffusion models of dynamical systems, the Hamilton-Jacobi-Bellman-Isaacs (HJBI) equation is typically employed to tackle stochastic optimal control problems. As our Chen equation relies on path signatures that may not be properly defined for a path of unbounded variation, connecting Chen equation to the HJBI equation would require additional insights and technical innovations.

Finally, although the MPC algorithms we proposed to use in this chapter for both signature control and for the traditional Bellman-based (waypoint-based) control have exhibited almost the same computational run time, we did not test them in the real robotics systems. Devising real-time MPC algorithm tailored to signature control is required to lift our work to the level of practical significance.

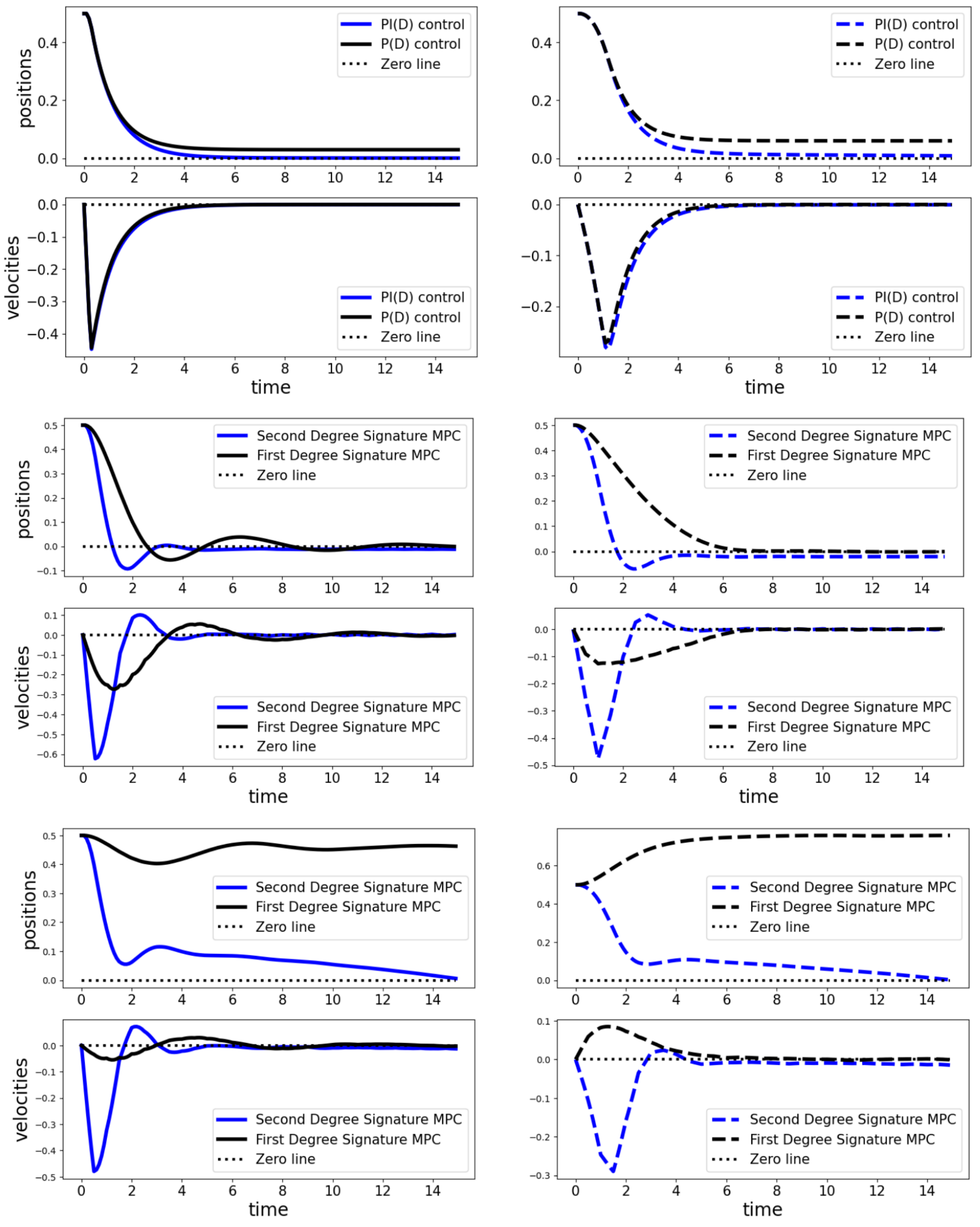


Figure 5.7: Two-mass spring, damper system. Solid lines are for the position and velocity of the first mass, and dashed lines are for the second mass. Black lines are without the second depth signatures, and the blue lines are with the elements of the second depth signatures corresponding to the integrated errors. Top: P control and PI control under no disturbance. Middle: Signature MPCs using signatures up to depth 1 and 2 under no disturbance case; both converge to the zero state well. Down: Signature MPCs under unknown disturbances; the one up to depth 2 converges to zero state while the one up to depth 1 does not because the depth 2 signature terms correspond to the integrated errors to mimic PI controls.

Table 5.3: Results on path tracking with a robotic manipulator end-effector. Comparing signature control, and baseline MPC and SAC RL with equally assigned waypoints under unknown fixed disturbance.

		Deviation (distance) from reference	
	Disturbance ($N \cdot m$)	Mean (10^{-2}m)	Variance (10^{-2}m)
signature control	+30	**1.674**	**0.002**
	+20	**1.022**	**0.002**
	+10	**0.615**	**0.001**
	±0	**0.458**	**0.001**
	−10	**0.605**	**0.001**
	−20	**0.900**	**0.001**
	−30	**1.255**	**0.002**
baseline MPC	+30	2.648	0.015
	+20	1.513	0.010
	+10	0.828	0.005
	±0	0.612	0.007
	−10	1.407	0.013
	−20	3.408	0.078
	−30	5.803	0.209
SAC RL	+30	15.669	0.405
	+20	10.912	0.224
	+10	6.252	0.102
	±0	3.853	0.052
	−10	6.626	0.243
	−20	12.100	0.557
	−30	16.019	0.743

Chapter 6

ALGORITHM DESIGN AT THE INTERSECTION OF STATISTICAL ML AND DYNAMICAL SYSTEMS

Chapter's key takeaways

This chapter presents the dynamic structure estimation problems for periodically behaved discrete dynamical system in the Euclidean space. The observations become sequentially available in a form of bandit feedback contaminated by a sub-Gaussian noise. To avoid losing the periodic structural information through concentration of measures, asymptotic lower bound results of exponential sums are applied to devise an estimation algorithm with statistical sample complexity guarantees. When the dynamics is linear, the Weyl sum, a variant of exponential sums studied in number theory community, is adopted. In this process, mathematical concepts, *nearly period*, and (θ, k)-*distinct eigenvalues*, are proposed out of necessity. Our algorithms are shown to correctly work in practical simulations.

6.1 Introduction

System identification has been of great interest in controls, economics, and statistical machine learning (cf. [247, 246, 142, 145, 119, 190, 165, 223, 71, 97, 222, 146]). In particular, estimations of periodic information, including eigenstructures for linear systems, under noisy and partially observable environments, are essential to a variety of applications such as biological data analysis (e.g., [105, 226, 274]; also see [88] for how gene oscillation affects differentiation of cells), earthquake analysis (e.g., [204, 213, 264]; see [15] for the connections of the frequencies and magnitude of earthquakes), chemical/asteroseismic analysis (e.g., [6]), and communication and information systems (e.g., [69, 76]), just to name a few.

Specifically, providing statistical guarantees for extractions of periodic information under perturbations following a general distribution is quite fundamental as it is connected to the information theory of communication capacities. Indeed, there has been an interest of studying novel paradigms for coding, transmitting, and processing information sent through optical communication systems [249]. When signals are coded digitally, erroneous signal transmission is no longer modeled with a simple Gaussian distribution. In fact, if the observation is contaminated by a noise following a more general distribution, a concentration of measures should be adopted (cf. [219]); and this approach suffers from a risk of making structural information of the underlying dynamics vanish as well.

In this chapter, we tackle this periodic structure estimation problem for *nearly* periodically behaved discrete dynamical systems (cf. [24]; we allow systems that are not *exactly* periodic) with sequentially available bandit feedback. Due to the presence of noise and partial observability, our problem setups do not permit the recovery of the full set of period/eigenvalues information in general; as such we ask the following question: *what subset of information on dynamic structures can be statistically efficiently estimated?* This chapter successfully answers this question by identifying and mathematically defining recoverable information, and proposes algorithms for efficiently extracting such information.

The technical novelty of our approach is highlighted by the careful adoption of the asymptotic bounds on the exponential sums that effectively cancel out noise effects while preserving the information to be estimated. When the dynamics is driven by a linear system, the use of the Weyl sum [260], a variant of exponential sums, enables us to extract more detailed information. To our knowledge, this is the first attempt of employing asymptotic results of the Weyl sum for statistical estimation problems, and further studies on the relations between statistical estimation theory and exponential sums (or even other number theoretical results) are of independent interests.

Contributions. The contributions of this work are three folds: (1) mathematically identify and define a recoverable set of periodic/eigenvalues information when the observations are available in a form of bandit feedback (the feedback is contaminated by a sub-Gaussian noise, which is more general than those usually considered in system identification work), (2) present provably correct algorithms for efficiently estimating such information; this constitutes the first attempt of adopting asymptotic results on the Weyl sum, (3) and implement our algorithms for toy examples to experimentally validate our claims.

6.2 Problem setups

In this section, we present our dynamical system model followed by some definitions on the properties of dynamical system and our problem setups.

6.2.1 Dynamic structure in bandit feedback

We define $D \subset \mathbb{R}^d$ as a (finite or infinite) collection of arms to be pulled. Let $(\eta_t)_{t=1}^{\infty}$ be a noise sequence. Let $\Theta \subset \mathbb{R}^d$ be a set of latent parameters. We assume that there exists a *dynamical system* f on Θ, equivalently, a map $f : \Theta \to \Theta$. At each time step $t \in \{1, 2, \ldots\}$, a learner pulls an arm $x_t \in D$ and observes a reward

$$r_t(x_t) := f^t(\theta)^\top x_t + \eta_t,$$

for some $\theta \in \Theta$. In other words, the hidden parameters for the rewards may vary over time but follow only a rule f with initial value θ. The function r_t could be viewed as the specific instance of partial observation (cf. [152]).

6.2.2 Nearly periodic sequence

Before delving into the setups, we define a general notion of nearly periodic sequence:

Definition 6.2.1 (Nearly periodic sequence)**.** Let $(\mathcal{X}, d)$ be a metric space. Let $\mu \geq 0$ and let $L \in \mathbb{Z}_{>0}$. We say a sequence $(y_t)_{t=1}^{\infty} \subset \mathcal{X}$ is *μ-nearly periodic* of length L if $d(y_{s+Lt}, y_s) < \mu$ for any $s, t \in \mathbb{Z}_{>0}$. We also call L the length of the μ-nearly period.

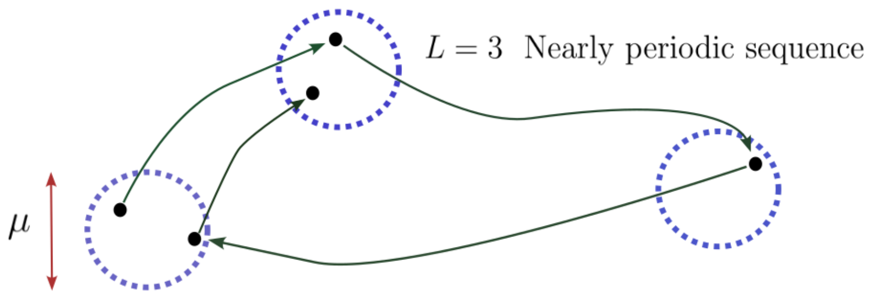


Figure 6.1: Illustration of nearly periodic sequence; it is of $L = 3$.

Intuitively, there exist L balls of diameter μ in $\mathcal{X}$ and the sequence $y_1, y_2, \dots$ moves in the balls in order if $(y_t)_{t=1}^{\infty}$ is μ-nearly periodic of length L (see Figure 6.1). Obviously, nearly periodic sequence of length L is also nearly periodic sequence of length mL for any $m \in \mathbb{Z}_{>0}$. We say a sequence is periodic if it is 0-nearly periodic. We introduce a notion of aliquot nearly period to treat estimation problems of period:

Definition 6.2.2 (Aliquot nearly period)**.** Let $(\mathcal{X}, d)$ be a metric space. Let $\rho > 0$ and $\lambda \geq 1$. Assume a sequence $\{y_t\}_{t=1}^{\infty} \subset \mathcal{X}$ is *μ-nearly periodic* of length L for some $\mu \geq 0$ and $L \in \mathbb{Z}_{>0}$. A positive integer ℓ is a *(ρ, λ)-aliquot nearly period* (*(ρ, λ)-anp*) of $(y_t)_{t=1}^{\infty}$ if $\ell | L$ and the sequence $(y_t)_{t=1}^{\infty}$ is $(\rho + 2\lambda\mu)$-nearly periodic.

We may identify the (ρ, λ)-anp with a $2\lambda\mu$-nearly period under an error margin ρ. When we estimate the length L of the nearly period of unknown sequence $(y_t)_t$, we sometimes cannot determine the L itself, but an aliquot nearly period.

Example 6.2.1. A trajectory of finite dynamical system is always periodic and it is the most simple but important example of (nearly) periodic sequence. We also emphasize that if we know the upper bound of the number of underlying space, the period is bounded above by the upper bound as well. These facts are summarized in Proposition 6.2.1. The cellular automata on finite cells is a specific example of finite dynamical systems. We will treat LifeGame [66], a special cellular automata, in our simulation experiment (see Section 6.4).

Proposition 6.2.1. *Let $f : \Theta \to \Theta$ be a map on a set Θ. If $|\Theta| < \infty$, then for any $t \geq |\Theta|$ and $\theta \in \Theta$, $f^{t+L}(\theta) = f^t(\theta)$ for some $1 \leq L \leq |\Theta|$.*

Proof. Since $|\{\theta, f(\theta), \ldots, f^{|\Theta|}(\theta)\}| > |\Theta|$, there exist $0 \leq i < j \leq |\Theta|$ such that $f^i(\theta) = f^j(\theta)$ by the pigeon hole principal. Thus, $f^t(\theta) = f^{j-i+t}(\theta)$ for all $t \geq |\Theta|$. □

If a linear dynamical system generates a nearly periodic sequence, we can show the linear system has a specific structure as follows:

Proposition 6.2.2. *Let $M : \mathbb{R}^d \to \mathbb{R}^d$ be a linear map. Let $\mathbb{C}^d = \oplus_\alpha V_\alpha$ be the decomposition via generalized eigenspaces of M, where α runs over the eigenvalues of M and $V_\alpha := \mathscr{N}((\alpha I - M)^d)$. Assume that there exists $\mu \geq 0$, for any $\theta \in \mathbb{R}^d$, $(M^t\theta)_{t=t_0}^\infty$ is μ-nearly periodic for some $t_0 \in \mathbb{N}$. Let $\theta = \sum_\alpha \theta_\alpha \in \oplus_\alpha V_\alpha$. Then, each eigenvalue α such that $\theta_\alpha \neq 0$ satisfies $|\alpha| \leq 1$, in addition, if $|\alpha| = 1$ and $\theta_\alpha \neq 0$, $M\theta_\alpha = \alpha\theta_\alpha$.*

Proof. We note that $\{M^t v\}_{t\geq 0}$ is bounded for any $v \in \mathbb{R}^d$ by the assumption on M. Thus, M cannot have an eigenvalue of magnitude greater than 1. We show that α is in the form of $\alpha = e^{i2\pi q}$ for some $q \in \mathbb{Q}$ if $|\alpha| = 1$. Suppose that $\alpha = e^{i2\pi\gamma}$ for an irrational number $\gamma \in \mathbb{R}$. Then, for an eigenvector w for θ_α with $\|w\|_{\mathbb{R}^d} > \mu$, $\{M^t w\}_{t > t_0}$ cannot become a μ-periodic sequence. Thus, we conclude $\alpha = e^{i2\pi q}$ for some $q \in \mathbb{Q}$. Next, we show $M\theta_\alpha = \alpha\theta_\alpha$ if $|\alpha| = 1$ and $\theta_\alpha \neq 0$. Suppose $(M - \alpha I)\theta_\alpha \neq 0$. Since $(M - \alpha I)^d\theta_\alpha = 0$, there exists $1 \leq d' < d$ such that $(M - \alpha I)^{d'+1}\theta_\alpha = 0$ but $(M - \alpha I)^{d'}\theta_\alpha \neq 0$. Let $w' := (M - \alpha I)^{d'-1}\theta_\alpha$. Then, we see that $(M - \alpha I)^2 w' = 0$ but $(M - \alpha I)w' \neq 0$. By direct computation, we see that

$$\|M^t w'\|_{\mathbb{R}^d} = \|(M - \alpha I + \alpha I)^t w'\|_{\mathbb{R}^d} \geq t\|(M - \alpha I)w'\|_{\mathbb{R}^d} - \|w'\|_{\mathbb{R}^d}.$$

Thus, we have $\|M^t w'\|_{\mathbb{R}^d} \to \infty$ as $t \to \infty$, which contradicts the fact that $\{M^t w'\}_{t\geq 0}$ is a bounded sequence. The last statement is obvious. □

Let W be a linear subspace of $\mathbb{C}^d$ generated by the trajectory $\{M\theta, M^2\theta, \ldots\}$ and denote $\dim(W)$ by d_0. Note that restriction of M to W induces a linear map from W to W. We denote by M_θ the induced linear map from W to W. Let $W = \oplus_{\alpha\in\Lambda} W_\alpha$ be the decomposition via the generalized eigenspaces of M_θ, where Λ is the set of eigenvalues of M_θ and $W_\alpha := \mathscr{N}((\alpha I - M_\theta)^{d_0})$. We define

$$W_{=1} := \oplus_{|\alpha|=1} W_\alpha,$$
$$W_{<1} := \oplus_{|\alpha|<1} W_\alpha.$$

Then, we have the following statement as a corollary of Proposition 6.2.2:

Corollary 6.2.3. *There exist linear maps $M_1, M_{<1} : W \to W$ such that*

1. $M_\theta = M_1 + M_{<1}$,
2. $M_1 M_{<1} = M_{<1} M_1 = O$,
3. *M_1 is diagonalizable and any eigenvalue of M_1 is of magnitude* 1, *and*
4. *any eigenvalue of $M_{<1}$ is of magnitude smaller than* 1.

Proof. Let $p : W \to W_{=1}$ be the projection and let $i : W_{=1} \to W$ be the inclusion map. We define $M_1 := i M_\theta p$. We can construct $M_{<1}$ in the similar manner and these matrices are desired ones. $\square$

Example 6.2.2. Let G be a finite group and let ρ be a finite dimensional representation of G, namely, a group homomorphism $\rho : G \to \mathrm{GL}_m(\mathbb{C})$, where $\mathrm{GL}_m(\mathbb{C})$ is the set of complex regular matrices of size m. Fix $g \in G$. Let $B \in \mathbb{C}^{n\times n}$ be a matrix whose eigenvalues have magnitudes smaller than 1. We define a matrix of size $m+n$ by

$$M := \begin{bmatrix} \rho(g) & 0 \\ 0 & B \end{bmatrix}.$$

Then, $(M^t x)_{t=t_0}^{\infty}$ is a μ-nearly periodic sequence for any $\mu > 0$, $x \in \mathbb{C}^m$, and sufficiently large t_0. Moreover, we know that the length of the nearly period is $|G|$. We treat the permutation of variables in $\mathbb{R}^d$ in the simulation experiment (see Section 6.4), namely the case where G is the symmetric group $\mathfrak{S}_d$ and ρ is a homomorphism from G to $\mathrm{GL}_d(\mathbb{C})$ defined by $\rho(g)((x_j)_{j=1}^d) := (x_{g(j)})_{j=1}^d$, which is the permutation of variable via g.

6.2.3 Problem setting

Here, we state our problem settings. We use the notation introduced in the previous sections. First, we summarize our technical assumptions as follows:

Assumption 6.2.4 (Conditions on arms)**.** *The set of arms D contains the unit hypersphere.*

Assumption 6.2.5 (Assumptions on noise)**.** *The noise sequence $\{\eta_t\}_{t=1}^{\infty}$ is conditionally R-sub-Gaussian ($R \in \mathbb{R}_{\geq 0}$), i.e., given t,*

$$\forall \lambda \in \mathbb{R},\ \mathbb{E}\left[e^{\lambda \eta_t} | \mathcal{F}_{t-1}\right] \leq e^{\frac{\lambda^2 R^2}{2}},$$

and $\mathbb{E}[\eta_t|\mathcal{F}_{t-1}] = 0$, $\mathrm{Var}[\eta_t|\mathcal{F}_{t-1}] \leq R^2$, where $\{\mathcal{F}_\tau\}_{\tau \in \mathbb{N}}$ is an ascending family and we assume that $x_1, \ldots, x_{\tau+1}, \eta_1, \ldots, \eta_\tau$ are measurable with respect to $\mathcal{F}_\tau$.

Assumption 6.2.6 (Assumptions on dynamical systems)**.** *There exists $\mu > 0$ such that for any $\theta \in \Theta$, the sequence $(f^t(\theta))_{t=t_0}^{\infty}$ is μ-nearly periodic of length L for some $t_0 \in \mathbb{N}$. We denote by B_θ the radius of the smallest ball containing $\{f^t(\theta)\}_{t=0}^{\infty}$.*

Remark 6.2.7. Assumption 6.2.4 excludes the lower bound arguments of the minimally required samples for this proposed work since taking sufficiently large vector (arm) makes the noise effect negligible. Considering more restrictive conditions for discussing lower bounds is out of scope of this chapter.

Then, our questions are described as follows:

- Can we estimate the length L from the collection of rewards $(r_t(x_t))_{t=1}^{T}$ efficiently ?
- If we assume the dynamical system is linear, can we further obtain the eigenvalues of f from a collection of rewards ?
- How many samples do we need to provably estimate the length L or eigenvalues of f ?

We will answer these questions in the following sections and via simulation experiments.

6.3 Algorithms and theory

With the above settings in place, we present a (computationally efficient) algorithm for each presented problem, and show its sample complexity for estimating certain information.

6.3.1 Period estimation

Here, we describe an algorithm for period estimation followed by its theoretical analysis. The overall procedure is summarized in Algorithm 4. To analyze the sample complexity of this estimation algorithm, we first introduce an exponential sum that plays a key role:

Definition 6.3.1. For a positive rational number $q \in \mathbb{Q}_{>0}$ and T complex numbers $a_1, \ldots, a_T \in \mathbb{C}$, we define

$$\mathcal{R}\big((a_t)_{t=1}^{T}; q\big) := \frac{1}{T}\sum_{j=1}^{T} a_j e^{i2\pi qj}.$$

For a μ-nearly periodic sequence $\mathbf{a} := (a_t)_{t=1}^{\infty}$ of length L, we define the supremum of the standard deviations of the L sequential data of $\mathbf{a}$:

$$\sigma_L(\mathbf{a}) := \sup_{t_0 \geq 1} \sqrt{\frac{1}{L}\sum_{t=t_0}^{t_0+L-1}\left|a_t - \frac{1}{L}\sum_{j=t_0}^{t_0+L-1} a_j\right|^2}.$$

The exponential sum $\mathcal{R}(\cdot;\cdot)$ can extract a divisor of the nearly period of a μ-nearly periodic sequence if μ is "sufficiently smaller" than the variance of the sequence even when the sequence is contaminated by noise; more precisely, we have the following lemma:

Lemma 6.3.1. *Let $\mathbf{a} := (a_j)_{j=1}^{\infty}$ be a μ-nearly periodic sequence of length L. Then, we have the following statements:*

1. *if $L > 1$, then there exists $s \in \mathbb{Z}_{>0}$ with $s < L$ such that*

$$\left|\mathcal{R}\left((a_j)_{j=1}^{T}; s/L\right)\right| > \sqrt{\frac{\sigma^2 - 2\mu\sigma}{L}} - \mu - \frac{L\sup_{t\geq 1}|a_t|}{T}, \tag{6.3.1}$$

Algorithm 4 Period estimation (DFT)

```
    **Input**: Current time $t_0 \in \mathbb{Z}_{>0}$; $T_p$; $\epsilon > 0$; $L_{\max} > 1$; orthogonal basis $\{\mathbf{u}_1, \ldots, \mathbf{u}_d\}$ of $\mathbb{R}^d$
    **Output**: Estimated length $\hat{L}$
 1: $\ell \leftarrow 1$; $\beta \leftarrow 1$
 2: for $m = 1, \ldots, d$ do
 3:     for $t = t_0, t_0 + 1, \ldots, t_0 + T_p - 1$ do
 4:         Sample arm $\mathbf{u}_m$ and observe $r_t(\mathbf{u}_m)$
 5:     end for
 6:     while $\ell \cdot \beta \leq L_{\max}$ do
 7:         $\ell \leftarrow \ell + 1$
 8:         for $(s, b) = (0,1), (0,2), \ldots, (0, \ell - 1), (1,1), (1,2), \ldots, (\beta - 1, \ell - 1)$ do
 9:             if $\left|\mathcal{R}\left((r_{t_0+s+\beta t}(\mathbf{u}_m))_{t=1}^{\lfloor T_p/\beta \rfloor}; b/\ell\right)\}\right|^2 > \epsilon$ then
10:                 $\beta \leftarrow \beta\ell$
11:                 $\ell \leftarrow 1$
12:                 Break
13:             end if
14:         end for
15:     end while
16:     $t_0 \leftarrow t_0 + T_p$
17: end for
18: $\hat{L} \leftarrow \beta$
```

2. if β is not a divisor of L, then for any $\alpha \in \mathbb{Z}_{>0}$,

$$\left|\mathcal{R}\left((a_j)_{j=1}^T; \alpha/\beta\right)\right| < \mu + \frac{L^2 \mathcal{C}_0 (\mu + \sup_{t \geq 1} |a_t|)}{T}, \tag{6.3.2}$$

where $\mathcal{C}_0 := 1 + 2/\sqrt{2}\pi(3/4)^{\pi^2/6} = 1.72257196806914\ldots$.

Proof. As $(a_t)_{t=1}^\infty$ is μ-almost periodic, there exist $(b_t)_{t=1}^\infty$ of period L and $(c_t)_{t=1}^\infty$ with $\sup_{t\geq 1} |c_t| < \mu$ such that $a_t = b_t + c_t$.

First, we prove (6.3.1). Let

$$\tilde{b} := \frac{1}{L}\sum_{t=1}^{L} b_t, \quad \hat{b}_q := \frac{1}{L}\sum_{t=1}^{L} b_t e^{i2\pi tq}, \quad (q \in \mathbb{Q}).$$

We claim that

$$L \cdot \sup\{|\hat{b}_{s/L}|^2\}_{s=1}^{L-1} > \sum_{s=1}^{L-1} |\hat{b}_{s/L}|^2 = \frac{1}{L}\sum_{t=1}^{L} |b_t - \tilde{b}|^2 \geq \sigma^2 - 2\mu\sigma. \tag{6.3.3}$$

In fact, the first inequality is obvious. The equality follows from the Plancherel formula for a finite abelian group (see, for example, [218, Excercise 6.2]). As for the last inequality, take arbitrary $t_0 \in \mathbb{Z}_{>0}$ and define $\tilde{a} = L^{-1}\sum_{t=t_0}^{t_0+L-1} a_t$ and $\tilde{c} = L^{-1}\sum_{t=t_0}^{t_0+L-1} c_t$. Then, we have

$$\begin{aligned}
\frac{1}{L}\sum_{t=1}^{L} |b_t - \tilde{b}|^2 &\geq \frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |a_t - \tilde{a}|^2 + \frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |c_t - \tilde{c}|^2 - \frac{2}{L}\sum_{t=t_0}^{t_0+L-1} |a_t - \tilde{a}| \cdot |c_t - \tilde{c}| \\
&\geq \frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |a_t - \tilde{a}|^2 + \frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |c_t - \tilde{c}|^2 \\
&\quad - 2\sqrt{\frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |a_t - \tilde{a}|^2} \cdot \sqrt{\frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |c_t - \tilde{c}|^2} \\
&\geq \frac{1}{L}\sum_{t=t_0}^{t_0+L-1} |a_t - \tilde{a}|^2 - 2\mu\sigma.
\end{aligned}$$

Here, we used the Cauchy-Schwartz inequality in the second inequality. Since t_0 is arbitrary, we have (6.3.3). Let $s \in \mathrm{argmax}_{s=1,\ldots,L-1}|\hat{b}_{s/L}|^2$ and let $q := s/L$. Let $T = LT' + \gamma$ for some $T', \gamma \in \mathbb{N}$ with $0 \leq \gamma < L$. Then, we have

$$\begin{aligned}
\left|\mathcal{R}\left((a_j)_{j=1}^{T}; q\right)\right| &\geq \left|\frac{1}{T'}\sum_{t=0}^{T'-1}\left[\frac{1}{L}\sum_{s=1}^{L} a_{Lt+s} e^{i2\pi qs}\right]\right| - \frac{L\sup_{t\geq 1}|a_t|}{T} \\
&> |\hat{b}_q| - \mu - \frac{L\sup_{t\geq 1}|a_t|}{T} \\
&\geq \sqrt{\frac{\sigma^2 - 2\mu\sigma}{L}} - \mu - \frac{L\sup_{t\geq 1}|a_t|}{T}.
\end{aligned}$$

Next, we prove (6.3.2). As in the same computation as above, we have

$$\begin{aligned}\left|\mathcal{R}\left((a_j)_{j=1}^T;\alpha/\beta\right)\right| &\leq \frac{LT'}{T}\left|\frac{1}{T'}\sum_{t=0}^{T'-1}e^{i2\pi\alpha Lt/\beta}\left[\frac{1}{L}\sum_{s=1}^{L}a_{Lt+s}e^{i2\pi\alpha s/\beta}\right]\right| + \frac{L\sup_{t>0}|a_t|}{T} \\ &< \frac{LT'}{T}\left|\frac{1}{T'}\sum_{t=0}^{T'-1}e^{i2\pi\alpha Lt/\beta}\right|\cdot\left|\frac{1}{L}\sum_{s=1}^{L}b_se^{i2\pi\alpha s/\beta}\right| + \mu + \frac{L\sup_{t>0}|a_t|}{T} \\ &< \frac{2}{|1-e^{i2\pi\alpha L/\beta}|}\cdot\left(\mu+\sup_{t\geq1}|a_t|\right)\cdot\frac{L}{T}+\mu+\frac{L\sup_{t>0}|a_t|}{T}.\end{aligned}$$

Since $|1-e^{i2\pi a}| \geq \sqrt{2}(1-a^2)^{\pi^2/6}a$ for $a \in (0,1)$ by [61, Proposition 3.2], we have

$$\begin{aligned}\left|\mathcal{R}\left((a_j)_{j=1}^T;\alpha/\beta\right)\right| &< \left(\mu+\sup_{t\geq1}|a_t|\right)\frac{2\beta L}{\sqrt{2}(3/4)^{\pi^2/6}T}+\mu+\frac{L\sup_{t\geq1}|a_t|}{T} \\ &< \mu + \frac{L\beta\mathcal{C}_0(\mu+\sup_{t\geq1}|a_t|)}{T}.\end{aligned}$$

□

Then, we obtain the explicit lower bound of the samples for period estimation:

Proposition 6.3.2. *Let* $\mathbf{a} := (a_t)_{t=1}^\infty$ *be a* μ*-nearly periodic sequence of length* L*. Fix a positive integer* $L_{\max} > 1$ *with* $L \leq L_{\max}$*,* $\delta \in (0,1)$*,* $\xi \in (0,1)$*, and* $\sigma_0 > 0$*. Let* $(\eta_t)_{t=1}^\infty$ *be a noise sequence satisfying Assumption 6.2.5. Put* $\gamma := 1/(1+\sqrt{4L_{\max}+1})$ *and* $\lambda := \mu/(\sigma_0\gamma)$*. We define*

$$\varepsilon := \sigma_0\gamma\xi.$$

If $\mu/(\gamma\xi) < \sigma_0 \leq \sigma_L(\mathbf{a})$*, then, for any*

$$T \geq \frac{8L_{\max}R^2\log(4/\delta)}{\sigma_0^2(\xi-\lambda)^2} + \frac{36L_{\max}^{5/2}\sup_{t\geq1}|a_t|}{\sigma_0(\xi-\lambda)},$$

the set of rational numbers

$$S_{T,\varepsilon} := \left\{q\in\mathbb{Q}\cap(0,1) : qL\in\mathbb{Z}_{>0} \text{ and } \left|\mathcal{R}\left((a_t+\eta_t)_{t=1}^T;q\right)\right|>\varepsilon\right\}$$

is nonempty with probability at least $1-\delta$*.*

If we apply several collections of rewards $(r_t(x_t))_{t=1}^T$ for sufficiently large T indicated in Proposition 6.3.2, we obtain various divisors of L. Finally, we provide the precise inputs and output of Algorithm 4 in the following Theorem:

Theorem 6.3.3. *Suppose Assumptions 6.2.4, 6.2.5 and 6.2.6 hold. Let $r \in [0,1)$ be a nonnegative real number, and suppose $\rho > 0$ and $\delta \in (0,1)$ are given. Fix a positive integer $L_{\max} > 1$ with $L \leq L_{\max}$. We define*

$$\varepsilon := \frac{\rho}{6\sqrt{d}L_{\max}}. \tag{6.3.4}$$

Assume that $r\varepsilon \geq \mu$. Let T_p be an integer satisfying

$$T_p \geq \frac{72dAL_{\max}^2}{\rho^2(1-r)^2} + \frac{108B_\theta\sqrt{d}L_{\max}^3}{\rho(1-r)}, \tag{6.3.5}$$

where $A := R^2 \log(4dL_{\max}^2 \log L_{\max}/\delta)$. Then, the output $\hat{L}$ of Algorithm 4 is a $(\rho, \sqrt{d})$-anp of $(f^t(\theta))_{t=t_0}^\infty$ with probability at least $1-\delta$.

If μ is sufficiently small, we may set r as a small positive number, in particular $r = 0$ if the system is periodic.

Remark 6.3.4. If random arm selection is adopted rather than the orthogonal basis, it may underestimate an error margin on some dimensions, which could lead to the nearly period with much larger error margin than expected; considering failure probability of such a case may potentially produce a variant of our algorithm.

6.3.2 Eigenvalue estimation

If the underlying system has certain structures, more detailed information about the system is expected to be obtained. In this section, we assume the following condition, linearity of the underlying dynamical system f on Θ, in addition to Assumption 6.2.4, 6.2.5, and 6.2.6:

Assumption 6.3.5 (Linear dynamical systems)**.** *The dynamical system $f : \Theta \to \Theta$ is linear and is represented by a matrix $M \in \mathbb{R}^{d\times d}$.*

Let $\mathbb{C}^d = \oplus_\alpha V_\alpha$ be the decomposition via generalized eigenspaces of M, where α runs over the eigenvalues of M and $V_\alpha := \mathscr{N}((\alpha I - M)^d)$. We describe $\theta = \sum_\alpha \theta_\alpha$ with $\theta_\alpha \in V_\alpha$. We remark that an eigenvalue α of M such that $\theta_\alpha \neq 0$ is in the form of $e^{2\pi i \ell / L}$ unless $|\alpha| < 1$ by Proposition 6.2.2.

Our objective is to estimate some of, if not all of, the eigenvalues of M with high probability within some error that decreases by the sample size. To this end, we define the meaningful subset of eigenvalues of M.

Definition 6.3.2 ((θ, k)-distinct eigenvalues)**.** For a vector $\theta \in \mathbb{C}^d$ and $k \in \mathbb{Z}_{>0}$, we define a (θ, k)-distinct eigenvalue by an eigenvalue β of M^k such that $|\beta| = 1$ and $\theta_\beta \neq 0$.

In our case, starting from a vector θ, the effect of the eigenvalues that are not of (θ, d)-distinct eigenvalues of M may not be observable. Basically, once being able to ignore the effects of eigenvalues of magnitudes less than 1, the system becomes nearly periodic and we aim at estimating (θ, d)-distinct eigenvalues as we obtain more samples.

Our eigenvalue estimation algorithm is summarized in Algorithm 5; it maintains the following matrices. For $N \in \mathbb{Z}_{>0}$, d random unit vectors $\tilde{x}_1, \ldots, \tilde{x}_d$, and $s = 0, 1$, we define the matrix $A_s(N) \in \mathbb{C}^{d \times d}$ so that its (k, ℓ) element is given by

$$\sum_{j=0}^{N-1} r_{2(k-1)d+sd+2d^2j+N+\ell}(\tilde{x}_k) e^{\frac{i2\pi j^2}{4L}}. \tag{6.3.6}$$

That is, after N steps, the reward multiplied by $\exp(i2\pi j^2/4L)$ is placed from the top row of A_0 and then the top row of A_1, followed by the second rows of them, and so on. Then, those values are summed up for every $2d^2$ steps or every jth cycle. Here, after throwing away N samples, the effects of eigenvalues of magnitude less than 1 become negligible, and the trajectory becomes nearly periodically behaved under Assumption 6.3.5. The rest of the samples is used to average out the observation noise while maintaining some meaningful information about M.

The aforementioned exponential sum can be characterized by the Weyl-type sum of matrices, a key machinery for our algorithm, which we define below:

Algorithm 5 Eigenvalue estimation

```
    Input: Effective sample size $N \in \mathbb{Z}_{>0}$; threshold $\gamma(N)$
    Output: Matrix $A_1(N)\hat{A}_0(N)^+$
 1: Independently draw random unit vectors $\tilde{x}_m,\ m \in \{1, \dots, d\}$, from uniform distribution over the unit sphere in $\mathbb{R}^d$.
 2: Wait $N$ time steps.
 3: for $t = N+1, \cdots, N+2Nd^2$ do
 4:     $m_0 \leftarrow \{(t-N-1) \mod 2d^2\} + 1$
 5:     $m \leftarrow \lceil m_0/2d \rceil$
 6:     Sample arm $x_t = \tilde{x}_m$ and observe $\tilde{r}_t := r_t(\tilde{x}_m)$.
 7: end for
 8: Construct matrices $A_0(N)$ and $A_1(N)$ as in (6.3.6), respectively.
 9: Obtain the low rank approximation $\hat{A}_0(N)$ of $A_0(N)$ via SVD with the threshold $\gamma(N)$.
10: Output $A_1(N)\hat{A}_0(N)^+$.
```

Definition 6.3.3. Let W be a linear space and let $M_1, \dots, M_N$ for $N \in \mathbb{Z}_{>0}$ be linear maps on W. For $L \in \mathbb{Z}_{>0}$, we define

$$\mathscr{W}\left((M_1, \dots, M_N)\right) := \frac{1}{N}\sum_{j=0}^{N-1} M_{j+1} e^{\frac{i2\pi j^2}{4L}}.$$

Remark 6.3.6. Let $E_{s,n} := (\eta_{2di+N+sd+j+1+2d^2n})_{i,j=0,\dots,d-1}$ be a noise matrix for $s = 0, 1$. Let

$$X := \begin{bmatrix} \tilde{x}_1^\top M^1 \\ x_2^\top M^{2d+1} \\ \vdots \\ \tilde{x}_d^\top M^{2(d-1)d+1} \end{bmatrix} \quad \text{and } K := (M\theta, \dots, M^d\theta).$$

Then, $A_s(N)$ has an alternative description as follows:

$$A_s(N) = X\mathscr{W}\left((M^{2d^2 j})_{j=0}^{N-1}\right) M^{sd+N-1}K + \mathscr{W}\left((E_{s,j})_{j=0}^{N-1}\right). \tag{6.3.7}$$

As in Proposition 6.3.8 below, the Weyl-type sum has a crucial property. Define $\kappa \geq 1$ by

$$\kappa := \inf_{P} \left\{ \|P\| \|P^{-1}\| : P^{-1} M P = J_M \right\},$$

where J_M is a Jordan normal form of M. Also, we define $\Delta \in (0, 1]$ to be a value such that, for any eigenvalue α of M satisfying $|\alpha| < 1$, $|\alpha| \leq 1 - \Delta$ (define $\Delta = 1$ if no such eigenvalue exists). Note by Proposition 6.2.2, the existence of such spectral gap is guaranteed without any further assumptions.

Roughly speaking, Algorithm 5 estimates "$A_1(N) A_0(N)^{-1} = X M^d X^{-1}$". Of course, the formula in "..." is not valid as X, K, and the Weyl-type sum are not necessarily invertible and we cannot recover full information of M^d in general. However, we can still reconstruct information of M^d restricted on the eigenspaces for (θ, d)-distinct eigenvalues.

To see this, we introduce the Weyl sum and its lower bound:

Lemma 6.3.7 (Lower bound on the Weyl sum [44, 187])**.** *Define the Weyl sum by*

$$\mathscr{W}(N, b, q) := \sum_{j=0}^{N} e^{i 2\pi \left(j \frac{2b}{4q} + j^2 \frac{1}{4q} \right)},$$

for some $b \in \mathbb{N},\ q \in \mathbb{Z}_{>0},\ b < q$ and $N \in \mathbb{Z}_{>0}$. Then, for $N \geq 16 q^2$, it holds that

$$|\mathscr{W}(N, b, q)| \in \Omega\left(\frac{N}{\sqrt{q}} \right).$$

Proof. It is immediate from Proposition 3.1 in [187] because $\gcd(1, 4q) = 1$, $4q \equiv 0 \mod 4$, and $2b$ is even. □

We define $C_{\mathrm{ws}}(L) > 0$ by

$$C_{\mathrm{ws}}(L) := \inf \left\{ C > 0 : C^{-1} < \left| \frac{1}{N} \mathscr{W}(N, b, L) \right| < C \text{ for any } N \geq 16 L^2, 0 \leq b < L \right\}.$$

Then, we have the following proposition:

Proposition 6.3.8. *Let M_1 and $M_{<1}$ be matrices as in Corollary 6.2.3. We define linear maps on W by*

$$\begin{aligned} Q_N(M_1) &:= \mathscr{W}\left((M_1^{2d^2j})_{j=0}^{N-1}\right) \\ Q_N(M_{<1}) &:= \mathscr{W}\left((M_{<1}^{2d^2j})_{j=0}^{N-1}\right). \end{aligned}$$

Then, we have the following statements:

1. *$\mathscr{W}\left((M_\theta^{2d^2j})_{j=0}^{N-1}\right) = Q_N(M_1) + Q_N(M_{<1})$,*

2. *for any $N \geq 16L^2$, $Q_N(M_1)$ is invertible on $W_{=1}$,*

3. *for any $r \geq 0$ and for any $N \geq 16L^2$, $\|M_1^r Q_N(M_1)|_{W_{=1}}\|, \|(M_1^r Q_N(M_1))|_{W_{=1}}^{-1}\| \leq \kappa C_{\mathrm{ws}}(L)$,*

4. *for any $r \geq 2(d-1)$, we have*

$$\|M_{<1}^r Q_N(M_{<1})\| \leq \frac{d^2 \kappa e^{-\Delta(r-d+1)}}{N\Delta^{d-1}}.$$

Proof. We prove 1. By the properties 1 and 2 in Corollary 6.2.3, we have $\mathscr{W}((M^{2d^2j})_{j=0}^{N-1}) = Q_N(M_1) + Q_N(M_{<1})$. Next, we prove 2. When we regard $Q_N(M_1)|_{W_{=1}}$ as a linear map on $W_{=1}$, it is represented as a diagonal matrix $\mathrm{diag}(\mathscr{W}(N, b_1, L), \ldots, \mathscr{W}(N, b_m, L))$, where $m = \dim W_{=1}$. Therefore, by Lemma 6.3.7, $Q_N(M_1)$ is a bijective linear map on $W_{=1}$. Next, we prove 3. We estimate $\|M_1^r Q_N(M_1)\|$. Let $p : \mathbb{C}^d \to W$ be the orthogonal projection and let $i : W \to \mathbb{C}^d$ be the inclusion map. Let $\tilde{M}_1 := iM_1p \in \mathbb{C}^{d\times d}$. Then, we have

$$\|Q_N(M_1)\| = \|Q_N(\tilde{M}_1)\| \leq \kappa C_{\mathrm{ws}}(L).$$

Next, we estimate $\|M_{<1}^r Q_N(M_{<1})\|$. Let $\tilde{M}_{<1} := iM_{<1}p$. Then, $iM_{<1}^r Q_N(M_{<1})p = \mathscr{W}((\tilde{M}_{<1}^{2d^2j+r})_{j=0}^{N-1})$ and $\|iM_{<1}^r Q_N(M_{<1})p\| = \|M_{<1}^r Q_N(M_{<1})\|$. Let P be a regular matrix such that

$$\tilde{M}_{<1} = PJP^{-1},$$

where J is the Jordan normal form. Then, for $r \geq 2(d-1)$, we see that

$$\begin{aligned}
\|M_{<1}^r Q_N(M_{<1})\| &= \|iM_{<1}^r Q_N(M_{<1})p\| \\
&\leq \frac{\kappa}{N}\sum_{j=0}^{N-1} \|J^{2d^2j+r}\| < \frac{d^2\kappa}{N}\sum_{j=0}^{\infty}\binom{2d^2j+r}{d-1}(1-\Delta)^{2d^2j+r-d+1} \\
&\leq \frac{d^2\kappa(1-\Delta)^{r-d+1}}{N\Delta^{d-1}} \leq \frac{d^2\kappa e^{-\Delta(r-d+1)}}{N\Delta^{d-1}}.
\end{aligned}$$

The second last inequality is proved as follows: for $|a| < 1$, $n \geq 1$ and $r \geq m \geq 0$,

$$\begin{aligned}
\left|\sum_{j=0}^{\infty}\binom{nj+r}{m}a^{nj-m+r}\right| &= \left|\frac{1}{m!}\frac{d^m}{dx^m}\sum_{j=0}^{\infty}x^{nj+r}\bigg|_{x=a}\right| \\
&\leq \frac{1}{m!}\sum_{j=0}^{m}\binom{m}{j}\frac{r!}{(r-m+j)!}|a|^{r-m+j}\left|\frac{d^j}{dx^j}\frac{1}{1-x^n}\bigg|_{x=a}\right| \\
&\leq \frac{1}{m!}\sum_{j=0}^{m}\binom{m}{j}\frac{r!}{(r-m+j)!}\frac{j!|a|^{r-m+j}}{(1-|a|)^j} = \sum_{j=0}^{m}\binom{r}{j}\frac{|a|^{r-m+j}}{(1-|a|)^j},
\end{aligned}$$

where, the last inequality follows from

$$\left|\frac{d^m}{dx^m}\frac{1}{1-x^n}\right| = \left|\sum_{\zeta^n=1}\frac{m!\zeta^{1-n}}{n(\zeta-x)^m}\right| \leq \frac{m!}{(1-|x|)^m}.$$

Thus, we have

$$\sum_{j=0}^{m}\binom{r}{j}\frac{|a|^{r-m+j}}{(1-|a|)^j} \leq |a|^{r-m}\sum_{j=0}^{r}\binom{r}{j}\left(\frac{|a|}{1-|a|}\right)^j \leq \frac{|a|^{r-m}}{(1-|a|)^m}.$$

□

Proposition 6.3.8 plays an essential role in our analysis and guarantees that the information in $A_s(N)$ about (θ, d)-distinct eigenvalues does not vanish while noise effects are canceled out. Now, we state our main theoretical result for the eigenvalue estimation algorithm. To this end, we introduce lower rank approximation via the singular value threshold.

Definition 6.3.4. Let $A \in \mathbb{C}^{n\times m}$ be a matrix. Let $A = U\,[D\ O]\,V^\dagger$ be a singular valued decomposition of A where $U \in \mathbb{C}^{n\times n}$ and $V \in \mathbb{C}^{m\times m}$ are unitary matrices and $D :=$

$\mathrm{diag}(\sigma_1, \ldots, \sigma_r, \mathbf{o}^\top)$ is a diagonal matrix with nonnegative components. Let $\gamma > 0$. We define a low rank approximation A_γ of A via the singular value threshold γ by the matrix A_γ defined by $A_\gamma = U\,[D_\gamma\ O]\,V^\dagger$, where, $D_\gamma := \mathrm{diag}(\mathbf{1}_{[\gamma,\infty)}(\sigma_1)\sigma_1, \ldots, \mathbf{1}_{[\gamma,\infty)}(\sigma_r)\sigma_r, \mathbf{o}^\top)$ and $\mathbf{1}_I$ is defined to be the characteristic function supported on $I \subset \mathbb{R}$.

Given this definition, we are ready to present the following main result:

Theorem 6.3.9. *Suppose Assumptions 6.2.4, 6.2.5, 6.2.6, and 6.3.5 hold. Given $\delta \in (0, 1]$, let the effective sample size*

$$N \geq \max\left\{16L^2, \frac{-(d-1)\log\Delta}{\Delta} + \frac{\log(B_\theta \kappa^2) + d + 6}{\Delta} + d\right\}, \tag{6.3.8}$$

and $\gamma(N) = (\sqrt{4d^2R^2\log(4d^2/\delta)} + 1)/\sqrt{N}$. Then, there exists a matrix A whose eigenvalues are zeros except for (θ, d)-distinct eigenvalues of M, such that the output of Algorithm 5, i.e. $A_1(N)\hat{A}_0(N)^+$, satisfies, with probability at least $1 - \delta$, that

$$\left\|A - A_1(N)\hat{A}_0(N)^+\right\| \leq C\left(\frac{R^2(\log(1/\delta) + 1) + 1}{\sqrt{N}}\right). \tag{6.3.9}$$

Here, $\hat{A}_0(N)^+$ is the Moore-Penrose pseudo inverse of a lower rank approximation of $A_0(N)$ via the singular value threshold $\gamma(N)$. The constant $C > 0$ depends on θ, M, d, $(\tilde{x}_m)_{m=1}^d$, and $C_{\mathrm{ws}}(L)$.

We mention that by using the results shown in [227], the bound on spectral norm (6.3.9) can be translated to the bounds on eigenvalues, where the constant depends on the form of A. As described in Theorem 6.3.9, the constant is not the absolute constant for any problem instance but depends on several factors; however, for the same execution, this rate is useful to judge how many samples one collects to estimate eigenvalues.

Remark 6.3.10. We note that we can reconstruct the $(\theta, 1)$-distinct eigenvalues of M via Algorithm 5 using the following trick: Fix nonnegative integer $r \geq 0$. Take $d + r$ random unit vectors $\tilde{x}_1, \ldots, \tilde{x}_{d+r}$. Then, for $s = 0, 1$, we may define a matrix $A_s(N; r) \in \mathbb{C}^{(d+r)\times(d+r)}$ so that its (k, ℓ) element is given by

$$\sum_{j=0}^{N-1} r_{2(k-1)(d+r)+s(d+r)+2(d+r)^2 j+N+\ell}(\tilde{x}_k)e^{\frac{i2\pi j^2}{4L}}.$$

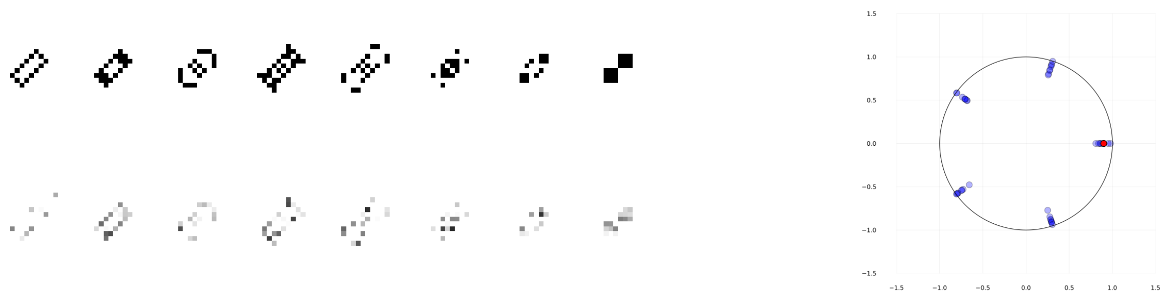


Figure 6.2: Left: Illustration of a period eight instance of LifeGame; (top) original transitions. (down) an instance of noisy observation. Right: μ-nearly periodic dynamics (6.4.1).

Then, we see that Algorithm 5 outputs a matrix $\tilde{A}(r)$ that well approximates the $(\theta, d+r)$-distinct eigenvalues of M since the matrix $A_s(N;r)$ coincides with $A_s(N)$ in the case when we replace M and θ with $M(r)$ and $\theta(r)$ defined by

$$M(r) := \begin{bmatrix} M & O \\ O & O \end{bmatrix} \in \mathbb{C}^{(d+r)\times(d+r)}, \theta(r) := \begin{bmatrix} \theta \\ \mathbf{o} \end{bmatrix} \in \mathbb{C}^{d+r}.$$

Let r be an integer such that $r+d$ is prime to L and fix a positive integer m such that $m(r+d) \equiv 1 \mod L$. Then, the eigenvalues of $\tilde{A}(r)^m$ is close to those of $(\theta, 1)$-distinct eigenvalues if we take sufficiently large N.

6.4 Simulated experiments

In this section, we present simulated experiments that complement the theoretical claims. In particular, we conducted period estimations for an instance of LifeGame [66], which is a special case of cellular automata [253], and for a nearly periodic toy system, and an eigenvalue estimation for a linear system, where some dimensions are for permutations and the rest is for shrinking.

