Towards Safe and Efficient Reinforcement Learning for Autonomous Systems: A Constrained Learning Approach with Safety Guarantees and System Dynamics Identification

Junxia Deng
University of Southern California
California, USA

Ceil Hong. Zhang
Massachusetts Institute of Technology
Massachusetts, USA
zhanghongceil@pku.org.cn

Abstract

Designing control policies for autonomous systems such as self-driving cars is complex. To this end, researchers are increasingly using reinforcement learning (RL) to design a policy. However, guaranteeing safe operation during real-world training and deployment is currently an unsolved issue, which is of vital importance for safety-critical systems. In addition, current RL approaches require accurate simulators (models) to learn policies, which is rarely the case in real-world applications. The thesis introduces a safe RL framework that provides safety guarantees and develops a constrained learning approach that learns system dynamics. We develop a safe RL algorithm that optimizes task rewards while satisfying safety constraints. We then consider a variant of safe RL problems when provided with a baseline policy. The baseline policy can arise from demonstration data and may provide useful cues for learning, but it is not guaranteed to satisfy the safety constraints. We propose a policy optimization algorithm to solve this problem. In addition, we apply a safe RL algorithm in the legged locomotion to show its real-world applicability. We propose an algorithm that switches between a safe recovery policy that keeps the robot away from unsafe states, and a learner policy that is optimized to complete the task. We further exploit the knowledge about the system dynamics to determine the switch of the policies. The results suggest that we can learn legged locomotion skills without falling in the real world. We then revisit the assumption of knowing system dynamics and develop a method that performs system identification from observations. Knowing the parameters of the system improves the quality of simulation and hence minimize unexpected behavior of the policy. Finally, while safe RL holds great promise for many applications, current approaches require domain expertise to specify constraints. We thus introduce a new benchmark with constraints specified in free-form text. We develop a model that can interpret and adhere to such textual constraints. We show that the method achieves higher rewards and fewer constraint violations than baselines.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Safe Reinforcement Learning and Constrained Learning for Dynamical Systems

Designing control policies for many autonomous systems such as self-driving cars, unmanned aerial vehicles, and personalized robotic assistants is inherently complex. For instance, to design a walking gait for quadruped robots, designers need to hand-tune the timing of the stance and swing phases of the gaits. During the stance phase, a model predicted control (MPC) is used to generate the desired torque commands for the legs. This requires approximating the system dynamics of the robot via a centroidal dynamics model. During the swing phase, swing trajectories and the corresponding landing positions of legs are generated based on the target walking speed and the current position of the legs. Such an approach requires knowledge about the complex system dynamics, which leads to a significant amount of time to design a control policy.

To deal with this challenge, practitioners are increasingly turning towards data-driven learning techniques such as reinforcement learning (RL) for designing sophisticated control policies. Reinforcement learning is concerned with how learning agents

ought to take actions in an environment to maximize the cumulative task reward. Recent advances in deep RL have demonstrated excellent performance on several domains ranging from games like Go [136] and StarCraft [9] to robotic control [101]. In these settings, agents are allowed to explore the entire state space and experiment with all possible actions during training to optimize the task reward. However, in many real-world applications such as self-driving cars and unmanned aerial vehicles, considerations of safety, fairness, and other costs prevent the agent from having complete freedom to explore when learning or fine-tuning a control policy in the real world. For instance, an autonomous car, while optimizing its driving policies, must not take any actions that could cause harm to pedestrians or property (including itself). Hence, it is prudent to critically examine the potential consequences and provide a safety guarantee. In particular, (1) when the RL algorithm is deployed in the real world on a physical robotic system, how do we ensure that the system will act as it did in the laboratory? (2) In addition, when the RL algorithm is used to learn a control policy, how do we streamline the learning process so that do not need to train the policy from scratch? (3) Furthermore, when the system dynamics that are used to train control policies do not faithfully reflect what happened in the real world, how do we improve the quality of the simulator (model)? (4) Finally, when provided with information about the task at hand or non-reward feedback to the system, how do we effectively exploit the information to improve reward performance while ensuring safety?

In this thesis, we take a step toward answering these questions by formulating the problem of *safe reinforcement learning*, in which we aim to learn control policies that maximize a reward function while satisfying predefined constraints in the form of the constrained optimization problem. One unique aspect of safe RL is that we would like the learning agent do not violate the safety constraints during the *entire* learning process, not just at the end of training. This is because when applying RL algorithms in

the real world to learn a control policy from scratch or fine-tune a control policy, the actions taken by the agent may cause damage to itself or humans. Specifically, we will study four aspects of safe RL: (1) **safety**: learning policy without violating the safety constraints, (2) **supervision of demonstration data**: exploiting the demonstration data or teacher agent into the learning process to improve learning efficiency while ensuring safety, (3) **constrained learning for system dynamics**: using knowledge about the physics to inform the learning system parameters of dynamics from observations when the unknown system parameters are essential for conducting safety analysis or synthesis of control policies, and (4) **multi-modal**: providing non-reward safety feedback (*e.g.,* text) to the system to correct its unsafe action while improving the task reward. In addition, we will test the proposed algorithms in a wide range of simulated and real-world applications to show their applicability, including (1) simulated robots manipulation tasks with safety constraints, (2) simulated traffic management tasks with fairness constraints of drivers, (3) simulated self-driving car policies with human supervision data, and (4) real quadruped robots with the safety constraints of falls. We hope these algorithms and applications will shed light on how to design control policies for robots to safely and efficiently learn tasks.

1.2 Related Work

In this thesis, we first focus on safe reinforcement learning in which we assume the system dynamics and their relevant system parameters are given to learn a control policy in the simulator. Then, in some applications, the system parameters that are useful for verifying the safety of the agent and simulating the environment are unknown due to partial knowledge about the system dynamics. We thus relax the assumption of knowing the system dynamics, and propose an approach of learning those dynamics for learning control policies in simulation or tracking the evolution of

the system parameters. Finally, current robotics systems are incapable of receiving feedback from humans for correcting their unsafe behaviors after the control policy is programmed or tuned in the factory. Even though they are able to accept feedback from humans, this is often achieved by directly typing the commands or using physical buttons, which is time-consuming and not easy for humans to use. We thus propose a method to use natural language to specify the safety constraints for the agents to correct their dangerous behavior for the ease of safe human-robot interactions. We now review the works in the field of (1) safe reinforcement learning, (2) physics-informed deep learning, and (3) RL for natural language processing.

1.2.1 Policy Learning with Constraints

The problem of policy learning with constraints is more challenging since directly optimizing for the reward, as in Q-Learning [112] or policy gradient [143], will usually violate the constraints. One approach is to incorporate constraints into the learning process by forming a constrained optimization problem. This approach has been explored in the context of safe RL [57]. The agent learns policies either by (1) exploration of the environment [6, 148, 37] or (2) through expert demonstrations [129, 126, 55]. For example, [149] use the sub-optimal demonstration to guide the learning. They obtain the safe policy by iteratively solving model predictive control. In addition, [180, 139, 150] pre-train a safe policy to guide the learning process. However, using expert demonstrations requires humans to label the constraint-satisfying behavior for every possible situation. The scalability of these rule-based approaches is an issue since many real autonomous systems such as self-driving cars and industrial robots are inherently complex. As a result, we design a safe reinforcement learning algorithm that does not rely on the pre-specified demonstration data. We show the learning agent is able to achieve better reward performance while satisfying the safety constraints compared to the prior works.

1.2.2 Leveraging Baseline Policies and Demonstration Data

Prior work has used baseline policies to provide initial information to RL algorithms to reduce or avoid undesirable situations. This is done by either: initializing the policy with the baseline policy [45, 138, 92, 3, 55, 95, 159, 83], or providing a teacher's advice to the agent [56, 124, 4, 181]. However, such works often assume that the baseline policy is constraint-satisfying [142, 15]. In many real-world applications, the baseline policy may not satisfy the constraints at hand. Hence we would like to propose an approach to safely learn from those constraint-violating policies.

1.2.3 Policy Optimization with the Initial Safe Set

[163, 141, 155] assume that the initial safe set is given, and the agent explores the environment and verifies the safety function from this initial safe set. However, such a safe set may not be known in advance in some applications, which limits their applicability. In this thesis, we focus on the assumption of given baseline policies that need not be safe, which is more practical in real world applications.

1.2.4 Reinforcement Learning for Quadrupedal Locomotion

Several recent works have applied RL in quadrupedal locomotion to acquire complex locomotion skills [119, 91, 145, 49, 64, 177, 63, 179]. The reason for using RL is that designing those control policies for robots usually requires explicit knowledge about the system dynamics, which are hard to obtain in practice. These approaches often directly learn the policy or fine-tune the learned policy in the real world without considering safety and thus require humans to recover robots when falling. In contrast, we focus on how to learn control policies safely for robots.

1.2.5 Physics-informed Learning and Differential Simulators

In the first three chapters of the thesis, we assume that the system dynamics are given to learn the control policy in simulation, or the system parameters that drive the system dynamics are known. However, for some applications, the knowledge of the system dynamics is not always known. Hence we would like to learn the system dynamics from the data. We now review this line of work. The approaches that learn system dynamics (*i.e.,* given the current state and the control input, we want to predict the next state) can be grouped into three categories: black-box methods, grey-box methods, and white-box methods. First, black-box methods [80, 171] do not get access to the system dynamics. This approach usually trains a neural network that directly predicts the states given the input of observations and actions. However, this method lacks interpretability due to the absence of constraints on the latent representations. Second, grey-box methods [41] exploit partial knowledge about the system dynamics and assume some parameters inside the system are unknown. This approach offers great flexibility in predicting system parameters and improves the interpretability of the model. Third, white box methods [122, 102] impose prior knowledge by using explicit dynamics models. Such an approach reduces the search space of neural networks but requires more knowledge about the system dynamics. In this thesis, we focus on the grey-box method since it balances interpretability and prior assumptions, and has many real-world applications.

Recent work on physics-based deep learning regularizes the latent representation by using Lagrangian and Hamiltonian dynamics [185, 50, 107]. This physics-informed deep learning framework is also being extended into autodifferentiation frameworks. This has enabled differentiation through system dynamics such as contact and friction models [185] and latent state models [82, 132, 60, 69]. These approaches lead to applications such as robot motion planning through differentiable models [7, 85], fluid simulation rendering [151, 76, 128], and system identification [96, 69]. However,

prior works assume that the data are from the same system parameters. This implies that neural networks do not learn to generalize to the new set of system parameters. In this thesis, we aim to design a model that is capable of identifying system parameters from multiple observations without re-training. This approach streamlines the identification process during testing or deployment.

1.2.6 Reinforcement Learning with Natural Language Processing

Several recent works have applied RL in the domain of natural language processing such as dialogue systems [182], instruction following [53], and human-robot interactions [118]. In this thesis, our work closely relates to the paradigm of instruction following in RL, which has previously been explored in several environments [108, 162, 28, 147, 13, 88, 12, 153, 106, 146, 165]. Prior work has also focused on creating realistic vision-language navigation datasets with visual urban or household environment [21, 31, 11, 42] and proposed computational models to learn multi-modal representations that fuse images with goal instructions [81, 22, 52, 105, 78, 54, 75, 53, 160]. In this thesis, we focus on using natural language to specify the safety constraints for correcting the unsafe behavior of the robot.

1.3 Organization and Contribution of the Thesis

The work in this thesis presents a framework for learning a control policy safely with testing on physical robotic platforms. The organization and contributions are summarized here, and can be visualized in Fig. 1.1. In Chapter 2, we formally introduce the problem of safe RL. We take a perspective from optimization theory and propose a constrained optimization algorithm. We provide a theoretical analysis of the learning performance for each policy update in terms of safety violations and reward

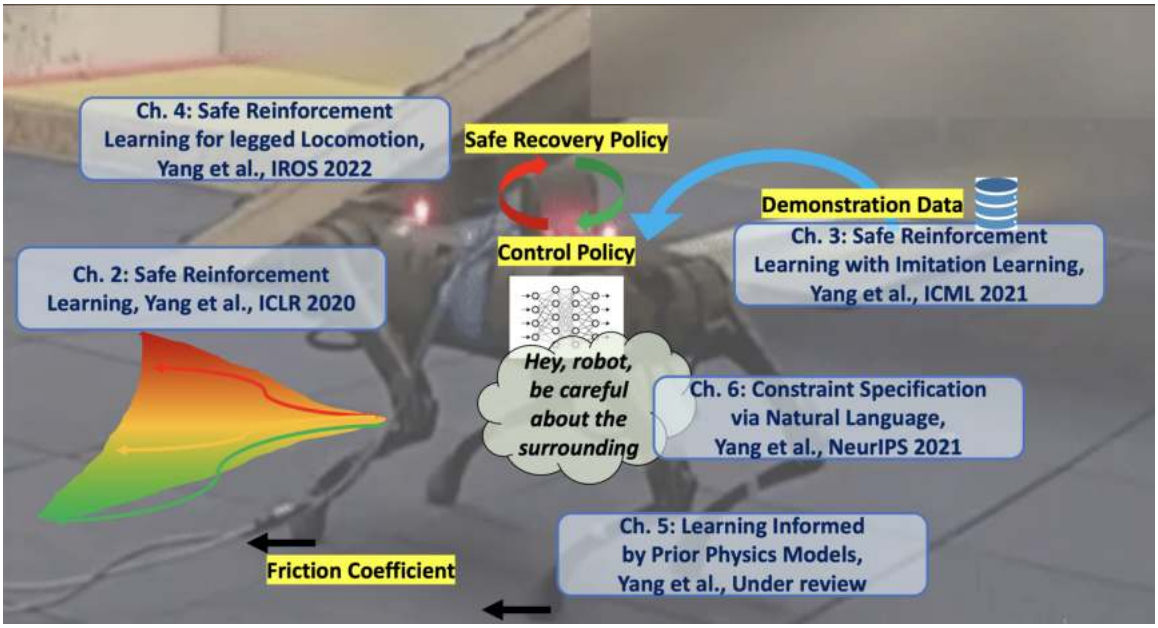


Figure 1.1: An overview of the thesis with an example of learning a control policy for quadruped in the real world.

improvement. Finally, we draw connections with related methods and conduct experiments with simulation tasks (robot manipulation and traffic management tasks) at the end of the chapter. Our contribution is a new safe reinforcement learning algorithm that improves safety constraint satisfaction using a projections approach over a prior state-of-the-art algorithm.

In Chapter 3, we aim to improve the learning efficiency of the algorithm proposed in Chapter 2. We assume that the baseline policy (demonstration data or teacher agent) is provided to the learning agent. The baseline policy contains a useful signal for learning, but it may not satisfy the safety constraints of the task at hand. We thus propose an algorithm that can safely exploit the baseline policy to improve the task reward performance. We provide theoretical analysis on the convergence of the algorithm, *i.e.,* how many policy updates are needed to achieve an optimal solution. Finally, we conduct experiments with simulation tasks (robot manipulation and traffic management tasks) and one human demonstration task for car simulation at the end of the chapter. The contribution is a new safe reinforcement learning algorithm that

is able to learn from constraint-violating baseline policies without violating the safety constraints.

In Chapter 4, we shift the focus from simulated applications to real-world applications and push the limit of the safe RL algorithm by applying it to real quadruped robots. Our goal is to understand the resilience of the algorithm in the more complex task due to the under-actuated and non-continuous robot dynamics. These complex dynamics make the robot fall easily. Our learning framework adopts a two-policy structure: a *safe recovery policy* that recovers robots from near-unsafe states, and a *learner policy* that is optimized to perform the desired control task. We exploit the knowledge about the system dynamics to determine the switch timing between these two policies. Finally, we experiment with the proposed framework in the real hardware and show that the robot never falls during the entire learning process. This is the first work to demonstrate that it is able to learn a locomotion skill autonomously and safely in the real world without manually resetting (*e.g.,* position the robot to a safe pose after falling).

In Chapter 5, we revisit the assumption of knowing system dynamics (*i.e.,* simulators or models) that are used to train control policies in simulation (Chapter 2, 3), and known system parameters of the system dynamics (Chapter 4). Many RL algorithms require simulators or models to faithfully simulate the true system dynamics in the real world. Without an accurate simulator, the policy trained in simulation could have low reward task performance in the real world due to environment distribution shift. This can cause the policy to have unseen or undesirable behaviors, leading to unsafe events. In addition, in many applications system designers do know the dynamic equations of the system (*i.e.,* physics) but do not know its state and the system parameters. Knowing the state and parameters of the system aids in learning a control policy safely and tracking parameter evolution over time. Hence, in this chapter, we relax the assumption of knowing the full system dynamics and assume

that the system equation is known but the system parameters are unknown. This setting is useful such as the system parameters (*e.g.,* friction coefficient) we used in Chapter 4 may change over time due to motor deterioration or even unknown to system designers. We thus propose and explore a model that integrates physics into an autoencoder to perform unsupervised state and physical parameter predictions from a sequence of observations. The networks that predict the states and system parameters from observations are constrained to follow the dynamics of the system (*i.e.,* physics) through a differentiable simulator. Finally, we evaluate the proposed model in three visual and one sensor measurement tasks. The contribution is that the proposed model imposes interpretability on latent states and achieves improved generalization performance for the long-term prediction of system dynamics over state-of-the-art baselines.

In Chapter 6, while safe RL holds great promise for many practical applications like robotics or autonomous cars, current approaches require specifying constraints in mathematical form. Such specifications demand domain expertise, limiting the adoption of safe RL. In addition, current autonomous systems are unable to receive feedback from humans after being programmed or designed in the factory. This prevents the agent from adapting to the changing environments. Even if the agent can understand the feedback from humans, this is often achieved by inputting the commands manually to the robot or pressing buttons on the robot, which is time-consuming. Such an approach prevents the autonomous agent to correct its unsafe behavior during deployment. To this end, we extend the algorithm in Chapter 2 and propose learning to interpret natural language constraints (*i.e.,* non-reward feedback) for safe RL. Natural language provides a flexible way for humans to specify the safety constraints of robots. In addition, we humans learn to be safe by receiving verbal feedback from parents without really trying unsafe behavior during our childhood. A robot with such a capability can greatly improve safety. We first introduce a new

multi-task benchmark that requires an agent to optimize reward while not violating constraints specified in free-form text. We then develop an agent with a modular architecture that can interpret and adhere to such textual constraints while learning new tasks. We show that our method achieves higher rewards (up to 11x) and fewer constraint violations (by 1.8x) compared to existing approaches. However, in terms of absolute performance, the dataset still poses significant challenges for agents to learn efficiently, motivating the need for future work.

In Chapter 7, we draw conclusions and review the contribution of the thesis. In addition, we discuss several possible research directions. Besides the theoretical analysis and empirical evaluation of methods and algorithms in the thesis, we also release the code to public.

In summary, the goal of the thesis is to enable safe learning for autonomous systems with a safety guarantee while demonstrating RL algorithms can be safely used in real-world applications. Chapter 2 introduces the framework of safe reinforcement learning (**safety**); Chapter 3 adds an expert policy to improve learning speed (**supervision of demonstration data**); Chapter 4 applies safe RL in the real hardware (**safety**); Chapter 5 relaxes the assumption of the knowing system dynamics and conducts system identification to learn the dynamics through a differential simulator (**constrained learning for system dynamics**), and finally, Chapter 6 considers the multi-modal scenario for optimizing the task reward under the textual constraints in safe RL (**multi-modal**).

Safely applying RL to autonomous systems (*e.g.,* self-driving cars, household robots) can have the enormous potential to streamline the deployment of systems and hence improve the quality of human life. To achieve this, this thesis develops a series of approaches to enable safe learning of control policies and demonstrate their applicability both in simulation and in the real world. Our safe learning approach allows the robot to accurately reason about its own safety and its impact on surround-

ing environments including humans. However, to achieve fully autonomous and safe robotic technologies, our understanding of safe autonomous systems needs to push further. We hope that this thesis could inspire researchers and practitioners in the field of reinforcement learning to develop next-generation safe autonomous systems.

Prior Publications. Parts of this thesis have been published in [173, 175, 176, 174].

Chapter 2

Projection-Based Constrained Policy Optimization

2.1 Introduction

Reinforcement learning (RL) has achieved impressive performance in the past decade on a wide variety of tasks across different domains such as video games [9], Go [136], and robotic control [101]. However, in many real-world applications such as self-driving cars and unmanned aerial vehicles, considerations of safety, fairness, and other costs prevent the agent from having complete freedom to explore. The agent is constrained to take actions that do not violate a specified set of constraints on state-action pairs. In this chapter, we formally address the problem of learning control policies that optimize a reward function while satisfying predefined constraints during the learning process.

The problem of policy learning with constraints is more challenging since directly optimizing for the reward, as in Q-Learning [112] or policy gradient [143], will usually violate the constraints. One approach is to incorporate constraints into the learning process by forming a constrained optimization problem. Then perform policy updates

using conditional gradient descent with line search to ensure constraint satisfaction [6]. However, the base optimization problem can become infeasible if the current policy violates the constraints. Another approach is to add a hyperparameter weighted copy of the constraints to the objective function [148]. However, this incurs the cost of extensive hyperparameter tuning.

To address the above issues, we propose projection-based constrained policy optimization (PCPO). This is an iterative algorithm that performs policy updates in two stages. The first stage maximizes reward using a trust region optimization method (*e.g.,* TRPO [133]) without constraints. This might result in a new intermediate policy that does not satisfy the constraints. The second stage reconciles the constraint violation (if any) by projecting the policy back onto the constraint set, *i.e.,* choosing the policy in the constraint set that is closest to the selected intermediate policy. This allows efficient updates to ensure constraint satisfaction without requiring a line search [6] or adjusting a weight [148]. Further, due to the projection step, PCPO offers efficient recovery from infeasible (*i.e.,* constraint-violating) states (*e.g.,* due to approximation errors), which existing methods do not handle well.

We analyze PCPO theoretically and derive performance bounds for the algorithm. Specifically, based on information geometry and policy optimization theory, we construct a lower bound on reward improvement, and an upper bound on constraint violations for each policy update. We find that with a relatively small step size for each policy update, the worst-case constraint violation and reward degradation are tolerable. We further analyze two distance measures for the projection step onto the constraint set. We find that the convergence of PCPO is affected by the smallest and largest singular values of the Fisher information matrix used during training. By observing these singular values, we can choose the appropriate projection best suited to the problem.

Empirically, we compare PCPO with state-of-the-art algorithms on four different

control tasks, including two Mujoco environments with safety constraints introduced by [6] and two traffic management tasks with fairness constraints introduced by [161]. In all cases, the proposed algorithm achieves comparable or superior performance to prior approaches, averaging more rewards with fewer cumulative constraint violations. For instance, across the above tasks, PCPO achieves 3.5 times fewer constraint violations and around 15% more reward. This demonstrates the ability of PCPO robustly learn constraint-satisfying policies and represents a step toward the reliable deployment of RL in real problems.

Prior Publications. Parts of this chapter have been published in [173].

Code. `https://sites.google.com/view/iclr2020-pcpo`

2.2 Problem Setup for Safe Reinforcement Learning

We frame our policy learning as a constrained Markov Decision Process (CMDP) [10], where policies will direct the agent to maximize the reward while minimizing the cost. We define CMDP as the tuple $< \mathcal{S}, \mathcal{A}, T, R, C >$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions that the agent can take, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability of the CMDP, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the cost function. Given the agent's current state s, the policy $\pi(a|s) : \mathcal{S} \to \mathcal{A}$ selects an action a for the agent to take. Based on s and a, the agent transits to the next state (denoted by s') according to the state transition model $T(s'|s, a)$, and receives the reward and pays the cost, denoted by $R(s, a)$ and $C(s, a)$, respectively.

We aim to learn a policy π that maximizes a cumulative discounted reward, de-

noted by

$$J^R(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\big], \tag{2.1}$$

while satisfying constraints, *i.e.,* making a cumulative discounted cost constraint[1] below a desired threshold h, denoted by

$$J^C(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)\big] \leq h, \tag{2.2}$$

where γ is the discount factor, τ is the trajectory ($\tau = (s_0, a_0, s_1, \cdots)$), and $\tau \sim \pi$ is shorthand for showing that the distribution over the trajectory depends on $\pi : s_0 \sim \mu, a_t \sim \pi(a_t|s_t), s_{t+1} \sim T(s_{t+1}|s_t, a_t)$, where μ is the initial state distribution. The formulation of the cost constraint in Eq. (2.2) can be used for many applications. For instance, in the stock market, we would like to learn a policy that trades stock within the budget of the cost (*i.e.,* we want the total cost to be smaller than the budget). Another example would be the case with self-driving cars. We would like to learn a policy that improves driving skills without hitting other vehicles. In this case, the threshold h for counting the number of collisions would be zero. Note that we want to develop an algorithm that ensures constraint satisfaction during the *entire* learning process, not just at the end of the training.

[86] give an identity to express the performance of policy π' in terms of the advantage function over another policy π :

$$J^R(\pi') - J^R(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi'}\\ a\sim\pi'}}[A^{\pi}_R(s,a)], \tag{2.3}$$

[1] In this thesis, we use the term "cost constraint" and "safety constraint" interchangeably.

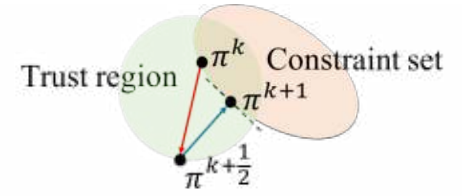


Figure 2.1: Update procedures for PCPO. In step one (red arrow), PCPO follows the reward improvement direction in the trust region (light green). In step two (blue arrow), PCPO projects the policy onto the constraint set (light orange).

where d^π is the discounted future state distribution, denoted by

$$d^\pi(s) \doteq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi), \tag{2.4}$$

and $A^\pi_R(s,a)$ is the reward advantage function, denoted by

$$A^\pi_R(s,a) \doteq Q^\pi_R(s,a) - V^\pi_R(s). \tag{2.5}$$

Here $Q^\pi_R(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0=s, a_0=a\big]$ is the discounted cumulative reward obtained by the policy π given the initial state s and action a, and $V^\pi_R(s) \doteq \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0=s\big]$ is the discounted cumulative reward obtained by the policy π given the initial state s. Similarly, we have the cost advantage function $A^\pi_C(s,a) = Q^\pi_C(s,a) - V^\pi_C(s)$, where $Q^\pi_C(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)|s_0=s, a_0=a\big]$, and $V^\pi_C(s) \doteq \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)|s_0=s\big]$.

2.3 Projection-Based Constrained Policy Optimization (PCPO)

To robustly learn constraint-satisfying policies, we develop PCPO–a trust region method that performs policy updates corresponding to reward improvement, followed by projections onto the constraint set. PCPO, inspired by *projected gradient descent*,

is composed of two steps for each update, a reward improvement step and a projection step (This is illustrated in Fig. 2.1).

Reward Improvement Step. First, we optimize the reward function by maximizing the reward advantage function $A_R^\pi(s,a)$ subject to a Kullback-Leibler (KL) divergence constraint. This constraints the intermediate policy $\pi^{k+\frac{1}{2}}$ to be within a δ-neighbourhood of π^k:

$$\begin{aligned} \pi^{k+\frac{1}{2}} = \arg\max_{\pi} \quad & \mathbb{E}_{\substack{s\sim d^{\pi^k} \\ a\sim\pi}}[A_R^{\pi^k}(s,a)] \\ \text{s.t.} \quad & \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi||\pi^k)[s]\right] \leq \delta. \end{aligned} \tag{2.6}$$

This update rule with the trust region, $\{\pi : \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi||\pi^k)[s]\right] \leq \delta\}$, is called Trust Region Policy Optimization (TRPO) [133]. It constrains the policy changes to a divergence neighborhood and guarantees reward improvement.

Projection Step. Second, we project the intermediate policy $\pi^{k+\frac{1}{2}}$ onto the constraint set by minimizing a distance measure D between $\pi^{k+\frac{1}{2}}$ and π:

$$\begin{aligned} \pi^{k+1} = \arg\min_{\pi} \quad & D(\pi, \pi^{k+\frac{1}{2}}) \\ \text{s.t.} \quad & J^C(\pi^k) + \mathbb{E}_{\substack{s\sim d^{\pi^k} \\ a\sim\pi}}[A_C^{\pi^k}(s,a)] \leq h. \end{aligned} \tag{2.7}$$

The projection step ensures that the constraint-satisfying policy π^{k+1} is close to $\pi^{k+\frac{1}{2}}$. We consider two distance measures D: L^2 norm and KL divergence. Importantly, using KL divergence projection in the probability distribution space allows us to provide provable guarantees for PCPO.

2.4 Theoretical Analysis

Performance Bound for PCPO with KL Divergence Projection. In safety-critical applications such as autonomous cars, one cares about how worse the performance of a system evolves when applying a learning algorithm. To this end, for PCPO with KL divergence projection, we analyze the worst-case performance degradation for each policy update when the current policy π^k *satisfies* the constraint. The following theorem provides a lower bound on reward improvement, and an upper bound on constraint violation for each policy update.

Theorem 2.4.1 (**Worst-case Bound on Updating Constraint-satisfying Policies**). *Define* $\epsilon_R^{\pi^{k+1}} \doteq \max_s \left|\mathbb{E}_{a\sim\pi^{k+1}}[A_R^{\pi^k}(s,a)]\right|$, *and* $\epsilon_C^{\pi^{k+1}} \doteq \max_s \left|\mathbb{E}_{a\sim\pi^{k+1}}[A_C^{\pi^k}(s,a)]\right|$. *If the current policy* π^k *satisfies the constraint, then under KL divergence projection, the lower bound on reward improvement, and upper bound on constraint violation for each policy update are*

$$J^R(\pi^{k+1}) - J^R(\pi^k) \geq -\frac{\sqrt{2\delta}\gamma\epsilon_R^{\pi^{k+1}}}{(1-\gamma)^2}, \text{ and } J^C(\pi^{k+1}) \leq h + \frac{\sqrt{2\delta}\gamma\epsilon_C^{\pi^{k+1}}}{(1-\gamma)^2},$$

where δ *is the step size in the reward improvement step.*

Proof. To prove the policy performance bound when the current policy is feasible (*i.e.,* constraint-satisfying), we prove the KL divergence between π^k and π^{k+1} for the KL divergence projection. We then prove our main theorem for the worst-case performance degradation.

Lemma 2.4.2. *If the current policy* π^k *satisfies the constraint, the constraint set is closed and convex, the KL divergence constraint for the first step is*

$$\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+\frac{1}{2}}||\pi^k)[s]\right] \leq \delta,$$

where δ *is the step size in the reward improvement step, then under KL divergence*

projection, we have

$$\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]\right] \leq \delta.$$

Proof. By the Bregman divergence projection inequality, π^k being in the constraint set, and π^{k+1} being the projection of the $\pi^{k+\frac{1}{2}}$ onto the constraint set, we have

$$\begin{aligned}
&\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^k||\pi^{k+\frac{1}{2}})[s]\right] \geq \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^k||\pi^{k+1})[s]\right] + \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^{k+\frac{1}{2}})[s]\right] \\
\Rightarrow \delta \geq &\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^k||\pi^{k+\frac{1}{2}})[s]\right] \geq \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^k||\pi^{k+1})[s]\right].
\end{aligned}$$

The derivation uses the fact that KL divergence is always greater than zero. We know that KL divergence is asymptotically symmetric when updating the policy within a local neighborhood. Thus, we have

$$\delta \geq \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+\frac{1}{2}}||\pi^k)[s]\right] \geq \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]\right].$$

□

Now we use Lemma 2.4.2 to prove our main theorem. By the theorem in [6] and Lemma 2.4.2, we have the following reward degradation bound for each policy update:

$$\begin{aligned}
J^R(\pi^{k+1}) - J^R(\pi^k) &\geq \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi^k}\\ a\sim\pi^{k+1}}}\left[A_R^{\pi^k}(s,a) - \frac{2\gamma\epsilon_R^{\pi^{k+1}}}{1-\gamma}\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]}\right] \\
&\geq \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi^k}\\ a\sim\pi^{k+1}}}\left[-\frac{2\gamma\epsilon_R^{\pi^{k+1}}}{1-\gamma}\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]}\right] \\
&\geq -\frac{\sqrt{2\delta}\gamma\epsilon_R^{\pi^{k+1}}}{(1-\gamma)^2}.
\end{aligned}$$

Again, we have the following constraint violation bound for each policy update:

$$J^C(\pi^k) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi^k}\\ a\sim\pi^{k+1}}}\left[A_R^{\pi^k}(s,a)\right] \leq h, \tag{2.8}$$

and

$$J^C(\pi^{k+1}) - J^C(\pi^k) \leq \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi^k} \\ a\sim\pi^{k+1}}}\Big[A_C^{\pi^k}(s,a) + \frac{2\gamma\epsilon_C^{\pi^{k+1}}}{1-\gamma}\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]}\Big]. \tag{2.9}$$

Combining Eq. (2.8) and Eq. (2.9), we have

$$\begin{aligned} J^C(\pi^{k+1}) &\leq h + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi^k} \\ a\sim\pi^{k+1}}}\Big[\frac{2\gamma\epsilon_C^{\pi^{k+1}}}{1-\gamma}\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]}\Big] \\ &\leq h + \frac{\sqrt{2\delta}\gamma\epsilon_C^{\pi^{k+1}}}{(1-\gamma)^2}. \end{aligned}$$

□

Theorem 2.4.1 indicates that if the step size δ is small, the worst-case performance degradation is tolerable.