Period estimation; LifeGame: We use a specific instance of LifeGame which is illustrated in Figure 6.2. As shown on the top eight pictures, starting from certain configuration of cells, it shows transitions of period eight. The sample size is computed as the smallest integer satisfying (6.3.5), and the threshold ε is given by (6.3.4). To prevent the dimension from becoming too large, we used five cells that correctly display period eight; that is $d = 5$. Noise η_t is given by i.i.d. Gaussian with proxy $R = 0.3$, and the down eight pictures of Figure 6.2 are some instances of noisy observations. We tested 50 different random seeds (i.e., 1, 51, 101, 151, ... , 2451), and computed the error rate (the number of runs producing a wrong estimate other than the fundamental period eight, which is divided by 50); and it was zero.

Period estimation; Simple μ-nearly periodic system: We consider the following μ-nearly periodic system that circulates over a circle with small variations:

$$r_{t+1} = \mu\left(\alpha\frac{r_t - 1}{\mu} - \lceil\alpha\frac{r_t - 1}{\mu}\rceil\right) + 1, \qquad \theta_{t+1} = \theta_t + \frac{2\pi}{L}, \tag{6.4.1}$$

where r and θ are the radius and angle, and $\alpha \notin \mathbb{Q}$. We use $\mu = 0.001$, $L = 5$, and $\alpha = \pi$. Noise η_t is drawn i.i.d. from the uniform distribution within $[-R, R]$ for $R = 0.3$. We tested 50 different random seeds (i.e., 1, 51, 101, 151, ... , 2451), and computed the error rate; and it was zero.

Eigenvalue estimation; Permutation and shrink: We use $d = 5$, and $M \in \mathbb{R}^{5\times 5}$ is made such that 1) the first four dimensions are for permutation (i.e., each of row and column of 4×4 sub-matrix has only one nonzero element that is one.), and 2) the last dimension is simply shrinking; we gave 0.7 for $(5,5)$-element of M. Initial vector θ_0 and each arm $\tilde{x}_m$, $m \in [d]$, are uniformly sampled from the unit sphere in $\mathbb{R}^5$. The value L is computed by $4! = 24$. We used the smallest integer N that satisfies (6.3.8), multiplied by $C_{\text{sim}} > 0$. The results are shown in Table 6.1; it is observed that the more samples we use the more accurate the estimates become to $(\theta_0, 5)$-distinct eigenvalues of M. Noise η_t is drawn i.i.d. from the uniform distribution within $[-R, R]$ for $R = 0.3$, and the Table 6.1 is of the random seed 1234. We also tested 50 different random seeds (i.e., 1, 51, 101, 151, ... , 2451) for $C_{\text{sim}} = 30$,

Table 6.1: Results for the eigenvalue estimation. The top-most row shows the true eigenvalues of M^5, and the second row shows its $(\theta_0, 5)$-distinct eigenvalues. From the third to sixth row, it shows the estimated eigenvalues for different values of C_{sim}.

eigenvalues of M^5	1.000	1.000	$-0.500 - 0.866i$	$-0.500 + 0.866i$	0.168
$(\theta_0, 5)$	1.000	0	$-0.500 - 0.866i$	$-0.500 + 0.866i$	0
when $C_{\text{sim}} = 1$	$1.000 + 0.008i$	0	$-0.489 - 0.860i$	$-0.510 + 0.869i$	0
when $C_{\text{sim}} = 5$	$0.999 - 0.000i$	0	$-0.495 - 0.867i$	$-0.501 + 0.862i$	0
when $C_{\text{sim}} = 10$	$1.000 + 0.003i$	0	$-0.497 - 0.867i$	$-0.501 + 0.864i$	0
when $C_{\text{sim}} = 30$	$1.000 + 0.001i$	0	$-0.500 - 0.866i$	$-0.500 + 0.864i$	0

and computed the mean absolute error between the true $(\theta_0, 5)$-distinct eigenvalues and their nearest estimated values; it was 0.0019, which is sufficiently small.

Computation: Throughout the experiment in Section 6.4, we used the following version of Julia [41]; for each experiment, the running time was less than a few minutes.

```
Julia Version 1.6.3
JULIA_NUM_THREADS = 12
OS: Linux (x86_64-pc-linux-gnu)
CPU: Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz
WORD_SIZE: 64
LIBM: libopenlibm
LLVM: libLLVM-11.0.1 (ORCJIT, broadwell)
```

6.5 Applications to bandit problems

We briefly cover the applicability of our proposed algorithms to bandit problems (e.g., regret minimization).

A naive approach is an explore-then-commit type algorithm (cf. [205, 20]). One employs our algorithm to estimate a nearly period, followed by a certain periodic bandit algorithm such as the work in [52] to obtain an asymptotic order of regret. Caveat here is, because our estimate is only an aliquot nearly period, one may need to take into account the regret caused by this misspecification when running bandit algorithms (e.g., ρ and μ may lead to (small) linear regret). Avoiding this small linear regret would require the system to be 0-nearly periodic and that there exists a sufficiently large *gap* ensuring μ-nearly period with sufficiently small μ implies 0-nearly period.

If one aims at designing an anytime algorithm, the straightforward application of our algorithms may not give near optimal asymptotic rate of *expected* regret because the failure probability of periodic structure estimations cannot be adjusted later. To remedy this, one can employ our algorithm repetitively, and gradually increase the span of such procedure. Importantly, samples from separated spans can contribute to the estimate together when the *surplus* beyond a multiple of period is properly dealt with. Since failure probability decreases exponentially with respect to sample size, we conjecture that increasing the span for bandit algorithm by a certain order will lead to the same rate (up to logarithm) of *expected* regret of the adopted bandit algorithm.

6.6 Chapter summary and discussion

This chapter proposed novel algorithms for estimating periodic information about the dynamical systems from bandit feedback contaminated by a sub-Gaussian noise. Statistical sample complexities for our algorithms have been provided which rely on the asymptotic results of exponential sums, including the Weyl sum. The obtainable (statistically important) estimates of the dynamic structures have been carefully and mathematically defined.

Our work is fundamental and should constitute the research on reconstruction of dynamical systems; however, because our problem setups are rather novel, its applicability to the real world problems is somewhat unknown. Also, the lower bound arguments or elimination of dependence on the upper bound $L_{\max}$ through systematic guessing should be investigated. Note, if the hyperparameters are correctly chosen for nearly periodic systems, the outputs of our algorithms are consistent and reasonable across runs, following the theoretical insights; therefore, practically, one can check if the hyperparameters are properly chosen by running the algorithms.

This work is a first attempt of applying the Weyl sum, a variant of exponential sums, to statistical estimation problems and the uses of other types of sums or number theoretic tools in this domain remain unexplored.

Finally, this chapter only considered (nearly) periodic dynamics, which is a mere fraction of interesting dynamical phenomena; and extending similar arguments to more general (random) dynamical systems including some nonlinear attractor dynamics will be an important direction of research.

Chapter 7

CONSTRUCTIVE APPROACH FOR ANALYZING COMPLEX MACHINE LEARNING ALGORITHMS AS DYNAMICAL SYSTEMS

Chapter's key takeaways

This chapter presents a constructive approach for analyzing some of the phenomena observed in deep RL algorithms employing different critic loss functions (including distributional RL). The key findings are as follows:

1. Theoretically, gradient w.r.t. quantile-Huber loss behaves the same as that of the Huber loss under certain conditions while minimization of the quantile-Huber loss, as an approximation of quantile loss, tends to better estimate the mean of the target than Huber loss does as a proxy for $\ell 1$ loss when the mean and median of the target differ
2. RL algorithms with MSE, Huber and quantile-Huber critic loss functions can show behavior separations even in very simple toy tabular MDP cases, for which the critical causes are the interaction of critic and action selection
3. For such a toy abstracted system that is made to consist of *guiding* rewards in addition to a sparse large reward/penalty, the use of (quantile-)Huber loss helps the critic and action selection evolve robustly, which leads to stable performance increase
4. For such a toy abstracted system that is made without *guiding* rewards, the use of MSE loss shows better performance; this observation enables straightforward creation of an environment where the use of MSE loss leads to a better performance
5. For more complex systems, metrics and visualizations hint at some connections to the behaviors of abstracted systems and help deepening the understanding; however the findings stay largely hypothetical

7.1 Introductions

Concurrently with the empirical success of distributional RL highlighted by the superhuman racing AI agent, Gran Turismo Sophy, theoretical investigations of convergence and comparisons against the traditional (expected) RL algorithm have been made.

However, elucidating the rationale behind this success has proven to be a challenge due to the inherent complexity of deep RL algorithms. This challenge penetrates through most of the modern complex ML domains, including large language models (cf. [208]). The complexity hinders straightforward deductive or inductive reasoning about the observable phenomena, and it is particularly pronounced in the domains such as RL, where dynamic interactions among the learner (including an actor and a critic), environment and even human users play a key role.

In this chapter, we aim at tackling this challenge by focusing on algorithmic behaviors generated by the soft-actor-critic (SAC) algorithm [94] with three types of critic loss functions, namely, the mean-squared-error (MSE), Huber [104], and quantile-Huber [72] (distributional RL) loss functions. Inspired by another set of results [55, 175] indicating the use of Huber loss for critic learning oftentimes (if not always) produces better learning curves, we ask the following question *"How does the choice of critic loss affect the learning curve? And how does it depend on the agent environment?"*. In particular, the novelty of our work lies in the methodology we employ, a constructive approach (see Chapter 1) tailored to our problem. See Figure 7.1 for illustration.

Our methodology begins with (approximately) categorizing the selected learning behaviors based on some observation. Subsequently, it aims at finding a model (1) involving *essential* components (such as the critic and action selection interaction) of the target systems, (2) having simplicity that allows clear reasoning about the behavior, and (3) showing behaviors of interest, based on which we can at least identify sufficient conditions for some class of learning behaviors to emerge. In our case, we naturally employ the learning curve of cumulative rewards for categorizing the behaviors, and consider simple forms of MDPs as

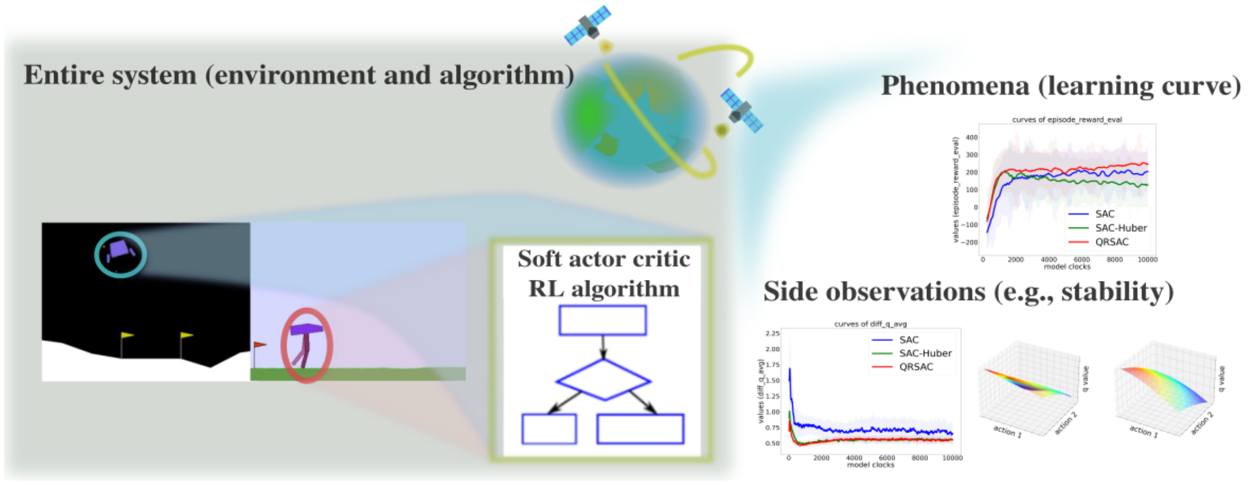

Entire system (environment and algorithm)
Phenomena (learning curve)
Soft actor critic RL algorithm
Side observations (e.g., stability)
curves of episode_reward_eval
SAC
SAC-Huber
QRSAC
model clocks
curves of diff_q_avg
q value
action 1
action 2

Figure 7.1: In this chapter, we treat the environment coupled with the RL algorithms (SAC with different critic loss) as the entire system to take constructive approach to analyze the phenomena (learning curves in our case) by using observable information.

candidates of such models; however, we mention that the process of selecting viable categorizations and models is the critical part of our methodology.

Finally we use a few measurable metrics and visualization that we propose to *retroductively* study about the phenomena observed in the target (more complex) systems. As there may not be a single common and valid cause across complex environments and due to the imperfect observations that can be made, our findings stay hypothetical and carry an attempt of *affirming the consequent* although we also conduct experiments aimed at eliminating or weakening other potential hypotheses; however, we believe our work will lead to further research on the methodology and for other ML domains.

Contributions: The contributions are five folds: (1) present theoretical analyses that expose some of the basic logic behind the behavior difference caused by the choice of critic loss, (2) present *behavior class* to categorize the learning curves of algorithms with different critic losses, (3) propose abstracted systems whose learning behaviors fall into the proposed behavior classes and are logically predictable, (4) propose a few metrics that help quantify the behaviors observed in more complex environments, and (5) present an array of experimental results.

7.2 Problem setups

In this chapter, we consider the MDP environment described in Section 2.1.3. The problem of interest in this chapter is described as follows: given an MDP $\mathcal{M}$ and an initial state distribution $\mu \in \Delta(\mathcal{S})$, find an algorithm A that returns a policy π^* which maximizes the expected return $\mathbb{E}\left[Q^\pi(s_0, a_0)\middle| s_0 \sim \mu, a_0 \sim \pi(s_0)\right]$.

Constructive approach

An illustration of constructive approach is depicted in Figure 1.9. A notable example of constructive approach may be seen in the Lorenz attractor [153]; although this abstract system is too simplified to represent the dynamics of atmospheric convection and weather, it shows that such a simple system can show chaotic behavior (i.e., the sensitivity to the initial conditions), which implies the inherent difficulty of weather forecast up to some extent. In this case, interactions of three *variables* are the essential components to look at in the target system while they are simplified in a way that the behavior can be analyzed thoroughly.

Dynamical system view of RL algorithms

To apply constructive approach to RLs, we are implicitly viewing an algorithm interacting with an environment as a dynamical system, resembling the natural phenomena, which might be expressed by an RDS. Let $\mathcal{X}$ be a set of the *entire state* which consists of the states of agent, environment, and the learning algorithm. The dynamics over $\mathcal{X}$ is assumed to be an RDS. This way, we will also be able to properly define the metrics for analyzing RL algorithms, which we will see later.

7.3 Basic theoretical analyses

In this section, we give some of the basic theoretical analyses about the Huber critic loss and quantile-Huber loss.

7.3.1 Quantile-Huber loss revisited

The Huber loss $\mathcal{L}_\kappa : \mathbb{R} \to \mathbb{R}$ for a given constant $\kappa > 0$ is defined by

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \le \kappa \\ \kappa\left(|u| - \frac{1}{2}\kappa\right), & \text{otherwise} \end{cases}$$

and the quantile-Huber loss for the quantile τ, namely, $\rho_\tau^\kappa(u)$ is given by

$$\rho_\tau^\kappa(u) = \frac{|\tau - \delta_{\{u<0\}}|\mathcal{L}_\kappa(u)}{\kappa}.$$

Equivalence between Huber and quantile-Huber loss algorithms under certain regime: The quantile-Huber loss using a single head (i.e., there is only one quantile point $\tau_1 = \frac{1}{2}$) is equivalent to the Huber loss within a constant scale factor because

$$\rho_{\frac{1}{2}}^\kappa(u) = \left|\frac{1}{2} - \delta_{\{u<0\}}\right| \frac{\mathcal{L}_\kappa(u)}{\kappa} = \frac{1}{2\kappa}\mathcal{L}_\kappa(u).$$

Next, assume the quantile-Huber loss uses $N \in \mathbb{Z}_{>0}$ heads with the initial values $\mathbf{q}_0 := [q_{0,1}, \cdots, q_{0,N}] \in \mathbb{R}^N$. Assume also that $\max_{i\in\{1,2,\cdots,N\}}\{|q_{0,i}|\} < M$ for some $M > 0$, and let $c = \frac{1}{N}\sum_{i=1}^N q_{0,i}$. Now, suppose one does bootstrap update for the target quantile $\mathbf{y} = [y_i, \cdots, y_N]$ that satisfies $|y_i| > M + \kappa$ for all i. The simple gradient update is given by

$$\mathbf{q}_1 = \mathbf{q}_0 - \alpha \cdot \frac{\partial \ell(\mathbf{q}_0, y)}{\partial \mathbf{q}}, \quad \alpha \in (0,1], \quad \ell(\mathbf{q}_0, y) = \frac{1}{N^2}\sum_i \sum_j \rho_{\tau_j}^\kappa(y_i - q_j).$$

Here, under assumption[1] $y_i > M + \kappa$, we obtain

$$\rho_{\tau_j}^\kappa(y_i - q_j) = \frac{\tau_j}{\kappa}\mathcal{L}_\kappa(y_i - q_j) \rightsquigarrow -\frac{\partial \rho_{\tau_j}^\kappa(y_i - q_j)}{\partial q_j} = \tau_j \rightsquigarrow -\frac{\partial \ell(\mathbf{q}_0, y)}{\partial q_j} = \frac{1}{N^2}\sum_i \tau_j = \frac{1}{N}\tau_j.$$

[1]A similar argument can be applied for the case $y_i < -M - \kappa$ too.

Finally, the mean of $\mathbf{q}_1$ is given by $\frac{1}{N}\sum_{i=1}^{N} q_{1,i} = c + \alpha \cdot \frac{1}{N^2}\sum_j \tau_j = c + \frac{\alpha}{2N}$. On the other hand, for the quantile-Huber loss algorithm with a single head, where $q_{0,1} = c$, we obtain $q_{1,1} = c + \alpha/2$. As such, it is again equivalent within a constant factor.

$\ell 1$ loss bootstrap leads to wrong solution for some MDPs: Let $\{y_n\}_{n\in\{1,\ldots,N\}} \subset \mathbb{R}$ be a set of N observations, and $\hat{y}$ be the estimate. Define

$$L_2 := \frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y})^2, \quad L_1 := \frac{1}{N}\sum_{n=1}^{N}|y_n - \hat{y}|.$$

Then, it is straightforward to see that L_2 and L_1 are minimized when

$$\frac{\partial L_2}{\partial \hat{y}} = -\frac{2}{N}\sum_{n=1}^{N}(y_n - \hat{y}), \quad \frac{\partial L_1}{\partial \hat{y}} = -\frac{1}{N}\sum_{n=1}^{N}\text{sign } (y_n - \hat{y})$$

are zeros, respectively, which implies the well-known facts that $\hat{y}$ should be the mean (median) of the observations under $\ell 2$ ($\ell 1$) loss. For the temporal-difference (TD) error minimization in the context of RL, the work [197] points out that the use of the Huber loss resembles that of a *mean Huber TD error* rather than the Bellman error. Therefore, using the Huber loss (or $\ell 1$ loss) may lead to a wrong critic or even to nonconvergent behaviors. We will see a (potential) outcome of this theoretical fact in a case study in Section 7.4.4.

In the next section, we present a categorization of the learning behaviors, and present simple MDPs which show respective behaviors when coupled with some algorithm employing different loss functions.

7.4 Categorizations

7.4.1 Behavior classes

In this chapter, we employ the learning curve of cumulative rewards as the basis for categorizing behaviors, and mainly focus on the following four classes of learning behaviors. Those classes are irrelevant to environments or function approximators. Throughout, we use the term "(quantile-)Huber" as a shorthand for quantile-Huber and Huber implementations.

Class 1: No significant performance separation For this class of behaviors, the learning curves for RL algorithms with MSE, Huber and quantile-Huber critic loss show no significant performance difference.

Class 2: MSE loss algorithm performs worse than (quantile-)Huber loss algorithms For this class of behaviors, RL algorithm with MSE critic loss struggles to increase performance while other two loss functions show no degradation of performance.

Class 3: MSE loss algorithm performs better than (quantile-)Huber loss algorithms For this class of behaviors, RL algorithms with Huber and quantile-Huber critic loss functions show worse performances than the MSE counterpart.

Class 4: MSE loss algorithm performs poorly at the beginning and better at later stage For this class of behaviors, RL algorithm with MSE critic loss struggles to increase performance at first, but catches up with other two to eventually perform better than the Huber and quantile-Huber loss counterparts.

Remark 7.4.1 (On behavior classes)**.** These classes obviously do not cover all of the possible combinations of behaviors; in fact, some Gym environments show behaviors that may not straightforwardly fall into the presented behavior classes. Rather, we presented them in order to gain insights about how the stability of critic/policy growth and the properties of reward/dynamics are coupled to produce certain behaviors.

Below, we propose some representative MDP environments that possess desirable properties while preserving simplicity to allow theoretical arguments to reason about the behaviors.

7.4.2 Representative environments

For RL problems, we naturally consider tabular MDPs as candidate abstracted models for understanding the behaviors of target complex systems. We present two abstracted toy MDP topologies in Figure 7.2. Moreover, we employ a simple toy algorithm given in Algorithm 6

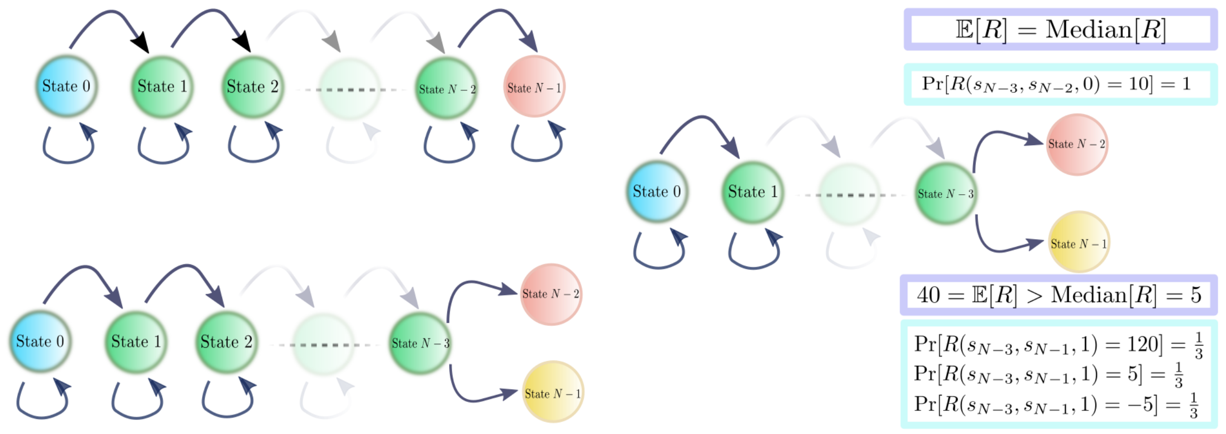

State 0
State 1
State 2
State $N-2$
State $N-1$
State 0
State 1
State 2
State $N-3$
State $N-2$
State $N-1$
$\mathbb{E}[R] = \text{Median}[R]$
$\Pr[R(s_{N-3}, s_{N-2}, 0) = 10] = 1$
State 0
State 1
State $N-3$
State $N-2$
State $N-1$
$40 = \mathbb{E}[R] > \text{Median}[R] = 5$
$\Pr[R(s_{N-3}, s_{N-1}, 1) = 120] = \frac{1}{3}$
$\Pr[R(s_{N-3}, s_{N-1}, 1) = 5] = \frac{1}{3}$
$\Pr[R(s_{N-3}, s_{N-1}, 1) = -5] = \frac{1}{3}$

Figure 7.2: An illustration of abstracted MDPs for categorization. Left Top: Topology for MDP Class 1 to MDP Class 3. Left Down: Topology for MDP Class 4 (omitting the self-transitions at the states $N-2$ and $N-1$). Right: An illustration of an MDP that demonstrates separation between ℓ_1 loss and Quantile loss algorithms.

(which is a modified ϵ-greedy Q learning algorithm), where the state and action spaces are $\mathcal{S} = \{s_0, s_1, \dots, s_{N-1}\}$ and $\mathcal{A} = \{0, 1\}$ at all of the states except for the final *done* states where it is the singleton $\mathcal{A} = \{0\}$. Assume, across all the classes, that $\alpha < 1/H$ and that $H > 2N$. In this section, we define *success conditions* to be those where the estimated Q values at each state lead to an optimal action selection that achieves the fastest reach to the *done* states. Note that the representative model of MDP Class 4 has two done states. We give the following classes of MDPs as representative environments for each behavior class; *we mention that those are a mere fraction of the entire space of MDPs but that they convey ideas of what causes the behavior separation.* For the four cases below, we suppose the reward function returns a scalar value deterministically.

MDP Class 1: *Guiding (smooth) rewards.* This class of MDPs is characterized by the abstracted system Figure 7.2 (left top) as follows: Action $a = 1$ increments the state. The

Algorithm 6 Algorithm for learning in the abstracted MDPs

```
1: Estimates of Q values ($\hat{Q}$) are set to zeros
2: for Episodes $t = 0, \dots$ do
3:     initial state $s_0$
4:     for Environment steps $h = 0, \dots H - 1$ do
5:         $\hat{Q}_{\text{copy}} \leftarrow \hat{Q}$
6:         Action selection $a = \min\{\arg\max_{a' \in \mathcal{A}} \hat{Q}(s_h, a')\}$ with $1 - \epsilon$ prob
7:         (otherwise uniformly select from $\mathcal{A} \setminus \{a\}$)
8:         Follow environment dynamics and observe data $d = (s_h, s_{h+1}, a, r_h)$
9:
           $$\hat{Q}_{\text{copy}}(s_h, a) \leftarrow \hat{Q}(s_h, a) + \alpha \nabla \ell \left( r_h + \max_{a'} \{\hat{Q}(s_{h+1}, a')\} - \hat{Q}(s_h, a) \right)$$
10:    end for
11:    $\hat{Q} \leftarrow \hat{Q}_{\text{copy}}$
12: end for
```

reward function R satisfies

$$R(s_i, s_i, 0) < R(s_i, s_{i+1}, 1), \quad R(s_{N-2}, s_{N-1}, 1) = R(s_{N-1}, s_{N-1}, 0) = 0, \quad \forall i \in \{0, \dots, N-2\},$$

and

$$R(s_i, s_{i+1}, 1) < R(s_{i+1}, s_{i+2}, 1), \quad \forall i \in \{0, \dots, N-3\},$$

meaning, the optimal action is *hinted at* in each step.

MDP Class 2: *Guiding (smooth) rewards + consistent sparse and large magnitude rewards.* This class of MDPs is characterized by the abstracted system Figure 7.2 (left top) as follows: Action $a = 1$ increments the state. The reward map satisfies

$$R = R_1 + R_2,$$

where R_1 satisfies the conditions of MDP Class 1 in addition to the bound $\|R_1\|_\infty < B$ for $B > 0$. R_2 is zero everywhere except for the transition at one of the states s_k in $\{s_0, \ldots, s_{N-3}\}$ giving

$$R_2(s_k, s_{k+1}, 1) < -BH.$$

MDP Class 3: *No guiding rewards.* This class of MDPs is characterized by the abstracted system Figure 7.2 (left top) as follows: Action $a = 1$ increments the state. The reward map satisfies

$$R(s_{N-3}, s_{N-2}, 1) \leq -3, \quad R(s_{N-2}, s_{N-1}, 1) > 4|R(s_{N-3}, s_{N-2}, 1)|,$$

and is zeros elsewhere, meaning, there is a *misleading* reward for the transition s_{N-3} to s_{N-2} to be overcome to reach $N - 1$.

MDP Class 4: *MDP Class 2 + necessity of fine-tuning of Q value.* This class is characterized by the abstracted system Figure 7.2 (left down) as follows: Action $a = 1$ increments the state except for s_{N-3} where either action leads to the final states s_{N-2} (action 0) or s_{N-1} (action 1). The reward function satisfies

$$R = R_1 + R_2 + R_4,$$

where R_1 follows the condition of that of MDP Class 1 up to the transition to s_{N-3} and zeros elsewhere, and R_2 is zero everywhere except for the transition at one of the states s_k in $\{s_0, \ldots, s_{N-4}\}$ giving

$$-2BH < R_2(s_k, s_{k+1}, 1) < -BH.$$

The function R_4 satisfies

$$R_4(s_{N-3}, s_{N-2}, 0) \geq 1, \quad R_4(s_{N-3}, s_{N-1}, 1) \geq cR_4(s_{N-3}, s_{N-2}, 0),$$

for some $c > 2$, meaning, the final transitions are similarly rewarding depending on c.

The illustrations of MDP topologies and schematic diagrams of reward shape are given in Figure 7.3.

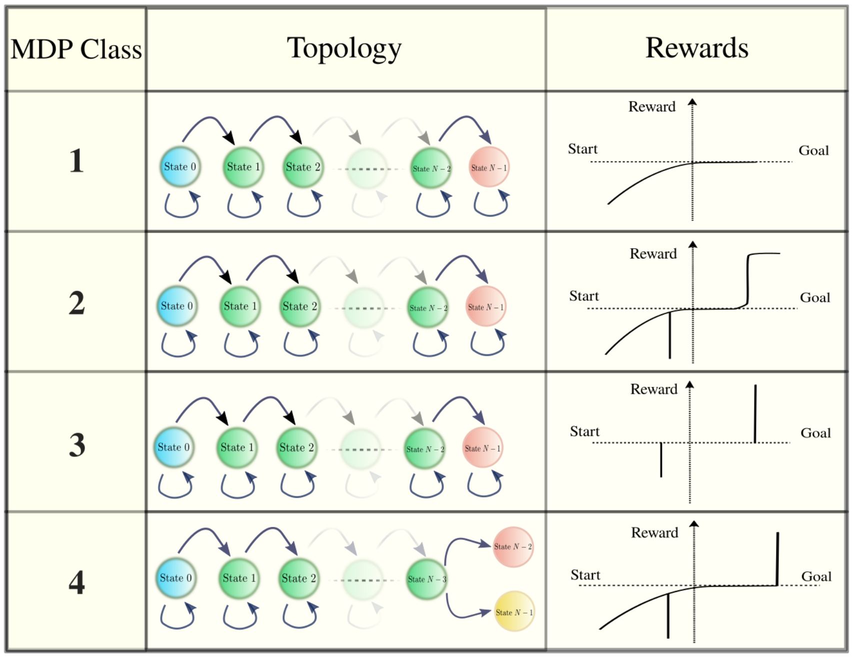


Figure 7.3: Illustrations of abstracted MDPs and their schematic diagrams of reward shape.

7.4.3 Theoretical result

Now, we present the performance separations of ℓ_1 loss algorithm and ℓ_2 loss algorithm, where $\ell_1(\cdot) = |\cdot|$ and $\ell_2(\cdot) = |\cdot|^2/2$, respectively.

Proposition 7.4.2 (Informal; see Appendix B.7 for a formal argument.)**.** *For each class of MDPs, ℓ_1 loss algorithm and ℓ_2 loss algorithm satisfy the followings:*

MDP Class 1: *Both algorithms will meet the success conditions equally faster when $\epsilon = 0$, which is of Class 1 behavior.*

MDP Class 2: *ℓ_1 loss algorithm meets the success conditions faster when $\epsilon = 0$, which is of Class 2 behavior.*

MDP Class 3: *Due to lack of guiding rewards, assume positive ϵ. ℓ_2 loss algorithm meets the success conditions faster more probably than ℓ_1 loss algorithm, which is of Class 3 behavior.*

MDP Class 4: *Suppose $\epsilon = 0$ except at the state s_{N-3} where $\epsilon = 0.5$ (i.e., completely random) until the agent explores that final transitions K times, where $H \geq K \geq 2$. Then, ℓ_1 loss algorithm meets the success conditions faster, but ℓ_2 loss algorithm becomes better at later stage of learning, which is of Class 4 behavior.*

Remark 7.4.3 (On ϵ in the algorithm)**.** In Proposition 7.4.2 above, we assume $\epsilon = 0$ for MDP Class 1 and MDP Class 2, which means there will be no exploration to simplify the arguments; which would be validated by the fact that they have guiding rewards which make it meet the success condition if it experiences sufficiently many episodes.

7.4.4 Other important cases

Here, we present some important cases that do not necessarily fall into the behavior classes that we defined. These behavioral observations may happen together with certain class of behavior.

Case study 1: *Separation between QRSAC and SAC-Huber ver. 1.* We only present a basic mechanism here. Consider the MDP illustrated in Figure 7.2 (right), where the probability of reward for the transitions to the final states is portrayed. In this case, the following holds:

$$10 = Q(s_{N-3}, 0) < 40 = Q(s_{N-3}, 1),$$

and therefore, the optimal action w.r.t. the Q function is 1. However, we have

$$10 = \text{Median}[R(s_{N-3}, s_{N-2}, 0)] > \text{Median}[R(s_{N-3}, s_{N-1}, 1)] = 5,$$

indicating that the median estimate will lead to a suboptimal action selection. As such, ℓ_1 loss may potentially lead to a wrong action selection in terms of the mean cumulative rewards.

Case study 2: *Separation between QRSAC and SAC-Huber ver. 2.* When the magnitudes of reward are very large across states, we see empirically that SAC-Huber outperforms QR-SAC in certain environments. We did not explore this phenomenon in detail; however, there is an evidence that SAC-Huber also degrades when the Q networks have multiple heads that are intentionally learned to have very distinct values but to be averaged to the true estimate. We explain this experimental results in Appendix A.5.4. This evidence may imply that the shared network having multiple heads that have too distinct values may cause troubles in learning.

7.5 Experimental evaluations

In this section, we present experimental results on (1) simple tabular environments with Q-learning, and (2) 1D environments, where the state and action spaces are both of one dimension, which enables us to fully visualize the growth of critic and actor, and on (3) some selected Gym environments [46].

7.5.1 Metrics

We define several metrics that are used to *hint at* the connection of algorithmic behaviors in the aforementioned environments to our proposed behavior classes.

Stability metrics

The oscillation of a function $f : \mathbb{N} \to \mathbb{R}$, generating a time sequence, over an interval $I \subset \mathbb{N}$ is defined by

$$\omega_f(I) = \sup_{\tau \in I} f(\tau) - \inf_{\tau \in I} f(\tau).$$

Inspired by this, we measure the oscillation of the Q value ω_Q over the interval $I_t := \{\max\{0, t - T_s + 1\}, \ldots, t\}$ at time step t by

$$\omega_Q^t := \mathbb{E}\left[\sum_{(s,a)\in\mathcal{D}_t}\left(\max_{\tau\in I_t}\{\hat{Q}_\tau(s,a)\} - \min_{\tau\in I_t}\{\hat{Q}_\tau(s,a)\}\right)/|\mathcal{D}_t|\right],$$

where $\mathcal{D}_t$ is the sampled batch at time t, which is a random variable.

Smoothness metrics

We also measure smoothness of the critic surface as well. To this end, we propose two smoothness metrics: expected determinants of Hessian over batch data, and variance of policy gradients.

Hessian: We compute the following:

$$\mathbb{E}\left[\sum_{(s,a)\in\mathcal{D}_t}\left|\det \mathbf{H}_a\left(\hat{Q}_t(s,a)\right)\right|/|\mathcal{D}_t|\right],$$

where $\mathbf{H}_a\left(\hat{Q}_t(s,a)\right) \in \mathbb{R}^{d_a\times d_a}$ is the Hessian matrix of $\hat{Q}_t$ at s and a with respect to a, and d_a is the dimension of action space. Conflict of policy update direction happens when the surface is *nonsmooth*; in particular, we would like to compute $|\det C|$, which indicates the variance of individually updated action points around a point (a^*, s^*), where

$$C := \mathbb{E}_{a\sim\mathcal{N}(a^\star,\sigma^2 I)}\left[z(a,s^\star,a^\star)z(a,s^\star,a^\star)^\top\right], \quad \sigma > 0,$$

and

$$z(a,s^\star,a^\star) = (a - a^\star) + \alpha\nabla_a Q(s^\star,a), \quad \alpha\in(0,1).$$

We can then show that the following approximation holds:

$$\det C \approx \det\{\sigma^2 I + 2\alpha\sigma^2\mathbf{H} + \alpha^2(\mathbf{g}\mathbf{g}^\top + \sigma^2\mathbf{H}^2)\}, \quad \mathbf{g} := \nabla_a Q(s^\star,a^\star), \quad \mathbf{H} := \mathbf{H}_a\left(Q(s^\star,a^\star)\right). \tag{7.5.1}$$

To see this, define $a' = a - a^\star$, and we write

$$C = \underbrace{\mathbb{E}[a'a'^\top]}_{=:\mathbf{A}} + \alpha\mathbb{E}[a'\nabla_a Q(s^\star, a)^\top + \nabla_a Q(s^\star, a)a'^\top] + \alpha^2\mathbb{E}[\nabla_a Q(s^\star, a)\nabla_a Q(s^\star, a)^\top].$$

Because

$$\nabla_a Q(s^\star, a) \approx \underbrace{\nabla_a Q(s^\star, a^\star)}_{=:\mathbf{g}} + \underbrace{\mathbf{H}\left(Q(s^\star, a^\star)\right)}_{=:\mathbf{H}} a',$$

we obtain

$$\begin{aligned} C &\approx \mathbf{A} + \alpha\left\{\mathbb{E}[a']\mathbf{g}^\top + \mathbf{AH} + \mathbf{g}\mathbb{E}[a'^\top] + \mathbf{HA}\right\} \\ &\quad + \alpha^2\left\{\mathbf{gg}^\top + \mathbf{g}\mathbb{E}[a'^\top]\mathbf{H} + \mathbf{H}\mathbb{E}[a']\mathbf{g} + \mathbf{HAH}\right\} \\ &= \mathbf{A} + \alpha(\mathbf{AH} + \mathbf{HA}) + \alpha^2(\mathbf{gg}^\top + \mathbf{HAH}), \end{aligned}$$

where we used $\mathbf{H} = \mathbf{H}^\top$ and the fact that $\mathbf{g}$ and $\mathbf{H}$ do not depend on a, and that $\mathbb{E}[a'] = 0$. Since $\mathbf{A} = \sigma^2 I$, we reach the approximation. As such, we use $|\det \mathbf{H}_a|$ as a simplified measure for the conflict of policy gradients.

Variance of gradients Typically, gradients of policy parameters, denoted by $\mathbf{g}_\pi \in \mathbb{R}^{d_\pi}$, are of very high dimension, which hinders direct computations of determinant of

$$\mathbb{E}\left[(\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi])(\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi])^\top\right].$$

We instead use the following trace computation:

$$\mathrm{tr}\mathbb{E}\left[(\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi])(\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi])^\top\right] = \mathbb{E}\left[\mathrm{tr}(\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi])(\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi])^\top\right] = \mathbb{E}\|\mathbf{g}_\pi - \mathbb{E}[\mathbf{g}_\pi]\|^2_{\mathbb{R}^{d_\pi}}.$$

Accuracy metrics

We test if the supervised learning and bootstrap learning under a fixed policy scenario with each critic loss function give accurate target value function estimate; to this end, if we know the target Q function Q^*, we use it and compute the leaning curve of the MSEs:

$$\mathbb{E}\left[\sum_{(s,a)\in\mathcal{D}_t}\left(Q^*(s,a) - \hat{Q}_t(s,a)\right)^2/|\mathcal{D}_t|\right].$$

For complex environments where we do not have the ground-truth Q function, we use the return (distribution) of QRSAC at some later learning stage; for such case of bootstrap learning scenario, we randomly draw *reward* from the distribution computed by

$$\hat{Z}^{\pi}(s,a) - \gamma \hat{Z}^{\pi}(s',a'),$$

where the policy π is the policy of QRSAC at the selected stage of learning which is used for bootstrap learning test associated with the return estimate $\hat{Z}^{\pi}$, s' is the observed next state and a' is sampled from $\pi(s')$.

7.5.2 Experimental setups

Common setups: Throughout, all RL runs are trained in a distributed manner with separated processes of a rollout worker that collects experiences and a trainer that follows the algorithm dynamics, asynchronously. For data storage and experience replay, we use Reverb [54]. Details of the neural networks and other hyperparameters are described in Appendix A.5.3. We also conduct bootstrap and supervised learning experiments under a fixed policy scenario to eliminate the effect of critic and policy interactions, and other minor experiments for testing the influence of multi-headed network.

Tabular MDPs: We tested on five different tabular MDPs; one of them, which we refer to as MDP 1, is illustrated in Figure 7.4 (left top), and other four are shown in Appendix A.5.1. All of our toy MDPs are intentionally created to possess similar properties to those of the representative MDP classes, except for the MDP 5 which is made to show the behavior introduced in Case study 1.

1D continuous environments: We tested 13 variants, which are summarized in Appendix A.5.2. For all of them, the state and action spaces are defined by $\mathcal{S} = [-5, 5]$ and $\mathcal{A} = [-1, 1]$ (see Figure 7.4 (left down)). The initial state is -4.5 or 4.5 with probability 0.5 each, and at every step, the state s is updated by action a. Those systems are created

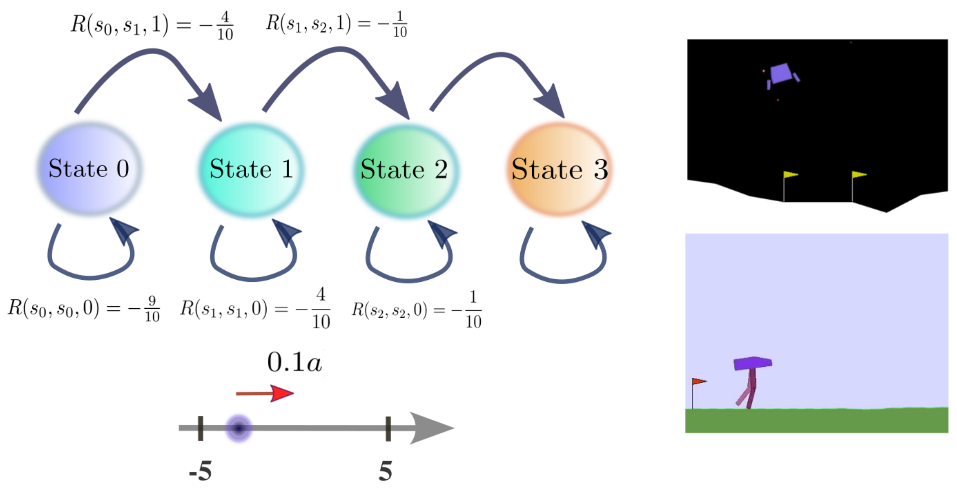

$R(s_0, s_1, 1) = -\frac{4}{10}$
$R(s_1, s_2, 1) = -\frac{1}{10}$
State 0
State 1
State 2
State 3
$R(s_0, s_0, 0) = -\frac{9}{10}$
$R(s_1, s_1, 0) = -\frac{4}{10}$
$R(s_2, s_2, 0) = -\frac{1}{10}$
$0.1a$
-5
5

Figure 7.4: Illustrations of selected environments for our simulation experiments. (Left top) Tabular MDP 1. (Left down) Simple 1D (continuous) environment. (Right) OpenAI Gym environments: Lunar Lander and Bipedal Walker are shown.

intentionally to possess some of the *conceptually* similar properties to the abstracted tabular MDP systems. We then observe evolution of our proposed metrics and full visualizations of critic/policy growth, from which we discuss if the cause of performance separations has some similarity to that of the abstracted systems (i.e., MDP Class 1 to MDP Class 4 etc.).

More complex environments We tested Lunar Lander, Bipedal Walker, and Hopper environments (see 7.4 (right)) for RL algorithms with different critic losses and settings. Although the visualizations of critic/policy growth are only partially possible, we conduct similar analyses to the case of 1D environments.

7.5.3 Results

We present only a set of selected results (see Appendix A.5.4 for the rest of the results).

7.5.4 Toy MDP

The learning curves for the toy MDPs are given in Figure 7.5. The curves of MDPs 1 to 4 are consistent to the behavior class definitions. For MDP 5, it shows noisy behavior due to the probabilistic nature of the system and shows the clear separation between Huber and quantile-Huber loss.

Figure 7.6 shows the evolution of Q value estimates of different loss algorithms for MDP 1, 2 and 4. MDP 1 shows similarly stable evolution of Q values for MSE and quantile-Huber cases; while in MDP 2 and 4, MSE loss algorithm shows radical changes of the Q estimate which leads to worse or better performance. Especially for MDP 4, MSE loss algorithm correctly estimates the optimal action at State 3. Figure 7.7 shows the same evolution for MDP 5. We observe that the values at State 2 differ between MSE/quantile-Huber and Huber loss, indicating that the Huber loss algorithm struggles to estimate the true order of values. This toy MDP is created intentionally so that the mean and median estimate of value differ, and the Huber loss algorithm shows inferior performance as expected (refer to Case study 1). Interestingly, quantile-Huber loss algorithm spreads the quantile buckets so that it can estimate the mean value better than the Huber loss counterpart.

All of these toy MDPs are created intentionally to follow similar properties to our theoretical MDP classes. Although they are not necessarily covered by our theoretical MDP classes, it is at least observed that creating such simple systems that show desired behaviors of different critic loss algorithms is no difficult task.

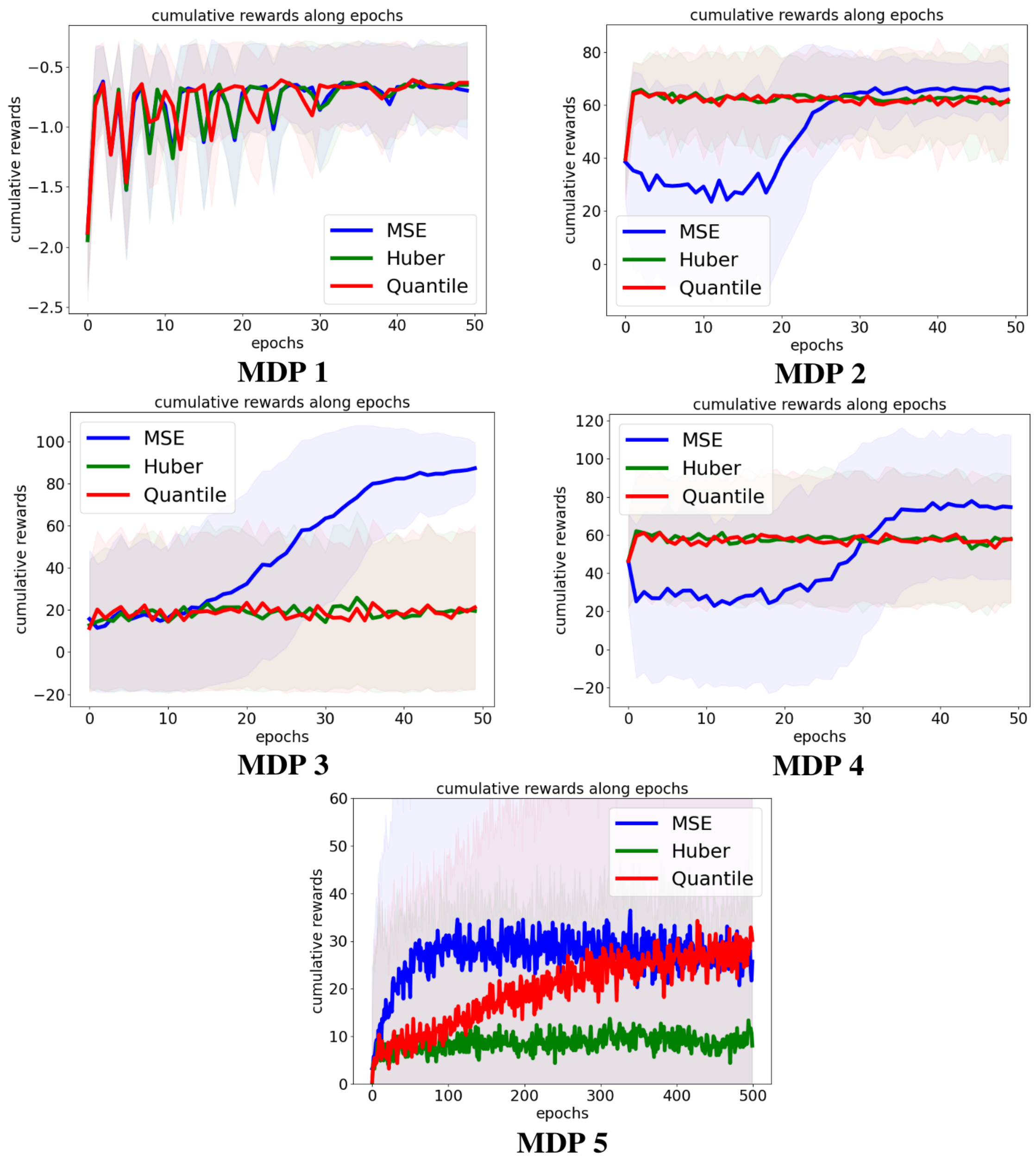


Figure 7.5: Reward curves averaged over 300 runs for MDPs 1 to 5 with standard deviation shading. All of them are consistent to the behavior class definitions; MDP 5 is noisy because of the probabilistic nature of the system but shows the clear separation between Huber and quantile-Huber loss.

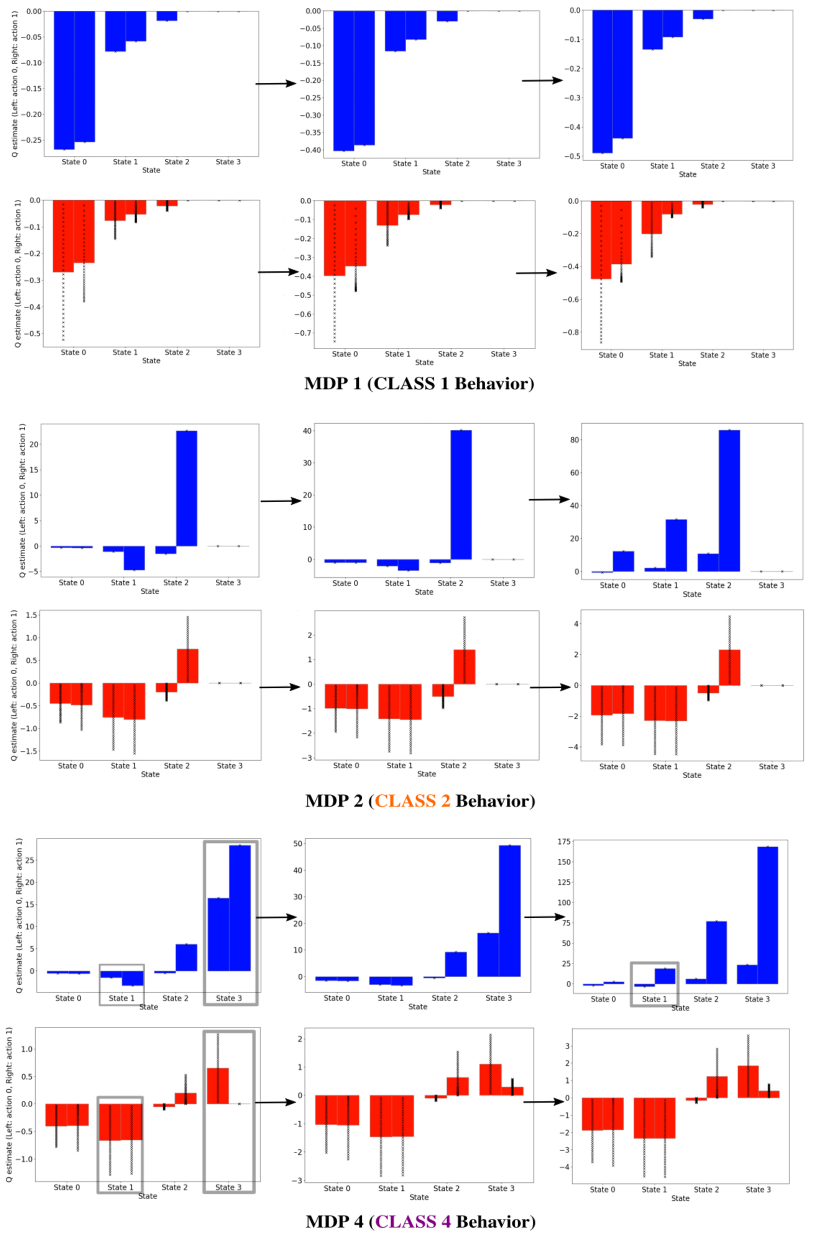


Figure 7.6: Evolution of Q value estimates for MSE (blue) and quantile-Huber (red) loss algorithms for MDP 1, 2 and 4; for each state (0 to 3), the left bin indicates the value for action 0 and the right is for action 1. Note the "×" marks in quantile-Huber cases correspond to the quantile estimates. MDP 1 shows similarly stable evolution of Q values for the two while MSE loss algorithm shows radical change for MDP 2. On the other hand, it is observed for MDP 4 that while MSE loss algorithm shows some radical change of the estimates, it finally reaches to the better state quicker.