Due to approximation errors or the random initialization of policies, PCPO may have a constraint-violating update. Theorem 2.4.1 does not give the guarantee on updating a *constraint-violating* policy. Hence we analyze worst-case performance degradation for each policy update when the current policy π^k *violates* the constraint. The following theorem provides a lower bound on reward improvement, and an upper bound on constraint violation for each policy update.

Theorem 2.4.3 (**Worst-case Bound on Updating Constraint-violating Policies**). *Define* $\epsilon_R^{\pi^{k+1}} \doteq \max_s \big|\mathbb{E}_{a\sim\pi^{k+1}}[A_R^{\pi^k}(s,a)]\big|$, $\epsilon_C^{\pi^{k+1}} \doteq \max_s \big|\mathbb{E}_{a\sim\pi^{k+1}}[A_C^{\pi^k}(s,a)]\big|$, $b^+ \doteq \max(0, J^C(\pi^k) - h)$, *and* $\alpha_{\mathrm{KL}} \doteq \frac{1}{2\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{a}}$, *where* $\boldsymbol{a}$ *is the gradient of the cost advantage function and* $\boldsymbol{H}$ *is the Hessian of the KL divergence constraint. If the current policy* π^k *violates the constraint, then under KL divergence projection, the lower bound on reward improvement and the upper bound on constraint violation for each policy*

update are

$$J^R(\pi^{k+1}) - J^R(\pi^k) \geq -\frac{\sqrt{2(\delta + {b^+}^2\alpha_{\mathrm{KL}})}\gamma\epsilon_R^{\pi^{k+1}}}{(1-\gamma)^2},$$
$$\textit{and } J^C(\pi^{k+1}) \leq h + \frac{\sqrt{2(\delta + {b^+}^2\alpha_{\mathrm{KL}})}\gamma\epsilon_C^{\pi^{k+1}}}{(1-\gamma)^2},$$

where δ is the step size in the reward improvement step.

Proof. To prove the policy performance bound when the current policy is infeasible (*i.e.,* constraint-violating), we prove the KL divergence between π^k and π^{k+1} for the KL divergence projection. We then prove our main theorem for the worst-case performance degradation.

Lemma 2.4.4. *If the current policy π^k violates the constraint, the constraint set is closed and convex, the KL divergence constraint for the first step is $\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+\frac{1}{2}}||\pi^k)[s]\right] \leq \delta$, where δ is the step size in the reward improvement step, then under the KL divergence projection, we have*

$$\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]\right] \leq \delta + {b^+}^2\alpha_{\mathrm{KL}},$$

where $\alpha_{\mathrm{KL}} \doteq \frac{1}{2\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{a}}$, $\boldsymbol{a}$ is the gradient of the cost advantage function, $\boldsymbol{H}$ is the Hessian of the KL divergence constraint, and $b^+ \doteq \max(0, J^C(\pi^k) - h)$.

Proof. We define the sublevel set of cost constraint functions for the current infeasible policy π^k:

$$L^{\pi^k} = \{\pi \mid J^C(\pi^k) + \mathbb{E}_{\substack{s\sim d^{\pi^k}\\ a\sim\pi}}[A_C^{\pi^k}(s,a)] \leq J^C(\pi^k)\}.$$

This implies that the current policy π^k lies in L^{π^k}, and $\pi^{k+\frac{1}{2}}$ is projected onto the constraint set: $\{\pi \mid J^C(\pi^k) + \mathbb{E}_{\substack{s\sim d^{\pi^k}\\ a\sim\pi}}[A_C^{\pi^k}(s,a)] \leq h\}$. Next, we define the policy π_l^{k+1} as the projection of $\pi^{k+\frac{1}{2}}$ onto L^{π^k}.

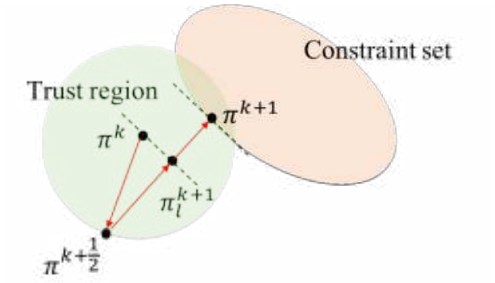


Figure 2.2: Update procedures for PCPO when the current policy π^k is infeasible. π_l^{k+1} is the projection of $\pi^{k+\frac{1}{2}}$ onto the sublevel set of the constraint set. We find the KL divergence between π^k and π^{k+1}.

By the Three-point Lemma, for these three polices π^k, π^{k+1}, and π_l^{k+1}, with $\varphi(\boldsymbol{x}) \doteq \sum_i x_i \log x_i$ (this is illustrated in Fig. 2.2), we have

$$\begin{aligned}
\delta \geq \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi_l^{k+1}||\pi^k)[s]\right] &= \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]\right] \\
&\quad - \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi_l^{k+1})[s]\right] \\
&\quad + \mathbb{E}_{s\sim d^{\pi^k}}\left[(\nabla\varphi(\pi^k) - \nabla\varphi(\pi_l^{k+1}))^T(\pi^{k+1} - \pi_l^{k+1})[s]\right] \\
\Rightarrow \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]\right] &\leq \delta + \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi_l^{k+1})[s]\right] \\
&\quad - \mathbb{E}_{s\sim d^{\pi^k}}\left[(\nabla\varphi(\pi^k) - \nabla\varphi(\pi_l^{k+1}))^T(\pi^{k+1} - \pi_l^{k+1})[s]\right].
\end{aligned} \tag{2.10}$$

The inequality $\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi_l^{k+1}||\pi^k)[s]\right] \leq \delta$ comes from that π^k and π_l^{k+1} are in L^{π^k}, and Lemma 2.4.2.

If the constraint violation of the current policy π^k is small, *i.e.,* b^+ is small, $\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi_l^{k+1})[s]\right]$ can be approximated by the second order expansion. By

the update rule in Eq. (2.14), we have

$$\begin{aligned}
\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi_l^{k+1})[s]\right] &\approx \frac{1}{2}(\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}_l^{k+1})^T\boldsymbol{H}(\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}_l^{k+1}) \\
&= \frac{1}{2}\Big(\frac{b^+}{\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{a}}\boldsymbol{H}^{-1}\boldsymbol{a}\Big)^T\boldsymbol{H}\Big(\frac{b^+}{\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{a}}\boldsymbol{H}^{-1}\boldsymbol{a}\Big) \\
&= \frac{{b^+}^2}{2\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{a}} \\
&= {b^+}^2\alpha_{\mathrm{KL}}, \qquad (2.11)
\end{aligned}$$

where $\alpha_{\mathrm{KL}} \doteq \frac{1}{2\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{a}}$.

And since δ is small, we have $\nabla\varphi(\pi^k) - \nabla\varphi(\pi_l^{k+1}) \approx \mathbf{0}$ given s. Thus, the third term in Eq. (2.10) can be eliminated.

Combining Eq. (2.10) and Eq. (2.11), we have

$$\mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi^{k+1}||\pi^k)[s]\right] \leq \delta + {b^+}^2\alpha_{\mathrm{KL}}.$$

□

Now we use Lemma 2.4.4 to prove our main theorem. Following the same proof in Theorem 2.4.1, we complete the proof. □

Theorem 2.4.3 indicates that when the policy has greater constraint violation (b^+ increases), its worst-case performance degradation increases. Note that Theorem 2.4.3 reduces to Theorem 2.4.1 if the current policy π^k satisfies the constraint ($b^+ = 0$).

2.5 PCPO Updates

For a large neural network policy with many parameters, it is impractical to directly solve for the PCPO update in Problem 2.6 and Problem 2.7 due to the computational cost. However, with a small step size δ, we can approximate the reward function and constraints with a first order expansion, and approximate the KL divergence

constraint in the reward improvement step, and the KL divergence measure in the projection step with a second order expansion. We now make several definitions: $\boldsymbol{g} \doteq \nabla_{\boldsymbol{\theta}} \mathbb{E}_{s\sim d^{\pi^k} a\sim\pi}[A_R^{\pi^k}(s,a)]$ is the gradient of the reward advantage function, $\boldsymbol{a} \doteq \nabla_{\boldsymbol{\theta}} \mathbb{E}_{s\sim d^{\pi^k} a\sim\pi}[A_C^{\pi^k}(s,a)]$ is the gradient of the cost advantage function, $\boldsymbol{H}_{i,j} \doteq \frac{\partial^2 \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi||\pi^k)[s]\right]}{\partial \boldsymbol{\theta}_j \partial \boldsymbol{\theta}_j}$ is the Hessian of the KL divergence constraint ($\boldsymbol{H}$ is also called the Fisher information matrix. It is symmetric positive semi-definite), $b \doteq J^C(\pi^k) - h$ is the constraint violation of the policy π^k, and $\boldsymbol{\theta}$ is the parameter of the policy.

2.5.1 Update Procedure

Reward Improvement Step. We linearize the objective function at π^k subject to second order approximation of the KL divergence constraint in order to obtain the following updates:

$$\begin{aligned} \boldsymbol{\theta}^{k+\frac{1}{2}} = \arg\max_{\boldsymbol{\theta}} \quad & \boldsymbol{g}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \\ \text{s.t.} \quad & \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \leq \delta. \end{aligned} \tag{2.12}$$

Projection Step. If the projection is defined in the parameter space, we can directly use L^2 norm projection. On the other hand, if the projection is defined in the probability space, we can use KL divergence projection. This can be approximated through the second order expansion. Again, we linearize the cost constraint at π^k. This gives the following update for the projection step:

$$\begin{aligned} \boldsymbol{\theta}^{k+1} = \arg\min_{\boldsymbol{\theta}} \quad & \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{2}})^T \boldsymbol{L}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{2}}) \\ \text{s.t.} \quad & \boldsymbol{a}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + b \leq 0, \end{aligned} \tag{2.13}$$

where $\boldsymbol{L} = \boldsymbol{I}$ for L^2 norm projection, and $\boldsymbol{L} = \boldsymbol{H}$ for KL divergence projection.

Algorithm 1 Projection-Based Constrained Policy Optimization (PCPO)

Initialize policy $\pi^0 = \pi(\boldsymbol{\theta}^0)$
for $k = 0, 1, 2, \cdots$ **do**
  Run $\pi^k = \pi(\boldsymbol{\theta}^k)$ and store trajectories in $\mathcal{D}$
  Compute $\boldsymbol{g}, \boldsymbol{a}, \boldsymbol{H}$, and b using $\mathcal{D}$
  Obtain $\boldsymbol{\theta}^{k+1}$ using update in Eq. (2.14)
  Empty $\mathcal{D}$

One may argue that using a linear approximation to the constraint set is not enough to ensure constraint satisfaction since the real constraint set is maybe non-convex. However, if the step size δ is small, then the linearization of the constraint set is accurate enough to locally approximate it.

Final PCPO Step. We solve Problem (2.12) and Problem (2.13) using convex programming. For each policy update, we have

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \sqrt{\frac{2\delta}{\boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g}}} \boldsymbol{H}^{-1} \boldsymbol{g} - \max \left(0, \frac{\sqrt{\frac{2\delta}{\boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g}}} \boldsymbol{a}^T \boldsymbol{H}^{-1} \boldsymbol{g} + b}{\boldsymbol{a}^T \boldsymbol{L}^{-1} \boldsymbol{a}} \right) \boldsymbol{L}^{-1} \boldsymbol{a}. \quad (2.14)$$

We assume that $\boldsymbol{H}$ does not have 0 as an eigenvalue and hence it is invertible. PCPO requires to invert $\boldsymbol{H}$, which is impractical for huge neural network policies. Hence we use the conjugate gradient method [133]. Algorithm 1 shows the pseudocode.

The derivation of the PCPO update in Eq. (2.14) is shown as follows.

Proof. For the first problem in Eq. (2.12), since $\boldsymbol{H}$ is the Fisher Information matrix, which automatically guarantees it is positive semi-definite. Hence it is a convex program with quadratic inequality constraints. Hence if the primal problem has a feasible point, then Slater's condition is satisfied and strong duality holds. Let $\boldsymbol{\theta}^*$ and λ^* denote the solutions to the primal and dual problems, respectively. In addition, the primal objective function is continuously differentiable. Hence the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for the optimality of $\boldsymbol{\theta}^*$ and λ^*.

We now form the Lagrangian:

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = -\boldsymbol{g}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + \lambda\Big(\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T\boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) - \delta\Big).$$

And we have the following KKT conditions:

$$-\boldsymbol{g} + \lambda^*\boldsymbol{H}\boldsymbol{\theta}^* - \lambda^*\boldsymbol{H}\boldsymbol{\theta}^k = 0 \qquad \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{2.15}$$

$$\frac{1}{2}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k)^T\boldsymbol{H}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) - \delta = 0 \qquad \nabla_{\lambda}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{2.16}$$

$$\frac{1}{2}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k)^T\boldsymbol{H}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) - \delta \leq 0 \qquad \text{primal constraints} \tag{2.17}$$

$$\lambda^* \geq 0 \qquad \text{dual constraints} \tag{2.18}$$

$$\lambda^*\Big(\frac{1}{2}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k)^T\boldsymbol{H}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) - \delta\Big) = 0 \qquad \text{complementary slackness} \tag{2.19}$$

By Eq. (2.15), we have $\boldsymbol{\theta}^* = \boldsymbol{\theta}^k + \frac{1}{\lambda^*}\boldsymbol{H}^{-1}\boldsymbol{g}$. And by plugging Eq. (2.15) into Eq. (2.16), we have $\lambda^* = \sqrt{\frac{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}{2\delta}}$. Hence we have our optimal solution:

$$\boldsymbol{\theta}^{k+\frac{1}{2}} = \boldsymbol{\theta}^* = \boldsymbol{\theta}^k + \sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{H}^{-1}\boldsymbol{g}, \tag{2.20}$$

which also satisfies Eq. (2.17), Eq. (2.18), and Eq. (2.19).

Following the same reasoning, we now form the Lagrangian of the second problem in Eq. (2.13):

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{2}})^T\boldsymbol{L}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{2}}) + \lambda(\boldsymbol{a}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + b).$$

And we have the following KKT conditions:

$$\boldsymbol{L}\boldsymbol{\theta}^* - \boldsymbol{L}\boldsymbol{\theta}^{k+\frac{1}{2}} + \lambda^*\boldsymbol{a} = 0 \qquad \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{2.21}$$

$$\boldsymbol{a}^T(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + b = 0 \qquad \nabla_{\lambda}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{2.22}$$

$$\boldsymbol{a}^T(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + b \leq 0 \qquad \text{primal constraints} \tag{2.23}$$

$$\lambda^* \geq 0 \qquad \text{dual constraints} \tag{2.24}$$

$$\lambda^*(\boldsymbol{a}^T(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + b) = 0 \qquad \text{complementary slackness} \tag{2.25}$$

By Eq. (2.21), we have $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{k+1} + \lambda^*\boldsymbol{L}^{-1}\boldsymbol{a}$. And by plugging Eq. (2.21) into Eq. (2.22) and Eq. (2.24), we have $\lambda^* = \max(0, \frac{\boldsymbol{a}^T(\boldsymbol{\theta}^{k+\frac{1}{2}} - \boldsymbol{\theta}^k) + b}{\boldsymbol{a}\boldsymbol{L}^{-1}\boldsymbol{a}})$. Hence we have our optimal solution:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^* = \boldsymbol{\theta}^{k+\frac{1}{2}} - \max(0, \frac{\boldsymbol{a}^T(\boldsymbol{\theta}^{k+\frac{1}{2}} - \boldsymbol{\theta}^k) + b}{\boldsymbol{a}^T\boldsymbol{L}^{-1}\boldsymbol{a}^T})\boldsymbol{L}^{-1}\boldsymbol{a}, \tag{2.26}$$

which also satisfies Eq. (2.23) and Eq. (2.25). Hence by Eq. (2.20) and Eq. (2.26), we have

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{H}^{-1}\boldsymbol{g} - \max(0, \frac{\sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{g} + b}{\boldsymbol{a}^T\boldsymbol{L}^{-1}\boldsymbol{a}})\boldsymbol{L}^{-1}\boldsymbol{a}.$$

□

2.5.2 Analysis of PCPO Update Rule

For a problem including multiple constraints, we can extend the update in Eq. (2.14) by using alternating projections. This approach finds a solution in the intersection of multiple constraint sets by sequentially projecting onto each of the sets. The update rule in Eq. (2.14) shows that the difference between PCPO with KL divergence and L^2 norm projections is the cost update direction, leading to a difference in reward

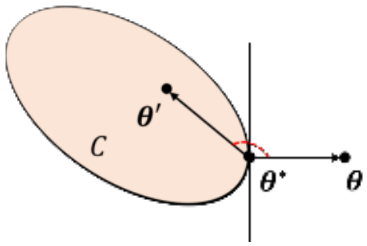


Figure 2.3: The projection onto the convex set with $\boldsymbol{\theta}' \in \mathcal{C}$ and $\boldsymbol{\theta}^* = \text{Proj}_{\mathcal{C}}^{\boldsymbol{L}}(\boldsymbol{\theta})$.

improvement. These two projections converge to different stationary points with different convergence rates related to the smallest and largest singular values of the Fisher information matrix shown in Theorem 2.5.1. For our analysis, we make the following assumptions: we *minimize* the negative reward objective function $f : \mathbb{R}^n \to \mathbb{R}$ (We follow the convention of the literature that authors typically minimize the objective function). The function f is L-smooth and twice continuously differentiable over the closed and convex constraint set $\mathcal{C}$.

Theorem 2.5.1 (**Reward Improvement Under L^2 Norm and KL Divergence Projections**). *Let $\eta \doteq \sqrt{\frac{2\delta}{\boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g}}}$ in Eq. (2.14), where δ is the step size for reward improvement, $\boldsymbol{g}$ is the gradient of f, and $\boldsymbol{H}$ is the Fisher information matrix. Let $\sigma_{\max}(\boldsymbol{H})$ be the largest singular value of $\boldsymbol{H}$, and $\boldsymbol{a}$ be the gradient of cost advantage function in Eq. (2.14). Then PCPO with KL divergence projection converges to a stationary point either inside the constraint set or in the boundary of the constraint set. In the latter case, the Lagrangian constraint $\boldsymbol{g} = -\alpha \boldsymbol{a}, \alpha \geq 0$ holds. Moreover, at step $k+1$ the objective value satisfies*

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^k) + ||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||^2_{-\frac{1}{\eta}\boldsymbol{H} + \frac{L}{2}\boldsymbol{I}}.$$

PCPO with L^2 norm projection converges to a stationary point either inside the constraint set or in the boundary of the constraint set. In the latter case, the Lagrangian constraint $\boldsymbol{H}^{-1}\boldsymbol{g} = -\alpha \boldsymbol{a}, \alpha \geq 0$ holds. If $\sigma_{\max}(\boldsymbol{H}) \leq 1$, then a step $k+1$ objective

value satisfies

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^k) + (\frac{L}{2} - \frac{1}{\eta})||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2.$$

Proof. We have the following lemma to characterize the projection and for the proof of Theorem 2.5.1. (See Fig. 2.3 for semantic illustration.)

Lemma 2.5.2. *For any* $\boldsymbol{\theta}$, $\boldsymbol{\theta}^* = \text{Proj}_{\mathcal{C}}^{\boldsymbol{L}}(\boldsymbol{\theta})$ *if and only if* $(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{L}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) \leq 0, \forall \boldsymbol{\theta}' \in \mathcal{C}$, *where* $\text{Proj}_{\mathcal{C}}^{\boldsymbol{L}}(\boldsymbol{\theta}) \doteq \arg\min_{\boldsymbol{\theta}' \in \mathcal{C}} ||\boldsymbol{\theta} - \boldsymbol{\theta}'||_{\boldsymbol{L}}^2$, *and* $\boldsymbol{L} = \boldsymbol{H}$ *if using the KL divergence projection, and* $\boldsymbol{L} = \boldsymbol{I}$ *if using the* L^2 *norm projection.*

Proof. ($\Rightarrow$) Let $\boldsymbol{\theta}^* = \text{Proj}_{\mathcal{C}}^{\boldsymbol{L}}(\boldsymbol{\theta})$ for a given $\boldsymbol{\theta} \notin \mathcal{C}$, $\boldsymbol{\theta}' \in \mathcal{C}$ be such that $\boldsymbol{\theta}' \neq \boldsymbol{\theta}^*$, and $\alpha \in (0, 1)$. Then we have

$$\begin{aligned} ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 &\leq ||\boldsymbol{\theta} - \big(\boldsymbol{\theta}^* + \alpha(\boldsymbol{\theta}' - \boldsymbol{\theta}^*)\big)||_{\boldsymbol{L}}^2 \\ &= ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 + \alpha^2||\boldsymbol{\theta}' - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 - 2\alpha(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{L}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) \\ \Rightarrow (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{L}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) &\leq \frac{\alpha}{2}||\boldsymbol{\theta}' - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2. \end{aligned} \tag{2.27}$$

Since the right hand side of Eq. (2.27) can be made arbitrarily small for a given α, and hence we have:

$$(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{L}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) \leq 0, \forall \theta' \in \mathcal{C}.$$

($\Leftarrow$) Let $\boldsymbol{\theta}^* \in \mathcal{C}$ be such that $(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{L}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) \leq 0, \forall \theta' \in \mathcal{C}$. We show that $\boldsymbol{\theta}^*$ must be the optimal solution. Let $\boldsymbol{\theta}' \in \mathcal{C}$ and $\boldsymbol{\theta}' \neq \boldsymbol{\theta}^*$. Then we have

$$\begin{aligned} ||\boldsymbol{\theta} - \boldsymbol{\theta}'||_{\boldsymbol{L}}^2 - ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 &= ||\boldsymbol{\theta} - \boldsymbol{\theta}^* + \boldsymbol{\theta}^* - \boldsymbol{\theta}'||_{\boldsymbol{L}}^2 - ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 \\ &= ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 + ||\boldsymbol{\theta}' - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 - 2(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{L}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) - ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2 \\ &> 0 \\ \Rightarrow ||\boldsymbol{\theta} - \boldsymbol{\theta}'||_{\boldsymbol{L}}^2 &> ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||_{\boldsymbol{L}}^2. \end{aligned}$$

Hence, $\boldsymbol{\theta}^*$ is the optimal solution to the optimization problem, and $\boldsymbol{\theta}^* = \text{Proj}_{\mathcal{C}}^{\boldsymbol{L}}(\boldsymbol{\theta})$. □

The proof of Theorem 2.5.1 is based on working in a Hilbert space and the non-expansive property of the projection. We first prove stationary points for PCPO with the KL divergence and L^2 norm projections, and then prove the change of the objective value.

When in stationary points $\boldsymbol{\theta}^*$, we have

$$\begin{aligned}
&a\boldsymbol{\theta}^* = \boldsymbol{\theta}^* - \sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{H}^{-1}\boldsymbol{g} - \max(0, \frac{\sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{g} + b}{\boldsymbol{a}^T\boldsymbol{L}^{-1}\boldsymbol{a}})\boldsymbol{L}^{-1}\boldsymbol{a}. \\
\Leftrightarrow &\sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{H}^{-1}\boldsymbol{g} = -\max(0, \frac{\sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}}\boldsymbol{a}^T\boldsymbol{H}^{-1}\boldsymbol{g} + b}{\boldsymbol{a}^T\boldsymbol{L}^{-1}\boldsymbol{a}})\boldsymbol{L}^{-1}\boldsymbol{a} \\
\Leftrightarrow &\boldsymbol{H}^{-1}\boldsymbol{g} \in -\boldsymbol{L}^{-1}\boldsymbol{a}. \qquad (2.28)
\end{aligned}$$

For the KL divergence projection ($\boldsymbol{L} = \boldsymbol{H}$), Eq. (2.28) boils down to $\boldsymbol{g} \in -\boldsymbol{a}$, and for the L^2 norm projection ($\boldsymbol{L} = \boldsymbol{I}$), Eq. (2.28) is equivalent to $\boldsymbol{H}^{-1}\boldsymbol{g} \in -\boldsymbol{a}$.

Now we prove the second part of the theorem. Based on Lemma 2.5.2, for the KL divergence projection, we have

$$\begin{aligned}
&(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+1})^T\boldsymbol{H}(\boldsymbol{\theta}^k - \eta\boldsymbol{H}^{-1}\boldsymbol{g} - \boldsymbol{\theta}^{k+1}) \leq 0 \\
&\quad \Rightarrow \boldsymbol{g}^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \leq -\frac{1}{\eta}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||^2_{\boldsymbol{H}}. \qquad (2.29)
\end{aligned}$$

By Eq. (2.29), and L-smooth continuous function f, we have

$$\begin{aligned}
f(\boldsymbol{\theta}^{k+1}) &\leq f(\boldsymbol{\theta}^k) + \boldsymbol{g}^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) + \frac{L}{2}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2 \\
&\leq f(\boldsymbol{\theta}^k) - \frac{1}{\eta}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||^2_{\boldsymbol{H}} + \frac{L}{2}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2 \\
&= f(\boldsymbol{\theta}^k) + (\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k)^T(-\frac{1}{\eta}\boldsymbol{H} + \frac{L}{2}\boldsymbol{I})(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \\
&= f(\boldsymbol{\theta}^k) + ||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||^2_{-\frac{1}{\eta}\boldsymbol{H} + \frac{L}{2}\boldsymbol{I}}.
\end{aligned}$$

For the L^2 norm projection, we have

$$\begin{aligned}
&(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+1})^T(\boldsymbol{\theta}^k - \eta \boldsymbol{H}^{-1}\boldsymbol{g} - \boldsymbol{\theta}^{k+1}) \leq 0 \\
\Rightarrow\ &\boldsymbol{g}^T \boldsymbol{H}^{-1}(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \leq -\frac{1}{\eta}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2. \qquad (2.30)
\end{aligned}$$

By Eq. (2.30), L-smooth continuous function f, and if $\sigma_{\max}(\boldsymbol{H}) \leq 1$, we have

$$\begin{aligned}
f(\boldsymbol{\theta}^{k+1}) &\leq f(\boldsymbol{\theta}^k) + \boldsymbol{g}^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) + \frac{L}{2}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2 \\
&\leq f(\boldsymbol{\theta}^k) + (\frac{L}{2} - \frac{1}{\eta})||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2.
\end{aligned}$$

To see why we need the assumption of $\sigma_{\max}(\boldsymbol{H}) \leq 1$, we define $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ as the singular value decomposition of $\boldsymbol{H}$ with $\boldsymbol{u}_i$ being the column vector of $\boldsymbol{U}$. Then we have

$$\begin{aligned}
\boldsymbol{g}^T \boldsymbol{H}^{-1}(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) &= \boldsymbol{g}^T \boldsymbol{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \\
&= \boldsymbol{g}^T(\sum_i \frac{1}{\sigma_i(\boldsymbol{H})}\boldsymbol{u}_i\boldsymbol{u}_i^T)(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \\
&= \sum_i \frac{1}{\sigma_i(\boldsymbol{H})}\boldsymbol{g}^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k).
\end{aligned}$$

If we want to have

$$\boldsymbol{g}^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \leq \boldsymbol{g}^T \boldsymbol{H}^{-1}(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k) \leq -\frac{1}{\eta}||\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k||_2^2,$$

then every singular value $\sigma_i(\boldsymbol{H})$ of $\boldsymbol{H}$ needs to be smaller than 1, and hence $\sigma_{\max}(\boldsymbol{H}) \leq 1$, which justifies the assumption we use to prove the bound. □

Theorem 2.5.1 shows that in the stationary point $\boldsymbol{g}$ is a line that points to the opposite direction of $\boldsymbol{a}$. Further, the improvement of the objective value is affected by the singular value of the Fisher information matrix. Specifically, the objective of KL

divergence projection decreases when $\frac{L\eta}{2}\boldsymbol{I} \prec \boldsymbol{H}$, implying that $\sigma_{\min}(\boldsymbol{H}) > \frac{L\eta}{2}$. And the objective of L^2 norm projection decreases when $\eta < \frac{2}{L}$, implying that condition number of $\boldsymbol{H}$ is upper bounded: $\frac{\sigma_{\max}(\boldsymbol{H})}{\sigma_{\min}(\boldsymbol{H})} < \frac{2||\boldsymbol{g}||_2^2}{L^2\delta}$. This is because that

$$
\begin{aligned}
&\eta = \sqrt{\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}} < \frac{2}{L} \\
\Rightarrow &\frac{2\delta}{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}} < \frac{4}{L^2} \\
\Rightarrow &\frac{\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}}{2\delta} > \frac{L^2}{4} \\
\Rightarrow &\frac{L^2\delta}{2} < \boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g} \\
&\leq ||\boldsymbol{g}||_2||\boldsymbol{H}^{-1}\boldsymbol{g}||_2 \\
&\leq ||\boldsymbol{g}||_2||\boldsymbol{H}^{-1}||_2||\boldsymbol{g}||_2 \\
&= \sigma_{\max}(\boldsymbol{H}^{-1})||\boldsymbol{g}||_2^2 \\
&= \sigma_{\min}(\boldsymbol{H})||\boldsymbol{g}||_2^2 \\
\Rightarrow &\sigma_{\min}(\boldsymbol{H}) > \frac{L^2\delta}{2||\boldsymbol{g}||_2^2}.
\end{aligned} \tag{2.31}
$$

By the definition of the condition number and Eq. (2.31), we have

$$
\begin{aligned}
&\frac{1}{\sigma_{\min}(\boldsymbol{H})} < \frac{2||\boldsymbol{g}||_2^2}{L^2\delta} \\
\Rightarrow &\frac{\sigma_{\max}(\boldsymbol{H})}{\sigma_{\min}(\boldsymbol{H})} < \frac{2||\boldsymbol{g}||_2^2\sigma_{\max}(\boldsymbol{H})}{L^2\delta} \\
&\leq \frac{2||\boldsymbol{g}||_2^2}{L^2\delta},
\end{aligned}
$$

which justifies what we discuss. In summary, observing the singular values of the Fisher information matrix allows us to adaptively choose the appropriate projection and hence achieve objective improvement.

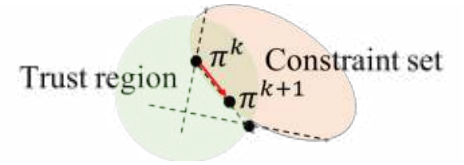


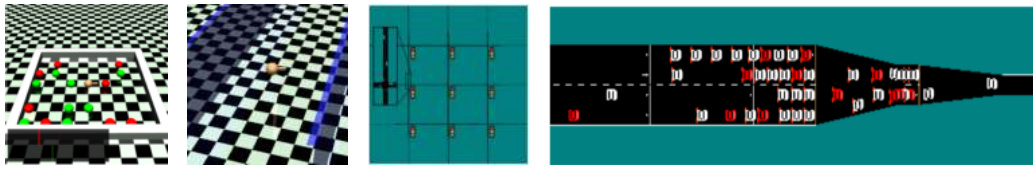
Figure 2.4: Update procedures for CPO [6]. CPO computes the update by simultaneously considering the trust region (light green) and the constraint set (light orange). CPO becomes infeasible when these two sets do not intersect.

2.5.3 Comparison to Constrained Policy Optimization (CPO)

Perhaps the closest work to ours is the approach of [6], which proposes the constrained policy optimization (CPO) algorithm to solve the following:

$$\boldsymbol{\theta}^{k+1} = \arg\max_{\boldsymbol{\theta}} \ \boldsymbol{g}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \quad \text{s.t. } \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \leq \delta, \ \boldsymbol{a}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + b \leq 0. \tag{2.32}$$

CPO simultaneously considers the trust region and the constraint, and uses the line search to select a step size (This is illustrated in Fig. 2.4). The update rule of CPO becomes infeasible when the current policy violates the constraint ($b > 0$). CPO recovers by replacing Problem (2.32) with an update to purely decrease the constraint value: $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \sqrt{\frac{2\delta}{\boldsymbol{a}^T \boldsymbol{H}^{-1} \boldsymbol{a}}} \boldsymbol{H}^{-1} \boldsymbol{a}$. This update rule may lead to slow progress in learning constraint-satisfying policies. In contrast, PCPO first optimizes the reward and uses the projection to satisfy the constraint. This ensures a feasible solution, allowing the agent to improve the reward while ensuring constraint satisfaction simultaneously.



(a) Gather (b) Circle (c) Grid (d) Bottleneck

Figure 2.5: The gather, circle, grid and bottleneck tasks. (a) Gather task: the agent is rewarded for gathering green apples but is constrained to collect a limited number of red fruit [6]. (b) Circle task: the agent is rewarded for moving in a specified wide circle, but is constrained to stay within a safe region smaller than the radius of the circle [6]. (c) Grid task: the agent controls the traffic lights in a grid road network and is rewarded for high throughput but constrained to let lights stay red for at most 7 consecutive seconds [161]. (d) Bottleneck task: the agent controls a set of autonomous vehicles (shown in red) in a traffic merge situation and is rewarded for achieving high throughput but constrained to ensure that human-driven vehicles (shown in white) have a low speed for no more than 10 seconds [161].

2.6 Experiments

2.6.1 Setup

Tasks. We compare the proposed algorithm with existing approaches on four control tasks in total: two tasks with safety constraints ((a) and (b) in Fig. 2.5), and two tasks with fairness constraints ((c) and (d) in Fig. 2.5). These tasks are briefly described in the caption of Fig. 2.5. The first two tasks–*Gather* and *Circle*–are Mujoco environments with state space constraints introduced by [6]. The other two tasks–*Grid* and *Bottleneck*–are traffic management problems where the agent controls either a traffic light or a fleet of autonomous vehicles. This is especially challenging since the dimensions of state and action spaces are larger, and the dynamics of the environment are inherently complex.

Baselines. We compare PCPO with four baselines outlined below.