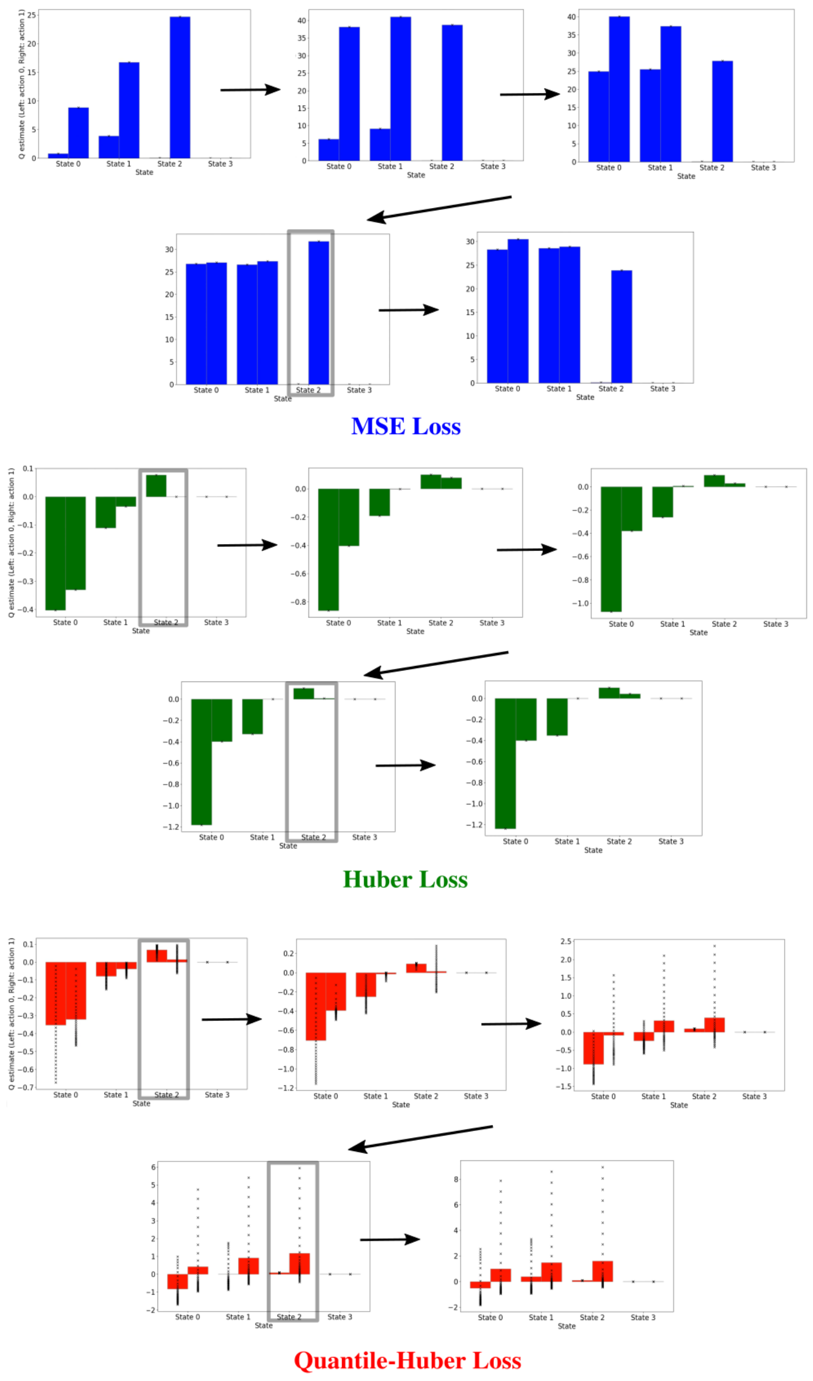


Figure 7.7: Evolution of Q value estimates for different loss algorithms for MDP 5; for each state (0 to 3), the left bin indicates the value for action 0 and the right is for action 1. Note the “×” marks in quantile-Huber cases correspond to the quantile estimates. It is observed that the values at State 1 differ between MSE/quantile-Huber and Huber loss, indicating that the Huber loss algorithm struggles to estimate the true order of values.

7.5.5 1D continuous environments

RL results

The learning curves, including reward, stability and smoothness curves, for the representative environments are shown in Figure 7.8, 7.9, 7.10 and 7.11. Note, in the plots, `diff_q_avg`, `det_Hessian_avg` and `policy_var_avg` are the stability, smoothness metric based on Hessian, and smoothness metric based on the variance of policy gradient directions.

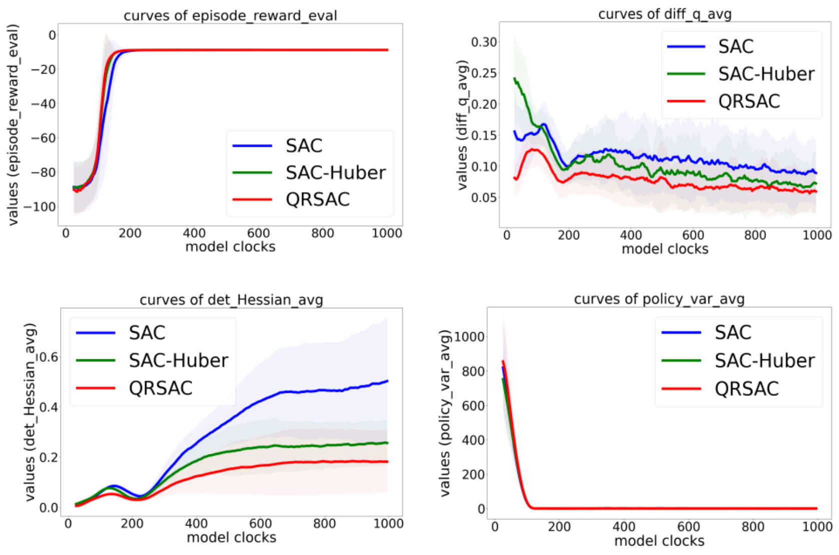


Figure 7.8: Reward curve, stability, and smoothness metrics (Hessian and policy gradient) curves for the representative 1D environment falling into Class 1.

There are a few observations.

1. SAC shows instability especially at the earlier stage of learning except for the Class 1 representative where all algorithms show similar stability

2. Smoothness seems to become worse following unstable critic evolution

3. At least, smoothness of critic and performance do not correlate exactly (Class 3 representative shows better performance for SAC but nonsmooth surface)

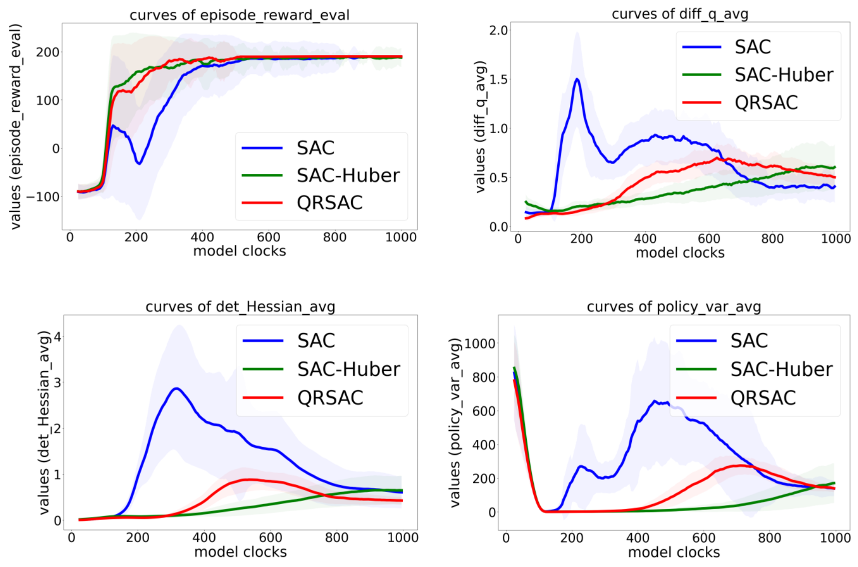


Figure 7.9: Reward curve, stability, and smoothness metrics (Hessian and policy gradient) curves averaged over 30 runs for the representative 1D environment falling into Class 2.

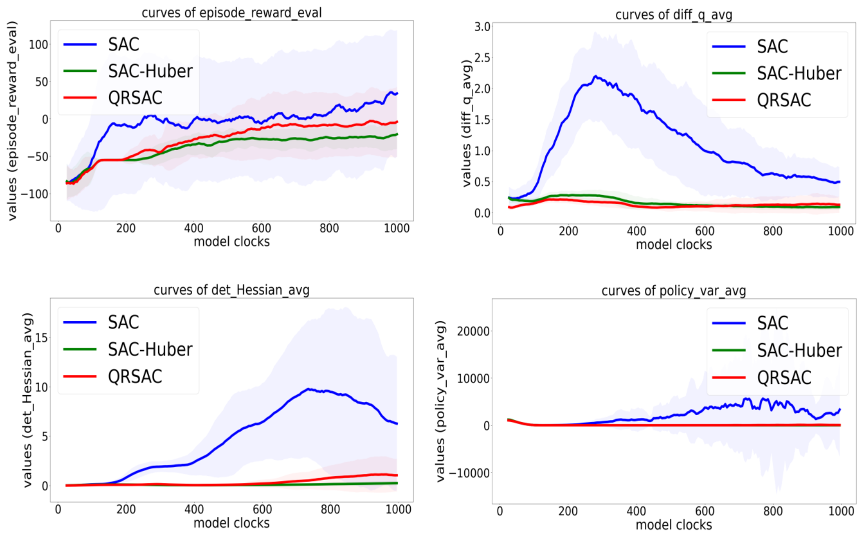


Figure 7.10: Reward curve, stability, and smoothness metrics (Hessian and policy gradient) curves averaged over 30 runs for the representative 1D environment falling into Class 3.

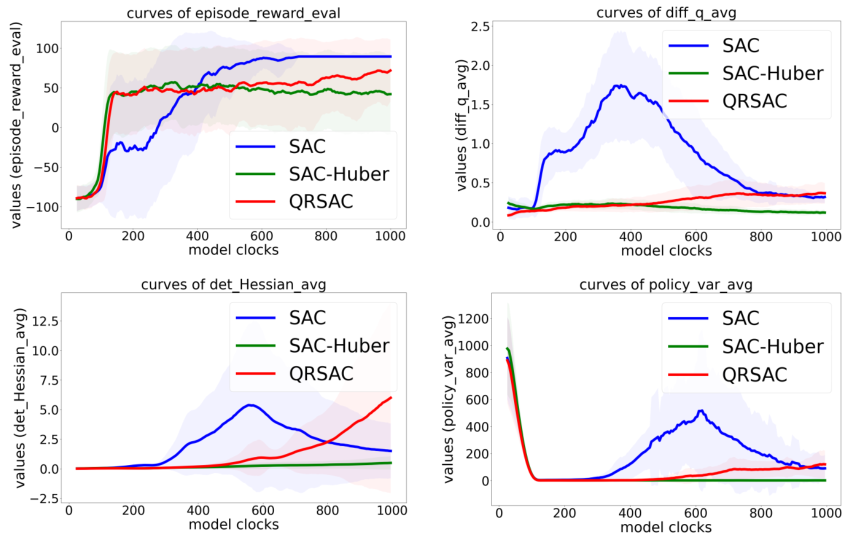


Figure 7.11: Reward curve, stability, and smoothness metrics (Hessian and policy gradient) curves averaged over 30 runs for the representative 1D environment falling into Class 4.

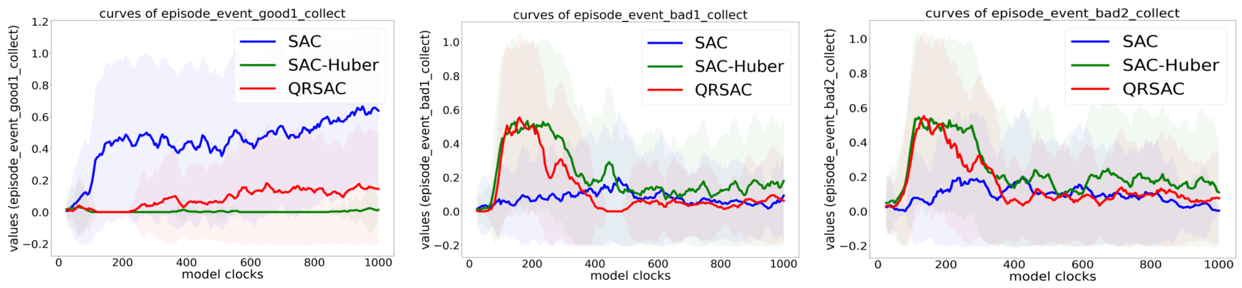


Figure 7.12: Event probabilities for *good1*, *bad1*, and *bad2* for the representative Class 3 1D example. Good event happens for SAC more often while SAC-Huber and QRSAC can decrease bad events.

For the Class 3 representative, we define *good1*, *bad1*, and *bad2* events as the transitions to exit states giving high reward and high costs respectively, and we plot the event occurrence probability in Figure 7.12. It is interesting to observe that SAC consistently shows higher chance of *good1* event while others do not but they can eventually decrease the chance of *bad* events. Stable growth of critic may help decrease the bad events here.

Figure 7.13 (top) shows the representative of Case study 2; with large magnitude of reward, QRSAC shows inferior performance. Also, for this environment, as mentioned in Appendix A.5.2, it has a flavor of Class 2 as well, and we can indeed observe that SAC-Huber slightly outperforms SAC.

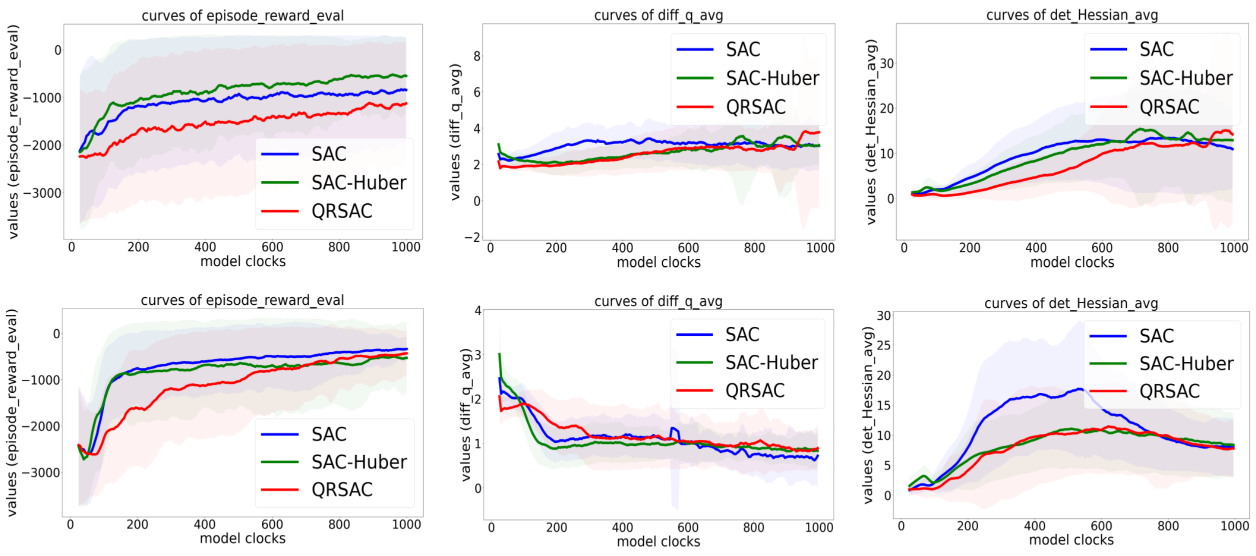


Figure 7.13: Reward curve, stability, and smoothness metric (Hessian) curves for the 1D environment showing what is discussed in Case study 2. Top ones are of the representative environment and the down ones are of env. #13.

We also visualize the evolution of critic surface and policy of the Class 1 to Class 4 representatives in Figure 7.14, 7.15, 7.16 and 7.17. We see that they match the intuition given by the stability curves. Class 1 representative shows similarly stable and smooth growth while SAC has some instability for Class 2 representative. For Class 3 one, it seems that SAC-Huber and QRSAC show relatively stable growth while SAC shows instability but faster learning of better critic surface. In the Class 4 case, SAC seems to reach relatively complicated critic surface faster while SAC-Huber and QRSAC show smoother critic surface which may not be the optimal surface in this environment.

We emphasize that these results are not comprehensive in the sense that one could potentially create other sets of environments which show different learning performance; however, this style of analysis is shown to be useful for identifying pathological and theoretically important environments and behaviors for further study.

Supervised learning and bootstrap learning

The results of bootstrap and supervised learning for the Class 2 and Class 4 representatives are shown in Figure 7.18. Most notably, the results of bootstrap learning *do not* correlate to the RL runs at all, and the Huber loss learning shows tremendously low performance. Therefore, we at least claim that the interaction between critic and policy plays critical role in the performance in some environments. For the supervised learning, the target critic generated by SAC-Huber seems to be learned well by the Huber loss algorithm while that generated by SAC is learned well by both QRSAC and SAC. While we do not delve into this, different loss algorithms for the regression could show distinct performance depending on the target function surface and data distribution. We mention that the data are generated by the policy learned by SAC or SAC-Huber and they may only cover a fraction of the state space.

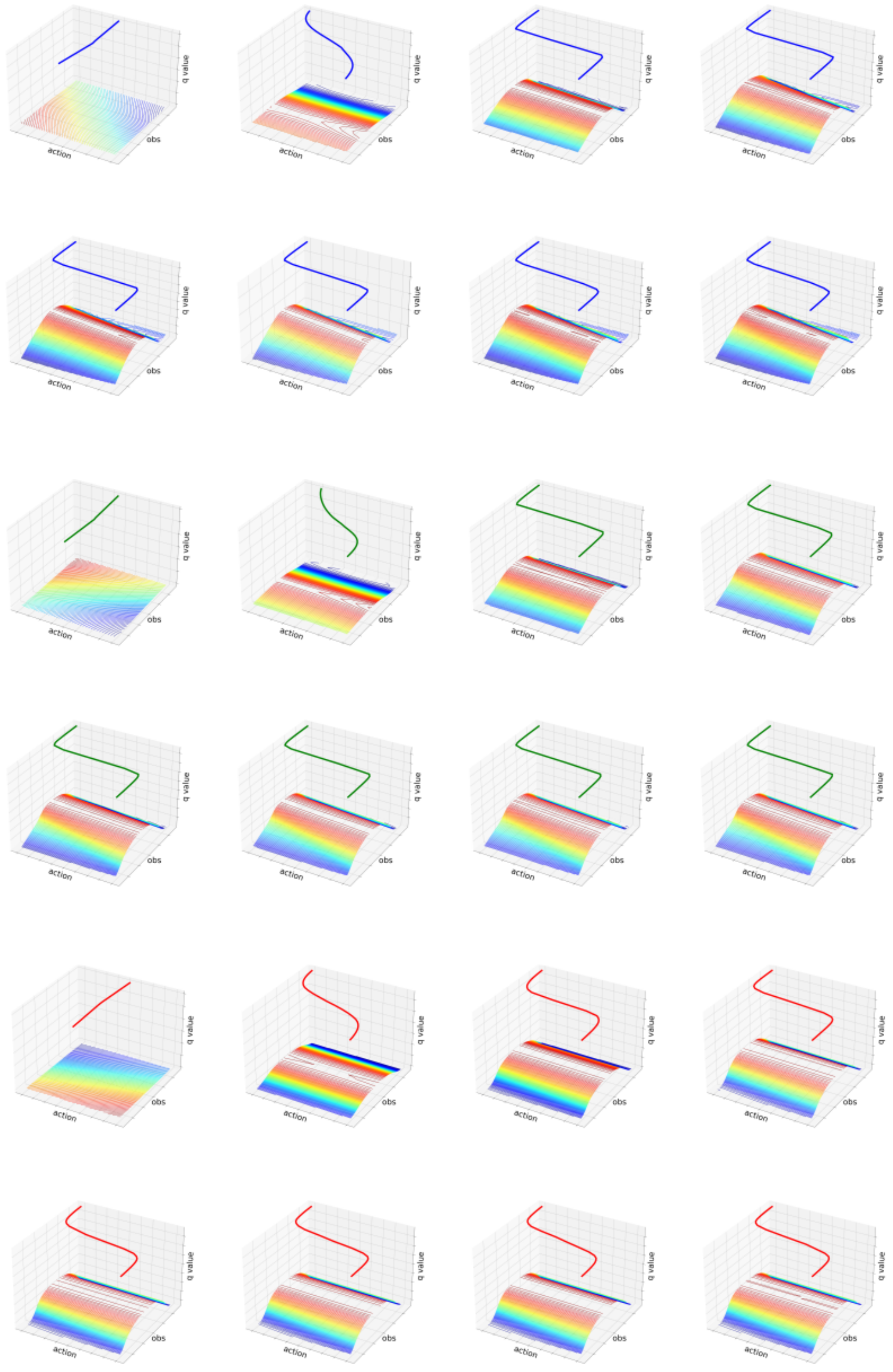


Figure 7.14: Evolution of critic surface and policy for SAC (blue), SAC-Huber (green) and QRSAC (red) for the representative 1D environment of Class 1. It is observed that all of them show stable evolution of policy and critic.

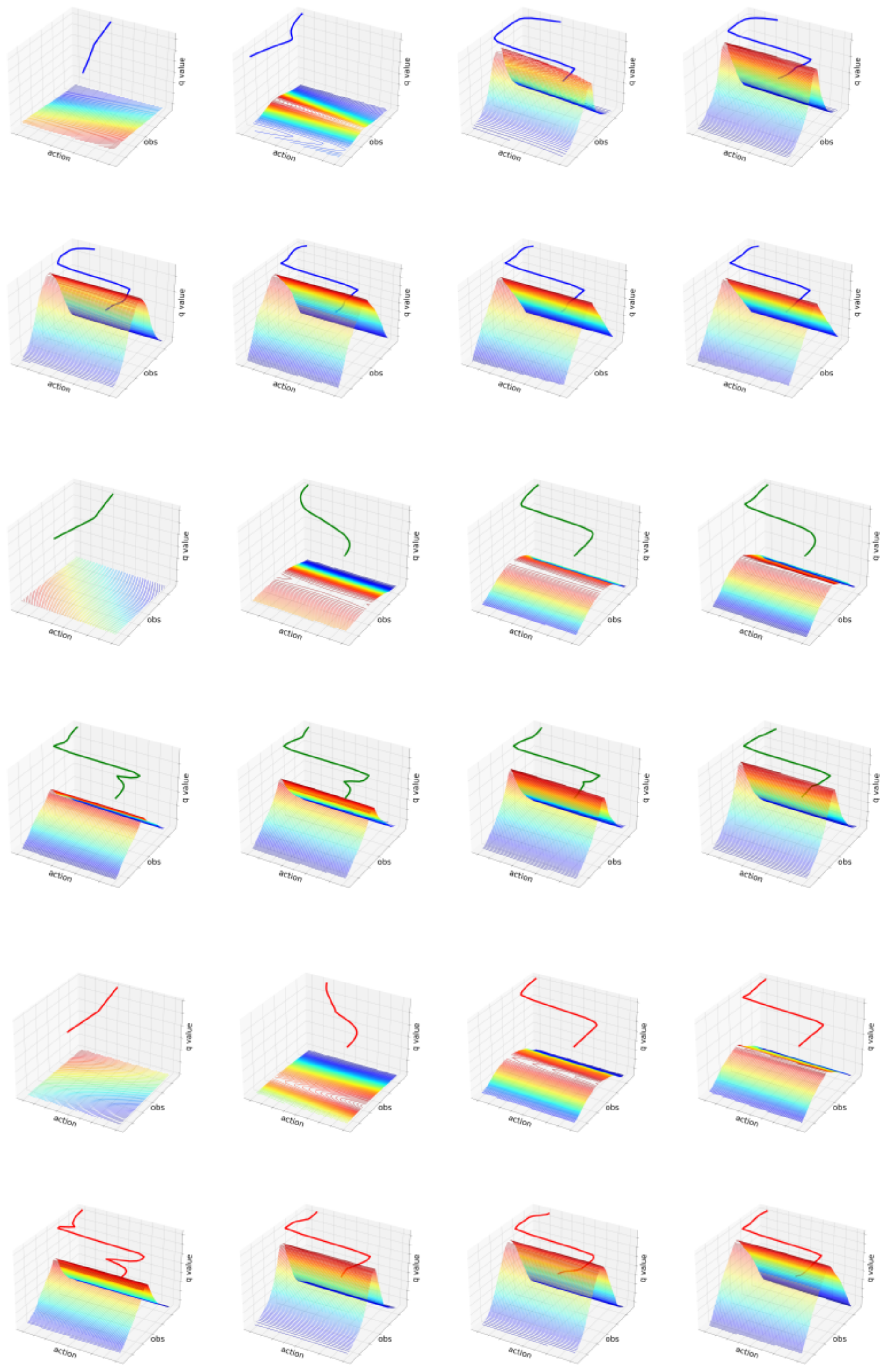


Figure 7.15: Evolution of critic surface and policy for SAC (blue), SAC-Huber (green) and QRSAC (red) for the representative 1D environment of Class 2. It is observed that SAC shows instability of policy growth.

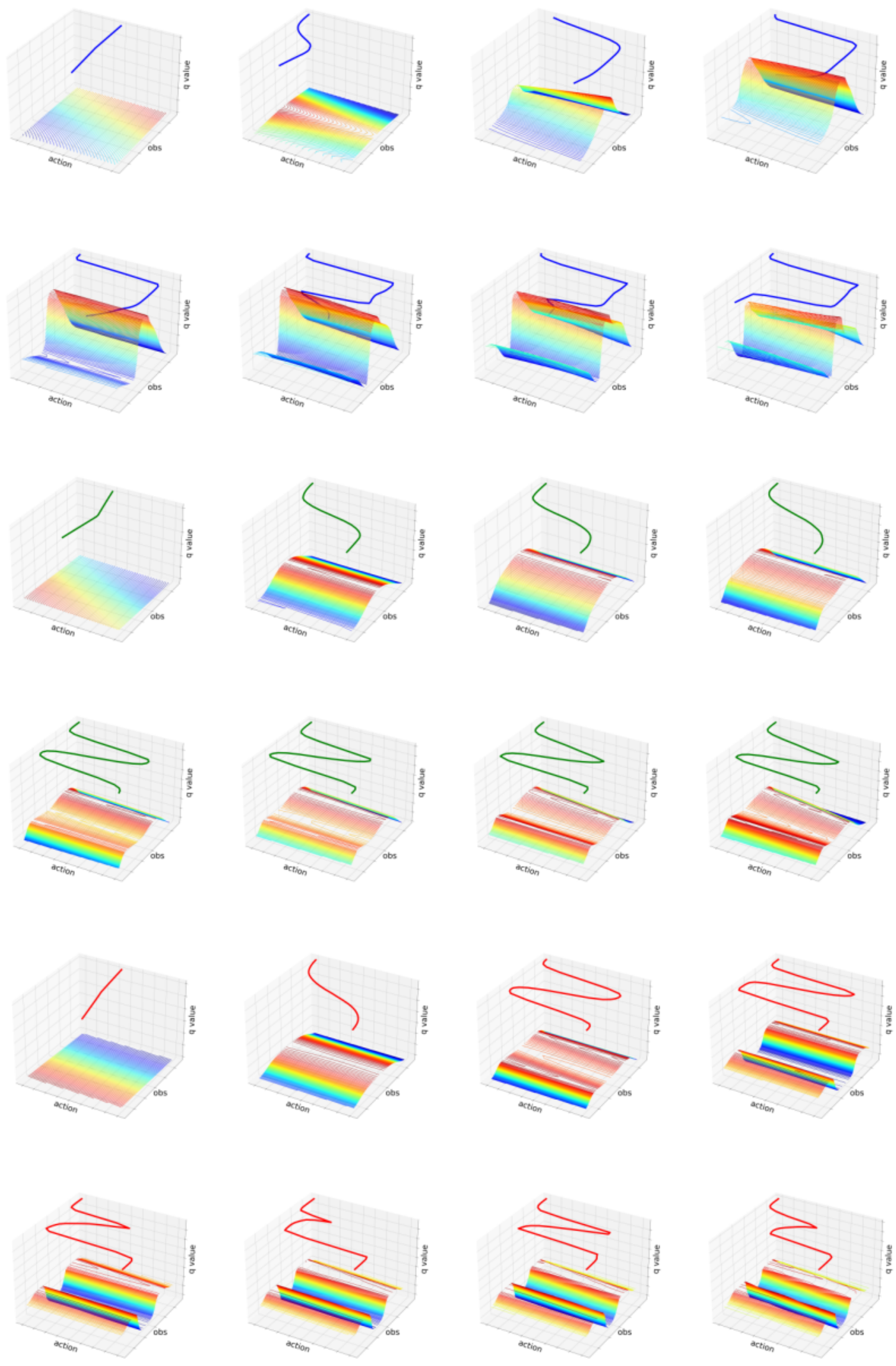


Figure 7.16: Evolution of critic surface and policy for SAC (blue), SAC-Huber (green) and QRSAC (red) for the representative 1D environment of Class 3. While SAC shows some instability, it helps learn the correct surface quicker.

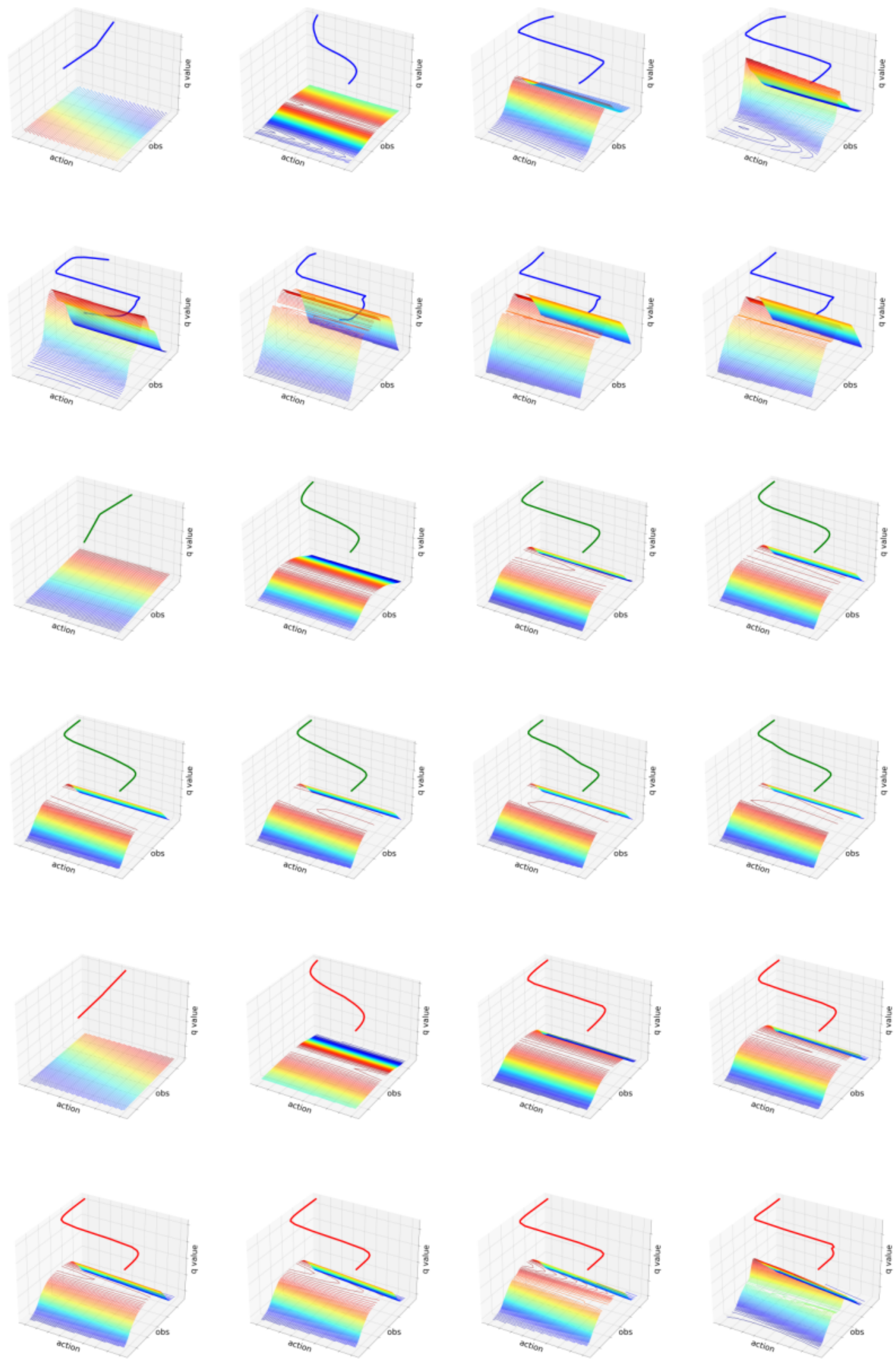
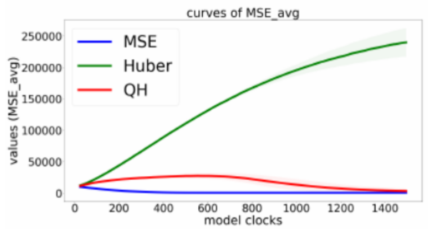
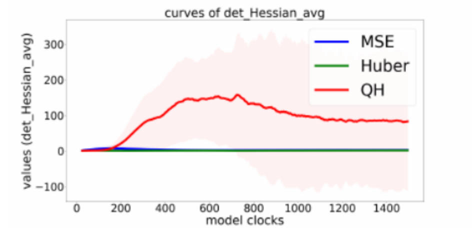


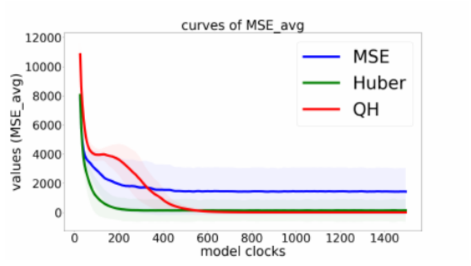
Figure 7.17: Evolution of critic surface and policy for SAC (blue), SAC-Huber (green) and QRSAC (red) for the representative 1D environment of Class 4. While SAC shows *well-tuned* critic surface at the end, others are not sufficiently capturing the correct surface.



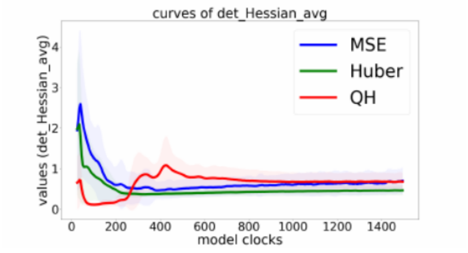
(a) MSE curves (bootstrap).



(b) Absolute value determinant of Hessian (bootstrap).

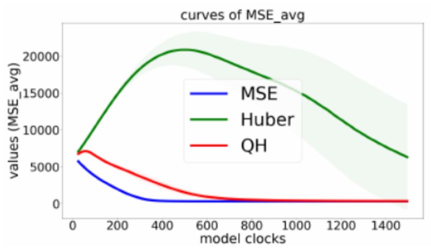


(c) MSE curves (supervised).

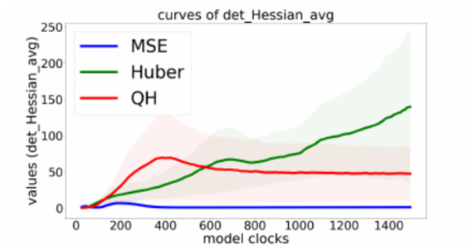


(d) Absolute value determinant of Hessian (supervised).

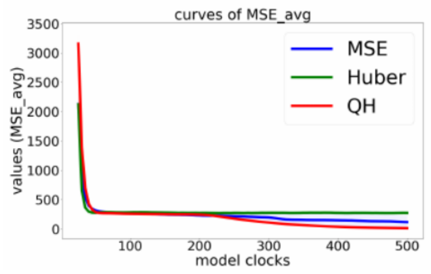
Representative env. for Class 2



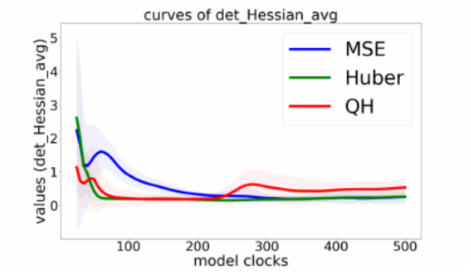
(a) MSE curves (bootstrap).



(b) Absolute value determinant of Hessian (bootstrap).



(c) MSE curves (supervised).



(d) Absolute value determinant of Hessian (supervised).

Representative env. for Class 4

Figure 7.18: Bootstrap/Supervised learning performance curves of the representative 1D examples for Class 2 and Class 4. We see that the curves do not appear to be consistent to those of the RL results.

7.5.6 Complex RL environments

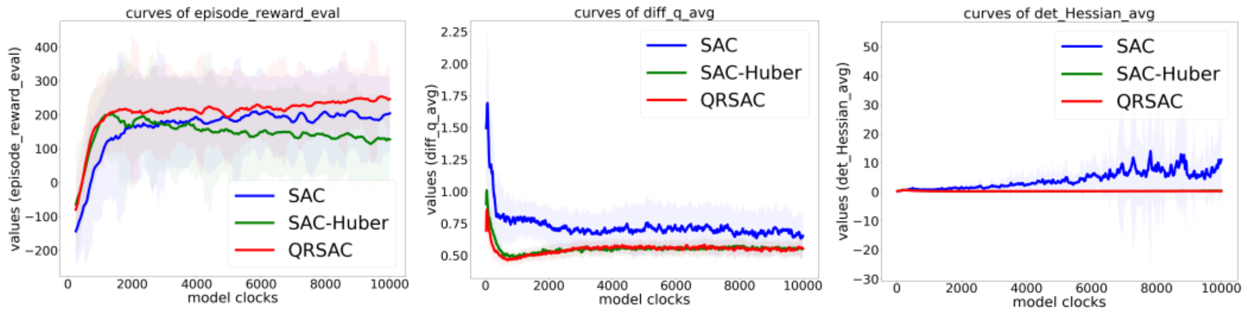
RL results

The learning curves, including reward, stability and smoothness curves, for Lunar Lander, Bipedal Walker and Hopper are shown in Figure 7.19. For all of the three, SAC shows instability in critic growth. We also visualize the evolution of critic surface and policy for SAC and QRSAC at the fixed state for Lunar Lander in Figure 7.20 to verify this observation.

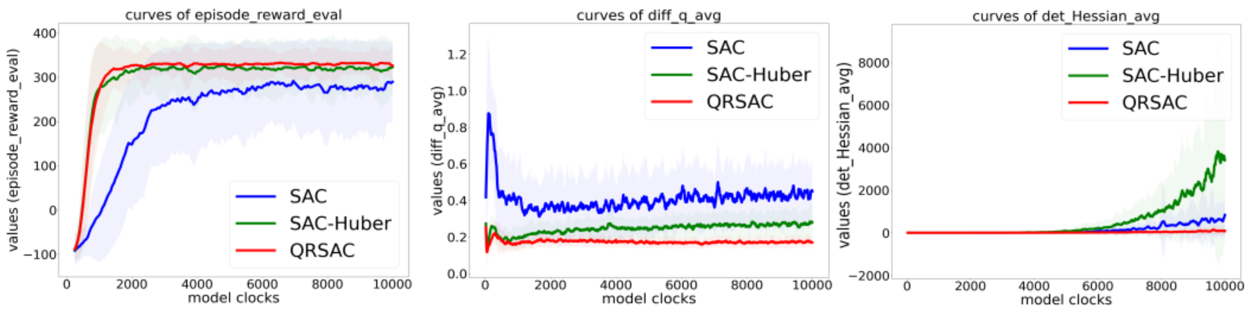
The rewards for all of them seem to *guide* the agent to achieve certain tasks; and for Bipedal Walker where there is large penalty when failing, which is similar to the property of MDP Class 2, it indeed shows the Class 2 behavior. For Lunar Lander, it also has the tendency of Class 2 behavior while the performance of SAC-Huber degrades at later stage. Although it stays largely hypothetical, the *crash* and *successful landing* can happen with similar action sequences compared to Bipedal Walker environment, and this probabilistic nature may cause this degradation with some flavor from Class 4 and Case study 1 representatives. The most interesting behavior emerges out of Hopper environment; it has some guiding reward, but the dynamics are very nonsmooth and discontinuous. As such, it may have Class 3 or Class 4 essence as well. However, we do not have clear reasoning about Hopper environment, and it is of future work for analyzing more behavior classes and the case where reward is smooth while dynamics is very nonsmooth.

Supervised learning and bootstrap learning

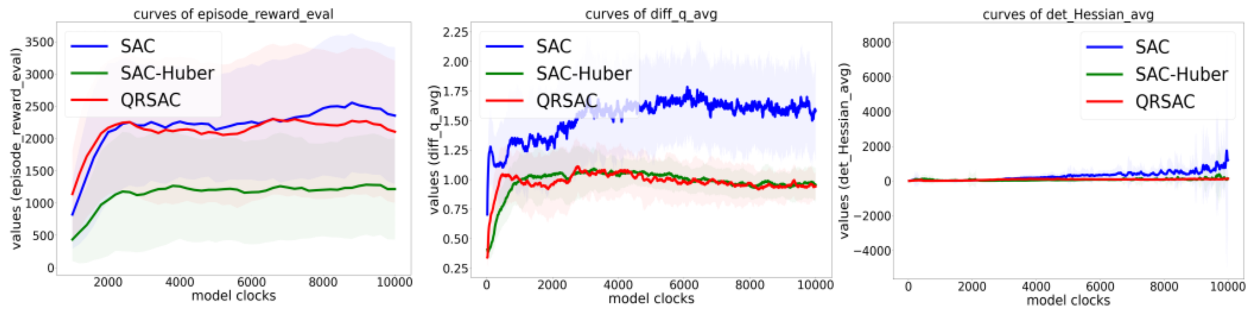
The results of bootstrap and supervised learning for Lunar Lander and Bipedal Walker environments are shown in Figure 7.21. Similar to the results of 1D environment cases, the performance of bootstrap learning *does not* correlate to the RL runs; however, the Huber loss algorithm is functioning well in these Gym environments. Although we used the target critic generated by QRSAC, the MSE loss algorithm shows better performance for both bootstrap and supervised learning in Bipedal Walker environment while quantile-Huber loss one shows better curve than the MSE loss counterpart in supervised learning for Lunar Lander.



Lunar Lander



Bipedal Walker



Hopper

Figure 7.19: Reward curve, stability, and smoothness metric (Hessian) curves for Lunar Lander, Bipedal Walker, and Hopper.

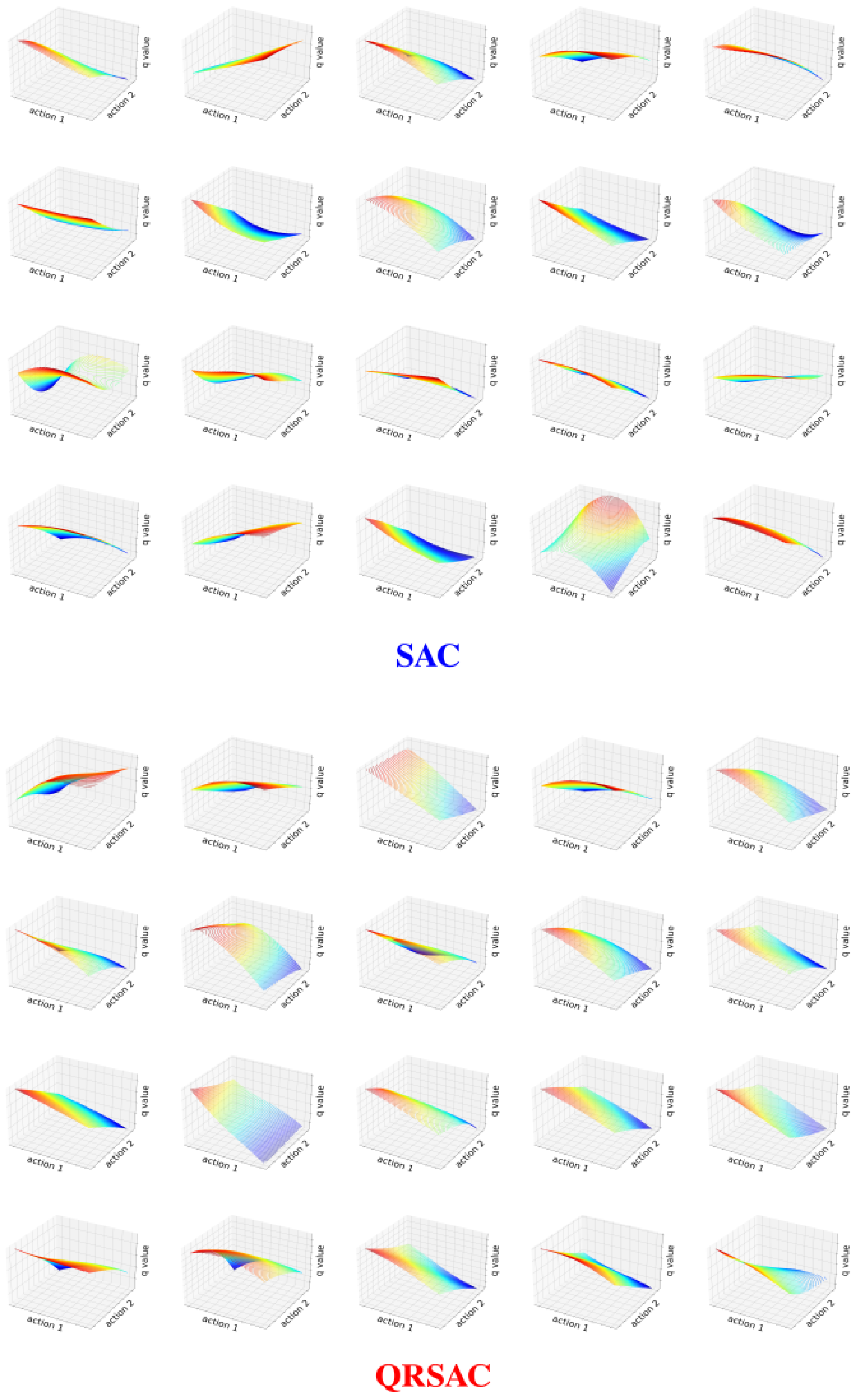
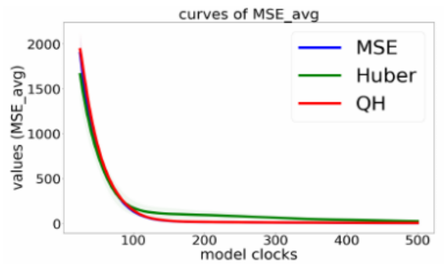
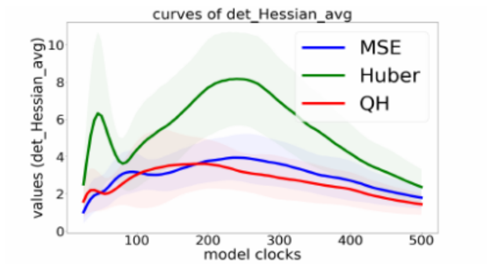


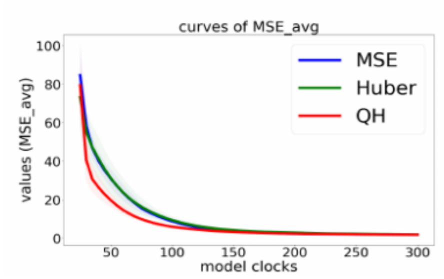
Figure 7.20: Evolution of critic surface for SAC and QRSAC at the fixed state for Lunar Lander; SAC shows more instability compared to QRSAC.



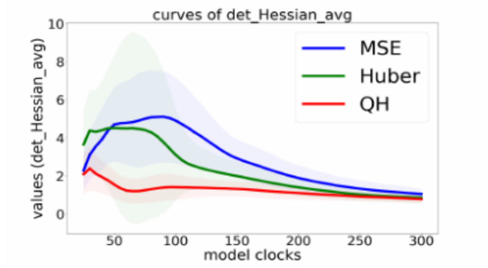
(a) MSE curves (bootstrap).



(b) Absolute value determinant of Hessian (bootstrap).

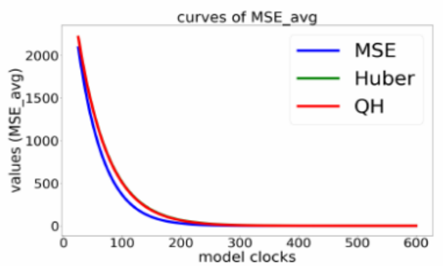


(c) MSE curves (supervised).

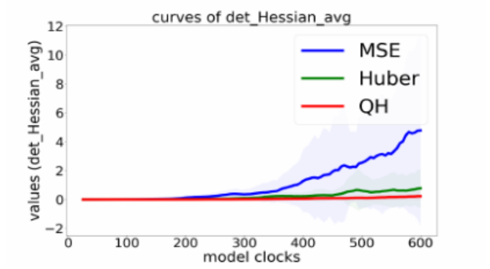


(d) Absolute value determinant of Hessian (supervised).

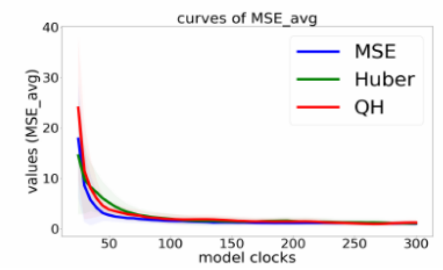
Lunar Lander



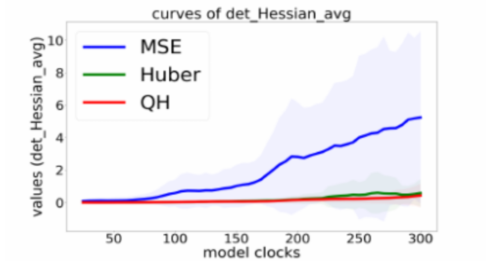
(a) MSE curves (bootstrap).



(b) Absolute value determinant of Hessian (bootstrap).



(c) MSE curves (supervised).



(d) Absolute value determinant of Hessian (supervised).

Bipedal Walker

Figure 7.21: Bootstrap/Supervised learning performance curves of Lunar Lander and Bipedal Walker. We see that the curves do not appear to be consistent to those of the RL results except that Huber loss one performs relatively poorly for bootstrap learning on Lunar Lander.

7.5.7 Findings

Below, we summarize our findings. We strongly emphasize that those results do not identify the clear mechanism of performance separation observed in complex environments but only help us understand some aspects of the behaviors.

1. For tabular MDPs and 1D environments, creating simple reward architectures that lead to the behaviors categorized in our proposed classes is a relatively straightforward task; which at least gives sufficient conditions for certain algorithmic behaviors to emerge

2. For all of the environments we tested except for the ones showing Class 1 behavior, SAC shows consistently unstable critic growth indicated by the visualization of critic/policy growth and by the stability measure

3. Smoothness of the critic surface does not necessarily correlate with better performance; and the critic seems to become nonsmooth after it shows unstable behaviors in most of our environments

4. At least for some environments, interaction of critic and policy produces remarkable behaviors that do not show up in the fixed policy bootstrap or supervised learning scenario; in particular, stability measures of critic in an RL setting show unique perspectives of critic and policy interaction

5. On the other hand, for some environments, the Huber loss algorithm shows significantly poor performance in the bootstrap learning

6. Although our behavior classes and (representative) abstracted models cannot explain every phenomenon observed in (complex) environments, it gives us some lessons about how the interactions among dynamics, reward structure, critic growth and policy growth affect the overall performance

7.6 Discussions on the relation to existing work

The work [158] proves that, with the squared Cramér distance loss, distributional RL behaves exactly the same as expected RL for the tabular and linear approximation settings, while the difference is pronounced in the nonlinear approximation settings. In this chapter, we provably showed the improvement with the use of quantile-Huber loss for a certain type of MDPs and retroductively argued similar phenomena may happen in more complex environments. In particular, *even the tabular case can show separation between distributional RL with quantile-Huber loss and expected RL*; this is no contradiction to [158] because the loss function for the distributional RL is different.

The authors of the work [197] claim that capacity constraints on approximators lead to distinct critic surfaces for the Huber and MSE loss cases. We showed, even for the simplified systems where no function approximator is used, the use of the Huber loss helps enhance performance under certain conditions. Moreover, in our toy abstracted environments, *the behavior difference emerges out of the interaction between critic and action selection and not necessarily of the learning accuracy of critic itself.*

Further, we showed that *smoothness of the surface does not necessarily correlate with an improvement of performance.* For some cases, instability of critic and policy evolution indeed causes inferior performance with nonsmooth critic surface; however, this instability (and consequent nonsmooth surface) could also correlate with better performance for certain environments.

Finally, although dynamical system perspective has been applied to RL in [160], we presented *patterns* or *classes* of algorithms' dynamic behaviors to take constructive approach; and *our approach does not separate critic and policy evolution unlike previous approach on learning dynamics.* In several environments having more complexity, we also saw that the interaction between critic and actor is playing a key role.

7.7 Chapter summary and discussion

In this chapter, we explored the performance difference of SAC algorithms with different critic loss functions including the quantile-Huber loss; in particular, we analyzed them by first categorizing the learning behaviors, and presenting MDP classes that are sufficiently simple but still show respective behaviors. In this way, we established some sufficient conditions that cause the performance separations. Then, we hint at the connection between this established logic and the mechanism of performance discrepancies observed in more complex RL environments through the lenses of some metrics and visualizations of critic/policy growth using a retroductive argument.

Most importantly, our approach does not *prove* the cause of behavior difference observed in complex RL environments as they have too many parameters to analyze or as there may not be a single common cause that is valid for different environments; therefore, only a massive amount of experiments conducted from a variety of angles can reinforce or fine-tune the arguments. In this regard, although we conducted some set of experiments, it can never be sufficient to explain phenomena in complex systems. We will need more detailed metrics, observations and respective experiments to strengthen the methodology.

Also, although tabular MDP can be a natural choice for the abstracted model, it is not yet clear if our constructive approach becomes a strong methodology for analyzing other domains of ML including large language models and generative models.

Chapter 8
CONCLUDING REMARKS

This thesis showcased multiple contributions of bridging the gap between dynamical systems (or control) and (statistical) ML from a variety of angles. Rather than patching them together, the beauty of this amalgamation is highlighted by the co-evolution of both fields through creations of novel and elegant concepts that fundamentally progress the understanding of each field, emerging out of necessity.

8.1 Summary

Chapter 3 bridged the control and ML by first coining a control theoretic term, namely, limited-duration safety, to utilize value function learned through RL with discount factor as a candidate for limited-duration control barrier function (LDCBF); then we exploited the guarantees given by LDCBFs in long-duration autonomy and in transfer learning tasks.

In Chapter 4, by exploiting the continuity property of continuous control problems and a state-of-the-art planning technique as a reasonable optimal control oracle embedded in the algorithm, we created a novel and sample efficient model-based RL algorithm, which is then shown to be effectively applied to practical complex control tasks. Further, the unique control perspectives, including pole assignment problems working in the spectral domain, are leveraged to devise a novel decision making framework which opens a new direction of research in RL.

Going further, Chapter 5 takes advantages of recent rough-path theoretical studies on path signatures, which have been developed to analyze controlled differential equations, to create a novel dynamic programming based decision making paradigm. This is a leap from the classical Bellman based approach, and its application to control tasks is shown to be effective for tasks such as trajectory following, having many desirable properties such as robustness against unknown disturbance and against misspecification of system models.

In Chapter 6, to address the loss of information when dealing with a dynamic environment in statistical estimation (ML) problems, we adopted asymptotic results of exponential sums studied in the number theory literature to devise novel estimation algorithms that have statistical guarantees. Also, this work successfully identified a statistically meaningful set of dynamic structure information which helps increase resolution of dynamical systems perspective as well.

Finally in Chapter 7, as a stepping stone towards the future direction of research which is aimed at treating the ML algorithm itself as a dynamical system to unify the perspectives, we took constructive approach to analyze the deep RL task, which is one of the most complex domains in ML.

8.2 Future direction of research

Before closing, I would like to list some of the future direction of research.

8.2.1 Extension of each contributing chapter

Each aspect in the contributing chapters should be further deepened in the future as there still exist a plenty of interesting ideas in each direction.

Chapter 3 explored control theoretic guarantees for learning systems. There exist plenty of exciting control theoretic results in, for example, multi-agent systems and network controls. We could extend the research to those domains of studies. More abstractly, the work has exploited the property of control theoretic tools that give guarantees on global characteristics of dynamics through local constraints (i.e., constraint on one step dynamics), and it stems from the state space representation of the dynamics. Coming up with novel representation of the dynamics so that some constraints on the learned behaviors can be efficiently encoded should be a promising future work.

In Chapter 4, especially for LC^3, we assumed the RKHS is known. Although using the simulator with different physics parameters as the featurizer is a promising approach, we should carefully study what kinds of representations could make the problem significantly simpler for complex practical tasks including contact-rich dynamics. Although related to Chapter 3, it could also be the case that considering representations for policy by following the idea of *muscle synergy* or the tension of *tendons* helps.

Moreover, the incorporation of the Koopman spectrum cost for decision making opens up a research direction for studying what kind of *duality* may be considered in the decision making or learning settings. Even within the realm of (data-driven) optimization, optimizing both for the primal and dual domains is essential in some applications and extending this idea to decision making problems should help them more flexible in terms of generative capability (recall many physical phenomena embrace duality). We believe the technical contribution of the chapter becomes a stepping stone. Also, it is critical to devise policy gradient type methods for the KSNR to scale up the methodology to more complex practical domains. It could be the case that we need to reconsider the way of sampling trajectories to properly incorporate this spectrum cost in the policy gradient type algorithms.

For the work of Chapter 5, as we briefly mentioned in the chapter, we should research an analogue of policy/value iteration and study its theoretical guarantees. We believe whether one can guarantee convergence to (sub)optimal solutions depends on the conditions on the signature cost; ranging from the linear assumption to convexity. Once this research turns out to be successful, we should be able to devise reliable RL algorithms based on Chen equation. At a meta level, it is pivotal to reconsider how we encode certain tasks executed by a single trajectory in the way that an efficient decision making is still possible. Aside from the path following tasks, our signature framework can encode *higher orders* of reward signals than the cumulative sum that is used in the traditional Bellman based settings, for example.

The work of Chapter 6 is somewhat exploratory for identifying the gap between dynamical systems theory and statistical approach. Digging deeper into other number theoretic results that are unknown in the ML community is a naive but exciting future direction. Although it is not really problem-oriented, naively merging existing results of estimation problems or analysis in signal processing, dynamical systems and potentially in number theory to statistical analysis should produce many novel results.

Finally for Chapter 7, we considered a constructive approach tailored to our specific problem setups; nevertheless, it is anticipated that the AI technology evolves around large scale data and models in the near future, and more general and systematic procedures of this constructive approach for wider domains of ML will be required. Therefore, formalizing this methodology should help the creation of some software framework that can be used to retroductively analyze some large scale complex systems. The abstraction of phenomena observed in the complicated target systems for the sake of *understanding* is a fundamental aspect of the human's cognitive process that the current ML systems may not be able to exercise well. Bringing the words of dynamical systems theory could potentially provide appropriate inductive bias for future ML designs.

8.2.2 Beyond this thesis

Among all, in the future, the following research inquiry should be addressed: *In what manner may we craft intricate "systems" that are endowed with unwavering reliability and robustness, while concurrently ensuring their intrinsic openness and emergent qualities?* This is the dilemma when one seeks optimization of behavior with respect to user-defined (or learned) costs to achieve desirable behaviors while aiming for preserving openness of the "system" and for keeping a room for emergent properties.

However, if looking closely, the biological systems are capable of producing "useful" dynamics and are at the same time flexibly interacting with environments while maintaining body conditions (e.g., homeostasis, homeorhesis [131, 120]). As such, in the long run, *universal biological* view (cf. [120]) may be adopted to (1) resolve inherent contradictions between emergent system quality and its exploitability, (2) establish the clear distinct formulations that stand apart from the prevailing ML paradigm, and (3) scale such novel paradigm to a level of practical viability.

A key idea would be to regard optimization as *hindsight*; in fact, so-called *inverse optimality* is well known in the control community. Instead of optimizing for some predefined cost function, studying the way of exploiting some extracted *stable* dynamics as something that can be described as a result of some cost optimization seems promising.

BIBLIOGRAPHY

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. *Proc. NeurIPS*, 24, 2011.

[2] N. Abe and P. M. Long. Associative reinforcement learning using linear probabilistic concepts. In *Proc. ICML*, pages 3–11, 1999.

[3] I. Abraham and T. D. Murphey. Active learning of dynamics for data-driven control using Koopman operators. *IEEE Trans. Robotics*, 35(5):1071–1083, 2019.

[4] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *Proc. ICML*, 2017.

[5] M. S. Advani, A. M. Saxe, and H. Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[6] C. Aerts, G. Molenberghs, M. Michielsen, M. G. Pedersen, R. Björklund, C. Johnston, J. S. G. Mombarg, D. M. Bowman, B. Buysschaert, P. I. Pápics, et al. Forward asteroseismic modeling of stars with a convective core from gravity-mode oscillations: parameter estimation and stellar model selection. *The Astrophysical Journal Supplement Series*, 237(1):15, 2018.

[7] A. Agarwal, N. Jiang, S. M. Kakade, and W. Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

[8] A. Agarwal, S. Kakade, A. Krishnamurthy, and W. Sun. FLAMBE: Structural complexity and representation learning of low rank MDPs. *Proc. NeurIPS*, 33:20095–20107, 2020.

[9] A. Agarwal, S. M Kakade, J. D. Lee, and G. Mahajan. Optimality and approximation with policy gradient methods in Markov decision processes. In *Conference on Learning Theory*, pages 64–66. PMLR, 2020.

[10] A. Agrawal and K. Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Proc. RSS*, 2017.

[11] A. P. Aguiar and J. P. Hespanha. Trajectory-tracking and path-following of under-actuated autonomous vehicles with parametric modeling uncertainty. *IEEE Trans. Automatic Control*, 52(8):1362–1379, 2007.

[12] A. K Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin. Reachability-based safe learning with Gaussian processes. In *IEEE Proc. CDC*, pages 1424–1431, 2014.

[13] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[14] M. Al Borno, M. De Lasa, and A. Hertzmann. Trajectory optimization for full-body movements with complex contacts. *IEEE Trans. Visualization and Computer Graphics*, 19(8):1405–1414, 2012.

[15] R. M. Allen and H. Kanamori. The potential for earthquake early warning in southern California. *Science*, 300(5620):786–789, 2003.

[16] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *IEEE Proc. ECC*, pages 3420–3431, 2019.

[17] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Automatic Control*, 62(8):3861–3876, 2017.

[18] H. B. Ammar, R. Tutunov, and E. Eaton. Safe policy search for lifelong reinforcement learning with sublinear regret. In *Proc. ICML*, pages 2361–2369, 2015.

[19] A. N. Andry, E. Y. Shapiro, and J. C. Chung. Eigenstructure assignment for linear systems. *IEEE Trans. Aerospace and Electronic Systems*, (5):711–729, 1983.

[20] F. J. Anscombe. Sequential medical trials. *Journal of the American Statistical Association*, 58(302):365–383, 1963.

[21] M. Arbel, D. J. Sutherland, M. Bińkowski, and A. Gretton. On gradient regularizers for MMD GANs. *Proc. NeurIPS*, 31, 2018.

[22] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[23] G. I. Arkhipov, V. N. Chubarikov, and A. A. Karatsuba. *Trigonometric Sums in Number Theory and Analysis*. De Gruyter, Berlin, New York, 2004.

[24] L. Arnold. *Random dynamical systems*. Springer, 1998.

[25] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[26] I. P. Arribas. Derivatives pricing using signature payoffs. *arXiv preprint arXiv:1809.09466*, 2018.

[27] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.

[28] P. Auer, Y. Chen, P. Gajane, C. Lee, H. Luo, R. Ortner, and C. Wei. Achieving optimal dynamic regret for non-stationary bandits without prior information. In *Conference on Learning Theory*, pages 159–163. PMLR, 2019.

[29] A. Ayoub, Z. Jia, C. Szepesvari, M. Wang, and L. Yang. Model-based reinforcement learning with value-targeted regression. In *Proc. ICML*, pages 463–474. PMLR, 2020.

[30] M. G. Azar, R. Munos, and B. Kappen. On the sample complexity of reinforcement learning with a generative model. *arXiv preprint arXiv:1206.6461*, 2012.

[31] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. *Proc. NeurIPS*, 30, 2017.

[32] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Systems, Man, and Cybernetics*, (5):834–846, 1983.

[33] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Proc. NeurIPS*, pages 1471–1479, 2016.

[34] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proc. ICML*, pages 449–458. PMLR, 2017.

[35] R. Bellman. An introduction to the theory of dynamic programming. Technical report, The Rand Corporation, Santa Monica, Calif., 1953.

[36] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[37] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. In *IEEE Proc. CDC*, pages 4661–4666, 2016.

[38] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Proc. NeurIPS*, 2017.

[39] A. Berlinet and C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[40] O. Besbes, Y. Gur, and A. Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Proc. NeurIPS*, 27, 2014.

[41] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[42] H. Boedihardjo and X. Geng. A non-vanishing property for the signature of a path. *Comptes Rendus Mathematique*, 357(2):120–129, 2019.

[43] H. Boedihardjo, X. Geng, T. Lyons, and D. Yang. The signature of a rough path: uniqueness. *Advances in Mathematics*, 293:720–737, 2016.

[44] J. Bourgain. Fourier transform restriction phenomena for certain lattice subsets and applications to nonlinear evolution equations. *Geometric & Functional Analysis GAFA*, 3(3):209–262, 1993.

[45] R. Brault, M. Heinonen, and F. Buc. Random Fourier features for operator-valued kernels. In *Proc. ACML*, pages 110–125, 2016.

[46] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[47] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Proc. NeurIPS*, 33:1877–1901, 2020.

[48] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

[49] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[50] D. Burov, D. Giannakis, K. Manohar, and A. Stuart. Kernel analog forecasting: Multiscale test problems. *Multiscale Modeling & Simulation*, 19(2):1011–1040, 2021.

[51] D. Bzdok, N. Altman, and M. Krzywinski. Statistics versus machine learning. *Nature Methods*, 15(4):233–234, 2018.

[52] H. Cai, Z. Cen, L. Leng, and R. Song. Periodic-GP: Learning periodic world with Gaussian process bandits. *arXiv preprint arXiv:2105.14422*, 2021.

[53] T. Cass, T. Lyons, and X. Xu. General signature kernels. *arXiv preprint arXiv:2107.00447*, 2021.

[54] A. Cassirer, G. Barth-Maron, E. Brevdo, S. Ramos, T. Boyd, T. Sottiaux, and M. Kroiss. Reverb: a framework for experience replay. *arXiv preprint arXiv:2102.04736*, 2021.

[55] J. S. O. Ceron and P. S. Castro. Revisiting Rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *Proc. ICML*, pages 1373–1383. PMLR, 2021.

[56] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Proc. NeurIPS*, pages 2249–2257, 2011.

[57] K. Chen. Iterated integrals and exponential homomorphisms. *Proceedings of the London Mathematical Society*, 3(1):502–512, 1954.

[58] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Proc. NeurIPS*, 31, 2018.

[59] R. T. Q. Chen, B. Amos, and M. Nickel. Learning neural event functions for ordinary differential equations. *Proc. ICLR*, 2021.

[60] Y. Chen, C. Lee, H. Luo, and C. Wei. A new algorithm for non-stationary contextual bandits: Efficient, optimal and parameter-free. In *Conference on Learning Theory*, pages 696–726. PMLR, 2019.

[61] C. Chesneau and Y. J. Bagul. A note on some new bounds for trigonometric functions using infinite products. *Malaysian Journal of Mathematical Sciences*, 14:273–283, 07 2020.

[62] W. Cheung, D. Simchi-Levi, and R. Zhu. Hedging the drift: Learning to optimize under nonstationarity. *Management Science*, 68(3):1696–1713, 2022.

[63] I. Chevyrev and A. Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[64] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Proc. NeurIPS*, 31, 2018.

[65] A. Cohen, T. Koren, and Y. Mansour. Learning linear-quadratic regulators efficiently with only $\sqrt{T}$ regret. In *Proc. ICML*, pages 1300–1309, 2019.

[66] J. Conway et al. The game of life. *Scientific American*, 223(4):4, 1970.

[67] J. Cortés and M. Egerstedt. Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6):495–503, 2017.

[68] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Trans. Robotics and Automation*, 20(2):243–255, 2004.

[69] R. Couillet and M. Debbah. *Random matrix methods for wireless communications.* Cambridge University Press, 2011.

[70] N. Črnjarić-Žic, S. Maćešić, and I. Mezić. Koopman operator spectrum for random dynamical systems. *Journal of Nonlinear Science*, pages 1–50, 2019.

[71] S. Curi, F. Berkenkamp, and A. Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. *Proc. NeurIPS*, 33:14156–14170, 2020.

[72] W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *AAAI*, volume 32, 2018.

[73] V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *21st Annual Conference on Learning Theory*, pages 355–366, 2008.

[74] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control. *IEEE Trans. Robotics*, 2023.

[75] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. *AAAI*, pages 761–768, 1998.

[76] S. A. Derevyanko, J. E. Prilepsky, and S. K. Turitsyn. Capacity estimates for optical transmission based on the nonlinear Fourier transform. *Nature Communications*, 7(1):12710, 2016.

[77] A. S. Dogra and W. Redman. Optimizing neural networks via Koopman operator theory. *Proc. NeurIPS*, 33:2087–2097, 2020.

[78] P. D'Oro and W. Jaśkowski. How to learn a useful critic? model-based action-gradient-estimator policy optimization. *Proc. NeurIPS*, 33:313–324, 2020.

[79] S. Du, S. Kakade, J. Lee, S. Lovett, G. Mahajan, W. Sun, and R. Wang. Bilinear classes: A structural framework for provable generalization in RL. In *Proc. ICML*, pages 2826–2836. PMLR, 2021.

[80] J. L. England. Dissipative adaptation in driven self-assembly. *Nature Nanotechnology*, 10(11):919–923, 2015.

[81] N. Etemadi. An elementary proof of the strong law of large numbers. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 55(1):119–122, 1981.

[82] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. *arXiv preprint arXiv:1710.08446*, 2017.

[83] A. Fermanian. *Learning time-dependent data with the signature transform.* PhD thesis, Sorbonne Université, 2021.

[84] R. P. Feynman. The principle of least action in quantum mechanics. In *Feynman's thesis*. Princeton University, 1942.

[85] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.

[86] R. A. Freeman and J. A. Primbs. Control Lyapunov functions: New ideas from an old source. In *IEEE Proc. CDC*, volume 4, pages 3926–3931, 1996.

[87] F. Fui-Hoon Nah, R. Zheng, J. Cai, K. Siau, and L. Chen. Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration. *Journal of Information Technology Case and Application Research*, 25(3):277–304, 2023.

[88] C. Furusawa and K. Kaneko. A dynamical-systems view of stem cell biology. *Science*, 338(6104):215–217, 2012.

[89] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021.

[90] P. Glotfelter, J. Cortés, and M. Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE Control Systems Letters*, 1(2):310–315, 2017.

[91] L. Grafakos. *Classical Fourier analysis*, volume 2. Springer, 2008.

[92] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. *Proc. NeurIPS*, 30, 2017.

[93] L. G. Gyurkó, T. Lyons, M. Kontkowski, and J. Field. Extracting information from the signature of a financial data stream. *arXiv preprint arXiv:1307.7244*, 2013.

[94] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. ICML*, pages 1861–1870. PMLR, 2018.

[95] B. Hambly and T. Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, pages 109–167, 2010.

[96] E. Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[97] E. Hazan, H. Lee, K. Singh, C. Zhang, and Y. Zhang. Spectral filtering for general linear dynamical systems. *Proc. NeurIPS*, 31, 2018.

[98] M. Hemati and H. Yao. Dynamic mode shaping for fluid flow control: New strategies for transient growth suppression. In *8th AIAA Theoretical Fluid Mechanics Conference*, page 3160, 2017.

[99] T. M. Heskes and B. Kappen. On-line learning processes in artificial neural networks. In *North-Holland Mathematical Library*, volume 51, pages 199–233. Elsevier, 1993.

[100] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Proc. NeurIPS*, 33:6840–6851, 2020.

[101] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[102] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. 2008.

[103] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[104] P. J. Huber. Robust estimation of a location parameter. In *Breakthroughs in Statistics*, pages 492–518. Springer, 1992.

[105] M. E. Hughes, K. C. Abruzzi, R. Allada, R. Anafi, A. B. Arpat, G. Asher, P. Baldi, C. De Bekker, D. Bell-Pedersen, J. Blau, et al. Guidelines for genome-scale analysis of biological rhythms. *Journal of Biological Rhythms*, 32(5):380–393, 2017.

[106] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[107] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[108] I. Ishikawa, K. Fujii, M. Ikeda, Y. Hashimoto, and Y. Kawahara. Metric on nonlinear dynamical systems with Perron-Frobenius operators. In *Proc. NeurIPS*, pages 2856–2866. 2018.

[109] T. Iwata and Y. Kawahara. Neural dynamic mode decomposition for end-to-end modeling of nonlinear dynamics. *arXiv:2012.06191*, 2020.

[110] D. H. Jacobson and D. Q. Mayne. *Differential dynamic programming.* American Elsevier Pub. Co., 1970.

[111] J. Jia and A. R. Benson. Neural jump stochastic differential equations. *Proc. NeurIPS*, 32, 2019.

[112] N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, and R. E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *Proc. ICML*, pages 1704–1713. PMLR, 2017.

[113] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[114] W. Jin, Z. Wang, Z. Yang, and S. Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

[115] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

[116] H. Kadri, E. Duflos, P. Preux, S. Canu, A. Rakotomamonjy, and J. Audiffren. Operator-valued kernels for learning from functional response data. *Journal of Machine Learning Research*, 17(20):1–54, 2016.

[117] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE Proc. ICRA*, pages 1470–1477, 2011.

[118] E. Kaiser, J. N. Kutz, and S. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2021.

[119] S. Kakade, A. Krishnamurthy, K. Lowrey, M. Ohnishi, and W. Sun. Information theoretic regret bounds for online nonlinear control. *Proc. NeurIPS*, 33:15312–15325, 2020.

[120] K. Kaneko. *Life: an introduction to complex systems biology.* Springer, 2006.

[121] Q. Kang, Y. Song, Q. Ding, and W. P. Tay. Stable neural ODE with Lyapunov-stable equilibrium points for defending against adversarial attacks. *Proc. NeurIPS*, 34:14925–14937, 2021.

[122] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[123] N. M. Katz. *Exponential Sums and Differential Equations. (AM-124).* Princeton University Press, 1990.

[124] Y. Kawahara. Dynamic mode decomposition with reproducing kernels for Koopman spectral analysis. *Proc. NeurIPS*, 29:911–919, 2016.

[125] H. Kervadec, J. Dolz, J. Yuan, C. Desrosiers, E. Granger, and I. B. Ayed. Constrained deep networks: Lagrangian optimization via log-barrier extensions. In *IEEE Proc. EUSIPCO*, pages 962–966, 2022.

[126] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.

[127] P. Kidger, P. Bonnier, I. Perez A., C. Salvi, and T. Lyons. Deep signature transforms. *Proc. NeurIPS*, 32, 2019.

[128] P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. *Proc. ICML*, 33:6696–6707, 2020.

[129] D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[130] F. J. Király and H. Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20, 2019.

[131] H. Kitano. Biological robustness. *Nature Reviews Genetics*, 5(11):826–837, 2004.

[132] S. Klus, I. Schuster, and K. Muandet. Eigendecompositions of transfer operators in reproducing kernel Hilbert spaces. *Journal of Nonlinear Science*, 30:283–315, 2020.

[133] M. Kobilarov. Cross-entropy motion planning. *International Journal of Robotics Research*, 31(7):855–871, 2012.

[134] R. Koenker. *Quantile regression*, volume 38. Cambridge University Press, 2005.

[135] J. Z. Kolter and G. Manek. Learning stable deep dynamics models. *Proc. NeurIPS*, 32, 2019.

[136] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proc. National Academy of Sciences of the United States of America*, 17(5):315, 1931.

[137] M. Korda and I. Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[138] M. Korda and I. Mezić. Optimal construction of Koopman eigenfunctions for prediction and control. *IEEE Trans. Automatic Control*, 65(12):5114–5129, 2020.

[139] V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation. In *IEEE Proc. ICRA*, pages 378–383, 2016.

[140] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic mode decomposition: Data-driven modeling of complex systems.* SIAM, 2016.

[141] V. Lakshmikantham and S. Leela. *Differential and Integral Inequalities: Theory and Applications: Volume I: Ordinary Differential Equations*. Academic press, 1969.

[142] S. Lale, K. Azizzadenesheli, B. Hassibi, and A. Anandkumar. Logarithmic regret bound in partially observable linear dynamical systems. *Proc. NeurIPS*, 33:20876–20888, 2020.

[143] T. Lattimore and M. Hutter. PAC bounds for discounted MDPs. In *ALT*, pages 320–334. Springer, 2012.

[144] T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[145] H. Lee. Improved rates for prediction and identification of partially observed linear dynamical systems. In *Algorithmic Learning Theory*, pages 668–698. PMLR, 2022.

[146] H. Lee and C. Zhang. Robust guarantees for learning an autoregressive filter. In *Algorithmic Learning Theory*, pages 490–517. PMLR, 2020.

[147] D. Levin, T. Lyons, and H. Ni. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260*, 2013.

[148] F. L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.

[149] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba. Learning compositional Koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[150] T. P. Lillicrap, J. Hunt, Jonathan, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[151] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C. Hsieh. Neural SDE: Stabilizing neural ODE networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

[152] L. Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1–12, 2010.

[153] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–141, 1963.

[154] K. Lowrey, A. Rajeswaran, S. M. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

[155] X. Lu and B. Van Roy. Information-theoretic confidence bounds for reinforcement learning. In *Proc. NeurIPS*, pages 2458–2466, 2019.

[156] H. Luo, C. Wei, A. Agarwal, and J. Langford. Efficient contextual bandits in non-stationary worlds. In *Conference on Learning Theory*, pages 1739–1776. PMLR, 2018.

[157] W. Luo, W. Sun, and A. Kapoor. Sample-efficient safe learning for online nonlinear control with control barrier functions. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 419–435. Springer, 2022.

[158] C. Lyle, M. G. Bellemare, and P. S. Castro. A comparative analysis of expected and distributional reinforcement learning. In *AAAI*, volume 33, pages 4504–4511, 2019.

[159] C. Lyle, M. Rowland, W. Dabney, M. Kwiatkowska, and Y. Gal. Learning dynamics and generalization in deep reinforcement learning. In *Proc. ICML*, pages 14560–14581. PMLR, 2022.

[160] C. Lyle, M. Rowland, G. Ostrovski, and W. Dabney. On the effect of auxiliary tasks on representation dynamics. In *Proc. AISTATS*, pages 1–9. PMLR, 2021.

[161] T. Lyons. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537*, 2014.

[162] T. J. Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2):215–310, 1998.

[163] T. J. Lyons, M. Caruana, and T. Lévy. *Differential equations driven by rough paths.* Springer, 2007.

[164] T. J. Lyons and N. Sidorova. Sound compression–a rough path approach. In *Proceedings of the 4th International Symposium on Information and Communication Technologies*, 2005.

[165] H. Mania, M. I. Jordan, and B. Recht. Active learning for nonlinear system identification with guarantees. *The Journal of Machine Learning Research*, 23(1):1433–1462, 2022.

[166] H. Mania, S. Tu, and B. Recht. Certainty equivalent control of LQR is efficient. *arXiv preprint arXiv:1902.07826*, 2019.

[167] A. Mauroy and I. Mezić. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Trans. Automatic Control*, 61(11):3356–3369, 2016.

[168] A. Mauroy, I. Mezić, and J. Moehlis. Isostables, isochrons, and Koopman spectrum for the action–angle representation of stable fixed point dynamics. *Physica D: Nonlinear Phenomena*, 261:19–30, 2013.

[169] A. Mauroy, Y. Susuki, and I. Mezić. *The Koopman operator in systems and control.* Springer, 2020.

[170] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[171] L. Meng and B. Zheng. The optimal perturbation bounds of the Moore-Penrose inverse under the Frobenius norm. *Linear algebra and its applications*, 432(4):956–963, 2010.

[172] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005.

[173] Z. Mhammedi, D. J. Foster, M. Simchowitz, D. Misra, W. Sun, A. Krishnamurthy, A. Rakhlin, and J. Langford. Learning the linear quadratic regulator from nonlinear observations. *Proc. NeurIPS*, 33:14532–14543, 2020.

[174] T. M. Mitchell. *Machine learning.* McGraw-Hill, 1997.

[175] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[176] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[177] C. Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(2):199, 1991.

[178] I. Mordatch, K. Lowrey, and E. Todorov. Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids. In *IEEE Proc. IROS*, pages 5307–5314, 2015.

[179] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):1–8, 2012.

[180] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka. Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*, 2012.

[181] J. Morrill, C. Salvi, P. Kidger, and J. Foster. Neural rough differential equations for long time series. In *Proc. ICML*, pages 7829–7838. PMLR, 2021.

[182] B. J. Morris, M. J. Powell, and A. D. Ames. Continuity and smoothness properties of nonlinear optimization-based feedback controllers. In *IEEE Proc. CDC*, pages 151–158, 2015.

[183] H. Nakao. Phase reduction approach to synchronization of nonlinear oscillators. *Contemporary Physics*, 57(2):188–214, 2017.

[184] Damianou A. Niekerk, B. V. and B. Rosman. Online constrained model-based reinforcement learning. In *Proc. AUAI*, 2017.

[185] G. Notomista, S. F. Ruf, and M. Egerstedt. Persistification of robotic tasks using control barrier functions. *IEEE Robotics and Automation Letters*, 3(2):758–763, 2018.

[186] S. Oh, A. M. Appavoo, and S. Gilbert. Periodic bandits and wireless network selection. *arXiv preprint arXiv:1904.12355*, 2019.

[187] T. Oh. Note on a lower bound of the Weyl sum in Bourgain's NLS paper (GAFA'93).

[188] M. Ohnishi, I. Akinola, J. Xu, A. Mandlekar, and F. Ramos. Signatures meet dynamic programming: Generalizing Bellman equations for trajectory following. *arXiv preprint arXiv:2312.05547 (accepted for publication in L4DC 2024)*, 2023.

[189] M. Ohnishi, I. Ishikawa, Y. Kuroki, and M. Ikeda. Dynamic structure estimation from bandit feedback. *arXiv preprint arXiv:2206.00861*, 2022.

[190] M. Ohnishi, I. Ishikawa, K. Lowrey, M. Ikeda, S. Kakade, and Y. Kawahara. Koopman spectrum nonlinear regulator and provably efficient online learning. *arXiv preprint arXiv:2106.15775*, 2021.

[191] M. Ohnishi, G. Notomista, M. Sugiyama, and M. Egerstedt. Constraint learning for control tasks with limited duration barrier functions. *Automatica*, 127, 2021.

[192] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Trans. Robotics*, 35(5):1186–1205, 2019.

[193] I. Osband and B. Van Roy. Model-based reinforcement learning and the Eluder dimension. In *Proc. NeurIPS*, pages 1466–1474, 2014.

[194] S. J. Pan, Q. Yang, et al. A survey on transfer learning. *IEEE Trans. Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[195] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

[196] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proc. ICML*, pages 16–17, 2017.

[197] A. Patterson, V. Liao, and M. White. Robust losses for learning value functions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 45(5):6157–6167, 2022.

[198] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, 2018.

[199] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt. The Robotarium: A remotely accessible swarm robotics research testbed. In *IEEE Proc. ICRA*, pages 1699–1706, 2017.

[200] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Proc. NeurIPS*, 29, 2016.

[201] M. Raginsky, A. Rakhlin, and M. Telgarsky. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703. PMLR, 2017.

[202] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proc. NeurIPS*, pages 1177–1184, 2008.

[203] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade. Towards generalization and simplicity in continuous control. *Proc. NeurIPS*, 2017.

[204] E. M. Rathje, N. A. Abrahamson, and J. D. Bray. Simplified frequency content estimates of earthquake ground motions. *Journal of Geotechnical and Geoenvironmental Engineering*, 124(2):150–159, 1998.

[205] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[206] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *IEEE Proc. CDC*, pages 3717–3724, 2020.

[207] I. D. J. Rodriguez, A. Ames, and Y. Yue. LyaNet: A Lyapunov framework for training neural ODEs. In *Proc. ICML*, pages 18687–18703. PMLR, 2022.

[208] A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2021.

[209] M. Rosca, T. Weber, A. Gretton, and S. Mohamed. A case for new neural network smoothness constraints. 2020.

[210] M. Rowland, M. Bellemare, W. Dabney, R. Munos, and Y. W. Teh. An analysis of categorical distributional reinforcement learning. In *Proc. AISTATS*, pages 29–37. PMLR, 2018.

[211] Y. Russac, C. Vernade, and O. Cappé. Weighted linear bandits for non-stationary environments. *Proc. NeurIPS*, 32, 2019.

[212] D. Russo and B. Van R. Eluder dimension and the sample complexity of optimistic exploration. In *Proc. NeurIPS*, pages 2256–2264, 2013.

[213] F. Sabetta and A. Pugliese. Estimation of response spectra and simulation of nonstationary earthquake ground motions. *Bulletin of the Seismological Society of America*, 86(2):337–352, 1996.

[214] C. Salvi, T. Cass, J. Foster, T. Lyons, and W. Yang. The signature kernel is the solution of a Goursat PDE. *SIAM Journal on Mathematics of Data Science*, 3(3):873–899, 2021.

[215] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[216] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe exploration for active learning with Gaussian processes. In *Proc. ECML PKDD*, pages 133–149, 2015.

[217] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

[218] J.-P. Serre. *Linear Representations of Finite Groups.* Springer New York, NY, 1977.

[219] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[220] S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[221] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[222] M. Simchowitz, R. Boczar, and B. Recht. Learning linear dynamical systems with semi-parametric least squares. In *Conference on Learning Theory*, pages 2714–2802. PMLR, 2019.

[223] M. Simchowitz and D. Foster. Naive exploration is optimal for online LQR. In *Proc. ICML*, pages 8937–8948. PMLR, 2020.

[224] B. F. Skinner. Two types of conditioned reflex: A reply to Konorski and Miller. *The Journal of General Psychology*, 16(1):272–279, 1937.

[225] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. ICML*, pages 2256–2265. PMLR, 2015.

[226] P. G. Sokolove and W. N. Bushell. The Chi square periodogram: its utility for analysis of circadian rhythms. *Journal of Theoretical Biology*, 72(1):131–160, 1978.

[227] Y. Song. A note on the variation of the spectrum of an arbitrary matrix. *Linear algebra and its applications*, 342(1-3):41–46, 2002.

[228] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[229] S. H. Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[230] C. Summers, K. Lowrey, A. Rajeswaran, S. Srinivasa, and E. Todorov. Lyceum: An efficient and scalable ecosystem for robot learning. In *Proc. L4DC*, pages 793–803. PMLR, 2020.

[231] W. Sun, N. Jiang, A. Krishnamurthy, A. Agarwal, and J. Langford. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pages 1–36, 2019.

[232] G. Sutanto, A. Wang, Y. Lin, M. Mukadam, G. Sukhatme, A. Rai, and F. Meier. Encoding physical constraints in differentiable Newton-Euler algorithm. In *Proc. L4DC*, pages 804–813. PMLR, 2020.

[233] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[234] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proc. NeurIPS*, pages 1057–1063, 2000.

[235] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[236] Emanuel T. and Weiwei L. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proc. ACC*, pages 300–306, 2005.

[237] Y. Tang, M. Rowland, R. Munos, B. Á. Pires, W. Dabney, and M. G. Bellemare. The nature of temporal difference errors in multi-step distributional reinforcement learning. *arXiv preprint arXiv:2207.07570*, 2022.

[238] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

[239] P. S. Thomas, W. C. Dabney, S. Giguere, and S. Mahadevan. Projected natural actor-critic. In *Proc. NeurIPS*, pages 2337–2345, 2013.

[240] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[241] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE Proc. IROS*, pages 23–30, 2017.

[242] E. Todorov. Linearly-solvable Markov decision problems. *Proc. NeurIPS*, 19, 2006.

[243] E. Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009.

[244] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *IEEE Proc. IROS*, pages 5026–5033, 2012.

[245] F. Trovo, S. Paladino, M. Restelli, and N. Gatti. Sliding-window Thompson sampling for non-stationary settings. *Journal of Artificial Intelligence Research*, 68:311–364, 2020.

[246] A. Tsiamis, N. Matni, and G. Pappas. Sample complexity of Kalman filtering for unknown systems. In *Proc. L4DC*, pages 435–444. PMLR, 2020.

[247] A. Tsiamis and G. J. Pappas. Finite sample analysis of stochastic system identification. In *IEEE Proc. CDC*, pages 3648–3654, 2019.

[248] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.

[249] S. K. Turitsyn, J. E. Prilepsky, S. T. Le, S. Wahls, L. L. Frumin, M. Kamalian, and S. A. Derevyanko. Nonlinear Fourier transform for optical data processing and transmission: advances and perspectives. *Optica*, 4(3):307–322, 2017.

[250] B. Tzen and M. Raginsky. Neural stochastic differential equations: Deep latent Gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

[251] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[252] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Proc. NeurIPS*, 30, 2017.

[253] J. Von Neumann, A. W. Burks, et al. Theory of self-reproducing automata. *IEEE Trans. Neural Networks*, 5(1):3–14, 1966.

[254] N. Wagener, C. Cheng, J. Sacks, and B. Boots. An online learning approach to model predictive control. *Proc. RSS*, 2019.

[255] L. Wang, A. D. Ames, and M. Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Trans. Robotics*, 2017.

[256] L. Wang, E. A. Theodorou, and M. Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *Proc. ICRA*, pages 2460–2465. IEEE, 2018.

[257] T. Wang and J. Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

[258] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[259] P. Wedin. Perturbation theory for pseudo-inverses. *BIT Numerical Mathematics*, 13(2):217–232, 1973.

[260] H. Weyl. Über die gleichverteilung von zahlen mod eins. *Math. Ann.*, 77:31–352, 1916.

[261] P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. *Proc. IFAC*, 40(12):462–467, 2007.

[262] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[263] A. T. Winfree. *The Geometry of Biological Time*. Springer, 2001.

[264] C. J. Wolfe. On the properties of predominant-period estimators for earthquake early warning. *Bulletin of the Seismological Society of America*, 96(5):1961–1965, 2006.

[265] Q. Wu, N. Iyer, and H. Wang. Learning contextual bandits in a non-stationary environment. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 495–504, 2018.

[266] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

[267] W. Xiao, T. Wang, R. Hasani, M. Lechner, Y. Ban, C. Gan, and D. Rus. On the forward invariance of neural ODEs. In *Proc. ICML*, pages 38100–38124. PMLR, 2023.

[268] Z. Xie, Z. Sun, L. Jin, H. Ni, and T. Lyons. Learning spatial-semantic context with fully convolutional recurrent network for online handwritten Chinese text recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 40(8):1903–1917, 2017.

[269] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. In *Proc. ICLR*, 2021.

[270] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. Robustness of control barrier functions for safety critical control. *Proc. IFAC*, 48(27):54–61, 2015.

[271] W. Yang, T. Lyons, H. Ni, C. Schmid, and L. Jin. Developing the path signature methodology and its application to landmark-based human action recognition. In *Stochastic Analysis, Filtering, and Stochastic Optimization: A Commemorative Volume to Honor Mark HA Davis's Contributions*, pages 431–464. Springer, 2022.

[272] Y. Yoshida and T. Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

[273] D. Zhou and M. Schwager. Vector field following for quadrotors using differential flatness. In *IEEE Proc. ICRA*, pages 6567–6572, 2014.

[274] T. Zielinski, A. M. Moore, E. Troup, K. J. Halliday, and A. J. Millar. Strengths and limitations of period estimation methods for circadian data. *PloS one*, 9(5):e96462, 2014.

Appendix A
ADDITIONAL EXPERIMENTAL RESULTS

Detailed experimental setups and additional results are presented here.

A.1 Simulation setups and results of LC^3

Below, we provide simulation setups, including the details of environments and parameter settings. Specifically, the hyperparameters, namely, 1) variance of random control variation for MPPI, 2) temperature parameter for MPPI, 3) planning horizon, 4) number of planning samples, 5) prior parameter λ, 6) posterior reshaping constant, 7) number of episodes between model updates, 8) number of features, and 9) RFF bandwidth, are presented.

Note parameters were tuned in the following way: we first tuned MPPI parameters on ground truth models, then we tuned number of RFFs, their bandwidth, prior parameter, and posterior reshaping constant.

A.1.1 Gym environments

The hyperparameters used for InvertedPendulum, Acrobot, CartPole, Mountain Car, Reacher, and Hopper are shown in Table A.1, A.2, A.3, A.4, A.5, and A.6, respectively. We used `JULIA_NUM_THREADS=12` for all the Gym experiments.

We mention that we tested many heuristics to improve performance such as input normalization, different prior parameter for each output dimension, using multiple bandwidth of RFFs, ensemble of RFF models, warm start of planner, experience replay, etc., however, none of them *consistently* improved the performance across tasks. Therefore we present the results with no such heuristics in this paper. Interestingly, increasing number of RFFs for some contact-rich dynamics such as Hopper did not reduce the modeling error significantly.

Table A.1: Hyperparameters used for InvertedPendulum environment.

MPPI hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.2^2	number of features	200
temperature parameter	0.1	RFF bandwidth	5.5
planning horizon	10	prior parameter	10^{-4}
number of planning samples	256	posterior reshaping constant	0
		episodes between model updates	1

Table A.2: Hyperparameters used for Acrobot environment.

MPPI hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.2^2	number of features	200
temperature parameter	0.3	RFF bandwidth	4.5
planning horizon	30	prior parameter	0.01
number of planning samples	256	posterior reshaping constant	10^{-3}
		episodes between model updates	1

Being able to model some of the critical interactions such as contacts seems to be the key for success of such a complicated environment.

A.1.2 Maze

In the Maze environment, states and controls are continuous and the agent plans over continuous spaces; however, the dynamics is given by 1) $x_{h+1} = x_h + [-0.5, 0]^\top$ (i.e., moving one step left) if $\lceil 2u_h \rceil = -1$, 2) $x_{h+1} = x_h + [0, -0.5]^\top$ (i.e., moving one step up) if $\lceil 2u_h \rceil = 0$, 3) $x_{h+1} = x_h + [0.5, 0]^\top$ (i.e., moving one step right) if $\lceil 2u_h \rceil = 1$, and 4) $x_{h+1} = x_h + [0, 0.5]^\top$ (i.e., moving one step down) if $\lceil 2u_h \rceil = 2$, except for the case there is a wall in the direction

Table A.3: Hyperparameters used for CartPole environment.

MPPI hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.2^2	number of features	200
temperature parameter	0.1	RFF bandwidth	1.5
planning horizon	50	prior parameter	5×10^{-4}
number of planning samples	128	posterior reshaping constant	10^{-4}
		episodes between model updates	1

Table A.4: Hyperparameters used for Mountain Car environment.

MPPI hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.3^2	number of features	100
temperature parameter	0.2	RFF bandwidth	1.3
planning horizon	110	prior parameter	0.01
number of planning samples	512	posterior reshaping constant	10^{-6}
		episodes between model updates	1

Table A.5: Hyperparameters used for Reacher environment.

MPPI hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.2^2	number of features	300
temperature parameter	0.3	RFF bandwidth	4.0
planning horizon	20	prior parameter	0.01
number of planning samples	256	posterior reshaping constant	0
		episodes between model updates	4

Table A.6: Hyperparameters used for Hopper environment.

MPPI hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.2^2	number of features	200
temperature parameter	0.2	RFF bandwidth	12.0
planning horizon	128	prior parameter	5×10^{-3}
number of planning samples	64	posterior reshaping constant	0.01
		episodes between model updates	1

of travel, which then ends up $x_{h+1} = x_h$.

The hyperparameters of Maze experiments are shown in Table A.7. Note the number of features is 100 because one hot vector (e.g., $\phi(x, u) = [1, 0, \ldots, 0]^\top$ if each coordinate of x is smaller than -0.75 and $u \leq -0.5$) in this maze environment is 100 dimension. Table A.7 also includes the parameters for PETS-CEM; we used the recommended values in the paper and the codebase, except for the planning horizon, which was set to be the same as the MPPI counterpart. We used `JULIA_NUM_THREADS=12` for all the Maze experiments.

A.1.3 Armhand with model ensemble features

In table A.8, we list the dynamical properties that were randomized to make our ensemble. We use uniform distributions to present a window of possible, realistic values for the parameters: for example, we randomize the objects mass between 0.1 and 1.0 kg. The center of mass distributions is the deviation from the center of the sphere, while the moments of inertia parameter is one value applied to all elements of a diagonal inertia matrix for the object. The contact parameters are specific to the MuJoCo dynamics simulator we use [244], and are the parameters of internal contact model of the simulation. The range of values of the parameters allow for objects in the ensemble to have different softness and rebound effects.

Table A.7: Hyperparameters used for Maze environment.

Planner hyperparameter	Value	LC^3 hyperparameter	Value
variance of controls	0.3^2	number of features	100
temperature parameter	0.05	prior parameter	0.01
MPPI planning horizon	50	posterior reshaping constant	10^{-3} (*best*)
MPPI planning samples	1024	episodes between model updates	1
PETS-CEM horizon	50		
PETS-CEM samples	500		
PETS-CEM elite size	50		

Also, Table A.9 lists learned model predictive error for different features, indicating that the ensemble of MuJoCo model successfully captured the true dynamics.

A.2 Simulation setups and results of KSNR

In this section, we provide simulation setups, including the details of environments (see also Figure A.1) and parameter settings.

A.2.1 Cross-entropy method

Throughout, we used CEM for dynamics parameter (policy) selection to approximately solve KSNR. Here, we present the setting of CEM.

First, we prepare some fixed feature (e.g., RFFs) for ϕ. Then, at each iteration of CEM, we generate many parameters to compute the loss (i.e., the sum of the Koopman spectrum cost and *negative* cumulative reward) by fitting the transition data generated by each parameter to the feature to estimate its Koopman operator $\mathscr{A}$. In particular, we used the following regularized fitting:

$$\mathscr{A} = YX^\top(XX^\top + I)^{-1},$$

Table A.8: Hyperparameters used for Armhand environment.

Hyperparameter	Value	Ensemble parameter	Value
variance of controls	0.2^2	models in ensemble	6
temperature parameter	0.08	mass	$\mathcal{U}(0.01, 1.0)$
planning horizon	50	center of mass	$\mathcal{U}(-0.04, 0.04) \times 3$
number of planning samples	64	moments of inertia	$\mathcal{U}(0.0001, 0.0004)$
prior parameter	10^{-4}	contact param. (solimp)	$[\mathcal{U}(0.5, 0.99),$
			$\mathcal{U}(0.4, 0.98),$
			$\mathcal{U}(0.0001, 0.01),$
			$\mathcal{U}(0.49, 0.51),$
			$\mathcal{U}(1.9, 2.1)]$
posterior reshaping constant	0.01	Contact Param. (solref)	$[\mathcal{U}(0.01, 0.03),$
			$\mathcal{U}(0.9, 1.1)]$
episodes between model updates	1		

Table A.9: Learned model predictive error for different features.

Feature method	Predictive error: $\lVert x_{h+1} - W\phi \rVert_2 / \lVert W\phi \rVert_2$
Random Fourier features, 2048	0.22
2 layer neural network, 2048 hidden, *relu* activation	0.41
Model ensemble of 6 models	0.09

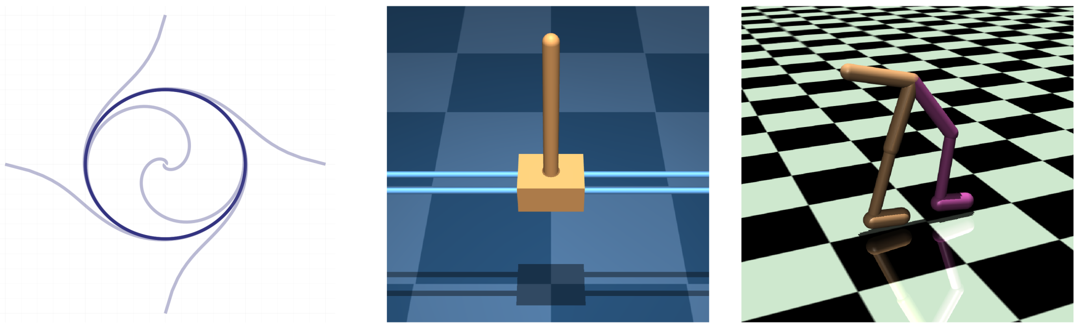


Figure A.1: Illustration of some dynamical systems we have used in Chapter 4. Left: Simple limit cycle represented effectively by the Koopman modes. Middle: DeepMind Control Suite [238] Cartpole showing stable cycle with spectral radius regularization. Right: OpenAI Gym [46] Walker2d showing simpler movement cycle when the Koopman eigenvalues are regularized.

where $Y := [\boldsymbol{\phi}_{x_{h_1+1,1}}, \boldsymbol{\phi}_{x_{h_2+1,2}}, \ldots, \boldsymbol{\phi}_{x_{h_n+1,n}}]$ and $X := [\boldsymbol{\phi}_{x_{h_1,1}}, \boldsymbol{\phi}_{x_{h_2,2}}, \ldots, \boldsymbol{\phi}_{x_{h_n,n}}]$.

If the feature spans a Koopman invariant space and the deterministic dynamical systems are considered, and if no regularization (i.e., the identity matrix I) is used, any sufficiently rich trajectory data may be used to exactly compute $\mathscr{K}(\pi)$ for π. However, in practice, the estimate depends on transition data although the regularization mitigates this effect. In our simulations, at each iteration, we randomly reset the initial state according to some distribution, and computed loss for each parameter generating trajectory starting from that state.

A.2.2 On Koopman modes

Suppose that the target Koopman operator $\mathscr{A}^\star$ has eigenvalues $\lambda_i \in \mathbb{C}$ and eigenfunctions $\xi_i : \mathcal{X} \to \mathbb{C}$ for $i \in \{1, 2, \ldots, d_\phi\}$, i.e.,

$$\mathscr{A}^\star \xi_i = \lambda_i \xi_i.$$

If the observable ϕ satisfies

$$\boldsymbol{\phi}_x = \sum_{i=1}^{d_\phi} \xi_i(x)\mathbf{m}_i^\star,$$

for $\mathbf{m}_i^\star \in \mathbb{C}^{d_\phi}$, then $\mathbf{m}_i^\star$s are called the Koopman modes. The Koopman modes are closely related to the concept *isostable*; interested readers are referred to [168] for example.

We mention the spectrum cost $\Lambda(\mathscr{A}) = \|\mathbf{m}_1 - \mathbf{m}_1^\star\|_1$ does not satisfy the Hölder condition (Assumption 4.3.11) in general; therefore this cost might not be used for KS-LC^3.

A.2.3 Setups: imitating target behaviors through Koopman operators

The discrete-time dynamics

$$r_{h+1} = r_h + v_{r,h}\Delta t,\ \theta_{h+1} = \theta_h + v_{\theta,h}\Delta t$$

is considered and the policy returns $v_{r,h}$ and $v_{\theta,h}$ given r_h and θ_h. In our simulation, we used $\Delta t = 0.05$. Note the ground-truth dynamics

$$\dot{r} = r(1-r^2),\ \dot{\theta} = 1,$$

is discretized to

$$r_{h+1} = r_h + r_h(1-r_h^2)\Delta t,\ \theta_{h+1} = \theta_h + \Delta t.$$

Figure A.2 plots the ground-truth trajectories of observations and x-y positions.

We trained the target Koopman operator using the ground-truth dynamics with random initializations; the hyperparameters used for training are summarized in Table A.10.

Then, we used CEM to select policy so that the spectrum cost is minimized; the hyperparameters are also summarized in Table A.10.

We tested two forms of the spectrum cost; $\Lambda_1(\mathscr{A}) = \|\mathbf{m} - \mathbf{m}^\star\|_1$ and $\Lambda_2(\mathscr{A}) = \|\mathscr{A} - \mathscr{A}^\star\|_{\mathrm{HS}}^2$. The resulting trajectories are plotted in Figure A.3 and A.4, respectively. It is interesting to observe that the top mode imitation successfully converged to the desirable

Table A.10: Hyperparameters used for limit cycle generation.

CEM hyperparameter	Value	Training target Koopman operator	Value
samples	200	training iteration	500
elite size	20	RFF bandwidth for ϕ	3.0
iteration	50	RFF dimension d_ϕ	80
planning horizon	80	horizon for each iteration	80
policy RFF dimension	50		
policy RFF bandwidth	2.0		

limit cycle while Frobenius norm imitation did not. Intuitively, the top mode imitation focuses more on reconstructing the practically and physically meaningful behavior while minimizing the error on the Frobenius norm has no immediately clear physical meaning.

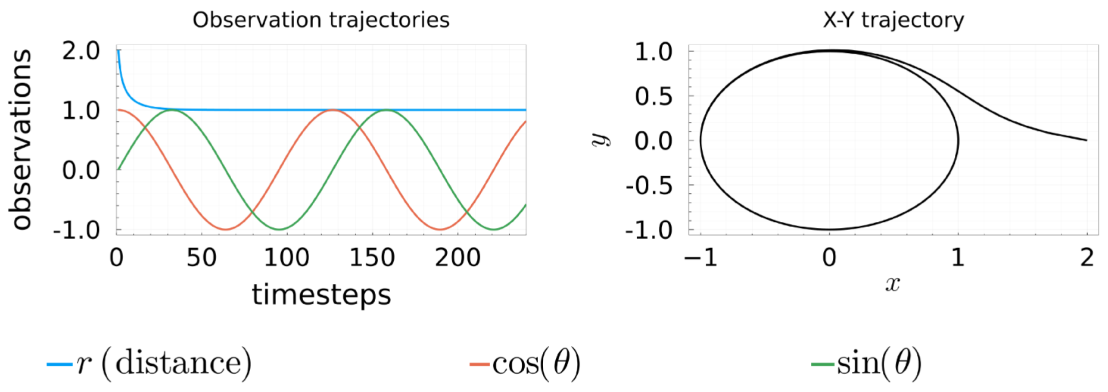


Figure A.2: The ground-truth trajectory of the limit cycle $\dot{r} = r(1 - r^2)$, $\dot{\theta} = 1$. Left: Observations r, $\cos(\theta)$, and $\sin(\theta)$. Right: x-y positions.

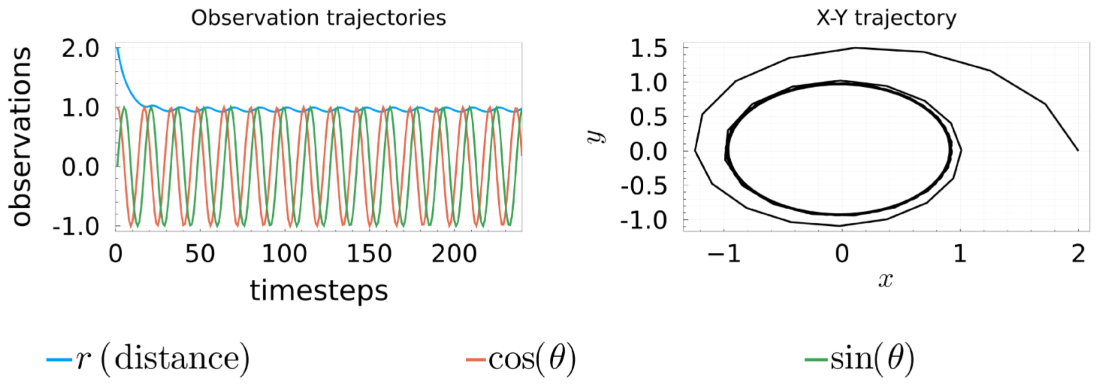


Figure A.3: The trajectory generated by RFF policies that minimize $\Lambda(\mathscr{A}) = \|\mathbf{m} - \mathbf{m}^\star\|_1$. Left: Observations r, $\cos(\theta)$, and $\sin(\theta)$. Right: x-y positions.