(1) Constrained Policy Optimization (CPO) [6].

(2) Primal-dual Optimization (PDO) [37]. In PDO, the weight (dual variables) is learned based on the current constraint satisfaction. A PDO policy update

solves:

$$\boldsymbol{\theta}^{k+1} = \arg\max_{\boldsymbol{\theta}} \quad \boldsymbol{g}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + \lambda^k \boldsymbol{a}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k), \tag{2.33}$$

where λ^k is updated using $\lambda^{k+1} = \lambda^k + \beta(J^C(\pi^k) - h)$. Here β is a fixed learning rate.

(3) Fixed-point Policy Optimization (FPO). A variant of PDO that solves Eq. (2.33) using a constant λ.

(4) Trust Region Policy Optimization (TRPO) [133]. The TRPO policy update is an *unconstrained* one:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \sqrt{\frac{2\delta}{\boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g}}} \boldsymbol{H}^{-1} \boldsymbol{g}.$$

Note that TRPO ignores any constraints. We include it to serve as an upper bound baseline on the reward performance. Since the main focus is to compare PCPO with the state-of-the-art algorithm (CPO), PDO and FPO are not shown in the ant circle, ant gather, grid, and bottleneck tasks for clarity.

Experimental Details. For the gather and circle tasks we test two distinct agents: a point-mass ($S \subseteq \mathbb{R}^9, A \subseteq \mathbb{R}^2$), and an ant robot ($S \subseteq \mathbb{R}^{32}, A \subseteq \mathbb{R}^8$). The agent in the grid task is $S \subseteq \mathbb{R}^{156}, A \subseteq \mathbb{R}^4$, and the agent in bottleneck task is $S \subseteq \mathbb{R}^{141}, A \subseteq \mathbb{R}^{20}$. For the simulations in the gather and circle tasks, we use a neural network with two hidden layers of size (64, 32) to represent Gaussian policies. For the simulations in the grid and bottleneck tasks, we use a neural network with two hidden layers of size (16, 16) and (50,25) to represent Gaussian policies, respectively. In addition, we use GAE-λ approach [134] to estimate $A^\pi_R(s,a)$ and $A^\pi_C(s,a)$. For the simulations in the gather and circle tasks, we use neural network baselines with the same architecture and activation functions as the policy networks. For the simulations

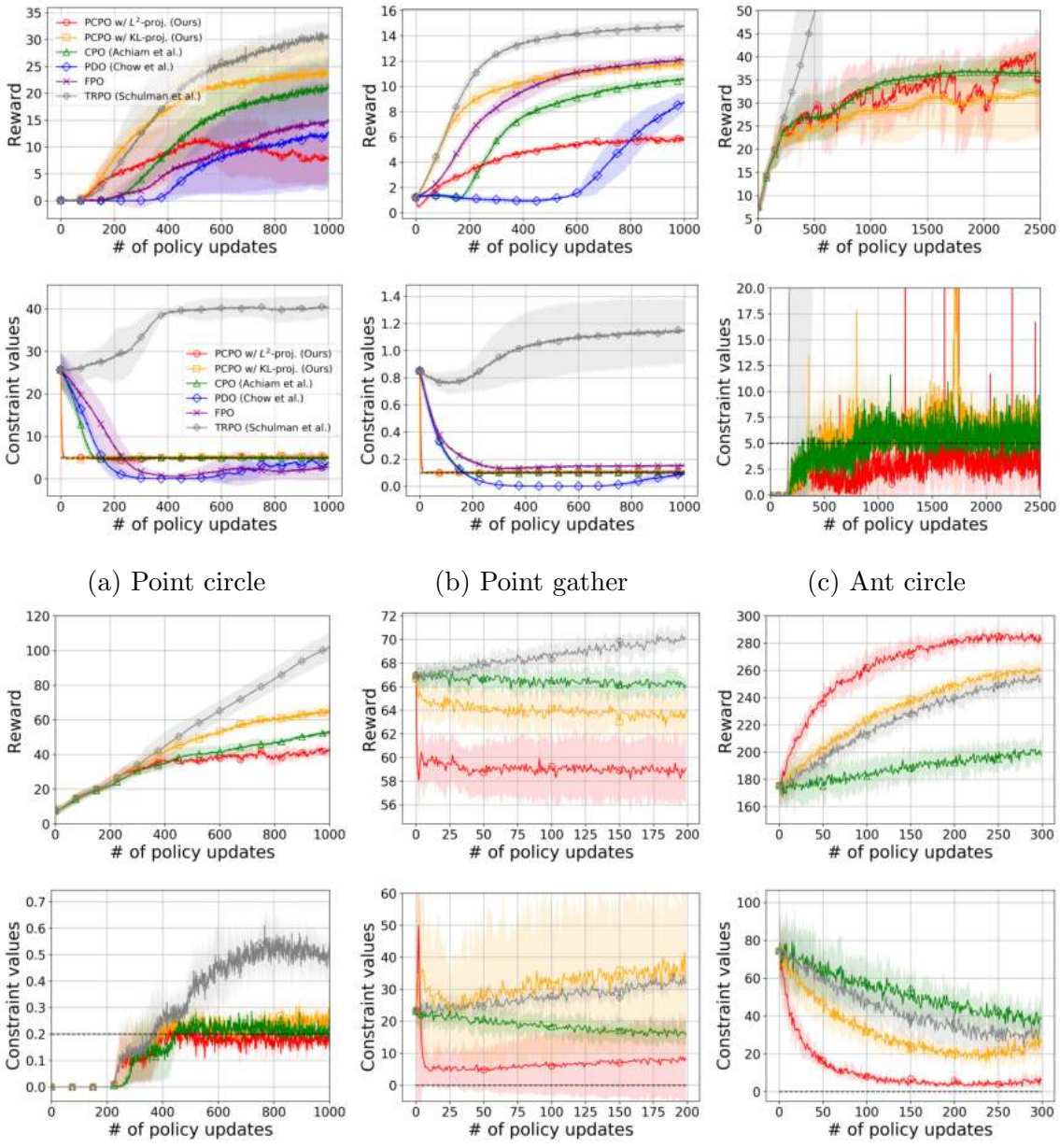
Parameter	PC	PG	AC	AG	Gr	BN
discount factor γ	0.995	0.995	0.995	0.995	0.999	0.999
step size δ	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}
λ_R^{GAE}	0.95	0.95	0.95	0.95	0.97	0.97
λ_C^{GAE}	1.0	1.0	0.5	0.5	0.5	1.0
Batch size	50,000	50,000	100,000	100,000	10,000	25,000
Rollout length	50	15	500	500	400	500
Cost constraint threshold h	5	0.1	10	0.2	0	0

Table 2.1: The hyperparameters used in the experiments.

in the grid and bottleneck tasks, we use linear baselines. In the experiments, since the step size is small, we reuse the Fisher information matrix of the reward improvement step in the KL projection step to reduce the computational cost. The step size δ is set to 10^{-4} for all tasks and all tested algorithms. For each task, we conduct 5 runs to get the mean and standard deviation for both the reward and the constraint value over the policy updates. The experiments are implemented in rllab [46], a tool for developing and evaluating RL algorithms. Table 2.1 shows the hyperparameters of each task for all algorithms are as follows (PC: point circle, PG: point gather, AC: ant circle, AG: ant gather, Gr: grid, and BN: bottleneck tasks): note that we do not use a learned model to predict the probability of entering an undesirable state within a fixed time horizon as CPO did for cost shaping.

2.6.2 Experiment Results

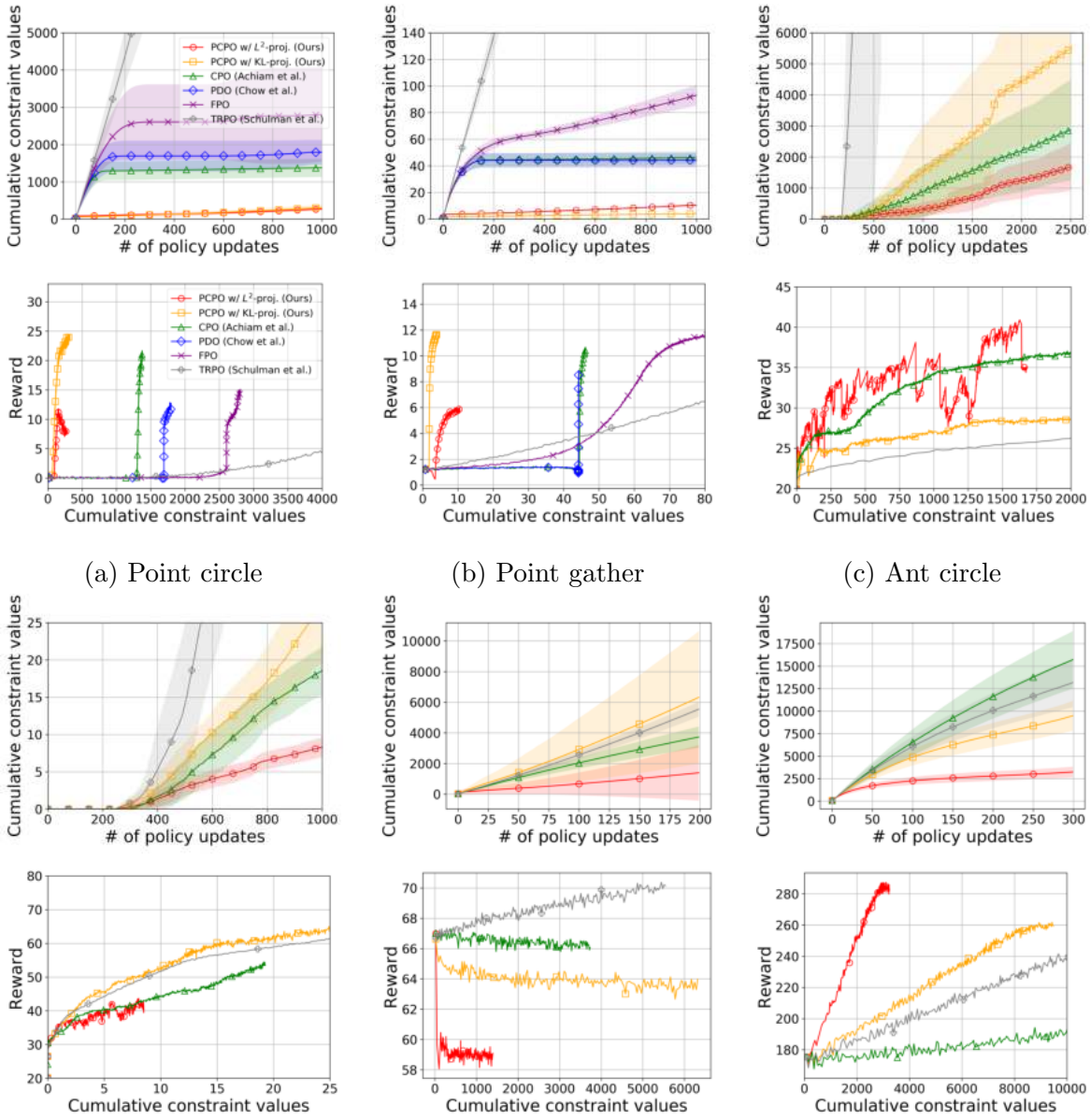
Overall Performance. The learning curves of the discounted reward and the undiscounted constraint value (the total number of constraint violations) over policy updates are shown for all tested algorithms and tasks in Fig. 2.6. The dashed line in the constraint figure is the cost constraint threshold h. The curves for baseline oracle, TRPO, indicate the reward and constraint value when the constraint is ignored. Overall, we find that PCPO is able to improve the reward while having the fastest constraint satisfaction in all tasks. In particular, PCPO is the only algorithm that



(a) Point circle (b) Point gather (c) Ant circle

(d) Ant gather (e) Grid (f) Bottleneck

Figure 2.6: The values of the discounted reward and the undiscounted constraint value (the total number of constraint violations) along with policy updates for the tested algorithms and task pairs. The solid line is the mean and the shaded area is the standard deviation, over five runs. The dashed line in the cost constraint plot is the cost constraint threshold h. The curves for baseline oracle, TRPO, indicate the reward and constraint violation values when the constraint is *ignored*. (Best viewed in color, and the legend is shared across all the figures.)



(a) Point circle (b) Point gather (c) Ant circle

(d) Ant gather (e) Grid (f) Bottleneck

Figure 2.7: The values of the cumulative constraint value over policy update, and the reward versus the cumulative constraint value for the tested algorithms and task pairs. The solid line is the mean and the shaded area is the standard deviation, over five runs. The curves for baseline oracle, TRPO, indicate the performance when the constraint is ignored. (Best viewed in color, and the legend is shared across all the figures.)

learns constraint-satisfying policies across all the tasks. Moreover, we observe that (1) CPO has more constraint violations than PCPO, (2) PDO is too conservative in optimizing the reward, and (3) FPO requires a significant effort to select a good value of λ.

We also observe that in the Grid and Bottleneck task, there is slightly more constraint violation than the easier task such as point circle and point gather. This is due to the complexity of the policy behavior and the non-convexity of the constraint set. However, even with a linear approximation of the constraint set, PCPO still outperforms CPO with 85.15% and 5.42 times less constraint violation in Grid and Bottleneck tasks, respectively.

These observations suggest that the projection step in PCPO drives the agent to learn the constraint-satisfying policy within a few policy updates, giving PCPO an advantage in applications.

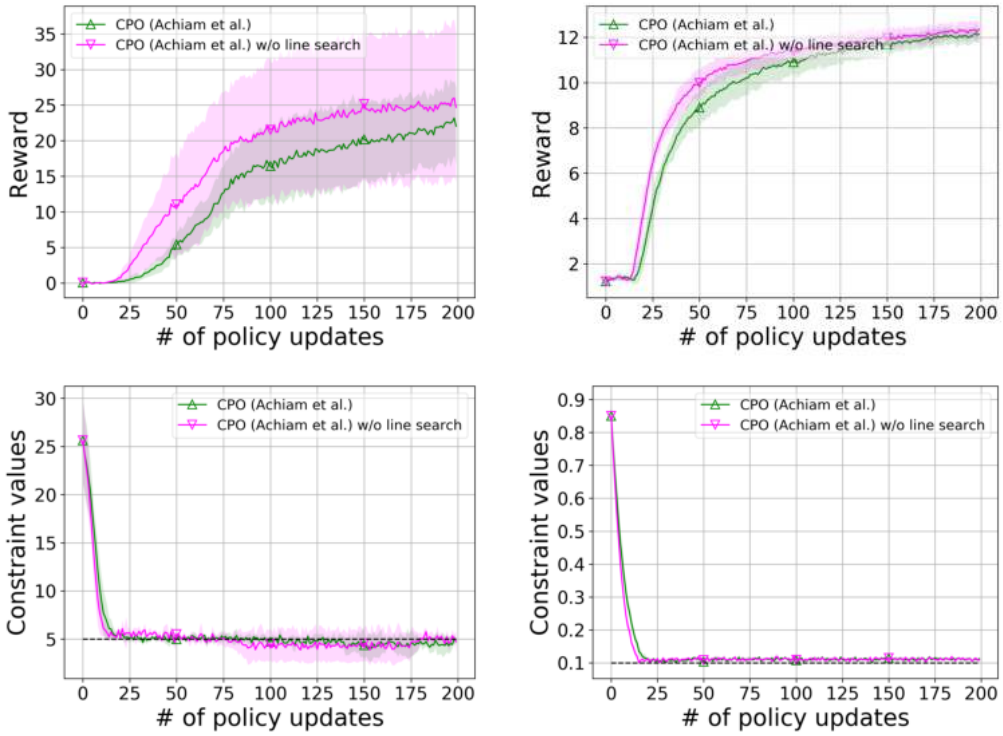
To show that PCPO achieves the same reward with less constraint violation, we examine the reward versus the cumulative constraint value and the learning curves of the cumulative constraint value over policy update for the tested algorithms in Fig. 2.7. We observe that PCPO outperforms CPO significantly with 66 times and 15 times less constraint violation under the same reward improvement in point circle and point gather tasks, respectively. Overall, we find that, (1) CPO has more cumulative constraint violations than PCPO; (2) PCPO with L^2 norm projection has less cumulative constraint violation than KL divergence projection except for the point circle and point gather tasks. This observation suggests that the Fisher information matrix is not well-estimated in the high dimensional policy space, leading to having more constraint violations; (3) PCPO has more reward improvement compared to CPO under the same number of cumulative constraint violations in point circle, point gather, ant circle, ant gather, and bottleneck task. This observation suggests that PCPO enables the agent to cautiously explore the environment under the con-

straints.

Comparison of PCPO with KL Divergence vs. L^2 Norm Projections. In Fig. 2.6 and 2.7, we observe that PCPO with L^2 norm projection is more constraint-satisfying than PCPO with KL divergence projection. In addition, PCPO with L^2 norm projection tends to have reward fluctuation (point circle, ant circle, and ant gather tasks), while PCPO with KL divergence projection tends to have more stable reward improvement (all the tasks).