A.2.4 Setups: Generating stable loops (Cartpole)

We used DeepMind Control Suite Cartpole environment with modifications; specifically, we extended the cart rail to $[-100, 100]$ from the original length $[-5, 5]$ to deal with divergent behaviors. Also, we used a combination of linear and RFF features; the first elements of the feature are simply the observation (state) vector, and the rest are Gaussian RFFs. That way, we found divergent behaviors were well-captured in terms of spectral radius. The hyperparemeters used for CEM are summarized in Table A.11.

Table A.11: Hyperparameters used for stable loop generation.

Hyperparameter	Value	Hyperparameter	Value
samples	200	elite size	20
iteration	100	planning horizon	100
dimension d_ϕ	50	RFF bandwidth for ϕ	2.0
policy RFF dimension	100	policy RFF bandwidth	2.0

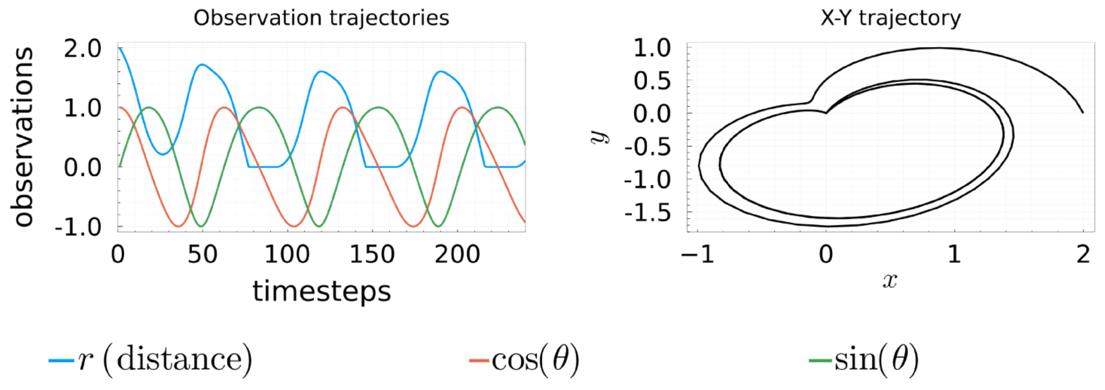


Figure A.4: The trajectory generated by RFF policies that minimize $\Lambda(\mathscr{A}) = \|\mathscr{A} - \mathscr{A}^\star\|_{\mathrm{HS}}^2$. Left: Observations r, $\cos(\theta)$, and $\sin(\theta)$. Right: x-y positions.

A.2.5 Setups: Generating smooth movements (Walker)

Because of the complexity of the dynamics, we used four random seeds in this simulation, namely, 100, 200, 300, and 400. We used a combination of linear and RFF features for both ϕ and the policy. Note, according to the work [203], linear policy is actually sufficient for some tasks for particular environments. The hyperparemeters used for CEM are summarized in Table A.12.

Table A.12: Hyperparameters used for Walker.

Hyperparameter	Value	Hyperparameter	Value
samples	300	elite size	20
iteration	50	planning horizon	300
dimension of d_ϕ	200	RFF bandwidth for ϕ	5.0
policy RFF dimension	300	policy RFF bandwidth	30.0

The resulting trajectories of Walker are illustrated in Figure A.5. The results are rather surprising; because we did not specify the height in reward, the dynamics with only cumulative cost showed rolling behavior (top figure) to go right faster most of the time. On the other hand, when the spectrum cost was used, the hopping behavior (down figure) emerged. Indeed this hopping behavior moves only one or two joints most of the time while fixing other joints, which leads to lower (absolute values of) eigenvalues.

The eigenspectrums of the resulting dynamics with/without the spectrum cost are plotted in Figure A.6. In fact, it is observed that the dynamics when the spectrum cost was used showed consistently lower (absolute values of) eigenvalues; for the hopping behavior, most of the joint angles converged to some values and stayed there.

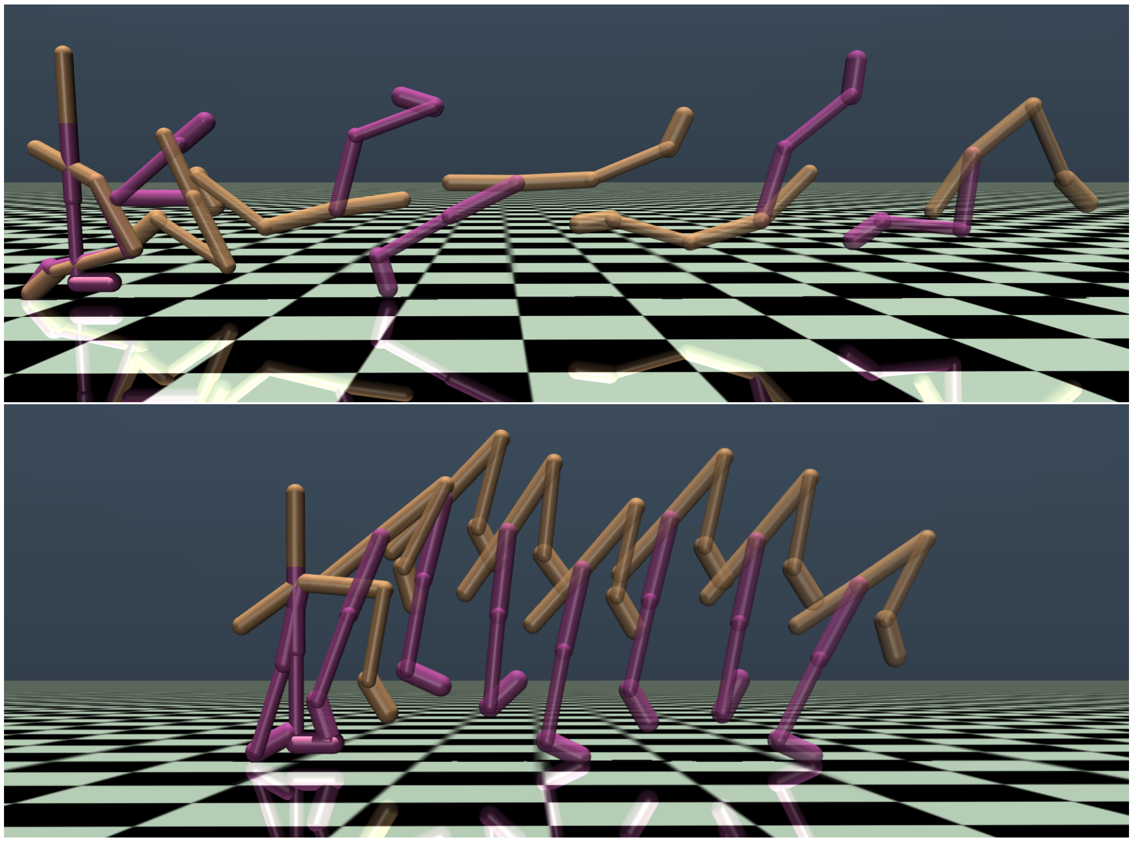


Figure A.5: Walker trajectories visualized via Lyceum. Top: When only (single-step) reward $v - 0.001\|a\|^2_{\mathbb{R}^6}$ is used, showing rolling behavior. Down: When the spectrum cost $\Lambda(\mathscr{A}) = 5\sum_{i=1}^{d_\phi} |\lambda_i(\mathscr{A})|$ is used together with the reward, showing simple hopping behavior.

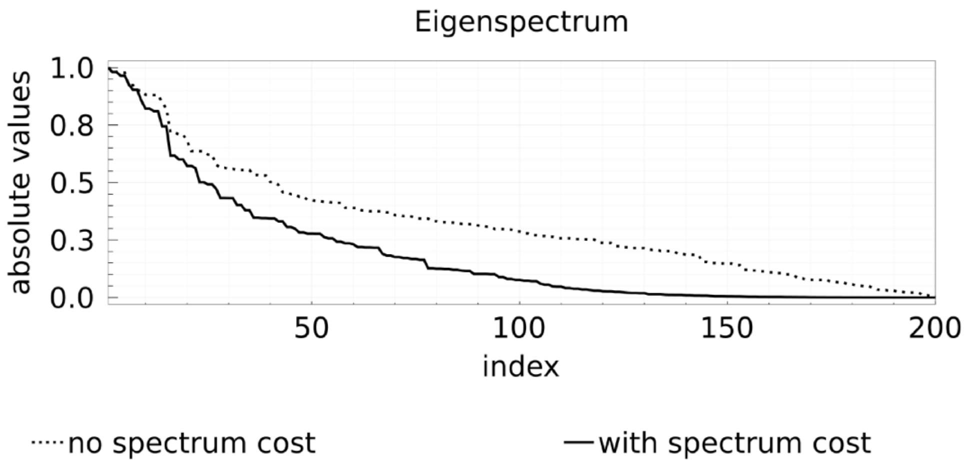


Figure A.6: Eigenspectrums showing absolute values of eigenvalues for the dynamics with/without the spectrum cost.

A.2.6 Additional experiments on smooth Walker motions

To investigate smooth motion generations studied in the main body of this thesis more, we conducted additional experiments. Especially, we also compare our KSNR for smoothness enhancements to the use of action costs in the Walker environment. In this experiment, we used the hyperparameters summarized in Table A.13. We again used a combination of linear and RFF features for both ϕ and the policy. Recall the default immediate reward is $v - 0.001\|a\|^2_{\mathbb{R}^6}$, where v is the velocity and a is the action vector of dimension 6. Here, in addition to KSNR, we tested increased action cost scenarios where the immediate rewards are $v - 0.01\|a\|^2_{\mathbb{R}^6}$ and $v - 0.1\|a\|^2_{\mathbb{R}^6}$ respectively. Across the six seed runs (of seed numbers of 100, 200, 300, 400, 500, and 600), we obtained the mean of the cumulative reward and the cumulative action cost (which is computed for the trajectories using $0.001\|a\|^2_{\mathbb{R}^6}$ for all of the cases), and the mean and standard deviation of the spectrum cost, all of which are summarized in Table A.14. As observed, increased action cost in our scenarios shows lower spectrum cost; while KSNR shows better cumulative reward with better spectrum cost. However, the motion generated by the increased action cost shows lower action penalty cost;

which implies that the spectrum cost and the action cost have some correlation while they qualitatively prefer different motions. We also measured the smoothness by another metric than the spectrum cost itself, which is defined by

$$\text{Smoothness}(\tau) := \frac{1}{d_x H} \sum_{h=0}^{H-1} \|x_{h+1} - x_h\|_1,$$

where $\tau := \{x_h\}_{h=0}^{H}$ is a trajectory. The mean smoothness values across the runs for the motions generated by the CEM algorithms with default action cost, 10 times more action cost, 100 times more action cost, and with the spectrum cost are 0.082, 0.033, 0.007, and 0.028 respectively, and they appear to be consistent to the spectrum cost in this case. The motions are visualized in Figure A.8; their joint trajectories are plotted in Figure A.7 and the eigenspectrums are given in Figure A.9. Note those motions are of those showing median values of the spectrum cost within the seed runs.

Table A.13: Hyperparameters used for additional Walker smoothness experiments.

Hyperparameters	Value	Hyperparameters	Value
samples	300	elite size	20
iteration	120	planning horizon	300
dimension of d_ϕ	200	RFF bandwidth for ϕ	5.0
policy RFF dimension	300	policy RFF bandwidth	30.0

Table A.14: Cumulative reward, cumulative action cost (penalty), and spectrum cost comparisons.

Method (Env. setting)	Reward	Penalty	Spectrum cost (mean)	Spectrum cost (std)
CEM (default action cost)	1011.5	177.3	317.0	±33.2
CEM (×10 action cost)	596.4	10.9	213.2	±50.0
CEM (×100 action cost)	63.4	0.5	88.8	±46.6
CEM (with spectrum cost)	737.8	78.1	186.6	±88.4

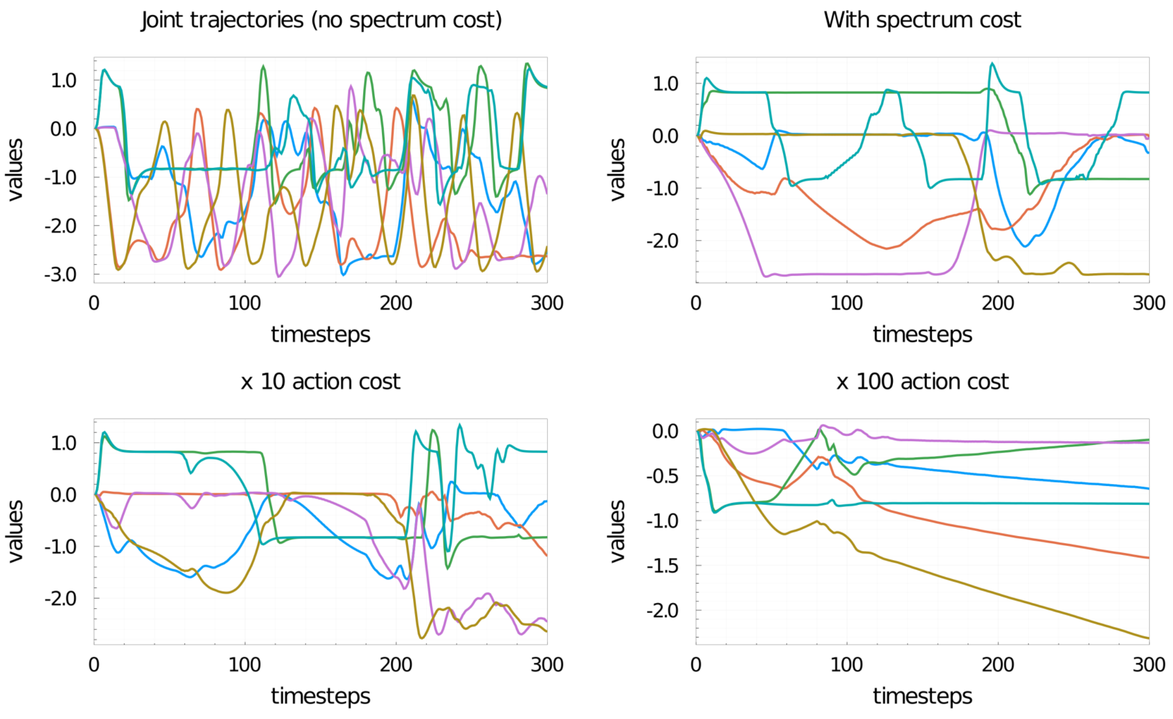


Figure A.7: Joint trajectories of Walker motions generated by the CEM algorithm with default action cost, 10 times more action cost, 100 times more action cost, and with the spectrum cost. They are of those showing median values of the spectrum cost within the seed runs.

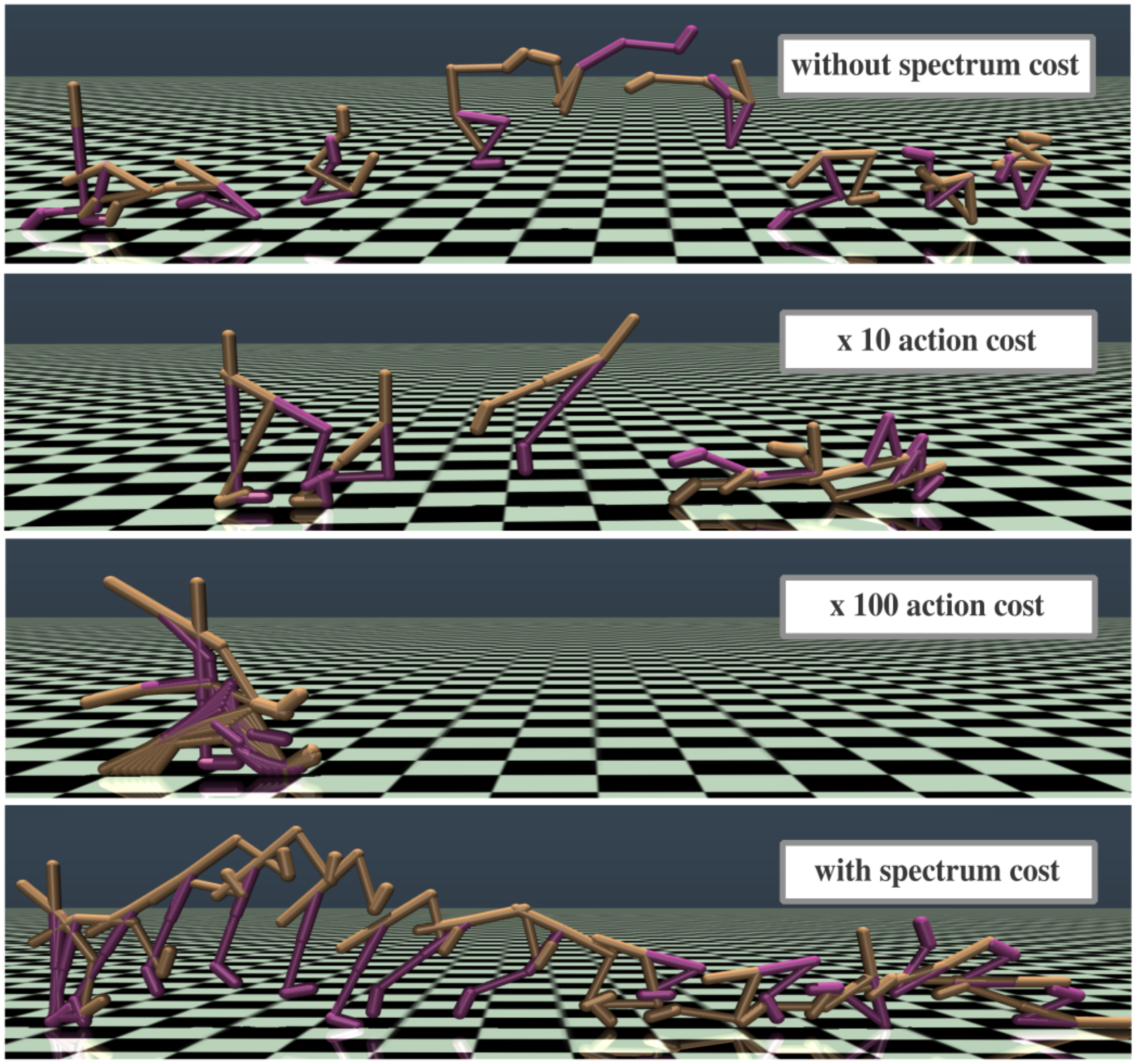


Figure A.8: Visualizations of Walker motions generated by the CEM algorithm with default action cost, 10 times more action cost, 100 times more action cost, and with the spectrum cost. The motions are of those showing median values of the spectrum cost within the seed runs. It is observed that the motion generated by the one with 10 times more action cost is smooth but uses two feet to hop, which would reduce the magnitudes of actions applied to the joints. The motion generated by the one with the spectrum cost again lifts one foot and hops; this specific visualized motion then shows a bit of rotation at the last moment.

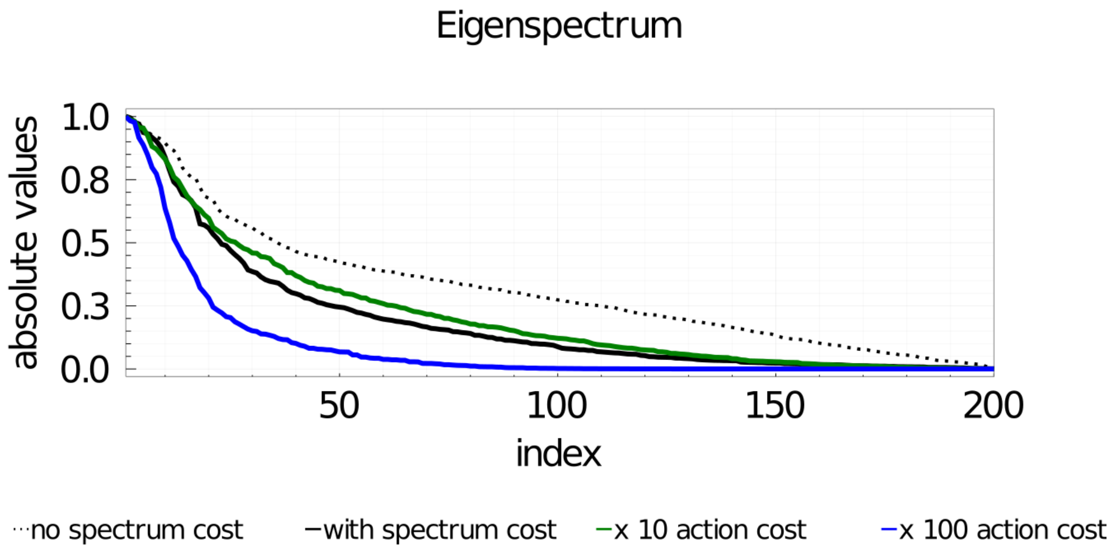


Figure A.9: Averaged eigenspectrums showing absolute values of eigenvalues for the dynamics with/without the spectrum cost and with 10 times more action cost and 100 times more action cost.

A.3 Additional numerical examples and setups for signature control

A.3.1 Details on the choice of terminal S-functions

In this section, we present the details of the choice of terminal S-functions. Other than the one used in the main text (illustrated in Figure A.10), another example of $\mathcal{TS}_m$ is given by

$$\mathcal{TS}_m(x, s, \sigma) \in \operatorname*{arg\,min}_{u \in T^m(\mathcal{X})} \ell\left(s \otimes_m S_m(\sigma) \otimes_m u\right) + \ell_{\text{reg}}(u). \tag{A.3.1}$$

If this computation is hard, one may choose $\mathcal{TS}_m(x, s, \sigma) = \mathbf{1}$; or for path tracking problem, one may choose the signature of a straight line between the endpoint of σ^* and the endpoint of σ. These three examples are shown in Figure A.11.

Terminal S-function and surrogate costs: For an application to MPC problems, we analyze the surrogate cost ℓ, regularizer ℓ_{reg}, and the terminal S-function $\mathcal{TS}$ in Algorithm 3. Suppose the problem is to track a given path with signature s^* ($m = \infty$). Fix the cost ℓ

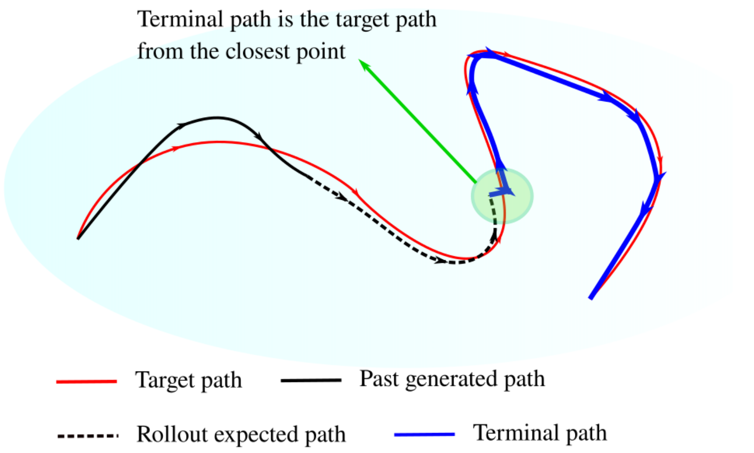


Figure A.10: Illustration of the terminal S-function used in this work.

to $\ell(s) = \|s - s^*\|^2 - w_1\|s\|^2$ and ℓ_{reg} to $\ell_{\text{reg}}(s) = w_2\|s\|^2$, where $w_1, w_2 \in \mathbb{R}_{\geq 0}$ are weights.

Here, ℓ_{reg} regularizes so that the terminal path becomes shorter, i.e., the agent prefers progressing more with accuracy sacrifice. The term $w_1\|s\|^2$ for ℓ is used to allow some deviations from the reference path. The norm $\|\cdot\|$ here is the one induced by the inner product defined in Section 2.1.7.

Fact A.3.1 (cf. [95, 42])**.** *For the signature $S(\sigma) = (1, s_1, s_2, \ldots)$ of a path σ of finite variation on $\mathcal{X}$ with the length $|\sigma| < \infty$, it follows that*

$$\|s_k\|_{\mathcal{X}^{\otimes k}} \leq \frac{|\sigma|^k}{k!}.$$

For sufficiently well-behaved *path (see [95, 42] for example), the limit exists:*

$$\lim_{k\to\infty} \||\sigma|^{-k} k! s_k\|^2_{\mathcal{X}^{\otimes k}} \leq 1.$$

If the norm is the projective norm, the limit is 1.

From this, we obtain

$$\|S(\sigma)\|^2 = \sum_{k=0}^{\infty} \|s_k\|^2_{\mathcal{X}^{\otimes k}} \leq \sum_{k=0}^{\infty} \left(\frac{|\sigma|^k}{k!}\right)^2 \leq \left(\sum_{k=0}^{\infty} \frac{|\sigma|^k}{k!}\right)^2 = e^{2|\sigma|},$$

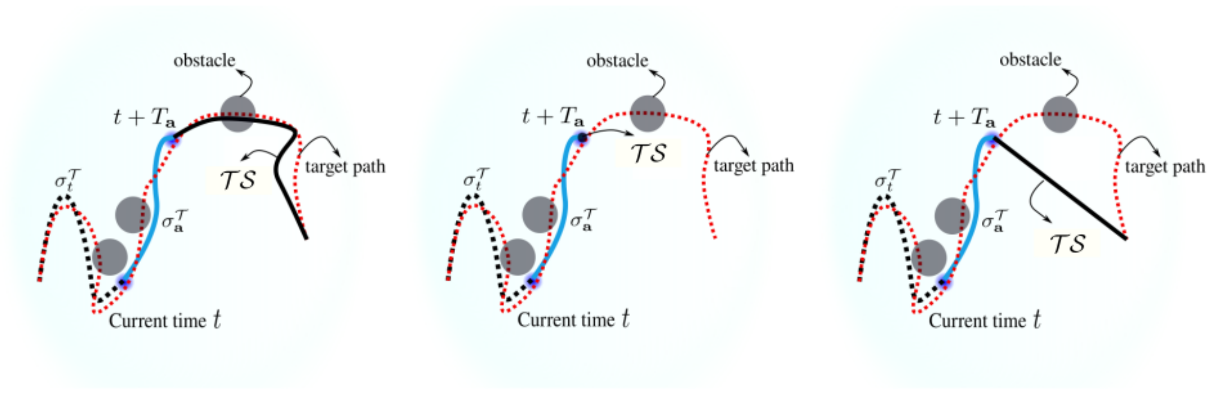


Figure A.11: Illustrations of example terminal S-functions when given target path to track. $\sigma_t^{\mathcal{T}}$ is the transformed past path whose signature is s_t. The left shows (A.3.1); the future path ignores any dynamic constraints and the optimal virtual path is computed. The middle is for the case that the signature is $\mathbf{1}$. The right is the case where straight line between the endpoint of the target path and the state at time $t+T_{\mathbf{a}}$ is the terminal path.

and for a zero length path, i.e., a point, we obtain $1 = \|S(\sigma)\|^2 = e^{2|\sigma|}$. Therefore, while one could use $(k!\|s_k\|_{\mathcal{X}^{\otimes k}})^{1/k}$ for large k as a proxy of $|\sigma|$, we simply use $\|S(\sigma)\|^2$.

We compare the following three different setups with the same surrogate cost and regularizer ($w_1 = 0,\ w_2 = 1$): (1) terminal path is the straight line between the endpoints of the rollout and the reference path, (2) terminal path is computed by nested optimization (see (A.3.1)), and (3) terminal path is given by the subpath of the reference path from the end time of the rollout. The comparisons are plotted in Figure A.12. In particular, for the reference path (linear $x = t$ or sinusoid $x = \sin(t\pi)$) over time interval $[0, 3]$, we use 20 out of 50 nodes to generate subpaths up to the fixed time 1.2. We use Adam optimizer with step size 0.1; and 300 update iterations for all but the type (2) above, which uses 30 iterations both for outer and inner optimizations.

From the figure, our example costs ℓ and ℓ_{reg} properly balance accuracy and length of the rollout subpath.

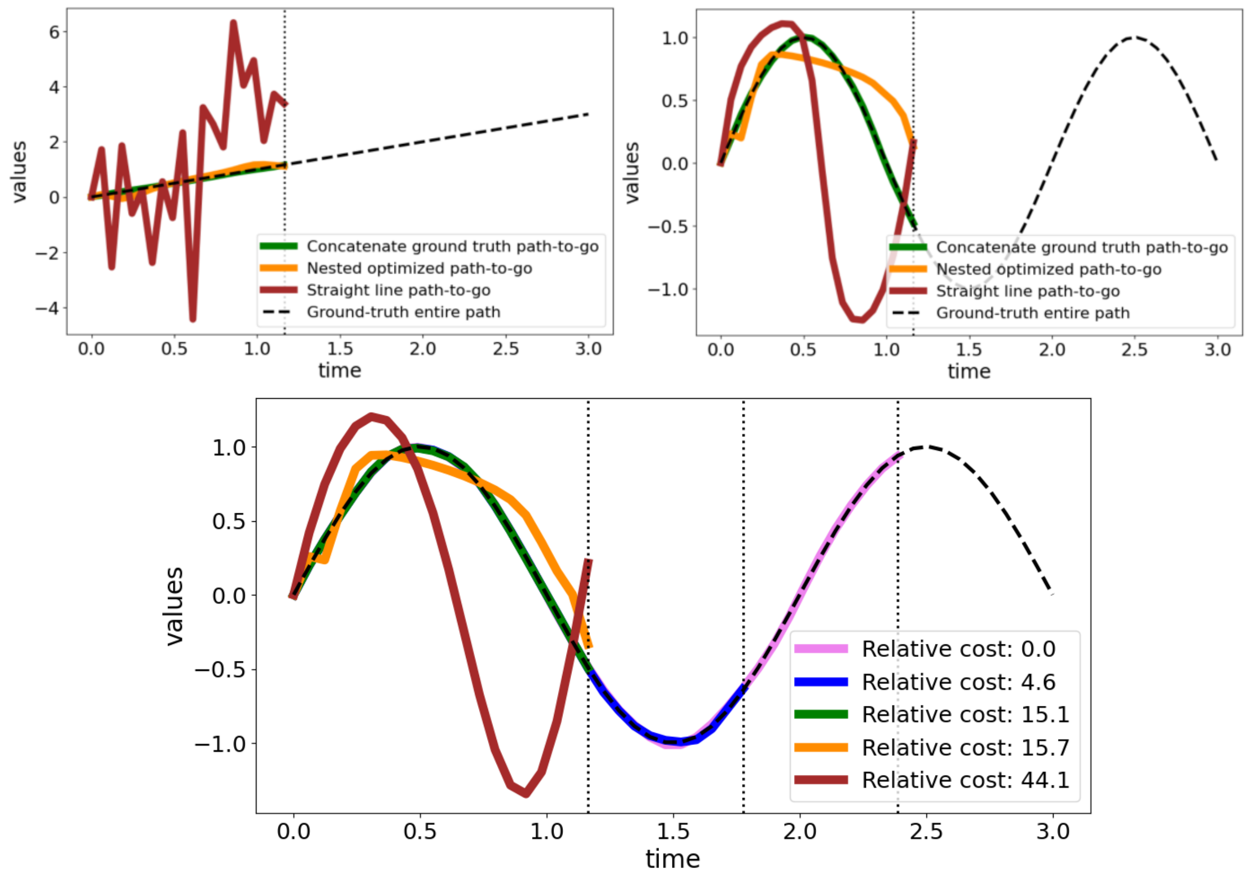


Figure A.12: Top: Green line is generated by the same approach as the one used in this work; orange one does inner optimization to obtain the optimal terminal S-function and the red line uses the straight line as the terminal path. Down: Comparison of the costs for the three subpaths and two other longer paths using the same choice of the terminal path as that in this work. As expected, longer subpath has lower score thanks to the regularizer cost.

A.3.2 Experimental setups

Here, we describe the detailed setups of each experiment and show some extra results.

Simple pointmass MPC

The parameters used for RRT* and CEM planner are shown in Table A.15.

The parameters for the signature MPC are given in Table A.16. Here, scaling of the states indicates that we multiply the path by this value and then compute the cost over the scaled path. Note we used torchdiffeq package [58, 59] of PyTorch [195] to compute rollout, and the evaluated points are of switching points of actions, and it replans when the current action repetition ends.

Table A.15: RRT* and CEM parameters.

Hyperparameter	Value	Hyperparameter	Value
max distance to the sample	10.0	goal state sample rate	0.2
safety margin to obstacle	0.0	γ to determine neighbors	1.0
CEM distance cost	quadratic	CEM obstacle penalty	1000
CEM elite number	3	CEM sample number	8
CEM iteration number	3	numpy random seed	1234

Integral control examples

We list the parameters used for signature MPCs; the execution horizon is 15.0 sec, and the reference is the signature of the linear path over the zero state along time interval from 0 to 25.0 (we extended the reference from 15.0 to 25.0 to increase stability). The control inputs are assumed to be fixed over planning horizon, and are actually executed over a planning interval. The number of evaluation points when planning is given so that the rollout path is

Table A.16: Parameters for the signature MPCs of point-mass (shared values for zero terminal S-function and the best choice ones).

Hyperparameter	Value	Hyperparameter	Value
static kernel	RBF/scale 0.5	dyadic order of PDE kernel	2
scaling of the states	0.05	update number of PyTorch	20
step size for update	0.2	number of actions N	3
weight w_1	0	regularizer weight w_2	8.0
maximum magnitude of control	1.0		

approximated by the piecewise linear interpolation of those points (e.g., for planning horizon of 1.0 sec with 5 eval points, a candidate of rollout path is evaluated evenly with 0.2 sec interval).

The signature cost is the squared Euclidean distance between the reference path signature and the generated path signature up to depth 1 or 2; the terminal path is just a straight line along the time axis, staying at the current state.

The parameters used for signature MPCs are listed in Table A.17; note the truncation depths for signatures are 1 and 2, respectively.

Path tracking with Ant

We use DiffRL package [269] Ant model.

Reference generation: We generate reference path over 2D plane with the points

$$[0.0, 0.0], [2.6, -3.9], [5.85, -1.95], [6.5, 0.0], [5.85, 1.95],$$
$$[2.6, 3.9], [0.0, 3.51], [-3.25, 0.0], [-6.5, -3.9], [-6.5, 4.55]$$

Table A.17: Parameters for the signature MPC for two-mass, spring, damper system.

Hyperparameter	Value	Hyperparameter	Value
kernel type	truncated linear	horizon for MPC	1.0 sec
number of eval points	5	update number per step	50
step size for update	0.1	planning interval	0.5 sec
number of actions	1	max horizon	25.0 sec
maximum magnitude of control	1.0		

and we obtain the splined path with 2000 nodes (skip number 1, weight for smoothness 0.5, iteration number 150 with Adam optimizer). Also, to use for the terminal path in MPC planning, we obtain rougher one with 200 nodes.

We run signature MPC (parameters are listed in Table A.18) and the simulation steps to reach the endpoint of the reference is found to be 880. Then, we run the baseline MPC; for the baseline, we generate 880 nodes for the spline (instead of 2000). Note these waypoints are evenly sampled from $t = 0$ to $t = 1$ of the obtained natural cubic spline. We also test slower version of baseline MPC with 1500 and 2500 simulation steps (i.e., number of waypoints).

Cost and reward: The baseline MPC uses time-varying waypoints by augmenting the state with time index. The time-varying instantaneous cost c_{baseline} to use is inspired by [198]:

$$c_{\text{baseline}}(x, t) = -\sum_{i=1}^{d_x} \exp\{-10(x_{(i)} - x^*_{(i)}(t))^2\}$$

for time step t, where $x^*(t)$ is the (scaled) waypoint at time t and $x_{(i)}$ is the ith dimension of (the scaled state) $x \in \mathbb{R}^{d_x}$. In addition to this cost, we add height reward for the baseline MPC:

$$r_{\text{height}}(z) = -100\text{LeakyReLU}_{0.001}(0.37 - z),$$

where $\text{LeakyReLU}_{0.001}$ is the Leaky ReLU function with negative slope 0.001 and z is the height of Ant. For signature MPC, in addition to the signature cost described in the main text, we also add the Bellman reward $10r_{\text{height}}$ (the scale 10 is multiplied to balance between signature cost and the height reward).

Experimental settings and evaluations: The parameters are listed in Table A.18. The parameters used for SAC RL are listed in Table A.19.

Since the model we use is differentiable, we use Adam optimizer to optimize rollout path by computing gradients through path. Surprisingly, with only 3 gradient steps per simulation step, it is working well; conceptually, this is similar to MPPI [262] approach where the distribution is updated once per simulation step and the computed actions are shifted and kept for the next planning. Also, we set the maximum points of the past path to obtain signatures to 50 (we skip some points when the past path contains more than 50 points).

To evaluate the accuracy, we generate 2000 nodes from the spline of the reference, and we compute the Euclidean distance from each node of the reference to the closest simulated point of the generated trajectory. The relative cumulative deviations are plotted in Figure A.13. For the same reaching time, signature MPC is significantly more accurate. When the speed is slowed, baseline MPC becomes a bit more accurate. Note our signature MPC can also tune the trade-off between accuracy and progress without knowing feasible waypoints.

Also, the performance curve of SAC RL is plotted in Figure A.15 (left) against the cumulative reward achieved by the signature MPC counterpart. We see that RL shows poor performance (refer to the discussions on difficulty of RL for path tracking problems in [198]).

Table A.18: Parameters for the signature MPC for Ant. Baseline MPC shares most of the common values except for scaling, which is 1.0 for baseline MPC.

Hyperparameter	Value	Hyperparameter	Value
static kernel	RBF/scale 0.5	dyadic order of PDE kernel	1
scaling of the states	0.2	update number of PyTorch	3
step size for update	0.1	number of actions N	64
weight w_1	0	regularizer weight w_2	3.0
maximum magnitude of control	1.0		

Table A.19: Parameters for the SAC RL for Ant path following.

Hyperparameter	Value	Hyperparameter	Value
number of steps per episode	128	initial alpha for entropy	1
step size for alpha	0.005	step size for actor	0.0005
step size for Q-function	0.0005	update coefficient to target Q-function	0.005
replay buffer size	10^6	number of actors	64
NN units for all networks	$[256, 128, 64]$	batch size	4096
activation function for NN	tanh	episode length	1000

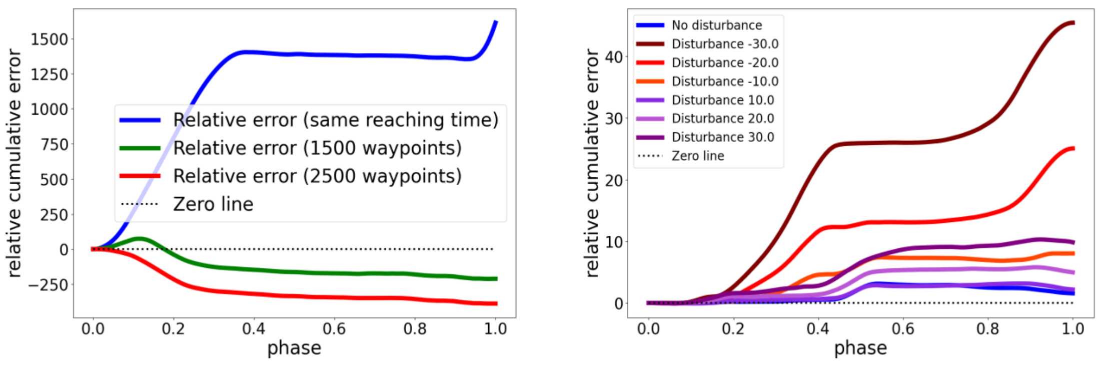


Figure A.13: For each node of the reference path, we compute the Euclidean distance from the closest simulated point of the generated trajectory. Phase is from 0 to 1, corresponding to the start and end points of the reference. Left shows the relative *cumulative* deviations along 2000 nodes of the reference for Ant. They are the errors of the baseline MPC compared to the errors of signature MPC; hence positive value shows the advantage of signature MPC. Right shows those for robotic arm experiments with different magnitudes of disturbances added to every joint.

Path tracking with Franka arm end-effector

We use DiffRL package again and a new Franka arm model is created from URDF model [232].

Model: The stiffness and damping for each joint are given by

$$\text{stiffness} : 400, 400, 400, 400, 400, 400, 400, 10^6, 10^6,$$
$$\text{damping} : 80, 80, 80, 80, 80, 80, 80, 100, 100,$$

and the initial positions of each joint are

$$1.157, -1.066, -0.155, -2.239, -1.841, 1.003, 0.469, 0.035, 0.035.$$

The action strength is 60.0 $\text{N} \cdot \text{m}$. Simulation step is 1/60 sec and the simulation substeps are 64.

Reference generation: We similarly generate reference path for the end-effector position with the points

$$[0.0, 0.0, 0.0], [0.1, -0.1, -0.1], [0.2, -0.15, -0.2], [0.18, 0.0, -0.18],$$
$$[0.12, 0.1, -0.12], [0.08, -0.1, 0.0], [0.05, -0.15, 0.1], [0.0, -0.12, 0.2],$$
$$[-0.05, -0.05, 0.25], [-0.1, 0.1, 0.15], [-0.05, 0.05, 0.08]$$

and we obtain the splined path with 1000 nodes (skip number 1, weight for smoothness 0.001, iteration number 150 with Adam optimizer). Also, to use for the terminal path in MPC planning, we obtain rougher one with 100 nodes. Similar to Ant experiments, we use 270 evenly assigned waypoints for the baseline MPC.

For the case with unknown disturbance, we use the same waypoints.

Experimental settings and evaluations: The parameters for MPCs are listed in Table A.20. The parameters used for SAC RL are listed in Table A.21.

We again set the maximum points of the past path to obtain signatures to 50, and the unknown disturbances are added to *every* joint.

The performance curve of SAC RL is plotted in Figure A.15 (right) against the cumulative reward achieved by the signature MPC counterpart. We see that the cumulative reward itself of SAC RL outperforms the signature MPC for no disturbance case for the robotic arm experiment; however it does not necessarily show better tracking accuracy along the path.

To evaluate the accuracy, we generate 1000 nodes from the spline of the reference. The relative cumulative deviations are plotted in Figure A.13. For all of the disturbance magnitude cases, signature MPC is more accurate. When the disturbance becomes larger, this difference becomes significant, showing robustness of our method.

Visually, the generated trajectories are shown in Figure A.14.

Table A.20: Parameters for the signature MPC for robotic arm. Baseline MPC shares most of the common values except for scaling, which is 1.0 for baseline MPC.

Hyperparameter	Value	Hyperparameter	Value
static kernel	RBF/scale 0.5	dyadic order of PDE kernel	1
scaling of the states	10.0	update number of PyTorch	3
step size for update	0.1	number of actions N	16
weight w_1	0.5	regularizer weight w_2	0.5
maximum magnitude of control	1.0		

Table A.21: Parameters for the SAC RL for robotic arm path following.

Hyperparameter	Value	Hyperparameter	Value
number of steps per episode	128	initial alpha for entropy	1
step size for alpha	0.005	step size for actor	0.0005
step size for Q-function	0.0005	update coefficient to target Q	0.005
replay buffer size	10^6	number of actors	64
NN units for all networks	$[256, 128, 64]$	batch size	2048
activation function for NN	tanh	episode length	300

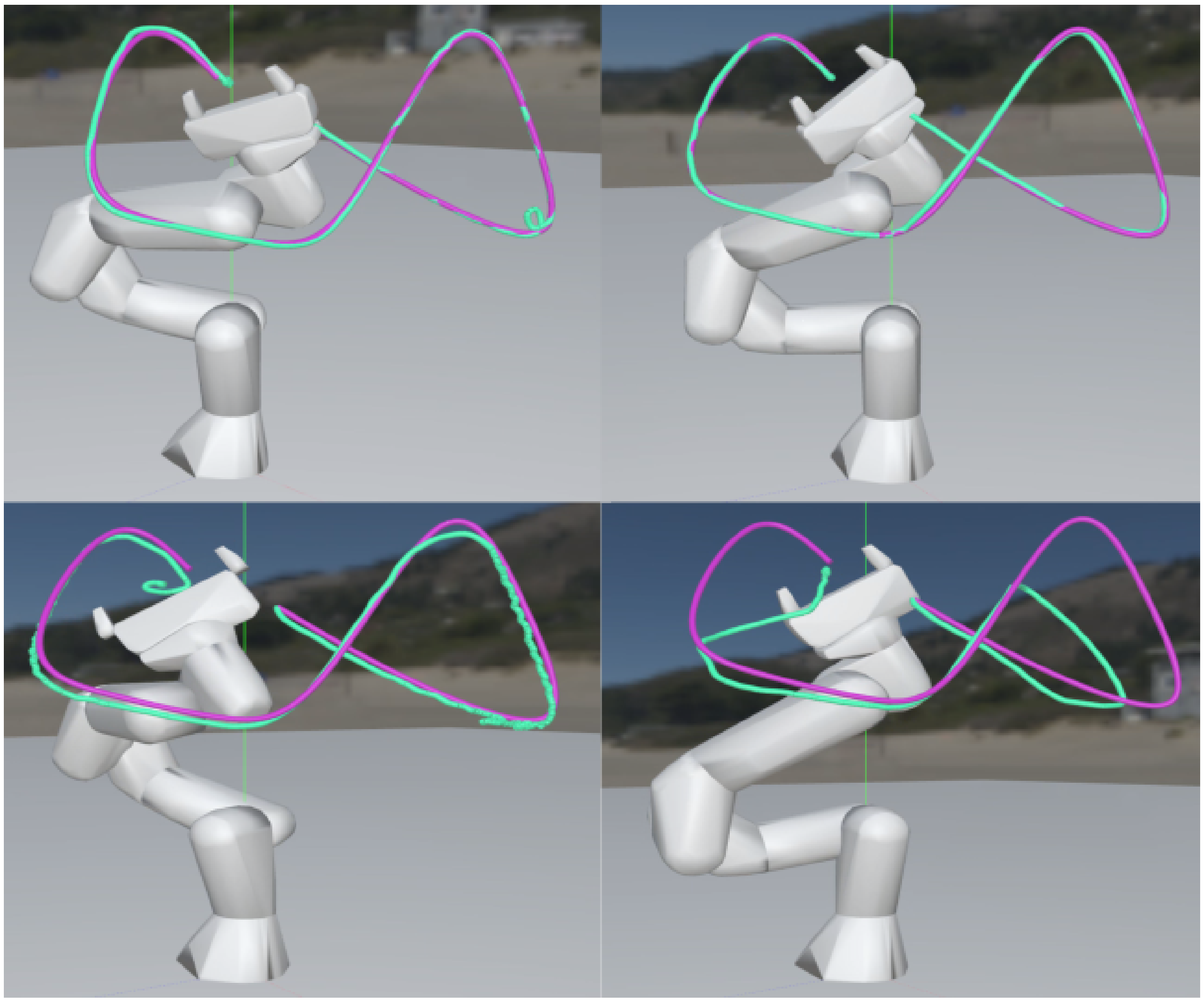
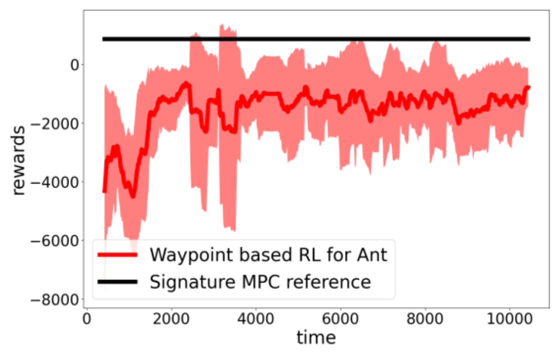


Figure A.14: Top: the left one shows signature control result for robotic arm manipulator end-effector path tracking and the right one shows the baseline MPC. They are tracking the path similarly well. Down: under disturbance -30.0. The baseline tracking accuracy is deteriorated largely while signature control is robust against disturbance. Signature MPC tries to track the first curve *stubbornly* by taking time there to retrace better. This is because the signature MPC is insensitive to waypoint designs but rather depends on the "distance" between the target path and the rollout path in the signature space.



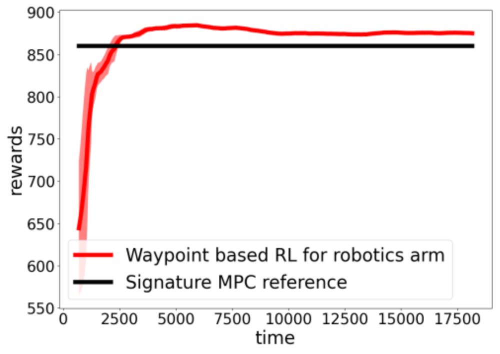


Figure A.15: Soft-actor-critic RL baseline for Ant (left) and robotic arm (right) path following experiment. The red curve shows the average cumulative rewards with standard deviation shade over five different seed runs, and the black line is for reference of the cumulative rewards achieved by signature control. The reward is the same (negative cost) as that used in the baseline MPC, and the state is augmented with the time step (maximum time step is 1000 and 300 given that the goal-reaching time of signature control is 880 and 270 steps, respectively). For the ant case, RL did not achieve comparable performance in terms of rewards. For the robotic arm case under no disturbance, the RL outperforms signature control slightly.

A.4 Simulations for dynamic structure estimation

This section presents the simulation setups and results for Chapter 6.

A.4.1 Period estimation: LifeGame

The hyperparameters of LifeGame environment and the algorithm are summarized in Table A.22. Note $\mu = 0$ because it is a periodic transition. Here, we used 12×12 blocks of cells and we focused on the five blocks surrounded by the red rectangle in Figure A.16. The transition rule is given by

1. If the cell is alive and two or three of its surrounding eight cells are alive, then the cell remains alive.

Table A.22: Hyperparameters used for period estimation of LifeGame.

LifeGame hyperparameter	Value	Algorithm hyperparameter	Value
height	12	accuracy for estimation ρ	0.98
width	12	failure probability bound δ	0.2
observed dimension	5	maximum possible period $L_{\max}$	10
observation noise proxy R	0.3		
ball radius B	$\sqrt{5}$		

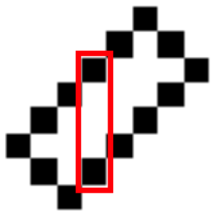


Figure A.16: The area we focus on for the cellular automata experiment.

2. If the cell is alive and more than three or less than two of its surrounding eight cells are alive, then the cell dies.

3. If the cell is dead and exactly three of its surrounding eight cells are alive, then the cell is revived.

A.4.2 Period estimation: Simple μ-nearly periodic system

The dynamical system

$$r_{t+1} = \mu\left(\alpha\frac{r_t - 1}{\mu} - \lceil\alpha\frac{r_t - 1}{\mu}\rceil\right) + 1, \qquad \theta_{t+1} = \theta_t + \frac{2\pi}{L},$$

is μ-nearly periodic. See Figure A.17 for the illustrations when $\mu = 0.2,\ L = 5,\ \alpha = \pi$. It is observed that there are five *clusters*. We mention that this system is not exactly periodic.

The hyperparameters of this system and the algorithm are summarized in Table A.23.

Table A.23: Hyperparameters used for μ-nearly periodic system.

System hyperparameter	Value	Algorithm hyperparameter	Value
dimension	2	accuracy for estimation ρ	0.3
μ	0.001	failure probability bound δ	0.2
α	π	maximum possible nearly period $L_{\max}$	8
observation noise proxy R	0.3		
ball radius B	2		

Table A.24: Hyperparameters used for eigenvalue estimation.

Hyperparameter	Value	Hyperparameter	Value
κ	6	a nearly period L	24
Δ	0.1	failure probability bound δ	0.2
dimension	5	observation noise proxy R	0.3
ball radius B	1		

A.4.3 Eigenvalue estimation

We used the matrix M given by

$$M := \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 \end{bmatrix}.$$

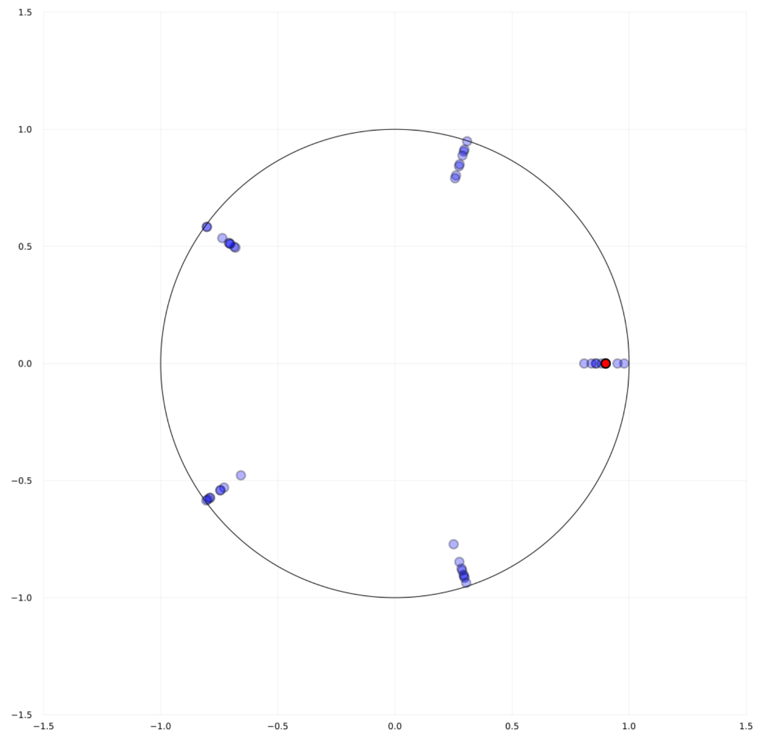


Figure A.17: An example of μ-nearly periodic system.

The first 4×4 block matrix is for permutation. After N steps, it is expected that the last dimension shrinks so that the system becomes nearly periodic. It follows that $4! = 24$ is a multiple of the length L. Eigenvalues of M^5 are given by $1.000,\ 1.000,\ -0.500 - 0.866i,\ -0.500 + 0.866i,\ 0.168$, and the $(\theta_0, 5)$-distinct eigenvalues are $1.000,\ -0.500 - 0.866i,\ -0.500 + 0.866i$.

The hyperparameters of the environment and the algorithm are summarized in Table A.24. Note we don't necessarily need κ, B, and Δ to run the algorithm as long as the effective sample size is sufficiently large; we used the values (satisfying the conditions) in Table A.24 for simplicity.

A.5 Array of experimental analyses of deep RL with different critic loss

We first define the RL environments studied in Chapter 7.

A.5.1 Toy MDPs

Our representative MDP classes are proven to show behaviors categorized as one of the behavior classes. Therefore, for our numerical experiment of toy MDPs, we construct MDPs that may not be necessarily covered by them, and we update the Q estimate every time step instead of every episode. Our toy MDPs, namely MDP 1, MDP 2, MDP 3, MDP4 and MDP 5, are depicted in Figure A.18.

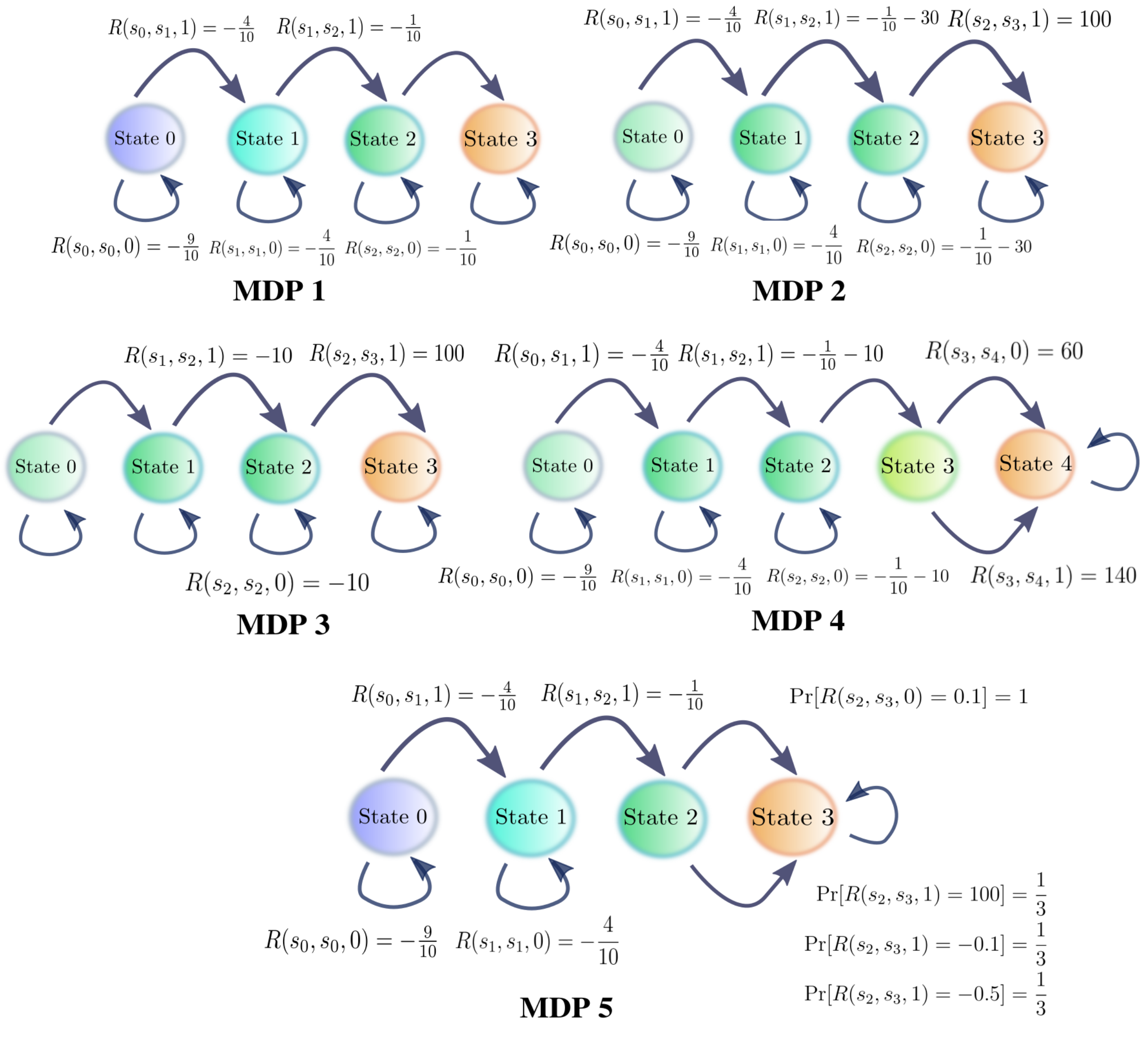
MDP 1: This MDP has *smooth* or *guiding* reward which tells the learner to reach State 3.

MDP 2: This MDP has additional large penalty for transitioning to State 2 and large reward for transitioning to State 3.

MDP 3: There are no *guiding* reward and is a large negative penalty for transitioning to State 2.

MDP 4: For this MDP, there exist two actions for transitioning to the *goal* State and the respective rewards are both positive but with some difference.

MDP 5: For this MDP, the reward is probabilistic which is designed to separate mean and median estimates of the value (refer to Case study 1).



$R(s_0, s_1, 1) = -\frac{4}{10}$
$R(s_1, s_2, 1) = -\frac{1}{10}$
State 0
State 1
State 2
State 3
$R(s_0, s_0, 0) = -\frac{9}{10}$
$R(s_1, s_1, 0) = -\frac{4}{10}$
$R(s_2, s_2, 0) = -\frac{1}{10}$
MDP 1
$R(s_0, s_1, 1) = -\frac{4}{10}$
$R(s_1, s_2, 1) = -\frac{1}{10} - 30$
$R(s_2, s_3, 1) = 100$
State 0
State 1
State 2
State 3
$R(s_0, s_0, 0) = -\frac{9}{10}$
$R(s_1, s_1, 0) = -\frac{4}{10}$
$R(s_2, s_2, 0) = -\frac{1}{10} - 30$
MDP 2
$R(s_1, s_2, 1) = -10$
$R(s_2, s_3, 1) = 100$
State 0
State 1
State 2
State 3
$R(s_2, s_2, 0) = -10$
MDP 3
$R(s_0, s_1, 1) = -\frac{4}{10}$
$R(s_1, s_2, 1) = -\frac{1}{10} - 10$
$R(s_3, s_4, 0) = 60$
State 0
State 1
State 2
State 3
State 4
$R(s_0, s_0, 0) = -\frac{9}{10}$
$R(s_1, s_1, 0) = -\frac{4}{10}$
$R(s_2, s_2, 0) = -\frac{1}{10} - 10$
$R(s_3, s_4, 1) = 140$
MDP 4
$R(s_0, s_1, 1) = -\frac{4}{10}$
$R(s_1, s_2, 1) = -\frac{1}{10}$
$\Pr[R(s_2, s_3, 0) = 0.1] = 1$
State 0
State 1
State 2
State 3
$\Pr[R(s_2, s_3, 1) = 100] = \frac{1}{3}$
$\Pr[R(s_2, s_3, 1) = -0.1] = \frac{1}{3}$
$\Pr[R(s_2, s_3, 1) = -0.5] = \frac{1}{3}$
$R(s_0, s_0, 0) = -\frac{9}{10}$
$R(s_1, s_1, 0) = -\frac{4}{10}$
MDP 5

Figure A.18: Illustrations of MDPs 1 to 5; the zero rewards are not depicted.

A.5.2 1D continuous environments

We created 13 1D environments. Note for this set of environments, we will use deep RL algorithms (SAC with different loss functions); the complexity is hence high even with the one-dimensional action and observation space. The observation (same as state in our examples) $s \in \mathcal{S} := [-5, 5] \cup \mathcal{S}_e$ evolves with the action $a \in \mathcal{A} := [-1, 1]$ by

$$M\left(P_{[-5,5]}[s + 0.1a + \epsilon], a\right),$$

where $P_{[-5,5]} : \mathbb{R} \to [-5, 5]$ projects back the state onto $[-5, 5]$, $M : [-5, 5] \times [-1, 1] \to [-5, 5] \cup \mathcal{S}_e$ is a map specific to each environment, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with an environment-specific σ.

Representative environments

We first present the four representative 1D environments showing each one of the behavior classes.

1D representative env. for Class 1: $M(s) = s$ for all s, $\sigma = 0$, and the reward is defined by

$$\forall s, s' \in \mathcal{S},\ \forall a \in \mathcal{A} :\ R(s, s', a) = -(0.001a^2 + 0.03s^2).$$

It is a simple system with smooth reward function.

1D representative env. for Class 2: M is defined by

$$M(s, a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.3, 0.3] \\ s & \text{otherwise} \end{cases},$$

where $\texttt{done} \in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s, s', a) = \begin{cases} -(0.001a^2 + 0.03s^2) + 200 & \text{if } s' \in \texttt{done} \\ -(0.001a^2 + 0.03s^2) & \text{otherwise} \end{cases}.$$

While it is similar to the Class 1 representative, it has a large reward around the center to exit the episode, which is expected to cause instability for SAC.

1D representative env. for Class 3: M is defined by

$$M(s,a) = \begin{cases} 2.5 & \text{if } s \in [-3.8, -3.5) \\ 0.8 & \text{if } s \in [-3.4, -3.2) \\ \texttt{bad1} & \text{if } s \in [-1.7, -1.4) \\ \texttt{good1} & \text{if } s \in [-0.2, 0.2] \\ \texttt{bad2} & \text{if } s \in [1.4, 1.7) \\ -0.8 & \text{if } s \in [3.2, 3.4) \\ -2.5 & \text{if } s \in [3.5, 3.8) \\ s & \text{otherwise} \end{cases},$$

where `bad1`, `bad2`, `good1` $\in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -(0.001a^2 + 0.03s^2) - 50 & \text{if } s' = \texttt{bad1} \\ -(0.001a^2 + 0.03s^2) + 100 & \text{if } s' = \texttt{good1} \\ -(0.001a^2 + 0.03s^2) - 50 & \text{if } s' = \texttt{bad2} \\ -(0.001a^2 + 0.03s^2) & \text{otherwise} \end{cases}.$$

There exist many *transports* and a few large penalties and reward without smooth guiding reward.

1D representative env. for Class 4: M is defined by

$$M(s,a) = \begin{cases} 1 & \text{if } s \in [-2.5, -2) \\ \texttt{bad1} & \text{if } s \in [-2.0, -1.5) \\ \texttt{good1} & \text{if } s \in [-0.1, 0.1] \\ \texttt{bad2} & \text{if } s \in [1.5, 2] \\ -1 & \text{if } s \in (2, 2.5] \\ s & \text{otherwise} \end{cases},$$

where `bad1`, `bad2`, `good1` $\in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -(0.1a^2 + 0.03s^2) - 100 & \text{if } s' = \texttt{bad1} \\ -(0.1a^2 + 0.03s^2) + 100 & \text{if } s' = \texttt{good1} \\ -(0.1a^2 + 0.03s^2) - 100 & \text{if } s' = \texttt{bad2} \\ -(0.1a^2 + 0.03s^2) & \text{otherwise} \end{cases}.$$

It is similar to the Class 3 representative while the highest reward can only be obtained when the agent tunes to the narrow interval of $[-0.1, 0.1]$.

1D representative env. for Case study 2: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.1, 0.1] \\ s & \text{otherwise} \end{cases},$$

where `done` $\in \mathcal{S}_e$, and $\sigma = 0.05$. The reward is defined by

$$\forall s, s' \in \mathcal{S},\ \forall a \in \mathcal{A}:\ R(s,s',a) = -(0.1a^2 + s^2).$$

It is mostly similar to that of the Class 1 representative while having much larger magnitudes of reward. The `done` state is reached from a narrow interval, which, as a result, may possess some *flavor* of the Class 2 representative.

Other environments

Here, we present other environments with similar properties to those in the representative environments.

1D env. #6: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.3, 0.3] \\ s & \text{otherwise} \end{cases},$$

where $\texttt{done} \in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$\forall s, s' \in \mathcal{S},\ \forall a \in \mathcal{A}:\ R(s, s', a) = -(0.001a^2 + 0.03s^2).$$

It is similar to the Class 1 representative but with `done` state which can be easily reached.

1D env. #7: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.3, 0.3] \\ s & \text{otherwise} \end{cases},$$

where $\texttt{done} \in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s, s', a) = \begin{cases} 1 & \text{if } s' < 0 \wedge a > 0 \\ -1 & \text{if } s' < 0 \wedge a < 0 \\ 1 & \text{if } s' > 0 \wedge a < 0 \\ -1 & \text{if } s' > 0 \wedge a > 0 \\ 200 & \text{if } s' \in \texttt{done} \end{cases}.$$

The action cost in this environment naturally guides the agent to reach the center having large reward, which resembles the Class 2 representative.

1D env. #8: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.3, 0.3] \\ \texttt{bad1} & \text{if } s \leq -4.9 \\ \texttt{bad2} & \text{if } s \geq 4.9 \\ s & \text{otherwise} \end{cases},$$

where `done`, `bad1`, `bad2` $\in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -(0.001a^2 + 0.03s^2) + 100 & \text{if } s' = \texttt{bad1} \\ -(0.001a^2 + 0.03s^2) + 100 & \text{if } s' = \texttt{bad2} \\ -(0.001a^2 + 0.03s^2) - 100 & \text{if } s' = \texttt{done} \wedge |a| \geq 0.8 \\ -(0.001a^2 + 0.03s^2) + 100 & \text{if } s' = \texttt{done} \wedge |a| < 0.8 \\ -(0.001a^2 + 0.03s^2) & \text{otherwise} \end{cases} \cdot$$

There is a smooth guiding reward while some large penalties and rewards are given at the exit states. This resembles the Class 2 representative but it has also some *flavor* from the Class 4 representative because of the necessity of tuning of action magnitude when entering `done` state.

1D env. #9: M is defined by

$$M(s,a) = \begin{cases} -3 & \text{if } s \in [-3.8, -3.5) \\ \texttt{bad1} & \text{if } s \in [-3.4, -3.2) \\ -1.5 & \text{if } s \in [-2.5, -2.2) \\ \texttt{bad2} & \text{if } s \in [-2.1, -1.9) \\ \texttt{bad3} & \text{if } s \in [1.9, 2.1) \\ 1.5 & \text{if } s \in [2.2, 2.5) \\ \texttt{bad4} & \text{if } s \in [3.2, 3.4) \\ 3 & \text{if } s \in [3.5, 3.8) \\ s & \text{otherwise} \end{cases},$$

where $\texttt{bad1}$, $\texttt{bad2}$, $\texttt{bad3}$, $\texttt{bad4} \in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -0.001a^2 - 50 & \text{if } s' = \texttt{bad1} \\ -0.001a^2 - 50 & \text{if } s' = \texttt{bad2} \\ -0.001a^2 - 50 & \text{if } s' = \texttt{bad3} \\ -0.001a^2 - 50 & \text{if } s' = \texttt{bad4} \\ -0.001a^2 + 50 & \text{if } s + 0.1a \in [-3.8, -3.5) \\ -0.001a^2 + 50 & \text{if } s + 0.1a \in [-2.5, -2.2) \\ -0.001a^2 + 50 & \text{if } s + 0.1a \in [2.2, 2.5) \\ -0.001a^2 + 50 & \text{if } s + 0.1a \in [3.5, 3.8) \\ -(0.001a^2 + 0.5s^2) + 3 & \text{if } s + 0.1a \in [-1.5, 1.5] \\ -0.001a^2 & \text{otherwise} \end{cases}.$$

While there are no significant guiding reward, showing some flavors of the Class 3 representative, it has some *guided route* to the center after the agent reaches to $s = -2.5$ or

$s = 2.5$ with some additional large penalty and reward, which is of the nature of the Class 2 representative.

1D env. #10: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.2, 0.2) \\ \texttt{bad1} & \text{if } s < 0 \wedge a < -0.3 \\ \texttt{bad2} & \text{if } s > 0 \wedge a > 0.3 \\ s & \text{otherwise} \end{cases},$$

where `done`, `bad1`, `bad2` $\in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -0.001a^2 - 50 & \text{if } s' = \texttt{bad1} \\ -0.001a^2 - 50 & \text{if } s' = \texttt{bad2} \\ -0.001a^2 + 200 & \text{if } s' \in \texttt{done} \\ -0.001a^2 & \text{otherwise} \end{cases}.$$

There is no significant guiding reward, but the transition to `bad` states somehow indicate which direction the agent should go. As such it has the Class 3 representative with a bit of the flavor of the Class 2 representative.

1D env. #11: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.1, 0.1] \\ s & \text{otherwise} \end{cases},$$

where `done` $\in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -(0.001a^2 + 0.03s^2) + 200 & \text{if } s' \in \texttt{done} \\ -(0.001a^2 + 0.03s^2) & \text{otherwise} \end{cases}.$$

While the setup is very similar to that of the Class 2 representative, the final `done` state is very narrow, requiring *fine-tune* of the control; therefore, it has the property of the Class 4 representative.

1D env. #12: M is defined by

$$M(s,a) = \begin{cases} \texttt{done} & \text{if } s \in [-0.3, 0.3] \wedge |a| > 0.95 \\ s & \text{otherwise} \end{cases},$$

where $\texttt{done} \in \mathcal{S}_e$, and $\sigma = 0$. The reward is defined by

$$R(s,s',a) = \begin{cases} -(0.001a^2 + 0.03s^2) + 200 & \text{if } s' \in \texttt{done} \\ -(0.001a^2 + 0.03s^2) & \text{otherwise} \end{cases}.$$

Similar to the 1D env. #11, it requires *fine-tune* to obtain large reward; as such, it has the property of the Class 4 representative.

1D env. #13: $M(s) = s$ for all s, $\sigma = 0$, and the reward is defined by

$$\forall s, s' \in \mathcal{S},\ \forall a \in \mathcal{A}:\ R(s,s',a) = -(0.1a^2 + s^2).$$

It is a simple system with smooth but large magnitude of reward, having the property of the Case study 2.

A.5.3 Experimental setups

Next, we present some parameter settings and explain the details of experiments conducted in Chapter 7

Common setups

The common hyperparameter setups are given in Table A.25.

Specific setups

The specific parameters used for the experiments, namely, toy MDPs, 1D environments, Lunar Lander, Bipedal Walker, and Hopper are given in Table A.26.

Table A.25: Common hyperparameters for the SAC algorithms.

Hyperparameter	Value	Hyperparameter	Value
discount factor	0.99	initial alpha for entropy	1
step size for critic	0.0003	step size for actor	0.0003
target update rate	0.005	activation function for NN	`tanh`
hidden units	$[256, 256]$	optimizer	Adam
# of QRSAC heads	32	window size for stability measure	32

For Lunar Lander, Bipedal Walker, and Hopper, we use OpenAI Gym [46] (for Lunar Lander environment, we introduce a feature to the observation vector that indicates the duration of agent being in a landing condition (i.e., zero velocities) to make it Markovian). We mention that Hopper uses MuJoCo [244] as an internal simulator.

Table A.26: Specific parameters for each environment considered in analysis of deep RL algorithms.

Toy MDPs			
horizon H	10	epsilon ϵ	0.1
gradient step size	0.05	# of independent runs	300
1D environment RL runs			
maximum time step	150	replay buffer capacity	20000
batch size	32	trainer steps per epoch	32
evaluation frequency (model clock)	1	# of independent runs	30
1D environment bootstrap and supervised learning runs			
maximum time step	150	replay buffer capacity	20000
batch size	32	trainer steps per epoch	32
evaluation frequency (model clock)	30	# of independent runs	30
Lunar Lander RL runs			
maximum time step	1000	replay buffer capacity	20000
batch size	32	trainer steps per epoch	200
evaluation frequency (model clock)	50	# of independent runs	30
Lunar Lander bootstrap and supervised learning runs			
maximum time step	1000	replay buffer capacity	20000
batch size	32	trainer steps per epoch	200
evaluation frequency (model clock)	20	# of independent runs	30
Bipedal Walker RL runs			
maximum time step	1000	replay buffer capacity	300000
batch size	256	trainer steps per epoch	64
evaluation frequency (model clock)	50	# of independent runs	30
Bipedal Walker bootstrap learning runs			
maximum time step	1000	replay buffer capacity	300000
batch size	256	trainer steps per epoch	200
evaluation frequency (model clock)	20	# of independent runs	30
Bipedal Walker supervised learning runs			
maximum time step	1000	replay buffer capacity	300000
batch size	256	trainer steps per epoch	200
evaluation frequency (model clock)	10	# of independent runs	30
Hopper RL runs			
maximum time step	1000	replay buffer capacity	1000000
batch size	256	trainer steps per epoch	200
evaluation frequency (model clock)	200	# of independent runs	30

Experimental details

RL: For toy MDPs, we use Algorithm 6 but update critic every time step (i.e., $\hat{Q}_{\text{copy}}$ is replaced by $\hat{Q}$), and use the MSE, Huber and quantile-Huber loss (which is similar to Quantile Regression Q learning in [72]).

For SAC algorithms (with different critic loss), we use the policy network that outputs the mean and diagonal log standard deviation of a squashed Gaussian distribution where the output is constrained within $(-1, 1)$. When exploration, randomly sampled vector from the Gaussian distribution is fed to `tanh` function to obtain the action. The temperature parameter of SAC algorithms is automatically tuned with the target entropy $-\dim(\mathcal{A})$. Throughout the experiments, the trained policy is evaluated by outputting the mean vector and by using 10 episodes. All runs are trained in a distributed manner with separated processes of a trainer and rollout worker that collects experiences, asynchronously. Rollout worker uses single CPU (2 GB memory) while trainer worker uses 7.7 CPU cores (8 GB memory) under virtual machine. We use Reverb [54] for experience replay.

To identify the properties causing degradation of QRSAC under environments with reward of large magnitudes, we consider *multi-headed* SAC-Huber to mimic some properties of QRSAC in the 1D representative environment for Case study 2. Figure A.19 illustrates (left) quantile regression and (right) multi-headed regression. Similar to the quantile case, multi-headed SAC-Huber uses different loss for each *head*; each head is optimized to estimate the average of target critic heads plus some distinct value assigned to each head while the mean of all heads stays the same. In particular, we assign very distinct value (i.e., -1000 to 1000) to mimic the scenario for QRSAC showing degradation of performance. Moreover, we also test QRSAC with 4 and 256 heads.

On the other hand, we test an effect of multi-headed network for some selected environments. Especially, we consider multi-headed SAC where in this case each head uses the same loss, and see if it improves performance in Class 2 and Class 4 representatives.

For the 1D environment experiments, we also conduct “Two Model experiments” as well;

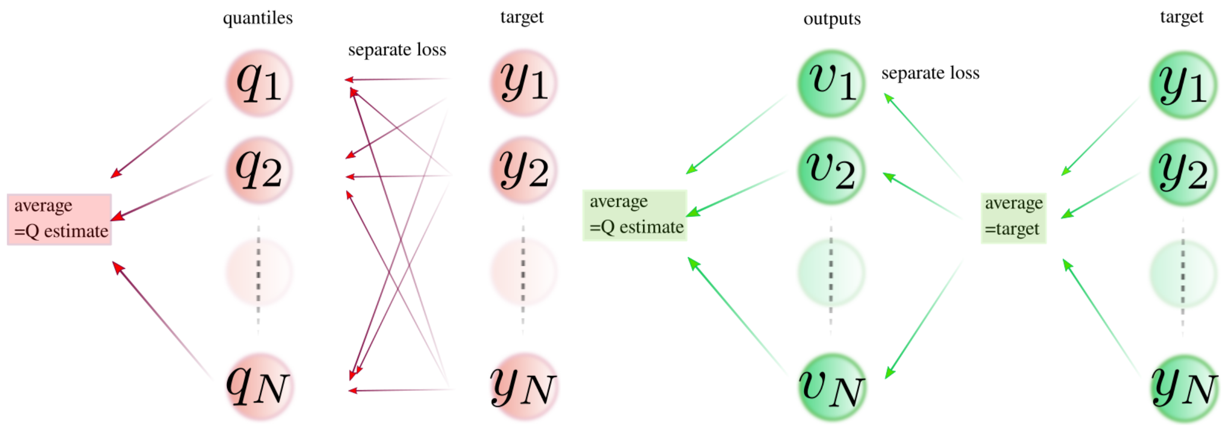


Figure A.19: An illustration of (Left) quantile regression and (Right) multi-headed regression.

where we maintain SAC and QRSAC critics and policies and data are collected by a randomly chosen policy from those two while evaluation is done for each one. For Lunar Lander, we conduct similar experiment but the data are collected by one of the policies of SAC and QRSAC.

Bootstrap and supervised learning: To eliminate the effect of critic and policy interaction, we use the fixed policy and train the action-value function by bootstrap (i.e., same as RL but without policy update) or by supervised learning.

For 1D environments, we test two cases.

1. The Class 2 representative environment with the policy and critic of SAC-Huber at the later stage of learning, which is the highest reward achieving algorithm out of three.

2. The Class 4 representative environment with the policy and critic of SAC at the later stage of learning, which is the highest reward achieving algorithm out of three.

The target observation is deterministic for 1D environments.

For Lunar Lander and Bipedal Walker, we use a policy and critic estimates of one QRSAC run at later stage of learning as the data generating policy and as the target critic. For the bootstrap learning in particular, we sample reward by the distribution:

$$\hat{Z}^{\pi}(s,a) - \gamma \hat{Z}^{\pi}(s',a'),$$

where the policy π and the return estimate $\hat{Z}^{\pi}$ are of QRSAC, s' is the observed next state and a' is sampled from $\pi(s')$. In the supervised learning, the observation is simply drawn from the fixed estimated return.

A.5.4 Additional experimental results

RL results for other 1D environments

1D env. #6: The reward curve shows mostly the Class 1 behavior (slight degradation of SAC) and the stability and smoothness are similar to the Class 1 representative (see Figure A.20 (top)).

1D env. #7: For this environment, reward curve shows interesting behaviors where SAC-Huber and QRSAC are better than SAC while they decrease the performance at the later stage to match that of SAC (see Figure A.20 (middle)). SAC shows unstable critic growth at the early stage as expected but other two show some notable instability at later stages. Furthermore, SAC consistently shows smoother critic surface. While we do not have clear reasoning of these behaviors, it might be the case that they are partially due to the *guiding but nonsmooth reward*. Nevertheless, instability of critic growth seems correlated to the degradation of performance in this environment.

1D env. #8: The reward curve shows mostly the Class 2 behavior (slight degradation of QRSAC) while SAC is showing a tiny advantage at the later stage which resembles Class 4 behavior (see Figure A.20 (down)). SAC indeed shows instability of critic growth and its nonsmooth surface. For this, we do not have clear reasoning about the degradation of QRSAC compared to SAC-Huber.

1D env. #9: Although there exists some fluctuation of reward curves compared to those showing the representative Class 1 behaviors, it is actually similar to that of the Class 1 representative but with clearer separation of stability measure (see Figure A.21 (first row)). As we discussed in Appendix A.5.2, we created this environment to have some flavors of the representative environments of Class 3 and Class 2 at the same time. While we are unable to identify the critical cause for the behavior, it maybe the case that those two conflicting properties cancel out each other.

1D env. #10: This environment shows one of the most difficult-to-analyze behaviors (see Figure A.21 (second row)). As we discussed in Appendix A.5.2, it may be expected to show similar behavior to 1D env. #9 as it has conflicting flavors of both of the Class 3 and Class 2 representatives. When looking at SAC and SAC-Huber, they show Class 3 behavior but QRSAC shows similar overall performance to SAC. While we do not have clear reasoning for the behavior in this environment, we see that some of the conflicting properties with some complication of transitions and rewards may show hard-to-predict behaviors.

1D env. #11: This environment shows somewhat expected behaviors which is of Class 2 with Class 4 flavor as we discussed in Appendix A.5.2 (see Figure A.21 (third row)). Stability and smoothness measures are seemingly consistent too.

1D env. #12: The behaviors of this environment are also not straightforward to analyze; however, we see Class 3 and Class 4 flavors together (see Figure A.21 (last row)). Conceptually the environment is created to have similar property to the Class 4 representative, and the condition for obtaining large reward is very strict which might lead to the case where existence of guiding reward is not helping much to achieve this condition. In such a case, Class 3 behavior may emerge.

1D env. #13: Figure 7.13 (down) shows the behavior of env. #13, and it is similar to that of the Case study 2 representative.

On Case study 2 and number of heads

The result is shown in Figure A.22 (left); which shows

1. Multi-headed SAC-Huber with distinct valued heads shows large degradation of performance as well

2. Fewer number of heads for QRSAC shows better performance

While it stays hypothetical, multi-headed critic network with very distinct values assigned to each head makes it harder to learn the critic surface which would be because of the conflict of gradient within the network.

On the other hand, the comparison of SAC and multi-headed SAC in the Class 2 and Class 4 representatives is shown in Figure A.23, implying having multi-heads does not improve the performance nor stability of critic growth significantly at least in this environment.

Two model example for 1D environments

The result for the two model experiment for the Class 3 representative 1D environment is plotted in Figure A.22 (right). It is observed that the QRSAC policy outperforms that of SAC under two model scenario, and QRSAC policy reaches the performance comparable to that of (single model) SAC. In this environment, we see that QRSAC can stably improve its performance without a *guiding* reward as long as the data covering *important* states are enriched. This observation may lead to better algorithm design in the future.

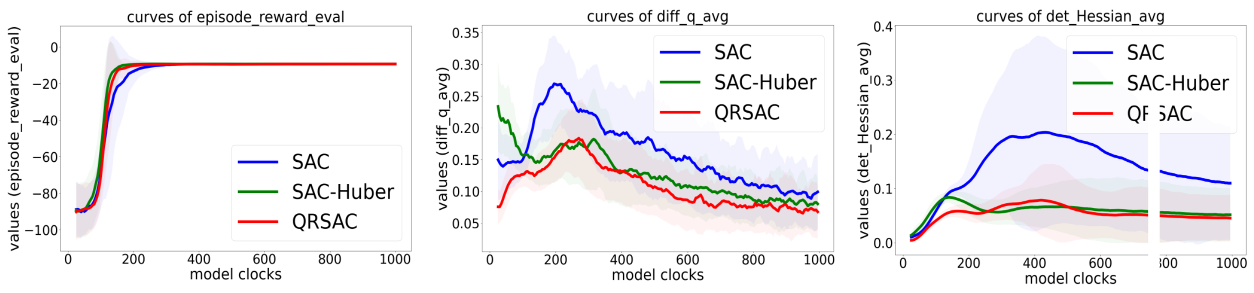
Two model example for Lunar Lander

The result for the two model experiment for Lunar Lander is shown in Table A.27, comparing the main policy (SAC policy for SAC and QRSAC policy for QRSAC) and side policy

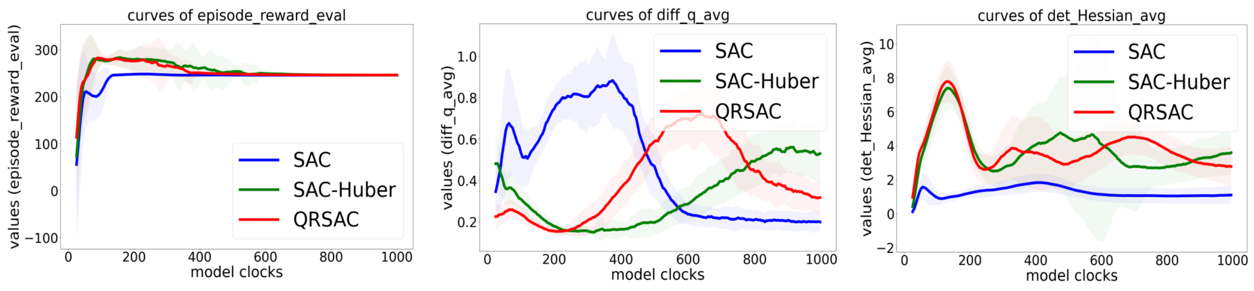
Table A.27: Lunar Lander two model experiments: cumulative discounted rewards for main and side policies (averaged over 10 runs).

Main algorithm	Main policy value	Side policy value
SAC	63.8	44.3
QRSAC	73.6	27.1

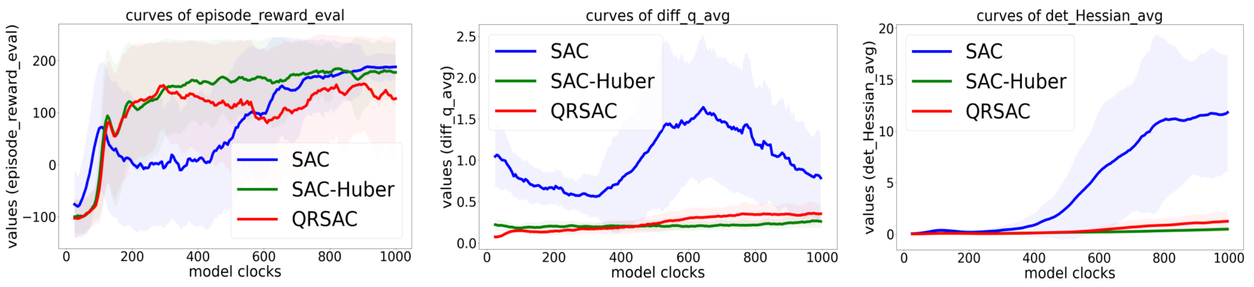
(QRSAC policy for SAC and SAC policy for QRSAC) performances. It is observed that the side policy is inferior in both cases; in this example we only used the main policy for data collection, so it might have had the data distribution inconsistent to the side policy, leading to degraded performance. As such, we observe at least in this environment that the interaction of critic and policy over epochs is critical for the performance separation.



1D env. # 6

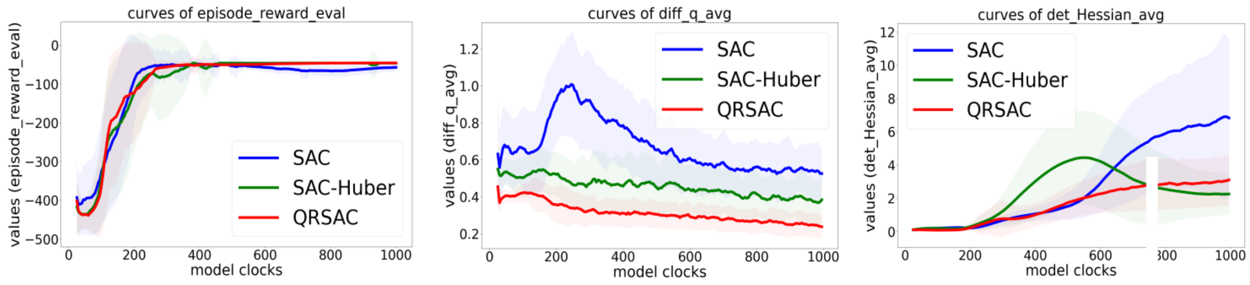


1D env. # 7

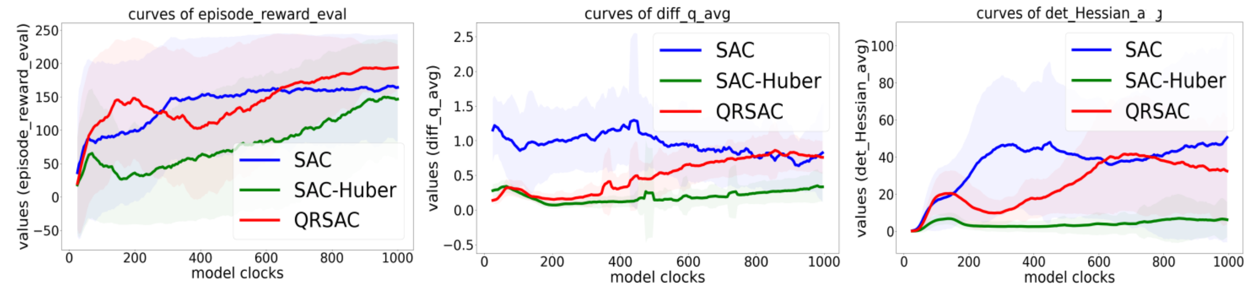


1D env. # 8

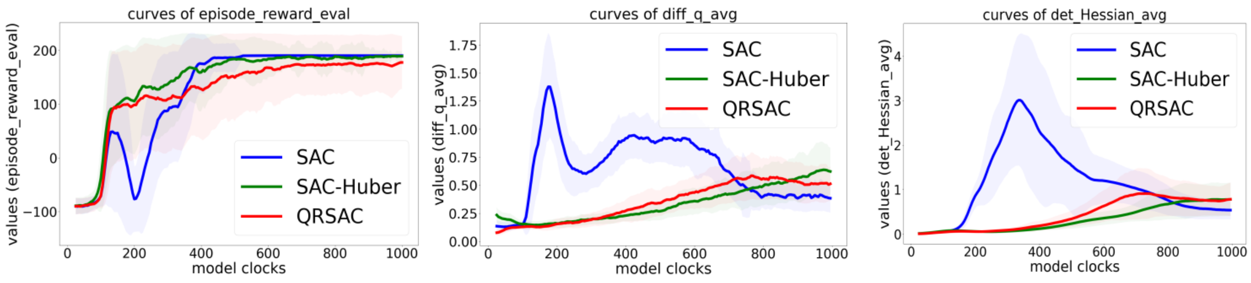
Figure A.20: Reward curve, stability, and smoothness metric (Hessian) curves for the variants of 1D environments #6 to #8.



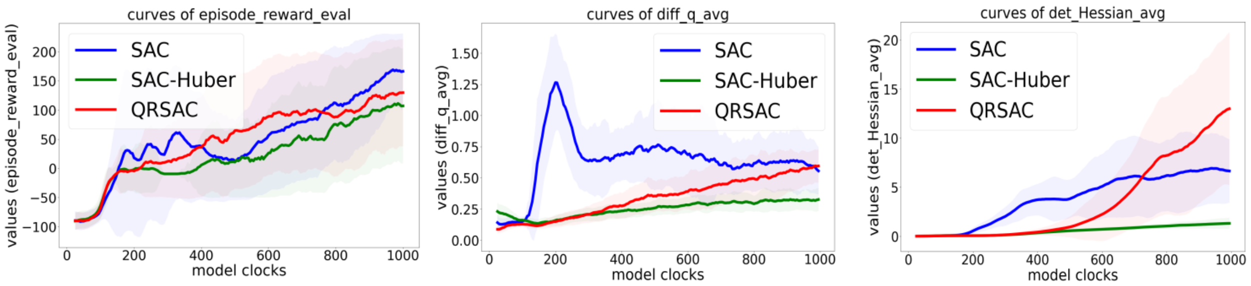
1D env. # 9



1D env. # 10



1D env. # 11



1D env. # 12

Figure A.21: Reward curve, stability, and smoothness metric (Hessian) curves for the variants of 1D environments #9 to #12.

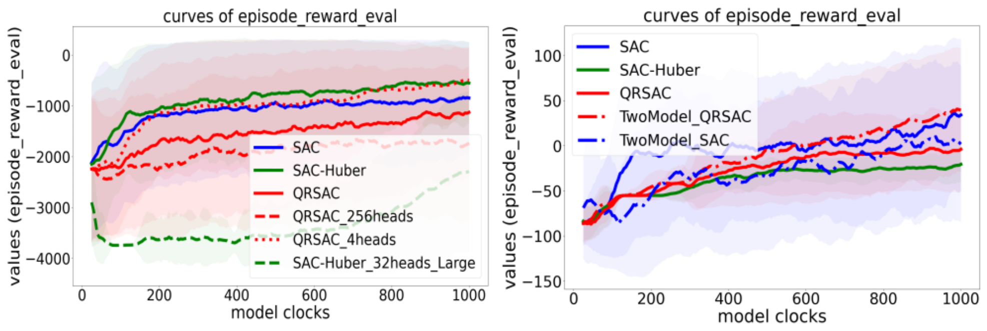
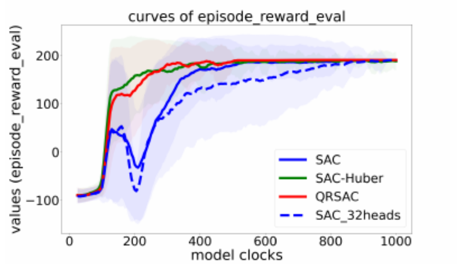
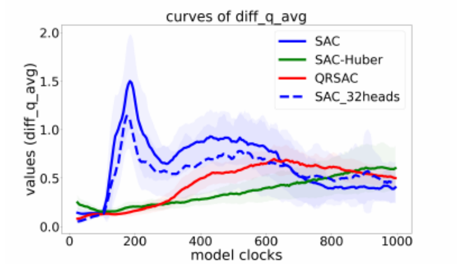


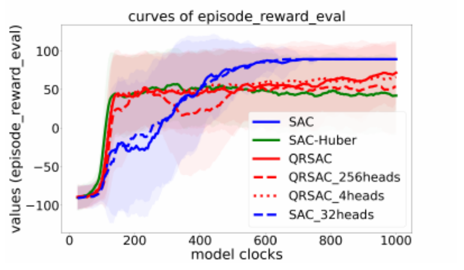
Figure A.22: Left: Learning curves of 1D example for Case study 2 with multi-headed SAC-Huber and QRSAC with different number of quantile heads. We observe that, under the existence of rewards of large magnitude, having more heads leads to poor performance. Right: Two model example for Class 3.



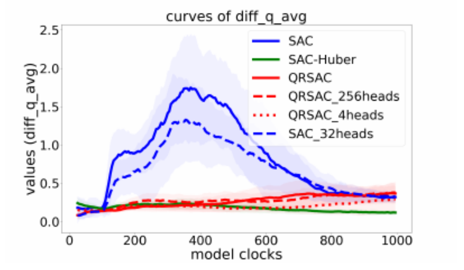
(a) Episode reward evaluations for Class 2 with multi-headed SAC.



(b) Oscillation of critic surface for Class 2 with multi-headed SAC.



(c) Episode reward evaluations for Class 4 with multi-headed SAC.



(d) Oscillation of critic surface for Class 4 with multi-headed SAC.

Figure A.23: Learning curves of the 1D example with multi-headed SAC, showing having multiple heads may not be the root cause of performance difference.

A.6 Software license

License information about the software used in this thesis is summarized below.

Julia: The MIT License; Copyright (c) 2009-2021: Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and other contributors: https://github.com/JuliaLang/julia/contributors

OpenAI Gym: The MIT License; Copyright (c) 2016 OpenAI (https://openai.com)

DeepMind Control Suite: Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

Lyceum: The MIT License; Copyright (c) 2019 Colin Summers, The Contributors of Lyceum

MuJoCo: MuJoCo Pro Lab license

sigkernel: Apache License 2.0; Copyright [2021] [Cristopher Salvi]

torchcubicspline: Apache License 2.0; Copyright [Patrick Kidger and others]

signatory: Apache License 2.0; Copyright [Patrick Kidger and others]

DiffRL: NVIDIA Source Code License

rl_games: MIT License; Copyright (c) 2019 Denys88

Franka URDF: MIT License; Copyright (c) Facebook, Inc. and its affiliates

Appendix B
ADDITIONAL THEORETICAL RESULTS

Additional theoretical contributions and some of the detailed proofs omitted in the main body of the thesis are presented here.

B.1 Proof of Theorem 4.2.4

In this section, we provide a proof of Theorem 4.2.4 in Chapter 4. Before delving into the proof, we give additional notations. Given a instantaneous cost function c, we define the cost (or the "cost-to-go") of a policy as:

$$J^{\pi}(x; c, W) = \mathbb{E}\left[\sum_{h=0}^{H-1} c(x_h, u_h)\middle|\pi, x_0 = x, W\right]$$

where the expectation is over trajectories sampled under π starting from x_0 in the model parameterized by W. The "cost-to-go" at state x at time $h \in [H]$ is denoted by:

$$J_h^{\pi}(x; c, W) = \mathbb{E}\left[\sum_{\ell=h}^{H-1} c(x_\ell, u_\ell)\middle|\pi, x_h = x\right].$$

When clear from context, we let the episode t index the policy, e.g., we write $J^t(x; c)$ to refer to $J^{\pi^t}(x, c)$. Subscripts refer to the time step within an episode and superscripts index the episode itself, i.e., ϕ_h^t will refer to the random vector which is the observed features during time step h within episode t. We let $\mathscr{H}_t$ denote the history up to the beginning of episode t.

Also, $\|x\|_M^2 := x^\top M x$ for a vector x and a matrix M. Note that the notation π for a policy is sometimes used as a map $\mathcal{X} \to \mathcal{U}$ or a generator of a trajectory when clear in the context, and that $P(\cdot|W, x, u)$ denotes the transition probability distribution at state x, control u, and under the model parameterization W.

B.1.1 Simulation analysis

We derive a novel self-bounding simulation lemma (Lemma B.1.3), using the Optional Stopping Theorem.

Lemma B.1.1 (Difference lemma). *Fix a policy π, cost function c, and model W. Consider any trajectory $\{x_h, u_h\}_{h=0}^{H-1}$ where $u_h = \pi(x_h)$ for all $h \in [H]$. For $h \in [H]$, let $\widehat{J}_h$ refer to the realized cost-to-go on this trajectory, i.e.,*

$$\widehat{J}_h = \sum_{\tau=h}^{H-1} c(x_\tau, u_\tau).$$

For all $\tau \in \{1, \dots H-1\}$, we have that:

$$\begin{aligned}\widehat{J}_0 - J_0^\pi(x_0; c, W) = \widehat{J}_\tau &- \mathbb{E}_{x'_\tau \sim P(\cdot|W, x_{\tau-1}, u_{\tau-1})} J_\tau^\pi(x'_\tau; c, W) \\ &+ \sum_{h=1}^{\tau-1} J_h^\pi(x_h; c, W) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} J_h^\pi(x'_h; c, W)\end{aligned}$$

Proof. Starting from $h = 0$, using $u_0 = \pi(x_0)$, we have:

$$\begin{aligned}\widehat{J}_0 - J_0^\pi(x_0; c, W) &= \widehat{J}_1 - \mathbb{E}_{x'_1 \sim P(\cdot|W, x_0, u_0)} J_1^\pi(x'_1; c, W) \\ &= \widehat{J}_1 - J_1^\pi(x_1; c, W) + J_1^\pi(x_1; c, W) - \mathbb{E}_{x'_1 \sim P(\cdot|W, x_0, u_0)} J_\ell^\pi(x'_1; c, W) \\ &= \widehat{J}_2 - \mathbb{E}_{x'_2 \sim P(\cdot|x_1, u_1, W)} J_2^\pi(x'_2; c, W) \\ &\quad + J_1^\pi(x_1; c, W) - \mathbb{E}_{x'_1 \sim P(\cdot|W, x_0, u_0)} J_1^\pi(x'_1; c, W).\end{aligned}$$

Recursion completes the proof, where, at each step of the recursion, we add and subtract $J_t^\pi(x_t; c, W)$ and apply the same operation on the term $\widehat{J}_t - J_t^\pi(x_t; c, W)$. □

Lemma B.1.2 ("Optional Stopping" simulation lemma). *Fix a policy π, cost function c, and model W. Consider the stochastic process over trajectories, where $\{x_h, u_h\}_{h=0}^{H} \sim \pi$ is sampled with respect to the model $W^\star$. With respect to this stochastic process, define a stopping time τ as:*

$$\tau = \min\left\{h \geq 0 : J_h^\pi(x_h; c, W) \geq J_h^\pi(x_h; c, W^\star)\right\}.$$

Define the random variable $\widetilde{J}_h^{\pi}(x_h)$ *as:*

$$\widetilde{J}_h^{\pi}(x_h) = \min\left\{J_h^{\pi}(x_h; c, W), J_h^{\pi}(x_h; c, W^\star)\right\}.$$

We have that:

$$\begin{aligned} & J_0^{\pi}(x_0; c, W^\star) - J_0^{\pi}(x_0; c, W) \\ & \leq \mathbb{E}\left[\sum_{h=0}^{H-1} 1\{h < \tau\}\left(\mathbb{E}_{x'_{h+1}\sim P(\cdot|W^\star, x_h, u_h)}\widetilde{J}_{h+1}^{\pi}(x'_{h+1}) - \mathbb{E}_{x'_{h+1}\sim P(\cdot|W, x_h, u_h)}\widetilde{J}_{h+1}^{\pi}(x'_{h+1})\right)\right] \end{aligned}$$

where the expectation is with respect to $\{x_h, u_h\}_{h=0}^{H} \sim \pi$ *sampled with respect to the model* $W^\star$.

Proof. Our filtration, $\mathscr{F}_h$, at time h will be the previous noise variables, i.e.,

$$\mathscr{F}_h := \{\epsilon_0, \epsilon_1, \ldots, \epsilon_{h-1}\},$$

and note that $\{x_1, u_1, c(x_1, u_1), \ldots, x_h, u_h, c(x_h, u_h)\}$ is fully determined by $\mathscr{F}_h$. Also, observe that τ is a valid stopping time with respect to the filtration $\mathscr{F}_h$.

Define:

$$M_h = \mathbb{E}\left[\widehat{J}_0 - J^\star(x_0; c, W) \mid \mathscr{F}_h\right]$$

which is a Doob martingale (with respect to our filtration), and so $\mathbb{E}[M_{h+1}|\mathscr{F}_h] = M_h$. By Doob's optional stopping theorem,

$$\mathbb{E}\left[\widehat{J}_0 - J^\star(x_0; c, W)\right] = \mathbb{E}[M_\tau] = \mathbb{E}\left[\mathbb{E}\left[\widehat{J}_0 - J^\star(x_0; c, W) \mid \mathscr{F}_\tau\right]\right]. \tag{B.1.1}$$

The proof consists in bounding M_τ.

Consider an $\mathscr{F}_\tau$, which is stopped at the random time τ. By Lemma B.1.1,

$$\begin{aligned}
M_\tau &= \mathbb{E}\left[\widehat{J}_0 - J^\star(x_0; c, W) \mid \mathscr{F}_\tau\right] \\
&= J_\tau\left(x_\tau; c, W^\star\right) - \mathbb{E}_{x'_\tau \sim P(\cdot|W, x_{\tau-1}, u_{\tau-1})} J_h(x'_\tau; c, W) \\
&\quad + \sum_{h=1}^{\tau-1} \Big(J_h(x_h; c, W) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} J_h(x'_h; c, W) \Big) \\
&= \sum_{h=1}^{\tau} \Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} J_h(x'_h; c, W) \Big) \\
&\le \sum_{h=1}^{\tau} \Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \Big) \\
&= \sum_{h=1}^{H} 1(h \le \tau) \Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \Big).
\end{aligned}$$

where the third equality follows using the definition of τ which implies that $J_\tau\left(x_\tau; c, W^\star\right) = \widetilde{J}_\tau\left(x_\tau\right)$ and that $J_h(x_h; c, W) = \widetilde{J}_h\left(x_h\right)$ for $h < \tau$; and the inequality is due to the definition of $\widetilde{J}$.

Using this bound on M_τ and (B.1.1), we have:

$$\mathbb{E}\left[\widehat{J}_0 - J^\star(x_0; c, W)\right] \le \sum_{h=1}^{H} \mathbb{E}\left[1(h \le \tau)\Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \Big)\right].$$

For the h-th term, observe:

$$\begin{aligned}
&\mathbb{E}\Big[1(h \le \tau)\Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \Big)\Big] \\
&= \mathbb{E}\Big[\mathbb{E}\left[1(h \le \tau)\Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \Big) \mid \mathscr{F}_{h-1}\right]\Big] \\
&= \mathbb{E}\Big[\mathbb{E}\left[1(h-1 < \tau)\Big(\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \Big) \mid \mathscr{F}_{h-1}\right]\Big] \\
&= \mathbb{E}\Big[1(h-1 < \tau)\mathbb{E}\left[\widetilde{J}_h\left(x_h\right) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h\left(x'_h\right) \mid \mathscr{F}_{h-1}\right]\Big] \\
&= \mathbb{E}\Big[1(h-1 < \tau)\left(\mathbb{E}_{x'_h \sim P(\cdot|W^\star, x_{h-1}, u_{h-1})} \widetilde{J}_h(x'_h) - \mathbb{E}_{x'_h \sim P(\cdot|W, x_{h-1}, u_{h-1})} \widetilde{J}_h(x'_h)\right)\Big].
\end{aligned}$$

where the second equality uses that $1(h \le \tau) = 1(h-1 < \tau)$, and the third equality uses that $1(h-1 < \tau)$ is measurable with respect to $\mathscr{F}_{h-1} = \{\epsilon_0, \ldots, \epsilon_{h-2}\}$. This completes the proof. $\square$

The previous lemma allows us to bound the difference in cost under two different models, i.e., $J^\pi(x;c,W^\star) - J^\pi(x;c,W)$, in terms of the second moment of the cumulative cost itself, i.e., in terms of $V^\pi(x;c,W^\star)$, where

$$V^\pi(x_0;c,W^\star) := \mathbb{E}\left[\left(\sum_{h=0}^{H-1} c(x_h,u_h)\right)^2 \middle| x_0,\pi,W^\star\right].$$

Lemma B.1.3 (Self-bounding, simulation lemma)**.** *For any policy π, model parameterization W, and nonnegative cost c, and for any state x_0, we have:*

$$\begin{aligned} & J^\pi(x_0;c,W^\star) - J^\pi(x_0;c,W) \\ & \leq \sqrt{HV^\pi(x_0;c,W^\star)}\sqrt{\mathbb{E}\left[\sum_{h=0}^{H-1} \min\left\{\frac{1}{\sigma^2}\left\|(W^\star - W)\,\phi(x_h,u_h)\right\|^2_{\mathbb{R}^{d_x}},1\right\}\right]}. \end{aligned}$$

where the expectation is with respect to π in $W^\star$ starting at x_0.