The above observations indicate that since the gradient of constraint is not multiplied by the Fisher information matrix, the gradient of the constraint is not aligned with the gradient of the reward. This reduces the reward improvement. However, when the Fisher information matrix is ill-conditioned or not well-estimated, especially in high dimensional policy space, a bad constraint update direction may hinder constraint satisfaction (ant circle, ant gather, grid, and bottleneck tasks). In addition, since the stationary points of KL divergence and L^2 norm projections are different, they converge to policies with different rewards (observe that PCPO with L^2 norm projection has a higher reward than the one with KL divergence projection around 2250 iterations in ant circle task, and has less reward in point gather task).

Discussion of Primal-dual Optimization (PDO) and Fixed Point Optimization (FPO). For the PDO baseline, we see that its constraint values fluctuate especially in the point circle task. This phenomena suggests that PDO is not able to adjust the weight λ^k quickly enough to meet the constraint threshold, which hinders the efficiency of learning constraint-satisfying policies. If the learning rate β is too big, the agent will be too conservative in improving the reward. For FPO, we also see that it learns near constraint-satisfying policies with slightly larger reward improvement compared to PDO. However, in practice, FPO requires a lot of engineering effort to select a good value of λ. Since PCPO requires no hyperparameter tuning, it has the advantage of robustly learning constraint-satisfying policies over PDO and FPO.



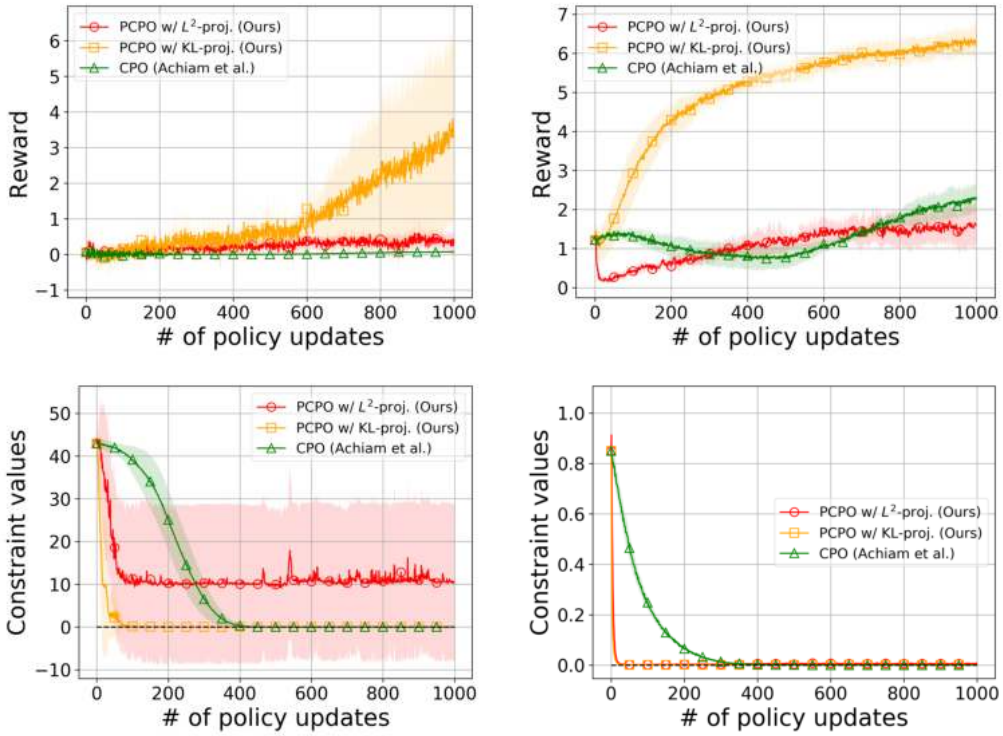
(a) Point circle (b) Point gather

Figure 2.8: The values of the reward and the constraint value for the tested algorithms and task pairs. The solid line is the mean and the shaded area is the standard deviation, over five runs. The dashed line in the cost constraint plot is the cost constraint threshold h. Line search helps to stabilize the training. (Best viewed in color)

2.6.3 Sensitivity Analysis

In this section, we examine various training aspects of the algorithms.

CPO without Line Search. Due to approximation errors, CPO performs a line search to check whether the updated policy satisfies the trust region and cost constraints. To understand the necessity of line search in CPO, we conducted the experiment with and without line search shown in Fig. 2.8. The step size δ is set to 0.01. We find that CPO without line search tends to (1) have large reward variance, especially in the point circle task, and (2) learn constraint-satisfying policies slightly faster. These observations suggest that line search is more conservative in optimizing the policies since it usually takes smaller steps. However, we conjecture that if using



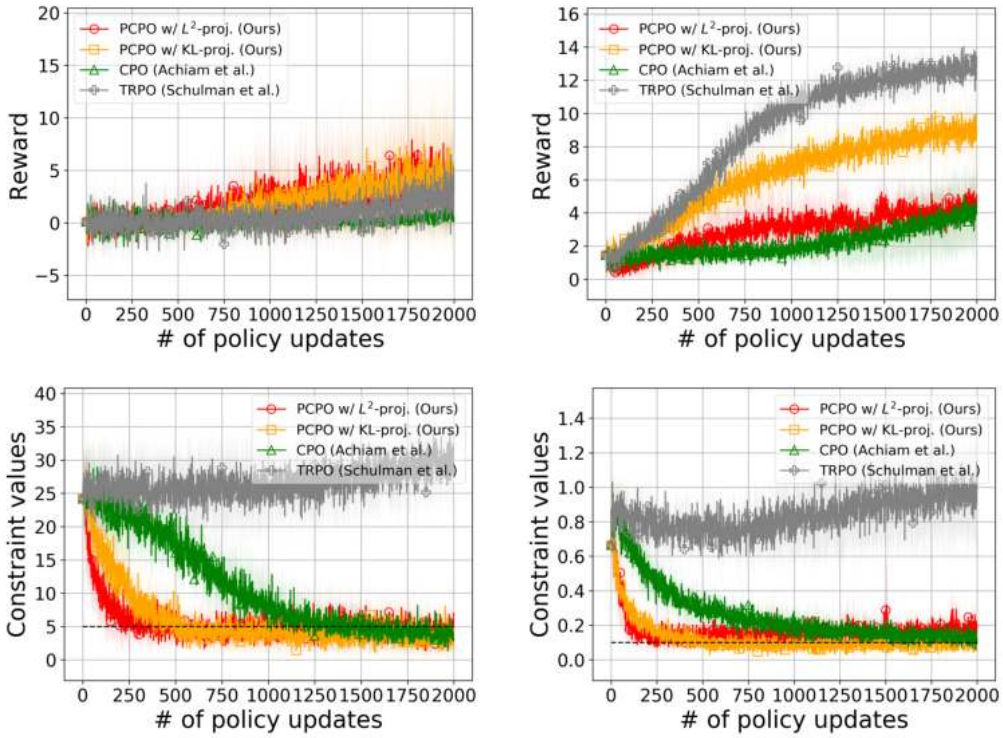
(a) Point circle (b) Point gather

Figure 2.9: The values of the reward and the constraint value for the tested algorithms and task pairs. The solid line is the mean and the shaded area is the standard deviation, over five runs. The dashed line in the cost constraint plot is the cost constraint threshold h. PCPO with KL divergence projection is the only one that can satisfy the constraint with the highest reward. (Best viewed in color)

smaller δ, the effect of line search is not significant.

Harder Constraints. To understand the stability of PCPO and CPO when deployed in more constraint-critical tasks, we increase the difficulty of the task by setting the constraint threshold to zero and reducing the safe area. The learning curve of discounted reward and constraint value over policy updates are shown in Fig. 2.9.

We observe that even with more difficult constraints, PCPO still has more reward improvement and constraint satisfaction than CPO, whereas CPO needs more feasible recovery steps to satisfy the constraint. In addition, we observe that PCPO with L^2 norm projection has high constraint variance in the point circle task, suggesting that



(a) Point circle (b) Point gather

Figure 2.10: The values of the reward and the constraint value for the tested algorithms and task pairs. The solid line is the mean and the shaded area is the standard deviation, over five runs. The dashed line in the cost constraint plot is the cost constraint threshold h. The curves for baseline oracle, TRPO, indicate the reward and constraint violation values when the constraint is ignored. We only use 1% of samples compared to the previous simulations for each policy update. PCPO still satisfies the constraints quickly even when the constraint set is not well-estimated. (Best viewed in color)

the reward update direction is not well aligned with the cost update direction. We also observe that PCPO with L^2 norm projection converges to a bad local optimum in terms of reward in point gather task, suggesting that in order to satisfy the constraint, the cost update direction destroys the reward update direction.

Smaller Batch Samples. To learn policies under constraints, PCPO and CPO require to have a good estimation of the constraint set. However, PCPO may project the policy onto the space that violates the constraint due to the assumption of approximating the constraint set by the linear half-space constraint. To understand

whether the estimation accuracy of the constraint set affects the performance, we conducted the experiments with a batch sample size reduced to 1% of the previous experiments (only 500 samples for each policy update) shown in Fig. 2.10.

We find that smaller training samples affect the performance of the algorithm, creating more reward and cost fluctuation. However, we observe that even with smaller training samples, PCPO still has more reward improvement and constraint satisfaction than CPO.

Analysis of the Approximation Error and the Computational Cost of the Conjugate Gradient Method. In the Grid task, we observe that PCPO with KL divergence projection does worse in reward than TRPO, which is expected since TRPO ignores constraints. However, TRPO actually outperforms PCPO with KL divergence projection in terms of constraint, which is unexpected since by trying to consider the constraint, PCPO with KL divergence projection has made constraint satisfaction worse.

The reason for this observation is that the Fisher information matrix is ill-conditioned, *i.e.,* the condition number $\lambda_{\max}(\boldsymbol{H})/\lambda_{\min}(\boldsymbol{H})$ ($\lambda_{\max}$ is the largest eigenvalue of the matrix) of the Fisher information matrix is large, causing conjugate gradient method that computes constraint update direction $\boldsymbol{H}^{-1}\boldsymbol{a}$ with small number of iteration output the inaccurate approximation. Hence the inaccurate approximation of $\boldsymbol{H}^{-1}\boldsymbol{a}$ causes PCPO with KL divergence projection to have more constraint violations than TRPO.

To solve this issue, one can have more epochs of the conjugate gradient method. This is because the convergence of the conjugate gradient method is controlled by the condition number [135]; the larger the condition number is, the more epochs the algorithm needs to get an accurate approximation. In our experiments, we set the number of iterations of the conjugate gradient method to be 10 to tradeoff between the computational efficiency and the accuracy across all tested algorithms and task

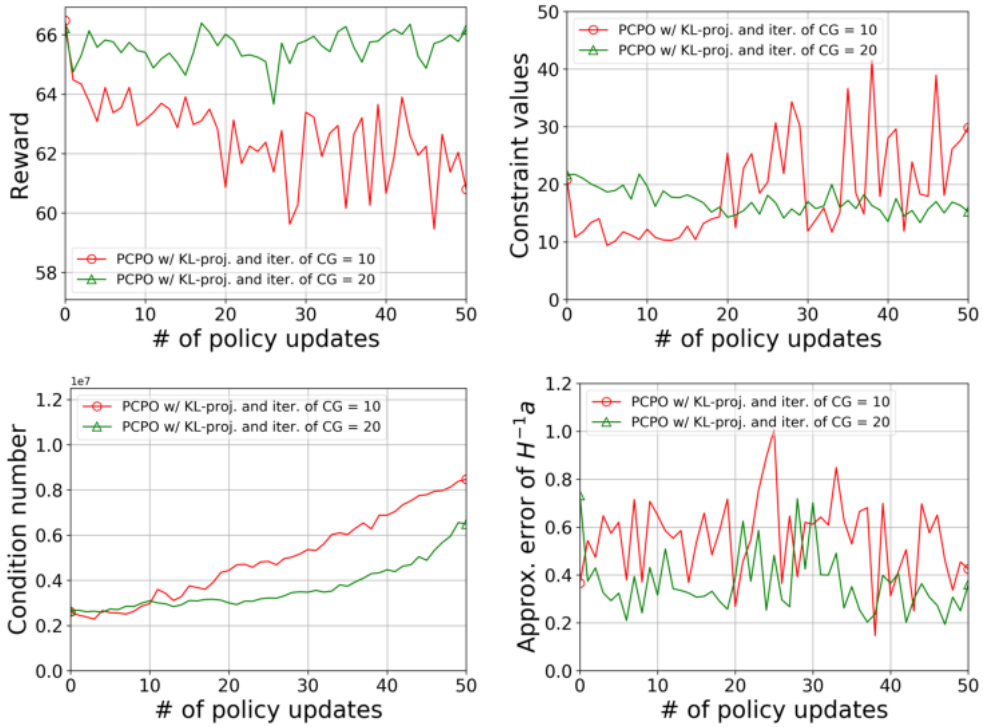


Figure 2.11: (1) The values of the reward and the constraint, (2) the condition number of the Fisher information matrix, and (3) the approximation error of the constraint update direction over training epochs with the conjugate gradient method's iteration of 10 and 20, respectively. The one with larger number of iteration has more constraint satisfaction since it has more accurate approximation. (Best viewed in color)

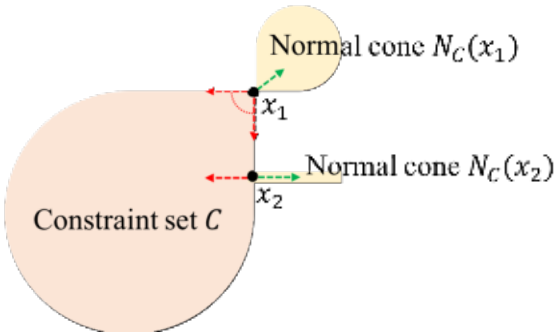


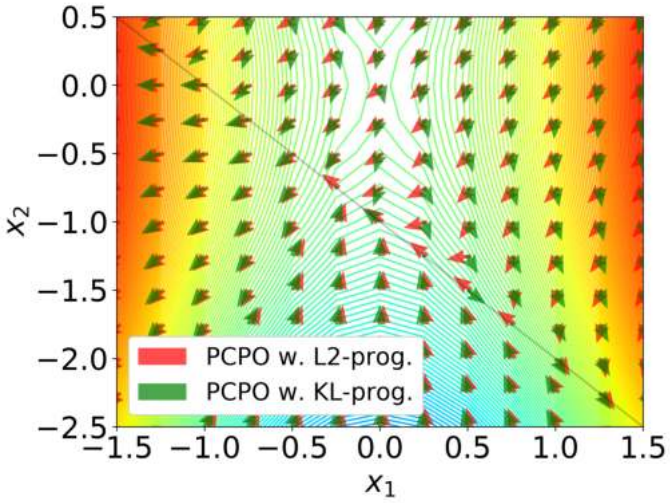
Figure 2.12: The semantic overview of stationery points of PCPO. The red dashed lines are negative directions of normal cones, and the green dashed lines are objective update directions. The objective update direction in a stationary point belongs to the negative normal cone.

pairs.

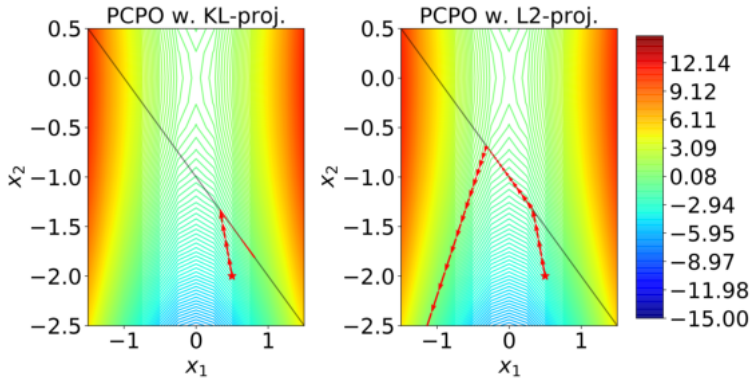
To verify our observation, we compare the condition number of the Fisher information matrix and the approximation error of the constraint update direction over training epochs with a different number of iterations of the conjugate gradient method shown in Fig. 2.11.

We observe that the Fisher information matrix is ill-conditioned, and the one with a larger number of iterations has less error and more constraint satisfaction. This observation confirms our discussion.

Comparison of Optimization Paths of PCPO with KL Divergence and L^2 Norm Projections. Theorem 2.5.1 states that a stationary point of PCPO with KL divergence projection is different from the one of PCPO with L^2 norm projection. See Fig. 2.12 for illustration. To compare both stationary points, we consider the following example shown in Fig. 2.13. We *maximize* a non-convex function $f(\boldsymbol{x}) = \boldsymbol{x}^T \text{diag}(\boldsymbol{y})\boldsymbol{x}$ subject to the constraint $\boldsymbol{x}^T\mathbf{1} \leq -1$, where $\boldsymbol{y} = [5, -1]^T$, and $\mathbf{1}$ is an all-one vector. An optimal solution to this constrained optimization problem is infinity. Fig. 2.13(a) shows the update direction that combines the objective and the cost constraint update directions for both projections. It shows that PCPO with KL divergence projection has stationary points with $\boldsymbol{g} \in -\boldsymbol{a}$ in the boundary of the



(a) Update direction



(b) Optimization path

Figure 2.13: The policy update direction that combines the objective and the constraint update directions of each point (top), and the optimization path of PCPO with KL divergence and L^2 norm projections with the initial point $[0.5, -2.0]^T$ (below). The red star is the initial point, the red arrows are the optimization paths, and the region that is below the black line is the constraint set. We see that both projections converge to different solutions.

constraint set (observe that the update direction is zero for PCPO with KL divergence projection at $\boldsymbol{x} = [0.75, -1.75]^T, [0.25, -1.25]^T$, and $[-0.25, -0.75]^T$), whereas PCPO with L^2 norm projection does not have stationary points in the boundary of the constraint set. Furthermore, Fig. 2.13(b) shows the optimization paths for both projections with one initial starting point. It shows that starting at the initial point $[0.5, -2.0]^T$, PCPO with KL divergence projection with the initial point $[0.5, -2.0]^T$ converges to a local optimum, whereas L^2 norm projection converges to infinity. However, the above example does not necessarily mean that PCPO with L^2 norm projection always finds a better optimum. For example, if the gradient direction of the objective is zero in the constraint set or the boundary, then both projections may converge to the same stationary point.

2.7 Discussion and Conclusion

We address the problem of finding constraint-satisfying policies. The proposed algorithm–projection-based constrained policy optimization (PCPO)–optimizes for the reward function while using the projections to ensure constraint satisfaction. This update rule allows PCPO to maintain the feasibility of the optimization problem of each update, addressing the issue of state-of-the-art approaches. The algorithm achieves comparable or superior performance to state-of-the-art approaches in terms of reward improvement and constraint satisfaction in all cases. We further analyze the convergence of PCPO, and find that certain tasks may prefer either KL divergence projection or L^2 norm projection. Future work will consider the following: (1) examining the Fisher information matrix to iteratively prescribe the choice of projection for policy update, and hence robustly learn constraint-satisfying policies with more reward improvement, and (2) using expert demonstration or other domain knowledge to reduce the sample complexity.

Acknowledgements. We thank Dr. Justinian Rosca for providing the funding and the helpful discussion, and Prof. Peter J. Ramadge and Prof. Karthik Narasimhan for the helpful discussion.

Chapter 3

Improving Constraint-mismatched Baseline Policies with Safety Constraints

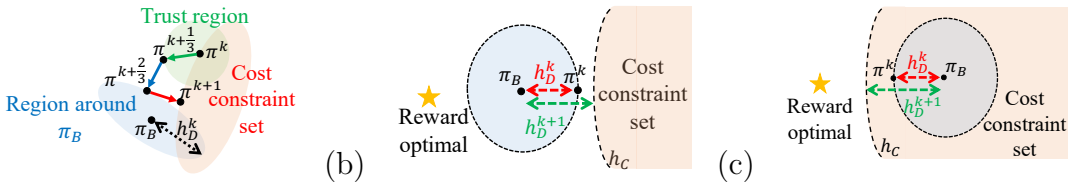
3.1 Introduction

In Chapter 2, we discussed a safe reinforcement learning (RL) algorithm that is trained without knowing the system dynamics of the task or prior baseline policies. However, in many real-world applications, system designers do have some knowledge about the task at hand. For instance, the designers of self-driving cars know the system dynamics of the car (*e.g.,* Newton's second law). Another case would be we have some demonstration data or teacher agents that are collected from the task that is similar to the task at hand. For example, while learning the control policy for a self-driving car is hard, we do collect a lot of human driving log data. If we can exploit this information, then we can streamline the learning process of the agent without learning from scratch. In this chapter, we focus on developing a safe reinforcement learning algorithm that can exploit the baseline policy to improve learning efficiency

without violating the safety constraints.

To this end, one would like to leverage a baseline policy available from demonstrations, a teacher, or a previous task. However, the baseline policy may be sub-optimal for the new application and may not be guaranteed to produce actions that satisfy desired constraints on safety, fairness, or other costs. For instance, when you drive an unfamiliar vehicle, you do so cautiously to ensure safety, while adapting your driving technique to the vehicle characteristics to improve your 'driving reward'. In effect, you (as the agent) gradually adopt a baseline policy (*i.e.,* prior driving skill) to avoid violating the constraints (*e.g.,* safety) while improving your driving reward (*e.g.,* travel time, fuel efficiency). The problem of safely learning from baseline policies is challenging because directly leveraging the baseline policy, as in DAGGER [129] or GAIL [73], may result in policies that violate the constraints since the baseline is not guaranteed to satisfy them. To ensure constraint satisfaction, prior work either adds a hyper-parameter weighted copy of the imitation learning (IL) objective (*i.e.,* imitating the baseline policy) to the RL objective [126, 55, 71], or pre-trains a policy with the baseline policy (*e.g.,* use a baseline policy as an initial policy) and then fine-tunes it through RL [116, 34]. However, both approaches do not ensure constraint satisfaction on *every* learning episode, which is an important feature of safe RL. In addition, the policy initialized by a low entropy baseline policy may never explore.

In this chapter, to learn from the baseline policy while satisfying constraints, we extend PCPO in Chapter 2 and propose an iterative algorithm that performs policy updates in three stages. The first step updates the policy to maximize expected reward using trust region policy optimization (*e.g.,* TRPO [133]). This can, however, result in a new intermediate policy that is too far from the baseline policy and may not satisfy the constraints. The second step performs a projection in policy space to control the distance between the current policy and the baseline policy. In contrast to the approach that regularizes the standard RL objective with the distance w.r.t. the



(a) (b) (c)

Figure 3.1: **(a)** Update procedures for SPACE. Step 1 (green) improves the reward in the trust region. Step 2 (blue) projects the policy onto an *adaptable* region around the baseline policy π_B. Step 3 (red) projects the policy onto the constraint set. **(b)** Illustrating when π_B is *outside* the constraint set. **(c)** Illustrating when π_B is *inside* the constraint set. The highest reward is achieved at the yellow star. h_D^k (the distance between π^k and π_B) is updated to h_D^{k+1} to ensure constraint satisfaction and exploration of the agent.

baseline policy and makes the regularization parameter fade over time, our approach allows the learning agent to update the distance when needed. In addition, this step allows the agent to explore without being overly restricted by the potentially constraint-violating baseline policy. This also enables the baseline policy to influence the learning even at later iterations without the computational burden of learning a cost function for the baseline policy [94]. The third step ensures constraint satisfaction at every iteration by performing a projection onto the set of policies that satisfy the given constraints. We call our algorithm *Safe Policy Adaptation with Constrained Exploration* (SPACE).

This paper's contributions are two-fold. **(1)** We explicitly examine how the baseline policy affects the cost violations of the agent and hence provide a method to safely learn from the baseline policy. This is done by controlling the distance between the learned policy at iteration k and the baseline policy to ensure both feasibility of the optimization problem and safe exploration by the learning agent (Fig. 3.1(b) and (c)). Such an approach, in contrast to non-adaptable constraint sets and learning a policy from scratch [173], leads to better sample efficiency and hence is more favorable in real applications. To our knowledge, prior work does not carry out such an analysis. We further provide a finite-time guarantee for the convergence of SPACE. **(2)** Second, we empirically show that SPACE can robustly learn from sub-optimal baseline

policies in a diverse set of tasks. These include two Mujoco tasks with safety constraints, and two real-world traffic management tasks with fairness constraints. We further show that our algorithm can safely learn from a *human demonstration* driving policy with safety constraints. In all cases, SPACE outperforms state-of-the-art safe RL algorithms, averaging 40% more reward with 10 times fewer cost violations. This shows that SPACE safely and efficiently leverages the baseline policy, and represents a step towards the safe deployment of RL in real applications.

Prior Publications. Parts of this thesis have been published in [175] **Code.** `https://sites.go`

3.2 Problem Setup for Safe Reinforcement Learning with Baseline Policies

We follow a similar setup as stated in Chapter 2. For completeness, we briefly restate the setup here. We frame our problem as a constrained Markov Decision Process (CMDP) [10], defined as a tuple $< \mathcal{S}, \mathcal{A}, T, R, C >$. Here $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and T specifies the conditional probability $T(s'|s,a)$ that the next state is s' given the current state s and action a. In addition, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a constraint cost function. The reward function encodes the benefit of using action a in state s, while the cost function encodes the corresponding constraint violation penalty.

A policy is a map from states to probability distributions on $\mathcal{A}$. It specifies that in state s the selected action is drawn from the distribution $\pi(s)$. The state then transits from s to s' according to the state transition distribution $T(s'|s,a)$. In doing so, a reward $R(s,a)$ is received and a constraint cost $C(s,a)$ is incurred, as outlined above.

Let $\gamma \in (0,1)$ denote a discount factor, and τ denote the trajectory $\tau = (s_0, a_0, s_1, \cdots)$ induced by a policy π. Normally, we seek a policy π that maximizes a cumulative dis-

counted reward:

$$J_R(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right], \tag{3.1}$$

while keeping the cumulative discounted cost below h_C

$$J_C(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t C(s_t,a_t)\right] \leq h_C. \tag{3.2}$$

Here we consider an additional objective. We are provided with a baseline policy π_B and at each state s we measure the divergence between $\pi(s)$ and $\pi_B(s)$. For example, this could be the KL-divergence $D(s) \doteq D_{\mathrm{KL}}(\pi(s)\|\pi_B(s))$. We then seek a policy that maximizes Eq. (3.1), satisfies Eq. (3.2), and ensures the discounted divergence between the learned and baseline policies is below h_D:

$$J_D(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t D(s_t)\right] \leq h_D. \tag{3.3}$$

We do not assume that the baseline policy satisfies the cost constraint. Hence we allow h_D to be adjusted during the learning of π to allow for reward improvement and constraint satisfaction.

Let $\mu_t(\cdot|\pi)$ denote the state distribution at time t under policy π. The discounted state distribution induced by π is defined to be $d^\pi(s) \doteq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mu_t(s|\pi)$. Now bring in the reward advantage function [86] defined by

$$A_R^\pi(s,a) \doteq Q_R^\pi(s,a) - V_R^\pi(s),$$

where $V_R^\pi(s) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0=s\right]$ is the expected reward from state s under policy π, and $Q_R^\pi(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|s_0=s,a_0=a\right]$ is the expected reward from state s and initial action a, and thereafter following policy π. These

definitions allow us to express the reward performance of one policy π' in terms of another π:

$$J_R(\pi') - J_R(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}, a\sim\pi'}[A_R^{\pi}(s,a)].$$

Similarly, we can define $A_D^{\pi}(s,a)$, $Q_D^{\pi}(s,a)$ and $V_D^{\pi}(s)$ for the divergence cost, and $A_C^{\pi}(s,a)$, $Q_C^{\pi}(s,a)$ and $V_C^{\pi}(s)$ for the constraint cost.

3.3 Safe Policy Adaptation with Constrained Exploration (SPACE)

We now describe the proposed iterative algorithm illustrated in Fig. 3.1. In what follows, π^k denotes the learned policy after iteration k, and M denotes a distance measure between policies. For example, M may be the 2-norm of the difference of policy parameters or some average over the states of the KL-divergence of the action policy distributions.

Step 1. We perform one step of trust region policy optimization [133]. This maximizes the reward advantage function $A_R^{\pi}(s,a)$ over a KL-divergence neighborhood of π^k:

$$\begin{aligned} \pi^{k+\frac{1}{3}} = \arg\max_{\pi} \ & \mathbb{E}_{s\sim d^{\pi^k},\ a\sim\pi}[A_R^{\pi^k}(s,a)] \\ \text{s.t. } & \mathbb{E}_{s\sim d^{\pi^k}}\left[D_{\mathrm{KL}}(\pi(s)\|\pi^k(s))\right] \leq \delta. \end{aligned} \tag{3.4}$$

Step 2. We project $\pi^{k+\frac{1}{3}}$ onto a region around π_B controlled by h_D^k to minimize

M:

$$\begin{aligned}
&\pi^{k+\frac{2}{3}} = \arg\min_{\pi} \; M(\pi, \pi^{k+\frac{1}{3}}) \\
&\text{s.t. } J_D(\pi^k) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi^k},\, a\sim\pi}[A_D^{\pi^k}(s)] \leq h_D^k.
\end{aligned} \tag{3.5}$$

Step 3. We project $\pi^{k+\frac{2}{3}}$ onto the set of policies satisfying the cost constraint to minimize M:

$$\begin{aligned}
&\pi^{k+1} = \arg\min_{\pi} \; M(\pi, \pi^{k+\frac{2}{3}}) \\
&\text{s.t. } J_C(\pi^k) + \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi^k},\, a\sim\pi}[A_C^{\pi^k}(s,a)] \leq h_C.
\end{aligned} \tag{3.6}$$

Remarks. Since we use a small step size δ, we can replace the state distribution d^π with d^{π^k} in Eq. (3.5) and (3.6) and hence compute $A_D^{\pi^k}$ and $A_C^{\pi^k}$.

3.3.1 Control the Distance Between the Agent and the Baseline Policy

We select h_D^0 to be small and gradually increase h_D^k at each iteration to expand the region around π_B. Specifically, we make $h_D^{k+1} > h_D^k$ if:

(a) $J_C(\pi^k) > J_C(\pi^{k-1})$: this increase is to ensure a nonempty intersection between the region around π_B and the cost constraint set (feasibility). See Fig. 3.1(b).

(b) $J_R(\pi^k) < J_R(\pi^{k-1})$: this increase gives the next policy more freedom to improve the reward and the cost constraint performance (exploration). See Fig. 3.1(c).

It remains to determine how to set the new value of h_D^{k+1}. Let $\mathcal{U}_1$ denote the set of policies satisfying the cost constraint, and $\mathcal{U}_2^k$ denote the set of policies in the region around π_B controlled by h_D^k. Then we have the following Lemma.

Lemma 3.3.1 (**Updating** h_D). *If at step* $k+1$*:* $h_D^{k+1} \geq \mathcal{O}\big((J_C(\pi^k) - h_C)^2\big) + h_D^k$, *then* $\mathcal{U}_1 \cap \mathcal{U}_2^{k+1} \neq \emptyset$ *(feasibility) and* $\mathcal{U}_2^{k+1} \cap \partial\mathcal{U}_1 \neq \emptyset$ *(exploration).*

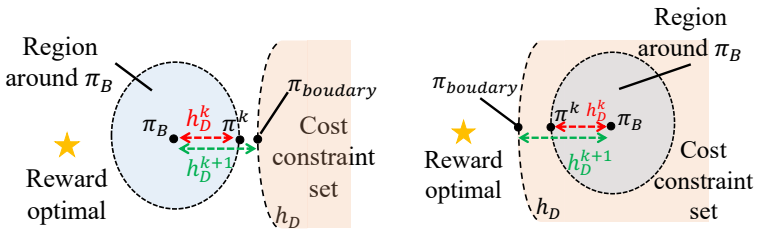


Figure 3.2: **(a)** Illustrating when π_B is *outside* the cost constraint set. **(b)** Illustrating when π_B is *inside* the cost constraint set. $\pi_{boundary}$ is the policy with $J_C(\pi_{boundary}) = h_C$. We aim to bound h_D^{k+1} (*i.e.*, the KL-divergence between $\pi_{boundary}$ and π_B) by using h_D^k.