Proof. For the proof, it is helpful to define the random variables:

$$\begin{aligned} \Delta_h &= \mathbb{E}_{x'_{h+1}\sim P(\cdot|W^\star,x_h,u_h)}\left[\widetilde{J}_{h+1}(x'_{h+1})\right] - \mathbb{E}_{x'_{h+1}\sim P(\cdot|W,x_h,u_h)}\left[\widetilde{J}_{h+1}(x'_{h+1})\right] \\ A_h &:= \mathbb{E}_{x'_{h+1}\sim P(\cdot|W^\star,x_h,u_h)}\left[\widetilde{J}_{h+1}(x'_{h+1})^2\right] \end{aligned}$$

By Lemma C.3.1 (which bounds the difference in means under two Gaussian distributions, using the chi-squared distance function), we have:

$$\begin{aligned} \Delta_h &\leq \sqrt{\mathbb{E}_{x_{h+1}\sim P(\cdot|W^\star,x_h,u_h)}\left[\widetilde{J}_{h+1}(x_{h+1})^2\right]}\min\left\{\frac{1}{\sigma}\left\|(W^\star - W)\,\phi(x_h,u_h)\right\|_{\mathbb{R}^{d_x}},1\right\} \\ &= \sqrt{A_h}\min\left\{\frac{1}{\sigma}\left\|(W^\star - W)\,\phi(x_h,u_h)\right\|_{\mathbb{R}^{d_x}},1\right\}. \end{aligned}$$

From Lemma B.1.2, we have:

$$J_0^\pi(x_0; c, W^\star) - J_0^\pi(x_0; c, W) \le \sum_{h=0}^{H-1} \mathbb{E}\left[1(h < \tau)\Delta_h\right]$$

$$\le \sum_{h=0}^{H-1} \mathbb{E}\left[\sqrt{A_h} \min\left\{\frac{1}{\sigma}\left\|(W^\star - W)\,\phi(x_h, u_h)\right\|_{\mathbb{R}^{d_x}}, 1\right\}\right]$$

$$\le \sum_{h=0}^{H-1} \sqrt{\mathbb{E}\left[A_h\right]}\sqrt{\mathbb{E}\left[\min\left\{\frac{1}{\sigma^2}\left\|(W^\star - W)\,\phi(x_h, u_h)\right\|^2_{\mathbb{R}^{d_x}}, 1\right\}\right]}$$

$$\le \sqrt{\mathbb{E}\left[\sum_{h=0}^{H-1} A_h\right]}\sqrt{\mathbb{E}\left[\sum_{h=0}^{H-1} \min\left\{\frac{1}{\sigma^2}\left\|(W^\star - W)\,\phi(x_h, u_h)\right\|^2_{\mathbb{R}^{d_x}}, 1\right\}\right]},$$

where in the second inequality we use $\mathbb{E}[ab] \le \sqrt{\mathbb{E}[a^2]\mathbb{E}[b^2]}$ and the Cauchy-Schwartz inequality in the last inequality. For the first term, observe that:

$$\mathbb{E}\left[A_h\right] = \mathbb{E}\left[\mathbb{E}_{x'_{h+1} \sim P(\cdot|W^\star, x_h, u_h)}\left[\widetilde{J}_{h+1}(x'_{h+1})^2\right]\right] = \mathbb{E}\left[\widetilde{J}_{h+1}(x_{h+1})^2\right] \le \mathbb{E}\left[J_{h+1}(x_{h+1})^2\right]$$

$$= \mathbb{E}\left[\left(\mathbb{E}\left[\sum_{\ell=h+1}^{H-1} c(x_\ell, u_\ell) \mid x_{h+1}\right]\right)^2\right] \le \mathbb{E}\left[\left(\sum_{\ell=h+1}^{H-1} c(x_\ell, u_\ell)\right)^2\right]$$

$$\le \mathbb{E}\left[\left(\sum_{\ell=0}^{H-1} c(x_\ell, u_\ell)\right)^2\right] = V^\pi$$

where the first inequality uses the definition of $\widetilde{J}$; the second inequality follows from Jensen's inequality; and the last inequality follows from our assumption that the instantaneous costs are nonnegative. The proof is completed by substitution. □

B.1.2 Regret analysis (and proofs of Theorem 4.2.4)

Throughout, let $\mathcal{E}_{t,cb}$ be the event that $W^\star \in \textsc{Ball}^t$ holds at episode t.

Lemma B.1.4 (Per-episode regret lemma)**.** *Suppose Assumptions 4.2.1 and 4.2.3 hold. Let*

$\mathscr{H}_{<t}$ be the history of events before episode t. For the LC^3, we have:

$$\mathbf{1}(\mathcal{E}_{t,cb})\Big(J^t(x_0;c^t) - J^\star(x_0;c^t)\Big)$$
$$\leq \sqrt{HV^t(x_0;c,W^\star)\left(\frac{4\beta^t}{\sigma^2}+H\right)}\sqrt{\mathbb{E}\left[\min\left\{\sum_{h=0}^{H-1}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}},1\right\}\;\Big|\;\mathscr{H}_{<t}\right]}.$$

Note that the expectation is with respect to the trajectory of LC^3, i.e., it is under π^t in $W^\star$.

Proof. Suppose $\mathcal{E}_{t,cb}$ holds, else the lemma is immediate. By construction of the LC^3 algorithm (the optimistic property) and by the self-bounding, simulation lemma (Lemma B.1.3), we have:

$$J^t(x_0;c^t,W^\star) - J^\star(x_0;c^t,W^\star) \leq J^t(x_0;c^t,W^\star) - J^t(x_0;c^t,\widehat{W}^t)$$
$$\leq \sqrt{HV^t(x_0;c,W^\star)}\sqrt{\mathbb{E}\left[\sum_{h=0}^{H-1}\min\left\{\frac{1}{\sigma^2}\left\|\left(W^\star-\widehat{W}^t\right)\phi_h^t\right\|^2_{\mathbb{R}^{d_x}},1\right\}\;\Big|\;\mathscr{H}_{<t}\right]}.$$

where the expectation is with respect to the trajectory of LC^3, i.e., of π^t in $W^\star$.

For $W^\star \in \textsc{Ball}^t$, we have

$$\left\|\left(\widehat{W}^t-W^\star\right)\phi_h^t\right\|_{\mathbb{R}^{d_x}} \leq \left\|\left(\widehat{W}^t-W^\star\right)(\Sigma^t)^{1/2}\right\|\,\|(\Sigma^t)^{-1/2}\phi_h^t\|$$
$$\leq \left(\left\|\left(\widehat{W}^t-\overline{W}^t\right)(\Sigma^t)^{1/2}\right\| + \left\|\left(\overline{W}^t-W^\star\right)(\Sigma^t)^{1/2}\right\|\right)\|\phi_h^t\|_{(\Sigma^t)^{-1}} \leq 2\sqrt{\beta^t}\|\phi_h^t\|_{(\Sigma^t)^{-1}}.$$

where we have also used that $\widehat{W}^t,\overline{W}^t \in \textsc{Ball}^t$, by construction.

This implies that:

$$\sum_{h=0}^{H-1}\min\left\{\frac{1}{\sigma^2}\|(W^\star-\widehat{W}^t)\phi_h^t\|^2_{\mathbb{R}^{d_x}},1\right\} \leq \sum_{h=0}^{H-1}\min\left\{\frac{4\beta^t}{\sigma^2}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}},1\right\}$$
$$\leq \min\left\{\frac{4\beta^t}{\sigma^2}\sum_{h=0}^{H-1}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}},H\right\} \leq \max\left\{\frac{4\beta^t}{\sigma^2},H\right\}\min\left\{\sum_{h=0}^{H-1}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}},1\right\}.$$

The proof is completed by substitution. □

Before we complete the proofs, the following two lemmas are helpful. The first lemma bounds the sum failure probability of $W^\star$ not being in all the confidence balls (over all the episodes); the lemma generalizes the argument from [1, 73] to matrix regression.

Lemma B.1.5 (Confidence ball)**.** *Let*

$$\beta^t = 2\lambda\|W^\star\|^2 + 8\sigma^2\left(d_x\log(5) + 2\log(t) + \log(4) + \log\left(\det(\Sigma^t)/\det(\Sigma^0)\right)\right).$$

We have:

$$\sum_{t=0}^{\infty}\Pr\left(\overline{\mathcal{E}}_{t,cb}\right) = \sum_{t=0}^{\infty}\Pr\left(\left\|\left(\overline{W}^t - W^\star\right)\left(\Sigma^t\right)^{1/2}\right\|^2 > \beta^t\right) \le \frac{1}{2}.$$

Proof. The center of the confidence ball, $\overline{W}^t$, is the minimizer of the ridge regression objective in (4.2.1); its closed-form expression is:

$$\overline{W}^t := \sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1} x_{h+1}^\tau(\phi_h^\tau)^\top(\Sigma^t)^{-1},$$

where $\Sigma^t = \lambda I + \sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1}\phi_h^\tau(\phi_h^\tau)^\top$. Using that $x_{h+1}^\tau = W^\star\phi_h^\tau + \epsilon_h^\tau$ with $\epsilon_h^\tau \sim \mathcal{N}(0,\sigma^2 I)$,

$$\begin{aligned}
\overline{W}^t - W^\star &= \sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1} x_{h+1}^\tau(\phi_h^\tau)^\top(\Sigma^t)^{-1} - W^\star \\
&= \sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1}(W^\star\phi_h^\tau + \epsilon_h^\tau)(\phi_h^\tau)^\top(\Sigma^t)^{-1} - W^\star \\
&= W^\star\left(\sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1}\phi_h^\tau(\phi_h^\tau)^\top\right)(\Sigma^t)^{-1} - W^\star + \sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1}\epsilon_h^\tau(\phi_h^\tau)^\top(\Sigma^t)^{-1} \\
&= -\lambda W^\star\left(\Sigma^t\right)^{-1} + \sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1}\epsilon_h^\tau(\phi_h^\tau)^\top(\Sigma^t)^{-1}.
\end{aligned}$$

For any $0 < \delta_t < 1$, using Lemma C.3.3, it holds with probability at least $1-\delta_t$,

$$\begin{aligned}
\left\|\left(\overline{W}^t - W^\star\right)\left(\Sigma^t\right)^{1/2}\right\| &\le \left\|\lambda W^\star\left(\Sigma^t\right)^{-1/2}\right\| + \left\|\sum_{\tau=0}^{t-1}\sum_{h=0}^{H-1}\epsilon_h^\tau(\phi_h^\tau)^\top(\Sigma^t)^{-1/2}\right\| \\
&\le \sqrt{\lambda}\|W^\star\| + \sigma\sqrt{8d_x\log(5) + 8\log\left(\det(\Sigma^t)\det(\Sigma^0)^{-1}/\delta_t\right)}.
\end{aligned}$$

where we have also used the triangle inequality. Therefore, $\Pr(\overline{\mathcal{E}}_{t,cb}) \le \delta_t$.

We seek to bound $\sum_{t=0}^{\infty}\Pr(\overline{\mathcal{E}}_{t,cb})$. Due to that at $t = 0$ we have initialized $\text{B}\small{\text{ALL}}^0$ to contain $W^\star$, we have $\Pr(\overline{\mathcal{E}}_{0,cb}) = 0$. For $t \ge 1$, let us assign failure probability $\delta_t = (3/\pi^2)/t^2$

for the t-th event, which, using the above, gives us an upper bound on the sum failure probability as $\sum_{t=1}^{\infty} \Pr(\overline{\mathcal{E}}_{t,cb}) < \sum_{t=1}^{\infty}(1/t^2)(3/\pi^2) = 1/2$. This completes the proof. $\square$

The next lemma provides a bound on the potential function used in our analysis. It is based on the elliptical potential function argument from [73, 228].

Lemma B.1.6 (Sum of potential functions). *For any sequence of ϕ_h^t, we have:*

$$\sum_{t=0}^{T-1} \min\left\{\sum_{h=0}^{H-1} \|\phi_h^t\|^2_{(\Sigma^t)^{-1}}, 1\right\} \leq 2\log\left(\det(\Sigma^T)\det(\Sigma^0)^{-1}\right).$$

Proof. Recall that $\Sigma^{t+1} = \Sigma^t + \sum_{h=0}^{H-1} \phi_h^t \left(\phi_h^t\right)^\top$ and $\Sigma^0 = \lambda I$. First use $x \leq 2\log(1+x)$ for $x \in [0,1]$, we have:

$$\min\left\{\sum_{h=0}^{H-1} \|\phi_h^t\|^2_{(\Sigma^t)^{-1}}, 1\right\} \leq 2\log\left(1 + \sum_{h=0}^{H-1} \|\phi_h^t\|^2_{(\Sigma^t)^{-1}}\right).$$

For Σ^{t+1}, using its recursive formulation, we have:

$$\log\det\left(\Sigma^{t+1}\right) = \log\det\left(\Sigma^t\right) + \log\det\left(I + \left(\Sigma^t\right)^{-1/2} \sum_{h=0}^{H-1} \phi_h^t (\phi_h^t)^\top \left(\Sigma^t\right)^{-1/2}\right).$$

Denote the eigenvalues of $\left(\Sigma^t\right)^{-1/2} \sum_{h=0}^{H-1} \phi_h^t (\phi_h^t)^\top \left(\Sigma^t\right)^{-1/2}$ as σ_i for $i \geq 1$. We have

$$\log\det\left(I + \left(\Sigma^t\right)^{-1/2} \sum_{h=0}^{H-1} \phi_h^t (\phi_h^t)^\top \left(\Sigma^t\right)^{-1/2}\right) = \log\prod_{i\geq 1}(1+\sigma_i) \geq \log\left(1 + \sum_{i\geq 1}\sigma_i\right),$$

where the last inequality uses that $\sigma_i \geq 0$ for all i. Using the above and the definition of the trace,

$$\log\det\left(I + \left(\Sigma^t\right)^{-1/2} \sum_{h=0}^{H-1} \phi_h^t (\phi_h^t)^\top \left(\Sigma^t\right)^{-1/2}\right) \geq \log\left(1 + \operatorname{tr}\left(\left(\Sigma^t\right)^{-1/2} \sum_{h=0}^{H-1} \phi_h^t (\phi_h^t)^\top \left(\Sigma^t\right)^{-1/2}\right)\right)$$
$$= \log\left(1 + \sum_{h=0}^{H-1} (\phi_h^t)^\top (\Sigma^t)^{-1} \phi_h^t\right).$$

By telescoping the sum,

$$2\sum_{t=0}^{T-1}\log\left(1+\sum_{h=0}^{H-1}(\phi_h^t)^\top(\Sigma^t)^{-1}\phi_h^t\right)\le 2\sum_{t=0}^{T-1}\left(\log\det\left(\Sigma^{t+1}\right)-\log\det\left(\Sigma^t\right)\right)$$
$$=2\log\left(\det(\Sigma^T)\det(\Sigma^0)^{-1}\right),$$

which completes the proof. □

Also, recall that LC^3 uses the setting of $\lambda=\sigma^2/\|W^\star\|^2$. We will also use that, for β^T as defined in Lemma B.1.5,

$$\beta^T=2\sigma^2+8\sigma^2\left(d_x\log(5)+2\log(T)+\log(4)+\log\left(\det(\Sigma^T)\det(\Sigma^0)^{-1}\right)\right)$$
$$\le 16\sigma^2\left(d_x+\log(T)+\log\left(\det(\Sigma^T)\det(\Sigma^0)^{-1}\right)\right). \tag{B.1.2}$$

In particular, we can take $C_1=16$ in LC^3. Also,

$$\mathbb{E}[\beta^T]\le 16\sigma^2\left(d_x+\log(T)+\gamma_T(\lambda)\right). \tag{B.1.3}$$

using the definition of the information gain.

We now conclude the proof of our main theorem (Theorem 4.2.4).

Proof of Theorem 4.2.4. Using the per-episode regret bound (Lemma B.1.4), our confidence ball, failure probability bound (Lemma B.1.5), and that $V^t \leq V_{\max}$,

$$
\begin{aligned}
\mathbb{E}\left[\textsc{Regret}_{\mathrm{LC}^3}\right] &= \mathbb{E}\left[\sum_{t=0}^{T-1}\left(J^t(x_0;c^t) - J^\star(x_0;c^t)\right)\right] \\
\leq\ & \mathbb{E}\left[\sum_{t=0}^{T-1}\mathbb{E}\left[\mathbf{1}(\mathcal{E}_{t,cb})\left(J^t(x_0;c^t) - J^\star(x_0;c^t)\right) \mid \mathscr{H}_t\right]\right] + \sqrt{V_{\max}}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathbf{1}(\overline{\mathcal{E}}_{t,cb})\right] \\
\leq\ & \sqrt{HV_{\max}}\sum_{t=0}^{T-1}\mathbb{E}\left[\sqrt{\frac{4\beta^t}{\sigma^2}+H}\ \sqrt{\mathbb{E}\left[\min\left\{\sum_{h=0}^{H-1}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}}, 1\right\}\ \Big|\ \mathscr{H}_{<t}\right]}\ \right] + \sqrt{V_{\max}}/2 \\
\leq\ & \sqrt{HV_{\max}}\sum_{t=0}^{T-1}\sqrt{\mathbb{E}\left[\frac{4\beta^t}{\sigma^2}+H\right]}\ \sqrt{\mathbb{E}\left[\min\left\{\sum_{h=0}^{H-1}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}}, 1\right\}\right]} + \sqrt{V_{\max}}/2 \\
\leq\ & \sqrt{HV_{\max}}\sqrt{\sum_{t=0}^{T-1}\mathbb{E}\left[\frac{4\beta^t}{\sigma^2}+H\right]}\ \sqrt{\mathbb{E}\left[\sum_{t=0}^{T-1}\min\left\{\sum_{h=0}^{H-1}\|\phi_h^t\|^2_{(\Sigma^t)^{-1}}, 1\right\}\right]} + \sqrt{V_{\max}}/2 \\
\leq\ & \sqrt{HV_{\max}}\sqrt{T\left(\frac{4\mathbb{E}[\beta^T]}{\sigma^2}+H\right)}\ \sqrt{\gamma_T(\lambda)} + \sqrt{V_{\max}}/2 \\
\leq\ & \sqrt{HV_{\max}}\sqrt{64T\left(d_x + \log(T) + \gamma_T(\lambda) + H\right)}\ \sqrt{\gamma_T(\lambda)} + \sqrt{V_{\max}}/2
\end{aligned}
$$

where the third inequality uses $\mathbb{E}[ab] \leq \sqrt{\mathbb{E}[a^2]\mathbb{E}[b^2]}$; the fourth uses the Cauchy-Schwartz inequality; the penultimate step uses that β_t is non-decreasing, along with the Lemma B.1.6 and the definition of the information gain; and the final step uses the bound on β^T in (B.1.3). This completes the proof. □

B.2 Proof of Theorem 4.3.16

In this section, we present the proof of Theorem 4.3.16 in Chapter 4. Throughout this section, suppose Assumptions 4.3.4 to 4.3.15 hold. Note that Assumption 4.3.9 is required for `OptDynamics`. We give some definitions of the values for the subsequent regret analysis.

B.2.1 Some definitions of the values

The value $J^\pi(X_0^t; M; c^t)$ in Algorithm 2 is defined by

$$J^\pi(X_0^t; M; c^t) := \sum_{n=0}^{N^t-1} \mathbb{E}_{\Omega_\pi}\left[\sum_{h=0}^{H_n^t-1} c^t(x_{h,n}^t)\middle|\pi, M, x_{0,n}^t\right],$$

where the expectation is taken over the trajectory following $\boldsymbol{\phi}_{x_{h+1}} = [\Psi(\pi)^\dagger \circ M]\boldsymbol{\phi}_{x_h} + \epsilon(\omega)$. Also the confidence ball at time instance t is given by

$$\text{BALL}_M^t := \left\{M\middle| \left\|(\Sigma_{\mathscr{A}}^t)^{\frac{1}{2}}\left(M - \overline{M}^t\right)\right\|^2 \le \beta_M^t\right\} \cap \text{BALL}_M^0, \;\; \Sigma_{\mathscr{A}}^t := \lambda I + \sum_{\tau=0}^{t-1}\sum_{n=0}^{N^\tau-1}\sum_{h=0}^{H_n^\tau-1} {\mathscr{A}_{h,n}^\tau}^\dagger \mathscr{A}_{h,n}^\tau,$$

and $\Sigma_{\mathscr{A}}^0 := \lambda I$, where $\beta_M^t := 20\sigma^2\left(d_\phi + \log\left(t\frac{\det(\Sigma_{\mathscr{A}}^t)}{\det(\Sigma_{\mathscr{A}}^0)}\right)\right)$ and

$$\overline{M}^t := \arg\min_M \left[\sum_{\tau=0}^{t-1}\sum_{n=0}^{N^\tau-1}\sum_{h=0}^{H_n^\tau-1}\left\|\boldsymbol{\phi}_{x_{h+1,n}^\tau} - \mathscr{A}_{h,n}^\tau(M)\right\|_{\mathbb{R}^{d_\phi}}^2 + \lambda\|M\|_{\text{HS}}^2\right].$$

Similar to the case of the analysis of LC^3, we define the expected maximum information gains as:

$$\gamma_{T,\mathscr{A}}(\lambda) := 2\max_{\mathcal{A}}\mathbb{E}_{\mathcal{A}}\left[\log\left(\frac{\det\left(\Sigma_{\mathscr{A}}^T\right)}{\det\left(\Sigma_{\mathscr{A}}^0\right)}\right)\right].$$

Here, det is a properly defined functional determinant of a bounded linear operator. Further,

$$\gamma_{2,T,\mathscr{A}}(\lambda) := 2\max_{\mathcal{A}}\mathbb{E}_{\mathcal{A}}\left[\left(\log\left(\frac{\det\left(\Sigma_{\mathscr{A}}^T\right)}{\det\left(\Sigma_{\mathscr{A}}^0\right)}\right)\right)^2\right].$$

Also, we define

$$\Sigma_{\mathscr{B}}^t := \lambda I + \sum_{\tau=0}^{t-1}{\mathscr{B}^\tau}^\dagger\mathscr{B}^\tau, \quad \Sigma_{\mathscr{B}}^0 := \lambda I, \quad \gamma_{T,\mathscr{B}}(\lambda) := 2\max_{\mathcal{A}}\mathbb{E}_{\mathcal{A}}\left[\log\left(\frac{\det\left(\Sigma_{\mathscr{B}}^T\right)}{\det\left(\Sigma_{\mathscr{B}}^0\right)}\right)\right].$$

Lemma B.2.1. *Assume that $\Psi(\pi) \in \mathbb{R}^{d_\Psi \times d_\phi}$ and that $\|\mathscr{B}^t\|_{\text{HS}} \le B_{\mathscr{B}}$, $\|\mathscr{A}_{h,n}^t\|_{\text{HS}} \le B_{\mathscr{A}}$ for all $t \in [T]$, $n \in [N^t]$, and $h \in [H_n^t]$ and some $B_{\mathscr{B}} \ge 0$ and $B_{\mathscr{A}} \ge 0$. Then, $\gamma_{T,\mathscr{A}}(\lambda) = O(d_\phi d_\Psi \log(1 + THB_{\mathscr{A}}^2/\lambda))$, and $\gamma_{T,\mathscr{B}}(\lambda) = O(d_\phi d_\Psi \log(1 + TB_{\mathscr{B}}^2/\lambda))$.*

Proof. For $\gamma_{T,\mathscr{A}}(\lambda)$, from the definition of Hilbert-Schmidt norm, we have

$$\operatorname{tr}\left(\sum_{t=0}^{T-1}\sum_{n=0}^{N^t-1}\sum_{h=0}^{H_n^t-1}\mathscr{A}_{h,n}^{t}{}^{\dagger}\mathscr{A}_{h,n}^{t}\right)\leq THB_{\mathscr{A}}^2,$$

and the result follows from Lemma C.3.4. The similar argument holds for $\gamma_{T,\mathscr{B}}(\lambda)$ too. $\square$

Now, we give the regret analysis below.

B.2.2 Regret analysis

We fist give the *positive operator norm bounding lemma* followed by another lemma based on it.

Lemma B.2.2 (Positive operator norm bounding lemma)**.** *Let $\mathcal{H}$ be a Hilbert space and $A_i,\ B_i \in \mathcal{L}(\mathcal{H};\mathcal{H})$ $(i \in \{1,2,\dots,n\})$. Assume, for all $i \in \{1,2,\dots,n\}$, that A_i is positive definite. Also, assume B_1 is positive definite, and for all $i \in \{2,3,\dots,n\}$, B_i is positive semi-definite. Then,*

$$\left\|\left(\sum_i B_i^{\frac{1}{2}}A_iB_i^{\frac{1}{2}}\right)^{-\frac{1}{2}}\left(\sum_i B_i\right)\left(\sum_i B_i^{\frac{1}{2}}A_iB_i^{\frac{1}{2}}\right)^{-\frac{1}{2}}\right\|\leq\max_i\left\|A_i^{-1}\right\|.$$

If $A_iB_i = B_iA_i$ for all $i \in \{1,2,\dots,n\}$, then

$$\left\|\left(\sum_i A_iB_i\right)^{-\frac{1}{2}}\left(\sum_i B_i\right)\left(\sum_i A_iB_i\right)^{-\frac{1}{2}}\right\|\leq\max_i\left\|A_i^{-1}\right\|.$$

Proof. Let $c := \max_i \left\|A_i^{-1}\right\|$. Then, we have, for all i,

$$I \preceq \left\|A_i^{-1}\right\| A_i \preceq cA_i,$$

from which it follows that

$$B_i \preceq cB_i^{\frac{1}{2}}A_iB_i^{\frac{1}{2}}.$$

Therefore, we obtain

$$\sum_i B_i \preceq c \sum_i B_i^{\frac{1}{2}} A_i B_i^{\frac{1}{2}}.$$

From the assumptions, $\left(\sum_i B_i^{\frac{1}{2}} A_i B_i^{\frac{1}{2}}\right)^{-1}$ exists and

$$\left(\sum_i B_i^{\frac{1}{2}} A_i B_i^{\frac{1}{2}}\right)^{-\frac{1}{2}} \left(\sum_i B_i\right) \left(\sum_i B_i^{\frac{1}{2}} A_i B_i^{\frac{1}{2}}\right)^{-\frac{1}{2}} \preceq cI,$$

from which we obtain

$$\left\| \left(\sum_i B_i^{\frac{1}{2}} A_i B_i^{\frac{1}{2}}\right)^{-\frac{1}{2}} \left(\sum_i B_i\right) \left(\sum_i B_i^{\frac{1}{2}} A_i B_i^{\frac{1}{2}}\right)^{-\frac{1}{2}} \right\| \leq c.$$

The second claim follows immediately. □

Lemma B.2.3. *Suppose Assumptions 4.3.4, 4.3.6, and 4.3.13 hold. Then, it follows that, for all $t \in [T]$,*

$$\left\| (\Sigma_{\mathscr{B}}^t)^{\frac{1}{2}} \left(M - \overline{M}^t\right) \right\|^2 \leq (1 + C^{-1}) \left\| (\Sigma_{\mathscr{A}}^t)^{\frac{1}{2}} \left(M - \overline{M}^t\right) \right\|^2.$$

Proof. Under Assumptions 4.3.4 and 4.3.6, define $\mathscr{C}^t \in \mathcal{L}\left(\mathcal{L}(\mathcal{H};\mathcal{H}'); \mathcal{L}(\mathcal{H};\mathcal{H}')\right)$ by

$$\mathscr{C}^t(M) = M \circ \left[\sum_{n=0}^{N^t-1} \sum_{h=0}^{H_n^t-1} \boldsymbol{\phi}_{x_{h,n}^t} \boldsymbol{\phi}_{x_{h,n}^t}^{\dagger} \right].$$

Also, define $\mathscr{X}^t := \sum_{n=0}^{N^t-1} \sum_{h=0}^{H_n^t-1} {\mathscr{A}_{h,n}^t}^{\dagger} \mathscr{A}_{h,n}^t$ and $\mathscr{Y}^t := \mathscr{B}^{t\dagger} \mathscr{B}^t$. We have $\mathscr{C}^t \mathscr{Y}^t = \mathscr{Y}^t \mathscr{C}^t = \mathscr{X}^t$ (and thus $\mathscr{X}^t \mathscr{Y}^t = \mathscr{Y}^t \mathscr{X}^t$), and

$$\Sigma_{\mathscr{A}}^t = \lambda I + \sum_{\tau=0}^{t-1} \mathscr{X}^\tau, \qquad \Sigma_{\mathscr{B}}^t = \lambda I + \sum_{\tau=0}^{t-1} \mathscr{Y}^\tau.$$

From Assumption 4.3.13, we obtain, for all $t \in [T]$, $(\mathscr{C}^t)^{-1}$ exists and

$$\|(\mathscr{C}^t)^{-1}\| \leq \left\| \left(\sum_{n=0}^{N^t-1} \sum_{h=0}^{H_n^t-1} \boldsymbol{\phi}_{x_{h,n}^t} \boldsymbol{\phi}_{x_{h,n}^t}^{\dagger} \right)^{-1} \right\| \leq \left\| \left(\sum_{n=0}^{N^t-1} \boldsymbol{\phi}_{x_{0,n}^t} \boldsymbol{\phi}_{x_{0,n}^t}^{\dagger} \right)^{-1} \right\| \leq C^{-1}.$$

Therefore, using Lemma B.2.2 by substituting I and $\mathscr{C}$ to A, λI and $\mathscr{Y}$ to B, it follows that

$$\left\|(\Sigma^t_{\mathscr{A}})^{-\frac{1}{2}}\Sigma^t_{\mathscr{B}}(\Sigma^t_{\mathscr{A}})^{-\frac{1}{2}}\right\| \leq \max\left\{1, \max_{\tau\in[t]}\{\|(\mathscr{C}^\tau)^{-1}\|\}\right\} \leq 1 + C^{-1},$$

and

$$\begin{aligned}
&\left\|(\Sigma^t_{\mathscr{B}})^{\frac{1}{2}}\left(M - \overline{M}^t\right)\right\|^2 = \left\|(\Sigma^t_{\mathscr{B}})^{\frac{1}{2}}(\Sigma^t_{\mathscr{A}})^{-\frac{1}{2}}(\Sigma^t_{\mathscr{A}})^{\frac{1}{2}}\left(M - \overline{M}^t\right)\right\|^2 \\
&\leq \left\|(\Sigma^t_{\mathscr{A}})^{\frac{1}{2}}\left(M - \overline{M}^t\right)\right\|^2\left\|(\Sigma^t_{\mathscr{A}})^{-\frac{1}{2}}(\Sigma^t_{\mathscr{B}})^{\frac{1}{2}}\right\|^2 = \left\|(\Sigma^t_{\mathscr{A}})^{-\frac{1}{2}}\Sigma^t_{\mathscr{B}}(\Sigma^t_{\mathscr{A}})^{-\frac{1}{2}}\right\|\left\|(\Sigma^t_{\mathscr{A}})^{\frac{1}{2}}\left(M - \overline{M}^t\right)\right\|^2 \\
&\leq (1 + C^{-1})\left\|(\Sigma^t_{\mathscr{A}})^{\frac{1}{2}}\left(M - \overline{M}^t\right)\right\|^2.
\end{aligned}$$

Here, the second equality used $\|\mathscr{A}\|^2 = \|\mathscr{A}\mathscr{A}^\dagger\|$. □

Now, let $\mathcal{E}^t$ be the event $M^\star \in \textsc{Ball}^t_M$. Assume $\bigcap_{t=0}^{T-1}\mathcal{E}^t$. Then, using Assumption 4.3.11,

$$\begin{aligned}
&\left(\Lambda[\mathscr{K}(\pi^t)] + J^{\pi^t}(X^t_0; c^t)\right) - \left(\Lambda[\mathscr{K}(\pi^{\star t})] + J^{\pi^{\star t}}(X^t_0; c^t)\right) \\
&= \left(\Lambda[\mathscr{K}(\pi^t)] - \Lambda[\mathscr{K}(\pi^{\star t})]\right) + \left(J^{\pi^t}(X^t_0; c^t) - J^{\pi^{\star t}}(X^t_0; c^t)\right) \\
&\leq \left(\Lambda[\mathscr{K}(\pi^t)] - \Lambda[\mathscr{B}^t \circ \hat{M}^t]\right) + \left(J^{\pi^t}(X^t_0; c^t) - J^{\pi^t}\left(X^t_0; \hat{M}^t; c^t\right)\right) \\
&\leq \left(\left|\Lambda[\mathscr{K}(\pi^t)] - \Lambda[\mathscr{B}^t \circ \hat{M}^t]\right|\right) + \left(J^{\pi^t}(X^t_0; c^t) - J^{\pi^t}\left(X^t_0; \hat{M}^t; c^t\right)\right) \\
&\leq \underbrace{\min\left\{L \cdot \max\left\{\left\|\mathscr{B}^t\left(M^\star - \hat{M}^t\right)\right\|^2, \left\|\mathscr{B}^t\left(M^\star - \hat{M}^t\right)\right\|^\alpha\right\}, 2\Lambda_{\max}\right\}}_{\text{term}_1} \\
&\qquad + \underbrace{\left(J^{\pi^t}(X^t_0; c^t) - J^{\pi^t}\left(X^t_0; \hat{M}^t; c^t\right)\right)}_{\text{term}_2}. \qquad \text{(B.2.1)}
\end{aligned}$$

Here, the first inequality follows because we assumed $\mathcal{E}^t$ and because the algorithm selects $\hat{M}^t$ and π^t such that

$$\Lambda[\mathscr{B}^t \circ \hat{M}^t] + J^{\pi^t}\left(X^t_0; \hat{M}^t; c^t\right) \leq \Lambda[\mathscr{B}^t \circ M] + J^\pi\left(X^t_0; M; c^t\right)$$

for any $M \in \textsc{Ball}^t_M$ and for any $\pi \in \Pi$. The third inequality follows from Assumption 4.3.11.

Using Lemma B.2.3, we have

$$
\begin{aligned}
&\left\|\mathscr{B}^t\left(M^\star-\hat{M}^t\right)\right\| \le \left\|(\Sigma_{\mathscr{B}}^t)^{\frac{1}{2}}\left(M^\star-\hat{M}^t\right)\right\|\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|\\
&\le \sqrt{(1+C^{-1})}\left\|(\Sigma_{\mathscr{A}}^t)^{\frac{1}{2}}\left(M^\star-\hat{M}^t\right)\right\|\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|\\
&\le \sqrt{(1+C^{-1})}\left(\left\|(\Sigma_{\mathscr{A}}^t)^{\frac{1}{2}}\left(M^\star-\overline{M}^t\right)\right\|+\left\|(\Sigma_{\mathscr{A}}^t)^{\frac{1}{2}}\left(\overline{M}^t-\hat{M}^t\right)\right\|\right)\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|\\
&\le 2\sqrt{(1+C^{-1})\beta_M^t}\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\| \qquad (\because \mathcal{E}^t).
\end{aligned}
$$

Therefore, if $\mathcal{E}^t$, it follows that

$$
\text{term}_1 \le \min\left\{L\left\{4(1+C^{-1})\beta_M^t+1\right\}\max\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^2, \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^\alpha\right\}, 2\Lambda_{\max}\right\}. \tag{B.2.2}
$$

Then, we use the following lemma which is an extension of Lemma B.1.6 to our Hölder condition.

Lemma B.2.4. *For any sequence of $\mathscr{B}^t$ and for any $\alpha \in (0,1]$, we have*

$$
\sum_{t=0}^{T-1}\min\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^{2\alpha}, 1\right\} \le 2T^{1-\alpha}\left[1+\log\left(\frac{\det\left(\Sigma_{\mathscr{B}}^T\right)}{\det\left(\Sigma_{\mathscr{B}}^0\right)}\right)\right].
$$

Proof. Using $x \leq 2\log(1+x)$ for $x \in [0,1]$,

$$\begin{aligned}
&\sum_{t=0}^{T-1} \min\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^{2\alpha}, 1\right\} \leq \sum_{t=0}^{T-1} \min\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|_{\mathrm{HS}}^{2\alpha}, 1\right\} \quad (\because \|\mathscr{A}\| \leq \|\mathscr{A}\|_{\mathrm{HS}}) \\
&= \sum_{t=0}^{T-1} \left(\min\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|_{\mathrm{HS}}^{2}, 1\right\}\right)^{\alpha} \leq \sum_{t=0}^{T-1} \left[2\log\left(1 + \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|_{\mathrm{HS}}^{2}\right)\right]^{\alpha} \\
&= \sum_{t=0}^{T-1} \left[2\log\left(1 + \mathrm{tr}\Big\{(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\mathscr{B}^{t\dagger}\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\Big\}\right)\right]^{\alpha} \\
&\leq 2^{\alpha}\sum_{t=0}^{T-1} \left[\log\det\left(I + (\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\mathscr{B}^{t\dagger}\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right)\right]^{\alpha} \\
&\leq 2^{\alpha}T^{1-\alpha} \left[\sum_{t=0}^{T-1} \log\det\left(I + (\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\mathscr{B}^{t\dagger}\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right)\right]^{\alpha} \\
&\leq 2^{\alpha}T^{1-\alpha} \left[\sum_{t=0}^{T-1} \left(\log\det\left(\Sigma_{\mathscr{B}}^{t+1}\right) - \log\det\left(\Sigma_{\mathscr{B}}^{t}\right)\right)\right]^{\alpha} \leq 2T^{1-\alpha}\left[\log\left(\frac{\det\left(\Sigma_{\mathscr{B}}^T\right)}{\det\left(\Sigma_{\mathscr{B}}^0\right)}\right)\right]^{\alpha} \\
&\leq 2T^{1-\alpha}\left(1 + \log\left(\frac{\det\left(\Sigma_{\mathscr{B}}^T\right)}{\det\left(\Sigma_{\mathscr{B}}^0\right)}\right)\right).
\end{aligned}$$

□

Here, the fifth inequality follows from [91, Exercise 1.1.4].

Corollary B.2.5. *For any sequence of $\mathscr{B}^t$ and for any $\alpha \in (0,1]$, we have*

$$\sum_{t=0}^{T-1} \min\left\{\max\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^{4}, \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^{2\alpha}\right\}, 1\right\} \leq 2T^{1-\alpha}\left[1 + \log\left(\frac{\det\left(\Sigma_{\mathscr{B}}^T\right)}{\det\left(\Sigma_{\mathscr{B}}^0\right)}\right)\right].$$

From (B.2.2) and Corollary B.2.5, we obtain

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{t=0}^{T-1} \mathrm{term}_1 \middle| \bigcap_{t=0}^{T-1} \mathcal{E}^t\right] \\
&\leq \mathbb{E}\left[\sum_{t=0}^{T-1} \min\left\{L\left\{4(1+C^{-1})\beta_M^t+1\right\}\max\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^2, \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^\alpha\right\}, 2\Lambda_{\max}\right\}\right] \\
&\leq \mathbb{E}\left[\sum_{t=0}^{T-1} \left\{L\left\{4(1+C^{-1})\beta_M^t+1\right\}+2\Lambda_{\max}\right\}\min\left\{\max\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^2, \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^\alpha\right\}, 1\right\}\right] \\
&\leq \sum_{t=0}^{T-1} \sqrt{\mathbb{E}\left[\,[L\left\{4(1+C^{-1})\beta_M^t+1\right\}+2\Lambda_{\max}]^2\right]}\sqrt{\mathbb{E}\left[\min\left\{\max\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^4, \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^{2\alpha}\right\}, 1\right\}\right]} \\
&\leq \sqrt{\sum_{t=0}^{T} \mathbb{E}\left[\,[L\left\{4(1+C^{-1})\beta_M^t+1\right\}+2\Lambda_{\max}]^2\right]} \\
&\quad \cdot \sqrt{\mathbb{E}\left[\sum_{t=0}^{T-1}\min\left\{\max\left\{\left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^4, \left\|\mathscr{B}^t(\Sigma_{\mathscr{B}}^t)^{-\frac{1}{2}}\right\|^{2\alpha}\right\}, 1\right\}\right]} \\
&\leq \sqrt{T\cdot \mathtt{value}}\sqrt{T^{1-\alpha}(2+\gamma_{T,\mathscr{B}}(\lambda))}. \qquad \text{(B.2.3)}
\end{aligned}
$$

Here, the first inequality is due to (B.2.2); the third inequality uses $\mathbb{E}[ab] \leq \sqrt{\mathbb{E}[a^2]\mathbb{E}[b^2]}$; the forth inequality uses the Cauchy-Schwartz inequality; the last is from Corollary B.2.5. Also,

$$
\begin{aligned}
\mathtt{value} &:= 16L^2(1+C^{-1})^2\mathbb{E}[(\beta_M^T)^2] + (8L^2+16\Lambda_{\max}L)(1+C^{-1})\mathbb{E}[\beta_M^T] + 4\Lambda_{\max}L + L^2 + 4\Lambda_{\max}^2 \\
&\leq C'\left\{L^2(1+C^{-1})^2\mathbb{E}[(\beta_M^T)^2] + (L^2+\Lambda_{\max}L)(1+C^{-1})\mathbb{E}[\beta_M^T] + \Lambda_{\max}^2 + L^2\right\},
\end{aligned}
$$

for some constant C'.

Next, we turn to bound the latter term term_2 of (B.2.1). Simple calculations show that

$$
M^\star - \overline{M}^t = \lambda(\Sigma_{\mathscr{A}}^t)^{-1}M^\star - (\Sigma_{\mathscr{A}}^t)^{-1}\sum_{\tau=0}^{t-1}\sum_{n=0}^{N^\tau-1}\sum_{h=0}^{H_n^\tau-1} {\mathscr{A}_{h,n}^\tau}^\dagger \epsilon_{h,n}^\tau,
$$

where $\epsilon_{h,n}^\tau$ is the sampled noise at τ-th episode, h-th time step, and n-th trajectory. Now, by introducing a Hilbert space containing an operator of $\mathcal{L}(\mathcal{L}(\mathcal{H};\mathcal{H}');\mathbb{R})$, which is a Hilbert-Schmidt operator, because of Assumption 4.3.4, we can apply Lemma C.3.3 to our problem too. Therefore, with probability at least $1-\delta_t$, it holds that

$$\left\|(\Sigma^t_{\mathscr{A}})^{\frac{1}{2}}\left(M-\overline{M}^t\right)\right\|^2 \le \lambda\|M^\star\|^2+\sigma^2(8d_\phi\log(5)+8\log(\det(\Sigma^t_{\mathscr{A}})\det(\Sigma^0_{\mathscr{A}})^{-1})/\delta_t),$$

and properly choosing δ_t leads to β^t_M defined in Section B.2.1, and we obtain the result

$$\Pr\left(\bigcup_{t=0}^{T-1}\overline{\mathcal{E}^t}\right)\le\frac{1}{2}.$$

In our algorithm, transition data are chosen from any initial states and the horizon lengths vary; however, slight modification of the analysis of LC^3 will give the following lemma.

Lemma B.2.6 (Modified version of Theorem 4.2.4)**.** *Suppose Assumptions 4.3.4, 4.3.6, 4.3.9, 4.3.13, and 4.3.15 hold. Then, the term* term$_2$ *is bounded by*

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=0}^{T-1}\mathrm{term}_2\middle|\bigcap_{t=0}^{T-1}\mathcal{E}^t\right]\\&\le\sqrt{HV_{\max}}\sqrt{64T(d_\phi+\log(T)+\gamma_{T,\mathscr{A}}(\lambda)+H)}\sqrt{\gamma_{T,\mathscr{A}}(\lambda)}.\end{aligned}$$

Combining all of the above results, we prove Theorem 4.3.16:

Proof of Theorem 4.3.16. Using (B.2.3) (which requires Assumptions 4.3.4, 4.3.6, 4.3.11, and 4.3.13), Lemma B.2.6 (which requires Assumptions 4.3.4, 4.3.6, 4.3.9, 4.3.13, and 4.3.15), and $\Pr\left(\bigcup_{t=0}^{T-1}\overline{\mathcal{E}^t}\right)\le\frac{1}{2}$, it follows that

$$\begin{aligned}&\mathbb{E}_{\mathrm{KS-LC^3}}\left[\textsc{Regret}_T\right]\\&\le\sqrt{T\left\{C'\left\{L^2(1+C^{-1})^2\mathbb{E}[(\beta^T_M)^2]+(L^2+\Lambda_{\max}L)(1+C^{-1})\mathbb{E}[\beta^T_M]+\Lambda^2_{\max}+L^2\right\}\right\}}\sqrt{T^{1-\alpha}(2+\gamma_{T,\mathscr{B}}(\lambda))}\\&+\sqrt{HV_{\max}}\sqrt{64T(d_\phi+\log(T)+\gamma_{T,\mathscr{A}}(\lambda)+H)}\sqrt{\gamma_{T,\mathscr{A}}(\lambda)}+\frac{1}{2}\cdot(\Lambda_{\max}+\sqrt{V_{\max}})\\&\le C_1T^{1-\frac{\alpha}{2}}(\tilde{d}_{T,1}+\tilde{d}_{T,2}),\end{aligned}$$

for some absolute constant C_1. Therefore, the theorem is proved. □

B.3 Separations of signature control from classical approach

This section discusses the separations of signature control presented in Chapter 5 from classical Bellman based approach. One may think that one can augment the state with signatures and give reward at the very end of the episode to encode the value over the entire trajectory within the classical Bellman based framework. There are obvious drawbacks for this approach; (1) for the infinite horizon case where the terminal state or time is unavailable, one cannot give any reward, and (2) input dimension for the value function becomes very large with signature augmentation. Here, in addition to the above, we show separations from the classical Bellman based approach from several point of views. Let $\mathcal{S}^{\pi}(a, y, t)$ $(\mathcal{ES}(y, t))$ be the S-function (expected S-function) and $Q^{\pi}(a, y, s, t)$ $(V^{\pi}(y, s, t))$ be the Q-function (value function) where s represents the signature of the past path.

B.3.1 Cost and expectation order

If the cost c is linear (e.g., the case of reduction to the Bellman equation), then the cost to be mimimized can be reformulated as

$$c\left(\mathbb{E}_{\Omega}\left[S_m\left(\sigma^{\mathcal{T}}_{\pi,F}\left(y_0, T, \omega\right)\right)\right]\right) = \mathbb{E}_{\Omega}\left[c\left(S_m\left(\sigma^{\mathcal{T}}_{\pi,F}\left(y_0, T, \omega\right)\right)\right)\right].$$

However, in general, the order is not exchangable. We saw that Chen equation reduces to the Bellman equation and therefore for any MDP over the state augmented by signatures (and horizon T), it is easy to see that there exists an interpolation, a transpotation, and a cost c such that

$$c\left(\mathcal{ES}^{\pi}_{m}\left(y_0, 0\right)\right) = V^{\pi}(y_0, \mathbf{1}, 0).$$

On the other hand, the opposite does not hold in general.

Claim B.3.1. *There exist a 3-tuple $(\mathcal{X}, \mathcal{A}, P_a)$ where P_a is the transition kernel for action $a \in \mathcal{A}$, a set of randomized policies ($\pi(a|x)$ is the probability of taking action $a \in \mathcal{A}$ at $x \in \mathcal{X}$*

under the policy π*)* Π*, an initial state* y_0*, and the cost function* c *of signature control, such that there is no immediate reward function* r *that satisfies*

$$\arg\min_{\pi\in\Pi} c\left(\mathcal{ES}_m^{\pi}\left(y_0, 0\right)\right) = \arg\min_{\pi\in\Pi} V^{\pi}(y_0, \mathbf{1}, 0).$$

Proof. Let $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, $\mathcal{A} = \{a_{-1}, a_1\}$, $\mathbb{T} = \mathbb{N}$, $y_0 = 0$, $T = 1$ and

$$P_{a_{-1}}(0, -1) = 1, \ P_{a_1}(0, 1) = 1.$$

Also, let $\Pi = \{\pi_1, \pi_2, \pi_3\}$ where

$$\pi_1(a_1|0) = 1, \ \pi_2(a_{-1}|0) = 1, \ \pi_3(a_1|0) = 0.5, \ \pi_3(a_{-1}|0) = 0.5,$$

and let $c : T^1(\mathcal{X}) \to \mathbb{R}_{\geq 0}$ be

$$c(s) = |s_1|.$$

The optimal policy for signature control is then π_3, i.e.,

$$\{\pi_3\} = \arg\min_{\pi\in\Pi} c\left(\mathcal{ES}_m^{\pi}\left(y_0, 0\right)\right).$$

Now, because we have

$$\begin{aligned}
V^{\pi_1}(y_0, \mathbf{1}, 0) &= \mathbb{E}_{\Omega_{a_{-1}}}\left[r(a_{-1}, y_0, \mathbf{1}, \omega)\right], \\
V^{\pi_2}(y_0, \mathbf{1}, 0) &= \mathbb{E}_{\Omega_{a_1}}\left[r(a_1, y_0, \mathbf{1}, \omega)\right], \\
V^{\pi_3}(y_0, \mathbf{1}, 0) &= \frac{\mathbb{E}_{\Omega_{a_{-1}}}\left[r(a_{-1}, y_0, \mathbf{1}, \omega)\right] + \mathbb{E}_{\Omega_{a_1}}\left[r(a_1, y_0, \mathbf{1}, \omega)\right]}{2},
\end{aligned}$$

possible immediate rewards to care about are only $\mathbb{E}_{\Omega_{a_{-1}}}\left[r(a_{-1}, y_0, \mathbf{1}, \omega)\right]$ and $\mathbb{E}_{\Omega_{a_1}}\left[r(a_1, y_0, \mathbf{1}, \omega)\right]$. It is straightforward to see that

$$\pi_3 \in \arg\min_{\pi\in\Pi} V^{\pi}(y_0, \mathbf{1}, 0)$$

only if

$$\mathbb{E}_{\Omega_{a_{-1}}}\left[r(a_{-1}, y_0, \mathbf{1}, \omega)\right] = \mathbb{E}_{\Omega_{a_1}}\left[r(a_1, y_0, \mathbf{1}, \omega)\right].$$

However, for any reward function satisfying this equation we obtain

$$\{\pi_3\} \neq \arg\min_{\pi\in\Pi} V^{\pi}(y_0, \mathbf{1}, 0).$$

□

B.3.2 Sample complexity

We considered stochastic policy above. What if the dynamics is deterministic (Ω is a singleton)? For a deterministic finite horizon case, technically, the cost over path can be represented by both the cost function with S-function and Q-function. The difference is the *steps* or sample complexity required to find an optimal path. Because S-function captures strictly more information than Q-function, it should show sample efficiency in certain problems even for deterministic cases. Here, in particular, we show that there exists a signature control problem which is more efficiently solved by the use of S-function than by Q-function. (We do not discuss typical lower bound arguments of RL sample complexity; giving certain convergence guarantees with lower bound arguments is an important future work.)

To this end, we define Signature MDP:

Definition B.3.1 (Finite horizon, time-dependent signature MDP)**.** Finite horizon, time-dependent signature MDP is the 8-tuple $(\mathcal{X}, \mathcal{A}, m, \{P\}_t, F, \{r\}_t, T, \mu)$ which consists of

- finite or infinite state space $\mathcal{X}$
- discrete or infinite action space $\mathcal{A}$
- signature depth $m \in \mathbb{Z}_{>0}$
- transition kernel $P_{a,t}$ on $\mathcal{X} \times \mathcal{X}$ for action $a \in \mathcal{A}$ and time $t \in [T]$
- signature is updated through concatenation of past path and the immediate path which is the interpolation of the current state and the next state by F
- reward r_t which is a time-dependent mapping from $\mathcal{X} \times T^m(\mathcal{X}) \times \mathcal{A}$ to $\mathbb{R}$ for time $t \in [T]$
- positive integer $T \in \mathbb{Z}_{>0}$ defining time horizon

- initial state distribution μ

Further, we call an algorithm Q-table ($\mathcal{S}$-table) based if it accesses state x exclusively through Q-table ($\mathcal{S}$-table) for all $x \in \mathcal{X}$. Now, we obtain the following claim.

Claim B.3.2. *There exists a finite horizon, time-dependent signature MDP with a set of deterministic policies* Π *and with a* known *reward* $\{r\}_t$ *such that the number of samples (trajectories) required in the worst case to determine an optimal policy is strictly larger for any Q-table based algorithm than a $\mathcal{S}$-table based algorithm.*

Proof. Let the first MDP $\mathcal{M}_1$ be given by $T = 3$, $\mathcal{X} = \mathcal{Y} := \{[0,0],[1,1],[2,2],[2,3],[-1,1],[0,1],[4,0]\} \subset \mathbb{R}^2$, $\mathcal{A} := \{a_1, a_2\}$, $m = 2$, F is linear interpolation of any pair of points, $\mu([0,0]) = \Pr[y_0 = [0,0]] = 1$, and

$$\begin{aligned} &P_{a_1,0}([0,0],[1,1]) = P_{a_1,1}([1,1],[2,2]) = P_{a_2,1}([1,1],[2,2]) = P_{a_1,2}([2,2],[2,3]) = \\ &= P_{a_2,0}([0,0],[-1,1]) = P_{a_1,1}([-1,1],[2,2]) = P_{a_2,1}([-1,1],[2,2]) = P_{a_2,2}([2,2],[4,0]) = 1 \end{aligned}$$

Also, let $\{r\}_t$ satisfy that

$$\forall t \in [T-1]: \; r_t = 0, \qquad r_{T-1}(x,s,a) = |s^+_{1,2}|,$$

where s^+ is the signature of entire path that is deterministically obtained from state x at time $T-1$, past path signature s, and action a (note we do not know the transition but only the output $|s^+_{1,2}|$). The possible deterministic trajectories (or policies) of state-action pairs are the followings:

$$(([0,0],a_1),([1,1],a_1),([2,2],a_1),([2,3]))$$
$$(([0,0],a_1),([1,1],a_2),([2,2],a_1),([2,3]))$$
$$(([0,0],a_1),([1,1],a_1),([2,2],a_2),([4,0]))$$
$$(([0,0],a_1),([1,1],a_2),([2,2],a_2),([4,0]))$$
$$(([0,0],a_2),([-1,1],a_1),([2,2],a_1),([2,3]))$$
$$(([0,0],a_2),([-1,1],a_2),([2,2],a_1),([2,3]))$$
$$(([0,0],a_2),([-1,1],a_1),([2,2],a_2),([4,0]))$$
$$(([0,0],a_2),([-1,1],a_2),([2,2],a_2),([4,0]))\,.$$

The optimal trajectories are the last two. Let the second MDP $\mathcal{M}_2$ be the same as $\mathcal{M}_1$ except that

$$P_{a_2,2}([2,2],[2,3]) = 1.$$

Suppose we obtain Q-table for the first six trajectories of $\mathcal{M}_1$. At this point, we cannot distinguish $\mathcal{M}_1$ and $\mathcal{M}_2$ exclusively from the Q-table; hence at least one more trajectory sample is required to determine the optimal policy for any Q-table based algorithm. On the other hand, suppose we obtain $\mathcal{S}$-table for the first six trajectories of $\mathcal{M}_1$. Then, the S-table at state $x = [2,2]$ with depth $m \geq 1$ determines the transition at $x = [2,2]$ for both actions; hence, we know the cost of the the last two trajectories without executing it. □

B.4 Proof of Proposition 6.3.2

Proof.

$$B := L_{\max} \sup_{t>0} |a_t|,$$
$$C := \sqrt{4R^2 \log(4/\delta)},$$
$$D := L_{\max}^2 \mathcal{C}_0 \left(\mu + \sup_{t\geq 1} |a_t|\right).$$

Then, $\sqrt{T} > 0$ satisfies the following inequality

$$\sqrt{T} \geq \max\left\{\frac{C + \sqrt{C^2 + 4(\varepsilon - \mu)B}}{2(\varepsilon - \mu)}, \frac{C + \sqrt{C^2 + 4(\varepsilon - \mu)D}}{2(\varepsilon - \mu)}\right\}$$

if and only if

$$\begin{aligned}
\varepsilon &\leq 2\varepsilon - \mu - \frac{L_{\max}\sup_{t>0}|a_t|}{T} - \sqrt{\frac{4R^2\log(4/\delta)}{T}}, \text{ and} \\
\varepsilon &\geq \mu + \frac{L_{\max}^2\mathcal{C}_0(\mu + \sup_{t\geq 1}|a_t|)}{T} + \sqrt{\frac{4R^2\log(4/\delta)}{T}}.
\end{aligned}$$

Since $L_{\max} \geq 2$, $D \geq B$, $\mu < \gamma\sup_{t\geq 1}|a_t|$, $\mathcal{C}_0(1+\gamma) < 3$, and

$$\begin{aligned}
\varepsilon - \mu &= \frac{\sigma_0(1-\lambda)}{\sqrt{L_{\max}}} \cdot \frac{\sqrt{L_{\max}}}{1 + \sqrt{4L_{\max} + 1}} \\
&\geq \frac{\sigma_0(1-\lambda)}{2\sqrt{2}\sqrt{L_{\max}}},
\end{aligned}$$

$$\begin{aligned}
&\max\left\{\frac{C + \sqrt{C^2 + 4(\varepsilon - \mu)B}}{2(\varepsilon - \mu)}, \frac{C + \sqrt{C^2 + 4(\varepsilon - \mu)D}}{2(\varepsilon - \mu)}\right\} \\
&= \frac{C + \sqrt{C^2 + 4(\varepsilon - \mu)D}}{2(\varepsilon - \mu)} \\
&\leq 2\sqrt{\frac{L_{\max}C^2}{4(\varepsilon - \mu)^2} + \frac{D}{(\varepsilon - \mu)}} \\
&\leq 2\sqrt{\frac{2L_{\max}^2C^2}{\sigma_0^2(\xi - \lambda)^2} + \frac{6\sqrt{2}L_{\max}^{5/2}\sup_{t\geq 1}|a_t|}{\sigma_0(\xi - \lambda)}} \\
&\leq 2\sqrt{\frac{2L_{\max}^2C^2}{\sigma_0^2(\xi - \lambda)^2} + \frac{9L_{\max}^{5/2}\sup_{t\geq 1}|a_t|}{\sigma_0(\xi - \lambda)}},
\end{aligned}$$

where we used $\sqrt{1 - 2\gamma} = 2\gamma\sqrt{L_{\max}}$. Since $2\varepsilon \leq \sqrt{(\sigma_0^2 - 2\mu\sigma_0)/L_{\max}}$, the statement follows from Lemma 6.3.1. □

B.5 Proof of Theorem 6.3.3

Proof. Let $\mathcal{K}$ be the set of all combinations (m, β, s, q) such that $m \in \{1, \ldots, d\}$, $\beta \in \{1, \ldots, L_{\max}\}$, $s \in [\beta - 1]$ and $\{q = \alpha/\ell : \alpha \in \{1, \ldots, \ell - 1\}\}$. We remark that

$$\begin{aligned}|\mathcal{K}| &\leq \sum_{m=1}^{d} \sum_{\beta=1}^{L_{\max}} \sum_{\ell \leq L_{\max}/\beta} \sum_{\alpha=1}^{\ell-1} \sum_{s=0}^{\beta-1} 1 = d \sum_{\beta=1}^{L_{\max}} \sum_{\ell \leq L_{\max}/\beta} \beta(\ell - 1) \\ &\leq d \sum_{\beta=1}^{L_{\max}} \frac{L_{\max}}{2} \left(\frac{L_{\max}}{\beta} - 1 \right) \\ &\leq \frac{d L_{\max}^2 \log L_{\max}}{2}.\end{aligned}$$

Also, let $\mathcal{E}_{\kappa}^{T_p}$ be the event such that, for the combination $\kappa \in \mathcal{K}$,

$$\left| \frac{1}{\lfloor T_p/\beta \rfloor} \sum_{j=0}^{\lfloor T_p/\beta \rfloor - 1} \left\{ \eta_{t_0 + \beta j + s} e^{\frac{i 2\pi \alpha j}{\ell}} \right\} \right| \leq \sqrt{\frac{4R^2 \log\left(4 d L_{\max}^2 \log L_{\max}/\delta\right)}{\lfloor T_p/\beta \rfloor}}.$$

Because the error sequence satisfies conditionally R-sub-Gaussian, using Lemma C.4.1 and from the fact that any subsequence of the filtration $\{\mathscr{F}_\tau\}$ is again a filtration, we obtain, for each $\kappa \in \mathcal{K}$,

$$\Pr\left[\mathcal{E}_{\kappa}^{T_p}\right] \geq 1 - \frac{\delta}{d L_{\max}^2 \log L_{\max}}.$$

Define $\mathcal{E}^{T_p} := \bigcap_{\kappa \in \mathcal{K}} \mathcal{E}_{\kappa}^{T_p}$. Then, it follows from the Fréchet inequality that

$$\begin{aligned}\Pr\left[\mathcal{E}^{T_p}\right] = \Pr\left[\bigcap_{\kappa \in \mathcal{K}} \mathcal{E}_{\kappa}^{T_p} \right] &\geq |\mathcal{K}| \left(1 - \frac{\delta}{d L_{\max}^2 \log L_{\max}} \right) - (|\mathcal{K}| - 1) \\ &= 1 - \frac{\delta |\mathcal{K}|}{d L_{\max}^2 \log L_{\max}} \geq 1 - \delta.\end{aligned}$$

Let $\widehat{L}$ be the output of Algorithm 4. We show that, with probability $1 - \delta$, $\widehat{L}$ is $(\rho, \sqrt{d})$-anp of L. In fact, suppose $\widehat{L}$ is not $(\rho, \sqrt{d})$-anp. We note that $\widehat{L} < L$. There exists $s \in [\widehat{L}]$ and $t_1, t_2 \in \mathbb{Z}_{>0}$,

$$\| f^{s + \widehat{L} t_1}(\theta) - f^{s + \widehat{L} t_2}(\theta) \|_{\mathbb{R}^d} > \rho + 2\sqrt{d}\mu.$$

Let $m' \in \operatorname{argmax}_{m=1,\dots,d} \left| f^{s+\widehat{L}t_1}(\theta)^\top \mathbf{u}_m - f^{s+\widehat{L}t_2}(\theta)^\top \mathbf{u}_m \right|$. Put $a_t := f^{s+\widehat{L}t}(\theta)^\top \mathbf{u}_{m'}$. Then, for any $t, t' \in \mathbb{Z}_{>0}$, we have

$$\begin{aligned} |a_t - a_{t'}| &\geq |a_{t_1} - a_{t_2}| - 2\mu \\ &\geq \frac{\| f^{s+\widehat{L}t_1}(\theta) - f^{s+\widehat{L}t_2}(\theta)\|_{\mathbb{R}^d}}{\sqrt{d}} - 2\mu \\ &> \frac{\rho}{\sqrt{d}}. \end{aligned}$$

Thus, by definition, we have

$$\sigma_{L/\widehat{L}}((a_t)_t) \geq \frac{\rho}{2\sqrt{dL/\widehat{L}}} \geq \frac{\rho}{2\sqrt{dL_{\max}}}.$$

Let $\sigma_0 = \rho/2\sqrt{dL_{\max}}$ and $\xi := \gamma^{-1}/3\sqrt{L_{\max}}$. Then, $\mu/(\gamma\xi) < \sigma_0 \leq \sigma_{L/\widehat{L}}((a_t)_t)$, and

$$\begin{aligned} &\frac{8L_{\max}R^2\log(4/\delta)}{\sigma_0^2(\xi-\lambda)^2} + \frac{36L_{\max}^{5/2}\sup_{t\geq 1}|a_t|}{\sigma_0(\xi-\lambda)} \\ &\leq \frac{32\xi^{-2}dL_{\max}^2R^2\log(4/\delta)}{\rho^2(1-r)^2} + \frac{72\xi^{-1}\sqrt{d}L_{\max}^3\sup_{t\geq 1}|a_t|}{\rho(1-r)} \\ &\leq \frac{72dL_{\max}^2R^2\log(4/\delta)}{\rho^2(1-r)^2} + \frac{108\sqrt{d}L_{\max}^3\sup_{t\geq 1}|a_t|}{\rho(1-r)}. \end{aligned}$$

The last inequality follows from $\xi \geq 2/3$. Thus, by Proposition 6.3.2, the algorithm finds $\beta > \widehat{L}$ in m'-th loop, and the output becomes an integer larger than $\widehat{L}$, which is contradiction. □

B.6 Proof of Theorem 6.3.9

Here, we provide the proof of Theorem 6.3.9. Let

$$\begin{aligned} K &:= [M\theta, \dots, M^d\theta] : \mathbb{C}^d \to W, \\ E_s(N) &:= \mathscr{W}\left((E_{s,j})_{j=0}^{N-1}\right) : \mathbb{C}^d \to \mathbb{C}^d. \end{aligned}$$

For a linear map $\mathcal{M} : W \to W$, we define linear maps:

$$X(\mathcal{M}) := \begin{bmatrix} \tilde{x}_1^\top \mathcal{M}^1 \\ \tilde{x}_2^\top \mathcal{M}^{2d+1} \\ \vdots \\ \tilde{x}_d^\top \mathcal{M}^{2(d-1)d+1} \end{bmatrix} : W \to \mathbb{C}^d,$$

$$Q_N(\mathcal{M}) := \mathscr{W}\left((\mathcal{M}^{2d^2 j})_{j=0}^{N-1}\right) : W \to W.$$

For $s = 0, 1$, we define linear maps on $\mathbb{C}^d$ by

$$A_s(N; \mathcal{M}) := X(\mathcal{M}) Q_N(\mathcal{M}) \mathcal{M}^{sd+N-1} K + E_s(N),$$
$$B_s(N; \mathcal{M}) := X(\mathcal{M}) Q_N(\mathcal{M}) \mathcal{M}^{sd+N-1} K.$$

We note that $A_s(N; M_\theta)$ is identical to $A_s(N)$ defined in (6.3.7). We impose the following assumption on $X(M_\theta)$:

Assumption B.6.1. *The kernel of the linear map $X(M_1)$ is the same as $\mathcal{N}(M_1)$.*

Note that this assumption holds with probability 1 if we randomly choose $\tilde{x}_1, \ldots, \tilde{x}_d$ (see Lemma C.1.3).

The following lemma provides an explicit description of $B_0(N; M_1)^+$.

Lemma B.6.2. *Suppose Assumption B.6.1 holds. Assume $N \geq 16L^2$. Let*

$$U_1 := \mathscr{I}(B_0(N; M_1)) \subset \mathbb{C}^d,$$
$$U_2 := \mathscr{N}(B_0(N; M_1)) \subset \mathbb{C}^d,$$
$$U_3 := \mathscr{I}(M_1) = W_{=1} \subset W.$$

Let $i : U_1 \to \mathbb{C}^d$ be the inclusion map and $p : \mathbb{C}^d \to U_2^\perp$ the orthogonal projection. Then, restriction of $X(M_1)$ (resp. $M_1 K$) to U_3 (resp. $U_2^\perp$) induces an isomorphism onto U_1 (resp. U_3). If we denote the isomorphism by $\tilde{X}$ (resp, $\tilde{K}$). Then, $B_0(N; M_1)^+$ is given by

$$B_0(N; M_1)^+ = p^\dagger \tilde{K}^{-1} M_1|_{U_3}^{2-N} Q_N(M_1)|_{U_3}^{-1} \tilde{X}^{-1} i^\dagger.$$

Proof. Since we have $\mathscr{N}(M_1) = \mathscr{N}(M_1^{r+1})$ for all $r \geq 0$ and Assumption B.6.1, surjectivity of K by Proposition C.1.2, and bijectivity of $Q_N(M_1)$ on U_3 by Proposition 6.3.8, we have

$$U_1 = \mathscr{I}(X(M_1)|_{U_3}),$$
$$U_2 = \mathscr{N}(M_1 K).$$

Thus, the restriction of $X(M_1)$ (resp. K) to U_3 (resp. $U_2^\perp$) induces an isomorphism onto U_1 (resp. U_3). Let us denote the isomorphism by $\tilde{X}$ (resp. $\tilde{K}$). Then, the last statement follows from Proposition C.1.1. □

Lemma B.6.3. *Assume $N \geq 16L^2$. Let*

$$A := B_1(N; M_1) B_0(N; M_1)^+.$$

Then, A is independent of N and its eigenvalues are zeros except for (θ, d)-distinct eigenvalues of M.

Proof. We use the notation as in Lemma B.6.2. By Lemma B.6.2, we obtain

$$B_1(N; M_1) B_0(N; M_1)^+ = i \tilde{X} M_1^d \tilde{X}^{-1} i^\dagger,$$

which is independent of N and its eigenvalues are zeros except for (θ, d)-distinct eigenvalues of M. □

The following result will be used in the proof of Theorem 6.3.9 (but not essential).

Lemma B.6.4. *We have*

$$\|X(M_1)\| \leq \kappa \sqrt{d},$$
$$\|X(M_{<1})\| \leq \kappa (d+1) 2^d.$$

Proof. Considering the Jordan normal form, the first inequality is obvious. As for the second inequality, we define $J \in \mathbb{R}^{d \times d}$ by the nilpotent matrix

$$J := \begin{bmatrix} 0 & 1 & & O \\ & \ddots & \ddots & \\ & & & 1 \\ O & & & 0 \end{bmatrix}.$$

Then, we have

$$\begin{aligned}
\kappa^{-1}\|X(M_1)\| &\leq \sum_{r=1}^{d} \|(I+J)\|_1 \\
&= \sum_{r=1}^{d}\sum_{j=0}^{d} \binom{r}{j}(d-j) \\
&= \sum_{j=0}^{d}(d-j)\sum_{r=1}^{d}\binom{r}{j} \\
&= d + \sum_{j=1}^{d}(d-j)\sum_{r=1}^{d}\binom{r}{j} \\
&= d + \sum_{j=1}^{d}(d-j)\binom{r+1}{j+1} \\
&= d + (d+1)2^d - (d+1)d - (d+1) \\
&< (d+1)2^d.
\end{aligned}$$

□

Proof of Theorem 6.3.9. Let A be the matrix introduced in Lemma B.6.3. Let

$$\gamma(N) := \frac{\sqrt{4d^2R^2\log(4d^2/\delta)} + 1}{\sqrt{N}}.$$

Let M_1 and $M_{<1}$ be matrices as in Corollary 6.2.3. Then, by Proposition 6.2.2, we see that

$$\begin{aligned}
A_s(N,M) &= A_s(N,M_1) + A_s(N,M_{<1}), \\
B_s(N,M) &= B_s(N,M_1) + B_s(N,M_{<1}).
\end{aligned}$$

We denote by $\hat{A}_s(N;M)$ (resp, $\hat{B}_s(N;M)$) the low rank approximation via the singular value threshold $\gamma(N)$ (see Definition 6.3.4). By direct computations, we have

$$\begin{aligned}
&\|A - A_1(N;M)\hat{A}_0(N;M)^+\| \\
&\leq \|A_1(N;M)\|\cdot\|\hat{A}_0(N;M)^+ - B_0(N;M_1)^+\| + \|A_1(N;M) - B_1(N;M_1)\|\cdot\|B_0(N;M_1)^+\| \\
&\leq (\|B_1(N;M_1)\| + \|B_1(N;M_{<1})\| + \|E_1(N)\|)\cdot\|\hat{A}_0(N;M)^+ - B_0(N;M_1)^+\| \\
&\qquad + \|B_1(N;M_{<1}) + E_1(N)\|\cdot\|B_0(N;M_1)^+\|.
\end{aligned}$$

By Proposition 6.3.8, for $s = 0, 1$ and for $N \geq \max\{2d, 16L^2\}$, we have

$$\begin{aligned}
\|B_s(N; M_1)\| &\leq \|X(M_1)\| \cdot \|M_1^{N+sd-1} Q_N(M_1)\| \cdot \|K\| \\
&\leq \kappa C_{\mathrm{ws}}(L) \|X(M_1)\| \|K\| \\
&\leq Bd\kappa^2 C_{\mathrm{ws}}(L) \\
\|B_s(N; M_{<1})\| &\leq \|X(M_{<1})\| \cdot \|M_{<1}^{N+sd-1} Q_N(M_{<1})\| \cdot \|K\| \\
&\leq \|X(M_{<1})\| \cdot \|K\| \cdot \frac{d^2 \kappa e^{-\Delta(N+sd-d)}}{N\Delta^{d-1}} \\
&\leq B\kappa\sqrt{d}(d+1)2^d \cdot \frac{d^2 \kappa e^{-\Delta(N+sd-d)}}{N\Delta^{d-1}} \\
&\leq \frac{B\kappa^2 e^{d+6-\Delta(N+sd-d)}}{N\Delta^{d-1}},
\end{aligned}$$

where we used $\|K\| \leq \sqrt{d}B$ (Assumption 6.2.6), and $\|X(M_{<1})\| \leq \kappa(d+1)2^d$ (Lemma B.6.4), and $\sqrt{d}(d+1)d^2 2^d < e^{d+6}$. By Lemma B.6.2 with Proposition 6.3.8, we see that

$$\|B_0(N; M_1)^+\| \leq \kappa C_{\mathrm{ws}}(L) \|\tilde{X}^{-1}\| \cdot \|\tilde{K}^{-1}\|.$$

By using Lemma C.4.1 and union bounds, we obtain

$$\max(\|E_0(N)\|_{\mathrm{HS}}, \|E_1(N)\|_{\mathrm{HS}}) \leq \gamma(N) - \frac{1}{\sqrt{N}},$$

with probability at least $1 - \delta$. Assume that

$$N \geq \frac{-(d-1)\log \Delta}{\Delta} + \frac{\log(B\kappa^2) + d + 6}{\Delta} + d.$$

Then, we see that $\|B_s(N; M_{<1})\| \leq 1/N$. Thus, by Lemma C.3.5, with probability at least $1 - \delta$, we have

$$\begin{aligned}
\|\hat{A}_0(N; M)^+ - B_0(N; M_1)^+\| &\leq 8(\|E_0(N)\| + \|B_0(N; M_{<1})\|) \cdot \|B_0(N, M_1)^+\|^2(\sqrt{d} + 1) \\
&\leq 8\gamma(N) \cdot \|B_0(N, M_1)^+\|^2(\sqrt{d} + 1) \\
&\leq 8(1 + \sqrt{d})\kappa^2 C_{\mathrm{ws}}(L)^2 \|\tilde{X}^{-1}\|^2 \cdot \|\tilde{K}^{-1}\|^2 \gamma(N).
\end{aligned}$$

Therefore, there exists $C > 0$ depending on $\|X(M)\|$, $\|K\|$, κ, $C_{\mathrm{ws}}(L)$, $\|\tilde{X}^{-1}\|$, $\|\tilde{K}^{-1}\|$, and d such that

$$\|A - A_1(N;M)\hat{A}_0(N;M)^{+}\| < C\left(\frac{R^2\left(\log\left(1/\delta\right)+1\right)+1}{\sqrt{N}}\right),$$

with probability at least $1-\delta$. □

B.7 Proof of Proposition 7.4.2

We will present a proof for each class.

MDP Class 1: We first show the behavior for ℓ_2 loss algorithm, where the magnitude of update is the absolute difference of the current estimate and the target value. We obtain

$$\hat{Q}_1(s_0,0) = \alpha H R(s_0,s_0,0),$$

and therefore, the agent will take $a=1$ at s_0 for the next episode. After episode 1, the Q estimate satisfies

$$\hat{Q}_2(s_0,1) = \alpha R(s_0,s_1,1),$$

and

$$\hat{Q}_2(s_1,0) = \alpha(H-1)R(s_1,s_1,0).$$

Repeating this process under $H > 2N$, $\alpha < 1/H$, and $R(s_i,s_{i+1},1) < R(s_{i+1},s_{i+2},1) \le 0$, we obtain, for all $0 \le i < N-1$ and $1 \le t < N$, as a rough bound,

$$\begin{aligned}
&\hat{Q}_t(s_i,0) = \delta(t-1 \ge i)\alpha(H-i)R(s_i,s_i,0),\\
&\hat{Q}_t(s_i,1) \ge \delta(t-2 \ge i)\alpha\sum_{j=i}^{N-2}\max\left\{(t-j-1),0\right\}\alpha^{j}R(s_j,s_{j+1},1)\\
&\ > \delta(t-2\ge i)\frac{\alpha(N-i)}{1-\alpha}R(s_i,s_{i+1},1) > \delta(t-2\ge i)2\alpha(N-i)R(s_i,s_{i+1},1),
\end{aligned}$$

where $\delta($`condition`$)$ returns 1 if the `condition` is met and otherwise zero. Hence, by $H > 2N$ and $0 \ge R(s_i,s_{i+1},1) > R(s_i,s_i,0)$ for all $i \in \{0,1,\ldots,N-2\}$, after time step $(N-1)H$, we have met the success condition.

For ℓ_1 loss algorithm where the magnitude of gradient is fixed to 1 almost everywhere, similar arguments as above hold. Therefore, for both ℓ_1 loss and ℓ_2 loss cases, we obtain

$$\inf_{\omega} \left[t \middle| \ell_1 \texttt{ loss meets success cond.}\right] = \inf_{\omega} \left[t \middle| \ell_2 \texttt{ loss meets success cond.}\right] = N-1.$$

MDP Class 2: It basically follows the same arguments as MDP Class 1, but the episode where the agent first reaches s_{k+1}, which we define to be t_{k+1}, it receives large penalty. For ℓ_1 loss, it does not affect the magnitude of update, and thus the behavior is the same as Class 1 until $t = N$. On the other hand, for ℓ_2 loss

$$\hat{Q}_{t_{k+1}+1}(s_k, 1) \le -\alpha BH < \alpha(H-k)R(s_k, s_k, 0),$$

and the agent will take wrong action at $t_{k+1}+1$. Therefore, we obtain

$$N-1 = \inf_{\omega} \left[t \middle| \ell_1 \texttt{ loss meets success cond.}\right] < \inf_{\omega} \left[t \middle| \ell_2 \texttt{ loss meets success cond.}\right].$$

MDP Class 3: First of all, to meet the success condition, the agent anyway needs to reach s_{N-2} to learn that action $a = 1$ must be chosen to reach the done state; and in this episode, it experiences the transition $s_{N-3} \to s_{N-2}$ which gives a penalty. This penalty does not propagate to the previous states because $\max_a\{\hat{Q}(s_{N-3}, a)\}$ stays zero. Given the reward conditions, the fastest possible case of meeting the success condition is thus by reaching s_{N-1} many times to cancel out the negative reward experienced when $s_{N-3} \to s_{N-2}$.

Consider ℓ_1 loss algorithm; because of the constant update magnitude of 1, the agent needs to reach s_{N-1} at least (as a loose bound) $\lceil |R(s_{N-3}, s_{N-2}, 1)|/\alpha \rceil$ epochs until the estimated Q value for the transition $s_{N-3} \to s_{N-2}$ becomes positive.

On the other hands, for ℓ_2 loss algorithm, assume that the agent reaches s_{N-1} at the initial episode. Then,

$$\hat{Q}_1(s_{N-3}, 1) = \alpha R(s_{N-3}, s_{N-2}, 1),$$
$$\hat{Q}_1(s_{N-2}, 1) > \alpha 4|R(s_{N-3}, s_{N-2}, 1)|.$$

Further, we assume the agent reaches to the final state for $\lceil 1/\alpha \rceil$ more consecutive episodes; then it gets

$$\hat{Q}_1(s_{N-3}, 1) \geq R(s_{N-3}, s_{N-2}, 1),$$
$$\hat{Q}_1(s_{N-2}, 1) > 2|R(s_{N-3}, s_{N-2}, 1)|.$$

Therefore, after at most $2\lceil 1/\alpha \rceil$ such consecutive episodes, it gets

$$\hat{Q}_1(s_{N-3}, 1) > 0.$$

The probability of such an event (i.e., experiencing at most $2\lceil 1/\alpha \rceil$ consecutive episodes of reaching to the final state) happening before $|R(s_{N-3}, s_{N-2}, 1)|/\alpha$ episodes is thus positive as long as

$$\frac{|R(s_{N-3}, s_{N-2}, 1)|}{\alpha} > 2 + \frac{2}{\alpha}.$$

Hence, under the assumptions, we obtain

$$\inf_{\omega}\left[t \middle| \ell_2 \texttt{ loss meets success cond.}\right] < \inf_{\omega}\left[t \middle| \ell_1 \texttt{ loss meets success cond.}\right].$$

MDP Class 4: We show the followings:

$$\inf_{\omega}\left[t \middle| \ell_1 \texttt{ loss meets success cond.}\right] < \inf_{\omega}\left[t \middle| \ell_2 \texttt{ loss meets success cond.}\right] \quad \text{(B.7.1)}$$

and

$$\exists T > 0 : \forall t \geq T,\ \Pr\left[\hat{Q}_{2,t}(s_{N-3}, 1) > \hat{Q}_{2,t}(s_{N-3}, 0)\right] > \Pr\left[\hat{Q}_{1,t}(s_{N-3}, 1) > \hat{Q}_{1,t}(s_{N-3}, 0)\right], \quad \text{(B.7.2)}$$

where $\hat{Q}_{1,t}$ ($\hat{Q}_{2,t}$) is the estimated Q function at episode t for ℓ_1 loss (for ℓ_2 loss).

Both ℓ_1 loss and ℓ_2 loss algorithms follow similarly the case of MDP Class 2, and thus the claim (B.7.1) holds.

First, consider ℓ_2 loss algorithm. Because of the action selection mechanism, and that the rewards are all negative except for the transitions to the final states s_{N-2} and s_{N-1}, and because $\alpha < 1/H$, it follows that the estimated Q values for action 0 at the intermediate states do not fall below the sum of all of the negative rewards, i.e.,

$$\hat{Q}_{2,t}(s_i, 1) \geq \sum_{j=i}^{N-4} R(s_j, s_{j+1}, 1).$$

This can be verified by inductions; $\hat{Q}_{2,t}(s_{N-3}, 0)$ or $\hat{Q}_{2,t}(s_{N-3}, 1)$ are at least zeros, and $\hat{Q}_{2,t}(s_{N-4}, 1)$ is at least $R(s_{N-4}, s_{N-3}, 1) + 0$ ($\because$ the step size is smaller than $1/H$ with the gradient of ℓ_2 loss loss, implying that the estimate does not go further than the target, and $\max_a\{\hat{Q}_{2,t}(s_{N-3}, a)\} \geq 0$), and then by inductions. Therefore, after sufficiently large number T of episodes, the agent selects action 1 at all the states except for s_{N-3}. As such, suppose T episodes include more than or equal to $K \geq 2$ reaches to the final states.

Now we know, for a fixed target value, the output of ℓ_2 loss algorithm after a given number of updates and is linear in the target value, and the magnitude of update strictly shrinks as the number of updates increases. Therefore, $\hat{Q}_{2,T}(s_{N-3}, 1)$ is larger than $\hat{Q}_{2,T}(s_{N-3}, 0)$ if the number of times action 0 at s_{N-3} is taken is smaller than $c > 2$ times the number of times action 1 is taken. It hence follows that after T episodes,

$$\Pr\left[\hat{Q}_{2,t}(s_{N-3}, 1) > \hat{Q}_{2,t}(s_{N-3}, 0)\right] > \frac{1}{2}, \quad \forall t \geq T.$$

On the other hand, consider ℓ_1 loss algorithm. After any number t of episodes, we have, for any $k \in \{0, 1, 2, \ldots, K\}$,

$$\Pr\left[\hat{Q}_t(s_{N-3}, 1) > \hat{Q}_t(s_{N-3}, 0) \middle| \texttt{agent has reached to the final state } k \texttt{ times}\right] \leq \frac{1}{2}$$

because the left hand side corresponds to the probability that random walk starting from the origin ends up in positive region at time k under the assumption $\alpha K < R(s_{N-3}, s_{N-2}, 0)$ and because of the reflection principle of random walk. After K reaches, the order of the estimates of Q values will not change, which now proves the claim (B.7.2).

Appendix C
MISCELLANEOUS

We provide some of the important technical lemmas here.

C.1 Linear algebra

Proposition C.1.1. *Let $A : \mathbb{C}^m \to \mathbb{C}^n$ be a linear map. Let $i : \mathscr{I}(A) \to \mathbb{C}^n$ be the inclusion map and let $p : \mathbb{C}^m \to \mathscr{N}(A)^\perp$ be the orthogonal projection. Let $\tilde{A} := A|_{\mathscr{N}(A)} : \mathscr{N}(A)^\perp \to \mathscr{I}(A)$ be an isomorphism. Then, the Moore-Penrose pseudo inverse A^+ coincides with $i^\dagger \tilde{A}^{-1} p^\dagger$.*

Proof. Let $B := p^\dagger \tilde{A}^{-1} i^\dagger$. We remark that $A = i\tilde{A}p$, $i^\dagger i = \mathrm{id}, pp^\dagger = \mathrm{id}$, we see that $ABA = A$, $BAB = B$, $(AB)^\dagger = AB$, and $BA = (BA)^\dagger$. By the uniqueness of Moore-Penrose pseudo inverse, $B = A^+$. □

Proposition C.1.2. *Let $A \in \mathbb{C}^{d\times d}$ be a matrix and let $v \in \mathbb{C}^d$ be a vector. Let $V \subset \mathbb{C}^d$ be a linear subspace generated by $\{A^j v\}_{j=1}^\infty$. Then, $V = \mathscr{I}\left([Av, A^2 v, \dots, A^d v]\right)$.*

Proof. Put $W = \mathscr{I}\left([Av, A^2 v, \dots, A^d v]\right)$. The inclusion $W \subset V$ is obvious, we prove the opposite inclusion. It suffices to show that $A^j v \in W$ for any positive integer $j > d$. By the Cayley-Hamilton theorem, $A^d = \sum_{j=1}^d c_j A^{d-j}$ for some $c_j \in \mathbb{C}$. Thus, by induction A^j is a linear combination of $A, A^2, \dots, A^d$, namely, $A^j v \in W$. □

Lemma C.1.3 (Null space of random matrix)**.** *Suppose $M_k \in \mathbb{R}^{d\times d}$, $k \in [d]$, have the same* *null space. Then, the null space of*

$$X := \begin{bmatrix} x_1^\top M_1 \\ x_2^\top M_2 \\ \vdots \\ x_d^\top M_d \end{bmatrix},$$

where $x_k,\ k \in [d]$, are unit vectors independently drawn from the uniform distribution over the unit hypersphere, is the same as those of M_k with probability one.

Proof. Given any $k-1$ dimensional linear subspace in $\mathscr{N}(M_k)^\perp$ for any $k \in [d]$, it holds that the probability that $x_k^\top M_k$ lies on that space is zero. Therefore, by union bound, and by the fact that the row space is the orthogonal complement of the null space, we obtain the result. □

C.2 On distributions

Lemma C.2.1 (Chi-squared distance between two Gaussians)**.** *For Gaussian distributions $\mathcal{N}(\mu_1, \sigma^2 I)$ and $\mathcal{N}(\mu_2, \sigma^2 I)$, the (squared) chi-squared distance between $\mathcal{N}_1$ and $\mathcal{N}_2$ is:*

$$\int \frac{(\mathcal{N}_1(z) - \mathcal{N}_2(z))^2}{\mathcal{N}_1(z)} dz = \exp\left(\frac{\|\mu_1 - \mu_2\|^2_{\mathbb{R}^{d_x}}}{2\sigma^2}\right) - 1.$$

Proof. Observe that:

$$\int \frac{(\mathcal{N}_1(z) - \mathcal{N}_2(z))^2}{\mathcal{N}_1(z)} dz = \int \mathcal{N}_1(z) - 2\mathcal{N}_2(z) + \frac{\mathcal{N}_2(z)^2}{\mathcal{N}_1(z)} dz = -1 + \int \frac{\mathcal{N}_2(z)^2}{\mathcal{N}_1(z)} dz.$$

Note that for $\mathcal{N}_2^2(z)/\mathcal{N}_1(z)$, we have:

$$\mathcal{N}_2^2(z)/\mathcal{N}_1(z) = \frac{1}{Z} \exp\left(-\frac{1}{2\sigma^2}\left(2\|z-\mu_2\|^2_{\mathbb{R}^{d_x}} - \|z-\mu_1\|^2_{\mathbb{R}^{d_x}}\right)\right),$$

where Z is the normalization constant for $\mathcal{N}(0, \sigma^2 I)$, i.e., $Z = \int \exp\left(-\frac{1}{2\sigma^2}\|z\|^2_{\mathbb{R}^{d_x}}\right) dz$.

For $2\|z-\mu_2\|^2_{\mathbb{R}^{d_x}} - \|z-\mu_1\|^2_{\mathbb{R}^{d_x}}$, we can verify that:

$$2\|z-\mu_2\|^2_{\mathbb{R}^{d_x}} - \|z-\mu_1\|^2_{\mathbb{R}^{d_x}} = \|z + (\mu_1 - 2\mu_2)\|^2_{\mathbb{R}^{d_x}} - 2\|\mu_1 - \mu_2\|^2_{\mathbb{R}^{d_x}}.$$

This implies that:

$$\int \frac{\mathcal{N}_2(z)^2}{\mathcal{N}_1(z)} dz = \frac{1}{Z} \int \exp\left(-\frac{1}{2\sigma^2}\left(\|z-(2\mu_2-\mu_1)\|^2_{\mathbb{R}^{d_x}} - 2\|\mu_1-\mu_2\|_{\mathbb{R}^{d_x}}\right)\right) dz$$
$$= \frac{1}{Z}\exp\left(\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{\sigma^2}\right) \int \exp\left(-\frac{1}{2\sigma^2}\|z-(2\mu_2-\mu_1)\|^2_{\mathbb{R}^{d_x}}\right) dz$$
$$= \exp\left(\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{\sigma^2}\right),$$

which concludes the proof. □

C.3 Bounding lemmas

Lemma C.3.1 (Expectation difference under two Gaussians)**.** *For Gaussian distributions $\mathcal{N}_1(\mu_1, \sigma^2 I)$ and $\mathcal{N}_2(\mu_2, \sigma^2 I)$, and for any (appropriately measurable) positive function g, it holds that:*

$$\mathbb{E}_{z\sim\mathcal{N}_1}[g(z)] - \mathbb{E}_{z\sim\mathcal{N}_2}[g(z)] \leq \min\left\{\frac{\|\mu_1-\mu_2\|_{\mathbb{R}^{d_x}}}{\sigma}, 1\right\} \sqrt{\mathbb{E}_{z\sim\mathcal{N}_1}\left[g(z)^2\right]}.$$

Proof. Define $m_i = \mathbb{E}_{z\sim\mathcal{N}_i}[g(z)]$ for $i \in \{1, 2\}$. We have:

$$m_1 - m_2 = \mathbb{E}_{z\sim\mathcal{N}_1}\left[g(z)(1-\frac{\mathcal{N}_2(z)}{\mathcal{N}_1(z)})\right]$$
$$\leq \sqrt{\mathbb{E}_{z\sim\mathcal{N}_1}[g(z)^2]}\sqrt{\int \frac{(\mathcal{N}_1(z)-\mathcal{N}_2(z))^2}{\mathcal{N}_1(z)} dz}$$
$$= \sqrt{\mathbb{E}_{z\sim\mathcal{N}_1}[g(z)^2]}\sqrt{\exp\left(\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{2\sigma^2}\right) - 1}$$

where we have used Lemma C.2.1. Also since m_2 is positive,

$$m_1 - m_2 \leq m_1 \leq \sqrt{\mathbb{E}_{z\sim\mathcal{N}_1}[g(z)^2]},$$

and so

$$m_1 - m_2 \leq \sqrt{\mathbb{E}_{z\sim\mathcal{N}_1}[g(z)^2]}\sqrt{\min\left\{\exp\left(\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{2\sigma^2}\right) - 1, 1\right\}}.$$

Now if the min is not achieved by 1, then $\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{2\sigma^2} \leq 1$. And since $\exp(x) \leq 1+2x$ for $0 \leq x \leq 1$, we have:

$$\min\left\{\exp\left(\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{2\sigma^2}\right)-1,1\right\} \leq \min\left\{1+\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{\sigma^2}-1,1\right\} = \min\left\{\frac{\|\mu_1-\mu_2\|^2_{\mathbb{R}^{d_x}}}{\sigma^2},1\right\}.$$

which completes the proof. $\square$

Lemma C.3.2 (Self-normalized bound for vector-valued martingales; [1]). *Let $\{\varepsilon_i\}_{i=1}^{\infty}$ be a real-valued stochastic process with corresponding filtration $\{\mathscr{F}_i\}_{i=1}^{\infty}$ such that ε_i is $\mathscr{F}_i$ measurable, $\mathbb{E}[\varepsilon_i|\mathscr{F}_{i-1}]=0$, and ε_i is conditionally σ-sub-Gaussian with $\sigma \in \mathbb{R}_{\geq 0}$. Let $\{X_i\}_{i=1}^{\infty}$ be a stochastic process with $X_i \in \mathcal{H}$ (some Hilbert space) and X_i being $\mathscr{F}_t$ measurable. Assume that a linear operator $V : \mathcal{H} \to \mathcal{H}$ is positive definite, i.e., $x^\top V x > 0$ for any $x \in \mathcal{H}$. For any t, define the linear operator $V_t = V + \sum_{i=1}^{t} X_i X_i^\top$ (here $xx^\top$ denotes outer-product in $\mathcal{H}$). With probability at least $1-\delta$, we have for all $t \geq 1$:*

$$\left\|\sum_{i=1}^{t} X_i\varepsilon_i\right\|^2_{V_t^{-1}} \leq 2\sigma^2 \log\left(\frac{\det(V_t)^{1/2}\det(V)^{-1/2}}{\delta}\right).$$

We generalize this lemma as follows:

Lemma C.3.3 (Self-normalized bound for matrix-valued martingales). *Let $\{\varepsilon_i\}_{i=1}^{\infty}$ be a d-dimensional vector-valued stochastic process with corresponding filtration $\{\mathscr{F}_i\}_{i=1}^{\infty}$ such that ε_i is $\mathscr{F}_i$ measurable, $\mathbb{E}[\varepsilon_i|\mathscr{F}_{i-1}]=0$, and ε_i is conditionally σ-sub-Gaussian with $\sigma \in \mathbb{R}_{\geq 0}$.*[1] *Let $\{X_i\}_{i=1}^{\infty}$ be a stochastic process with $X_i \in \mathcal{H}$ (some Hilbert space) and X_i being $\mathscr{F}_t$ measurable. Assume that a linear operator $V : \mathcal{H} \to \mathcal{H}$ is positive definite. For any t, define the linear operator $V_t = V + \sum_{i=1}^{t} X_i X_i^\top$ Then, with probability at least $1-\delta$, we have for all t, we have:*

$$\left\|\sum_{i=1}^{t} \epsilon_i X_i^\top V_t^{-1/2}\right\|^2 \leq 8\sigma^2 d\log(5) + 8\sigma^2\log\left(\frac{\det(V_t)^{1/2}\det(V)^{-1/2}}{\delta}\right).$$

[1]We say a vector-valued, random variable z is σ-sub-Gaussian if $w \cdot z$ is σ-sub-Gaussian for every unit vector w.

Proof. Denote $S = \sum_{i=1}^{t} \epsilon_i X_i^\top$. Let us form an ϵ-net, in ℓ_2 distance, $\mathcal{C}$ over the unit ball $\{w : \|w\|_{\mathbb{R}^d} \leq 1, w \in \mathbb{R}^d\}$. Via a standard covering argument (e.g., [219]), we can choose $\mathcal{C}$ such that $\log(|\mathcal{C}|) \leq d\log(1 + 2/\epsilon)$.

Consider a fixed $w \in \mathcal{C}$ and $w^\top S = \sum_{i=1}^{t} w^\top \epsilon_i X_i^\top$. Note that $w^\top \epsilon_i$ is a σ-sub Gaussian due to $\|w\|_{\mathbb{R}^d} \leq 1$. Hence, Lemma C.3.2 implies that with probability at least $1-\delta$, for all t,

$$\left\| V_t^{-1/2} \sum_{i=1}^{t} X_i \left(w^\top \epsilon_i\right) \right\| \leq \sqrt{2}\sigma \sqrt{\log\left(\frac{\det(V_t)^{1/2}\det(V)^{-1/2}}{\delta}\right)}.$$

Now apply a union bound over $\mathcal{C}$, we get that with probability at least $1-\delta$:

$$\forall w \in \mathcal{C} : \left\| V_t^{-1/2} \sum_{i=1}^{t} X_i \left(w^\top \epsilon_i\right) \right\| \leq \sqrt{2}\sigma \sqrt{d\log(1 + 2/\epsilon) + \log\left(\frac{\det(V_t)^{1/2}\det(V)^{-1/2}}{\delta}\right)}.$$

For any w with $\|w\|_{\mathbb{R}^d} \leq 1$, there exists a $w' \in \mathcal{C}$ such that $\|w - w'\|_{\mathbb{R}^d} \leq \epsilon$. Hence, for all w such that $\|w\|_{\mathbb{R}^d} \leq 1$,

$$\begin{aligned}\left\| V_t^{-1/2} \sum_{i=1}^{t} X_i \left(w^\top \epsilon_i\right) \right\| &\leq \sqrt{2}\sigma \sqrt{d\log(1 + 2/\epsilon) + \log\left(\frac{\det(V_t)^{1/2}\det(V)^{-1/2}}{\delta}\right)} \\ &\quad + \epsilon \left\| \sum_{i=1}^{t} \epsilon_i X_i^\top V_t^{-1/2} \right\|.\end{aligned}$$

By the definition of the spectral norm, this implies that:

$$\left\| \sum_{i=1}^{t} \epsilon_i X_i^\top V_t^{-1/2} \right\| \leq \frac{1}{1-\epsilon}\sqrt{2}\sigma \sqrt{d\log(1 + 2/\epsilon) + \log\left(\frac{\det(V_t)^{1/2}\det(V)^{-1/2}}{\delta}\right)}.$$

Taking $\epsilon = 1/2$ concludes the proof. $\square$

Lemma C.3.4. *For any sequence $x_0, \ldots, x_{T-1}$ such that, for $t < T$, $x_t \in \mathbb{R}^d$ and $\|x_t\|_{\mathbb{R}^d} \leq B \in \mathbb{R}_{\geq 0}$, we have:*

$$\log\det\left(I + \frac{1}{\lambda}\sum_{t=0}^{T-1} x_t x_t^\top\right) \leq d\log\left(1 + \frac{TB^2}{d\lambda}\right).$$

Proof. Denote the eigenvalues of $\sum_{t=0}^{T-1} x_t x_t^\top$ as $\sigma_1, \ldots, \sigma_d$, and note:

$$\sum_{i=1}^{d} \sigma_i = \operatorname{tr}\left(\sum_{t=0}^{T-1} x_t x_t^\top\right) \leq TB^2.$$

Using the AM-GM inequality,

$$\log\det\left(I+\frac{1}{\lambda}\sum_{t=0}^{T-1}x_tx_t^\top\right)=\log\left(\prod_{i=1}^{d}(1+\sigma_i/\lambda)\right)$$
$$=d\log\left(\prod_{i=1}^{d}(1+\sigma_i/\lambda)\right)^{1/d}\le d\log\left(\frac{1}{d}\sum_{i=1}^{d}(1+\sigma_i/\lambda)\right)\le d\log\left(1+\frac{TB^2}{d\lambda}\right),$$

which concludes the proof. □

Lemma C.3.5 (Perturbation bounds of the Moore-Penrose inverse)**.** *Suppose $A\in\mathbb{C}^{d\times d}$ is a matrix. Let $E\in\mathbb{C}^{d\times d}$ be a matrix satisfying*

$$\exists C>0,\quad \forall N\in\mathbb{Z}_{>0},\quad \|E\|\le C\frac{1}{\sqrt{N}},$$

and let $\hat{A}_E\in\mathbb{C}^{d\times d}$ be the low rank approximation of $A+E$ via SVD with the singular value threshold $C/\sqrt{N}$. Then, we obtain

$$\left\|A^+-\hat{A}_E^+\right\|\le\frac{8C\|A^+\|^2\left(\sqrt{d}+1\right)}{\sqrt{N}}.$$

Proof. Let $\sigma_{\min}=\|A^+\|^{-1}$ be the minimal singular value of A. From [171, Theorem 1.1] (or originally [259]) and from the fact

$$\left\|\hat{A}_E-A\right\|\le\|E\|+\frac{\sqrt{d}C}{\sqrt{N}},$$

we obtain

$$\left\|A^+-\hat{A}_E^+\right\|\le\frac{1+\sqrt{5}}{2}\max\left\{\|A^+\|^2,\left\|\hat{A}_E^+\right\|^2\right\}\left(\sqrt{d}+1\right)\frac{C}{\sqrt{N}}.$$

For N such that $1\le N\le 4C^2/\sigma_{\min}^2$, we have $\left\|\hat{A}_E^+\right\|\le\sqrt{N}/C\le 2/\sigma_{\min}$. Suppose singular values $\sigma_k,\ k\in[d]$, of A, and $\hat{\sigma}_k,\ k\in[d]$, of $A+E$ are sorted in descending order. Then, it holds that

$$|\sigma_k-\hat{\sigma}_k|\le\|E\|.$$

Therefore, for $N > 4C^2/\sigma_{\min}^2$, the minimum singular value $\hat{\sigma}_d$ is greater than $\sigma_{\min}/2$. In this case, $\hat{A}_E = A + E$, and it follows that $\left\|\hat{A}_E^+\right\| \leq 2/\sigma_{\min}$. Hence, for all $N \geq 1$, we obtain

$$\left\|\hat{A}_E^+\right\| \leq \frac{2}{\sigma_{\min}},$$

from which, it follows that

$$\left\|A^+ - \hat{A}_E^+\right\| \leq \frac{2C\left(1+\sqrt{5}\right)\left(\sqrt{d}+1\right)}{\sigma_{\min}^2\sqrt{N}} \leq \frac{8C\|A^+\|^2\left(\sqrt{d}+1\right)}{\sqrt{N}}.$$

□

C.4 Azuma-Hoeffding inequality for exponential sum

Lemma C.4.1. *Let $\{X_j\}_{j=1}^n$ for $n \in \mathbb{Z}_{>0}$ be sub-Gaussian martingale difference with variance proxy R^2 and a filtration $\{\mathscr{F}_j\}$. Also, let $\{a_j\} \subset \mathbb{C}$ be a sequence of complex numbers satisfying $|a_j| \leq 1$ for all $j \in [n]$. Then, the followings hold, where $*[\cdot]$ stands for $\mathfrak{Re}[\cdot]$ or $\mathfrak{Im}[\cdot]$.*

$$\Pr\left[\frac{1}{n}\left|*\left[\sum_{j=1}^n a_j X_j\right]\right| \leq \sqrt{\frac{2R^2\log(2/\delta)}{n}}\right] \geq 1-\delta,$$
$$\Pr\left[\frac{1}{n}\left|\sum_{j=1}^n a_j X_j\right| \leq \sqrt{\frac{4R^2\log(4/\delta)}{n}}\right] \geq 1-\delta.$$

Proof. For a filtration $\{\mathscr{F}_i\}_{i\leq n}$, we have

$$\mathbb{E}\left[e^{\lambda\mathfrak{Re}\left[\sum_{j=1}^n a_jX_j\right]}\right] \leq \mathbb{E}\left[e^{\lambda\mathfrak{Re}\left[\sum_{j=1}^{n-1} a_jX_j\right]}\mathbb{E}\left[e^{\lambda\mathfrak{Re}[a_nX_n]}\middle|\mathscr{F}_{n-1}\right]\right] \leq e^{\frac{\lambda^2R^2}{2}}\mathbb{E}\left[e^{\lambda\mathfrak{Re}\left[\sum_{j=1}^{n-1} a_jX_j\right]}\right],$$

where the first inequality follows from the assumption of filtration, and the second inequality follows from

$$\mathbb{E}\left[e^{\lambda\mathfrak{Re}[a_nX_n]}\middle|\mathscr{F}_{n-1}\right] = \mathbb{E}\left[e^{\lambda X_n\mathfrak{Re}[a_n]}\middle|\mathscr{F}_{n-1}\right] \leq e^{\frac{\lambda^2R^2}{2}}.$$

By induction, we obtain

$$\mathbb{E}\left[e^{\lambda\mathfrak{Re}\left[\sum_{j=1}^n a_jX_j\right]}\right] \leq e^{\frac{n\lambda^2R^2}{2}}.$$

By using Markov inequality and the union bound, it follows that

$$\Pr\left[\frac{1}{n}\left|\mathfrak{Re}\left[\sum_{j=1}^{n} a_j X_j\right]\right| > \epsilon\right] \le 2e^{-\frac{n\epsilon^2}{2R^2}}.$$

Similarly, we have

$$\Pr\left[\frac{1}{n}\left|\mathfrak{Im}\left[\sum_{j=1}^{n} a_j X_j\right]\right| > \epsilon\right] \le 2e^{-\frac{n\epsilon^2}{2R^2}}.$$

Therefore, we obtain

$$\begin{aligned}
&\Pr\left[\frac{1}{n}\left|\sum_{j=1}^{n} a_j X_j\right| \le \sqrt{\frac{4R^2\log(4/\delta)}{n}}\right] \\
&\ge \Pr\left[\frac{1}{n}\left|\mathfrak{Re}\left[\sum_{j=1}^{n} a_j X_j\right]\right| \le \sqrt{\frac{2R^2\log(4/\delta)}{n}}\right] \\
&\quad + \Pr\left[\frac{1}{n}\left|\mathfrak{Im}\left[\sum_{j=1}^{n} a_j X_j\right]\right| \le \sqrt{\frac{2R^2\log(4/\delta)}{n}}\right] - 1 \\
&\ge \left(1-\frac{\delta}{2}\right) + \left(1-\frac{\delta}{2}\right) - 1 \ge 1-\delta,
\end{aligned}$$

where the first inequality follows from the Fréchet inequality. □