Proof. Based on Theorem 1 in [6], for any two policies π and π' we have

$$
\begin{aligned}
& J_C(\pi') - J_C(\pi) \geq \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^\pi \\ a\sim\pi'}}\left[A_C^\pi(s,a) - \frac{2\gamma\epsilon_C^{\pi'}}{1-\gamma}\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi'(s)||\pi(s))}\right] \\
\Rightarrow \quad & \frac{2\gamma\epsilon_C^{\pi'}}{(1-\gamma)^2}\mathbb{E}_{s\sim d^\pi}\left[\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi'(s)||\pi(s))}\right] \geq -J_C(\pi') + J_C(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^\pi \\ a\sim\pi'}}\left[A_C^\pi(s,a)\right] \\
\Rightarrow \quad & \frac{2\gamma\epsilon_C^{\pi'}}{(1-\gamma)^2}\mathbb{E}_{s\sim d^\pi}\left[\sqrt{\frac{1}{2}D_{\mathrm{KL}}(\pi'(s)||\pi(s))}\right] \geq -J_C(\pi') + J_C(\pi) \\
\Rightarrow \quad & \frac{\sqrt{2}\gamma\epsilon_C^{\pi'}}{(1-\gamma)^2}\sqrt{\mathbb{E}_{s\sim d^\pi}\left[D_{\mathrm{KL}}(\pi'(s)||\pi(s))\right]} \geq -J_C(\pi') + J_C(\pi) \\
\Rightarrow \quad & \mathbb{E}_{s\sim d^\pi}\left[D_{\mathrm{KL}}(\pi'(s)||\pi(s))\right] \geq \frac{(1-\gamma)^4(-J_C(\pi') + J_C(\pi))^2}{2\gamma^2{\epsilon_C^{\pi'}}^2}.
\end{aligned} \tag{3.7}
$$

The fourth inequality follows from Jensen's inequality. We then define $\varphi(\pi(s)) \doteq \sum_i \pi(a(i)|s)\log\pi(a(i)|s)$. By Three-point Lemma [29], for any three policies π, π', and $\hat{\pi}$ we have

$$
\begin{aligned}
\mathbb{E}_{s\sim d^\pi}\left[D_{\mathrm{KL}}(\pi'(s)||\hat{\pi}(s))\right] = \mathbb{E}_{s\sim d^\pi}\left[D_{\mathrm{KL}}(\pi'(s)||\pi(s))\right] + \mathbb{E}_{s\sim d^\pi}\left[D_{\mathrm{KL}}(\pi(s)||\hat{\pi}(s))\right] \\
-\mathbb{E}_{s\sim d^\pi}\left[(\nabla\varphi(\hat{\pi}(s)) - \nabla\varphi(\pi(s)))^T(\pi'(s) - \pi(s))\right].
\end{aligned} \tag{3.8}
$$

Let $\pi_{boundary}$ denote a policy satisfying $J_C(\pi_{boundary}) = h_C$ (*i.e.*, $\pi_{boundary}$ is in the boundary of the set of the policies which satisfy the cost constraint $J_C(\pi) \leq h_C$). Let

$\pi' = \pi_{boundary}, \hat{\pi} = \pi_B$ and $\pi = \pi^k$ in Eq. (3.7) and Eq. (3.8) (this is illustrated in Fig. 3.2). Then we have

$$
\begin{aligned}
&\mathbb{E}_{s\sim d^{\pi^k}}\Big[D_{\mathrm{KL}}(\pi_{boundary}(s)||\pi_B(s))\Big] - \mathbb{E}_{s\sim d^{\pi^k}}\Big[D_{\mathrm{KL}}(\pi^k(s)||\pi_B(s))\Big] \\
=\ &\mathbb{E}_{s\sim d^{\pi^k}}\Big[D_{\mathrm{KL}}(\pi_{boundary}(s)||\pi^k(s))\Big] \\
&\qquad - \mathbb{E}_{s\sim d^{\pi^k}}\Big[(\nabla\varphi(\pi_B(s)) - \nabla\varphi(\pi^k(s)))^T(\pi_{boundary}(s) - \pi^k(s))\Big] \\
\geq\ &\frac{(1-\gamma)^4(-J_C(\pi_{boundary}) + J_C(\pi^k))^2}{2\gamma^2{\epsilon_C^{\pi'}}^2} \\
&\qquad - \mathbb{E}_{s\sim d^{\pi^k}}\Big[(\nabla\varphi(\pi_B(s)) - \nabla\varphi(\pi^k(s)))^T(\pi_{boundary}(s) - \pi^k(s))\Big] \\
=\ &\frac{(1-\gamma)^4(-h_C + J_C(\pi^k))^2}{2\gamma^2{\epsilon_C^{\pi'}}^2} \\
&\qquad - \mathbb{E}_{s\sim d^{\pi^k}}\Big[(\nabla\varphi(\pi_B(s)) - \nabla\varphi(\pi^k(s)))^T(\pi_{boundary}(s) - \pi^k(s))\Big] \\
=\ &\mathcal{O}\Big(\big(-h_C + J_C(\pi^k)\big)^2\Big),
\end{aligned}
\tag{3.9}
$$

where $J_C(\pi_{boundary}) = h_C$.

For the first case in Fig. 3.2(a), we would like to have $\mathcal{U}_1 \cap \mathcal{U}_2^{k+1} \neq \emptyset$ (feasibility). For the second case in Fig. 3.2(b), we would like to have $\mathcal{U}_2^{k+1} \cap \partial\mathcal{U}_1 \neq \emptyset$ (exploration). These implies that the policy in step $k+1$ is $\pi_{boundary}$ which satisfies $\mathcal{U}_1 \cap \mathcal{U}_2^{k+1} \neq \emptyset$ and $\mathcal{U}_2^{k+1} \cap \partial\mathcal{U}_1 \neq \emptyset$.

Now let $h_D^{k+1} \doteq \mathbb{E}_{s\sim d^{\pi^k}}\Big[D_{\mathrm{KL}}(\pi_{boundary}(s)||\pi_B(s))\Big]$ and $h_D^k \doteq \mathbb{E}_{s\sim d^{\pi^k}}[D_{\mathrm{KL}}(\pi^k(s)||\pi_B(s))]$. Then Eq. 3.9 implies

$$
h_D^{k+1} \geq \mathcal{O}\Big(\big(-h_C + J_C(\pi^k)\big)^2\Big) + h_D^k.
$$

□

Remarks. Two values are in the big $\mathcal{O}$. The first value depends on the discounted factor γ, and the second value depends on relative distances between π^k, π_B, and the policy in $\partial\mathcal{U}_1$. The intuition is that the smaller the distances are, the smaller the

update of h_D^k is.

Importantly, Lemma 3.3.1 ensures that the boundaries of the region around π_B determined by h_D and the set of policies satisfying the cost constraint intersect. Note that h_D will become large enough to guarantee feasibility during training. This *adaptable* constraint set, in contrast to the *fixed* constraint set in PCPO, allows the learning algorithm to explore policies within the cost constraint set while still learning from the baseline policy. Compared to other CMDP approaches, the step of projecting close to π_B allows the policy to quickly improve. Compared to behavior cloning, the steps of reward optimization and constraint projection allow the policy to achieve good final performance. We examine the importance of updating h_D in the following section.

3.4 SPACE Updates

We will implement a policy as a neural network with fixed architecture parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$. We then learn a policy from the achievable set $\{\pi(\cdot|\boldsymbol{\theta}) \colon \boldsymbol{\theta} \in \mathbb{R}^n\}$ by iteratively learning $\boldsymbol{\theta}$. Let $\boldsymbol{\theta}^k$ and $\pi^k \doteq \pi(\cdot|\boldsymbol{\theta}^k)$ denote the parameter value and the corresponding policy at step k. In this setting, it is impractical to solve for the policy updates in Eq. (3.4), (3.5) and (3.6). Hence we approximate the reward function and constraints with first order Taylor expansions, and KL-divergence with a second order Taylor expansion. We will need the following derivatives:

(1) $\boldsymbol{g}^k \doteq \nabla_{\boldsymbol{\theta}} \mathbb{E}_{s\sim d^{\pi^k},\ a\sim\pi}[A_R^{\pi^k}(s,a)]$,

(2) $\boldsymbol{a}^k \doteq \nabla_{\boldsymbol{\theta}} \mathbb{E}_{s\sim d^{\pi^k},\ a\sim\pi}[A_D^{\pi^k}(s)]$,

(3) $\boldsymbol{c}^k \doteq \nabla_{\boldsymbol{\theta}} \mathbb{E}_{s\sim d^{\pi^k},\ a\sim\pi}[A_C^{\pi^k}(s,a)]$, and

(4) $\boldsymbol{F}^k \doteq \nabla^2_{\boldsymbol{\theta}} \mathbb{E}_{s\sim d^{\pi^k}} \left[D_{\mathrm{KL}}(\pi(s)\|\pi^k(s))\right]$.

Each of these derivatives are taken w.r.t. the neural network parameter and evaluated at $\boldsymbol{\theta}^k$. We also define $b^k \doteq J_D(\pi^k) - h_D^k$, and $d^k \doteq J_C(\pi^k) - h_C$. Let

Algorithm 2 Safe Policy Adaptation with Constrained Exploration

Initialize a policy $\pi^0 = \pi(\cdot|\boldsymbol{\theta}^0)$ and a trajectory buffer $\mathcal{B}$
for $k = 0, 1, 2, \cdots$ **do**
  Run $\pi^k = \pi(\cdot|\boldsymbol{\theta}^k)$ and store trajectories in $\mathcal{B}$
  Obtain $\boldsymbol{\theta}^{k+1}$ using the update in Eq. (3.13)
  If $J_C(\pi^k) > J_C(\pi^{k-1})$ or $J_R(\pi^k) < J_R(\pi^{k-1})$
    Update h_D^{k+1} using **Lemma 3.3.1**
  Empty $\mathcal{B}$

$u^k \doteq \sqrt{\frac{2\delta}{\boldsymbol{g}^{k^T}\boldsymbol{F}^{k^{-1}}\boldsymbol{g}^k}}$, and $\boldsymbol{L} = \boldsymbol{I}$ for the 2-norm projection and $\boldsymbol{L} = \boldsymbol{F}^k$ for the KL-divergence projection.

3.4.1 Update Procedure

Step 1. Approximating Eq. (3.4) yields

$$\begin{aligned} \boldsymbol{\theta}^{k+\frac{1}{3}} = \arg\max_{\boldsymbol{\theta}} \ & \boldsymbol{g}^{k^T}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \\ \text{s.t. } & \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^k) \leq \delta. \end{aligned} \tag{3.10}$$

Step 2 & 3. Approximating Eq. (3.5) and (3.6), similarly yields

$$\begin{aligned} \boldsymbol{\theta}^{k+\frac{2}{3}} = \arg\min_{\boldsymbol{\theta}} \ & \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{3}})^T \boldsymbol{L} (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{3}}) \\ \text{s.t. } & \boldsymbol{a}^{k^T}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + b^k \leq 0, \end{aligned} \tag{3.11}$$

$$\begin{aligned} \boldsymbol{\theta}^{k+1} = \arg\min_{\boldsymbol{\theta}} \ & \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}})^T \boldsymbol{L} (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\ \text{s.t. } & \boldsymbol{c}^{k^T}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + d^k \leq 0, \end{aligned} \tag{3.12}$$

where $\boldsymbol{L} = \boldsymbol{I}$ for the 2-norm projection and $\boldsymbol{L} = \boldsymbol{F}^k$ for the KL-divergence projection. We solve these problems using convex programming, then we have ($(\cdot)^+$ is $\max(0, \cdot)$)

$$\begin{aligned}\boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}^k + u^k {\boldsymbol{F}^k}^{-1}\boldsymbol{g}^k \\ &- \Big(\frac{u^k {\boldsymbol{a}^k}^T {\boldsymbol{F}^k}^{-1}\boldsymbol{g}^k + b^k}{{\boldsymbol{a}^k}^T \boldsymbol{L}^{-1}\boldsymbol{a}^k}\Big)^+ \boldsymbol{L}^{-1}\boldsymbol{a}^k \\ &- \Big(\frac{u^k {\boldsymbol{c}^k}^T {\boldsymbol{F}^k}^{-1}\boldsymbol{g}^k + d^k}{{\boldsymbol{c}^k}^T \boldsymbol{L}^{-1}\boldsymbol{c}^k}\Big)^+ \boldsymbol{L}^{-1}\boldsymbol{c}^k.\end{aligned} \tag{3.13}$$

We now prove the update rule in Eq. (3.13).

Proof. For the first problem in Eq. (3.10), since $\boldsymbol{F}^k$ is the Fisher Information matrix, it is positive semi-definite. Hence it is a convex program with quadratic inequality constraints. If the primal problem has a feasible point, then Slater's condition is satisfied and strong duality holds. Let $\boldsymbol{\theta}^*$ and λ^* denote the solutions to the primal and dual problems, respectively. In addition, the primal objective function is continuously differentiable. Hence the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for the optimality of $\boldsymbol{\theta}^*$ and λ^*. We now form the Lagrangian:

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = -{\boldsymbol{g}^k}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + \lambda\Big(\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k(\boldsymbol{\theta} - \boldsymbol{\theta}^k) - \delta\Big).$$

And we have the following KKT conditions:

$$-\boldsymbol{g}^k + \lambda^* \boldsymbol{F}^k\boldsymbol{\theta}^* - \lambda^* \boldsymbol{F}^k\boldsymbol{\theta}^k = 0 \qquad \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{3.14}$$

$$\frac{1}{2}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) - \delta = 0 \qquad \nabla_{\lambda}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{3.15}$$

$$\frac{1}{2}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) - \delta \leq 0 \qquad \text{primal constraints} \tag{3.16}$$

$$\lambda^* \geq 0 \qquad \text{dual constraints} \tag{3.17}$$

$$\lambda^*\Big(\frac{1}{2}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) - \delta\Big) = 0 \qquad \text{complementary slackness} \tag{3.18}$$

By Eq. (3.14), we have $\boldsymbol{\theta}^* = \boldsymbol{\theta}^k + \frac{1}{\lambda^*}{\boldsymbol{F}^k}^{-1}\boldsymbol{g}^k$. And by plugging Eq. (3.14) into

Eq. (3.15), we have $\lambda^* = \sqrt{\frac{\boldsymbol{g}^{k^T}\boldsymbol{F}^{k^{-1}}\boldsymbol{g}^k}{2\delta}}$. Hence we have a solution

$$\boldsymbol{\theta}^{k+\frac{1}{3}} = \boldsymbol{\theta}^* = \boldsymbol{\theta}^k + \sqrt{\frac{2\delta}{\boldsymbol{g}^{k^T}\boldsymbol{F}^{k^{-1}}\boldsymbol{g}^k}}\boldsymbol{F}^{k^{-1}}\boldsymbol{g}^k, \tag{3.19}$$

which also satisfies Eq. (3.16), Eq. (3.17), and Eq. (3.18).

For the second problem in Eq. (3.11), we follow the same procedure for the first problem to form the Lagrangian:

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{3}})^T\boldsymbol{L}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{3}}) + \lambda(\boldsymbol{a}^{k^T}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + b^k).$$

And we have the following KKT conditions:

$$\boldsymbol{L}\boldsymbol{\theta}^* - \boldsymbol{L}\boldsymbol{\theta}^{k+\frac{1}{3}} + \lambda^*\boldsymbol{a}^k = 0 \qquad \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{3.20}$$

$$\boldsymbol{a}^{k^T}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + b^k = 0 \qquad \nabla_{\lambda}\mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{3.21}$$

$$\boldsymbol{a}^{k^T}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + b^k \leq 0 \qquad \text{primal constraints} \tag{3.22}$$

$$\lambda^* \geq 0 \qquad \text{dual constraints} \tag{3.23}$$

$$\lambda^*(\boldsymbol{a}^{k^T}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + b^k) = 0 \qquad \text{complementary slackness} \tag{3.24}$$

By Eq. (3.20), we have $\boldsymbol{\theta}^* = \boldsymbol{\theta}^k + \lambda^*\boldsymbol{L}^{-1}\boldsymbol{a}^k$. And by plugging Eq. (3.20) into Eq. (3.21) and Eq. (3.23), we have $\lambda^* = \max(0, \frac{\boldsymbol{a}^{k^T}(\boldsymbol{\theta}^{k+\frac{1}{3}} - \boldsymbol{\theta}^k) + b^k}{\boldsymbol{a}^k\boldsymbol{L}^{-1}\boldsymbol{a}^k})$. Hence we have a solution

$$\boldsymbol{\theta}^{k+\frac{2}{3}} = \boldsymbol{\theta}^* = \boldsymbol{\theta}^{k+\frac{1}{3}} - \max(0, \frac{\boldsymbol{a}^{k^T}(\boldsymbol{\theta}^{k+\frac{1}{3}} - \boldsymbol{\theta}^k) + b^k}{\boldsymbol{a}^{k^T}\boldsymbol{L}^{-1}\boldsymbol{a}^{k^T}})\boldsymbol{L}^{-1}\boldsymbol{a}^k, \tag{3.25}$$

which also satisfies Eq. (3.22) and Eq. (3.24).

For the third problem in Eq. (3.12), instead of doing the projection on $\pi^{k+\frac{2}{3}}$ which is the intermediate policy obtained in the second step, we project the policy $\pi^{k+\frac{1}{3}}$ onto the cost constraint. This allows us to compute the projection without too much computational cost. We follow the same procedure for the first and second

problems to form the Lagrangian:

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{3}})^T \boldsymbol{L}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{3}}) + \lambda(\boldsymbol{c}^{k^T}(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + d^k).$$

And we have the following KKT conditions:

$$\boldsymbol{L}\boldsymbol{\theta}^* - \boldsymbol{L}\boldsymbol{\theta}^{k+\frac{1}{3}} + \lambda^* \boldsymbol{c}^k = 0 \qquad \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{3.26}$$

$$\boldsymbol{c}^{k^T}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + d^k = 0 \qquad \nabla_{\lambda} \mathcal{L}(\boldsymbol{\theta}^*, \lambda^*) = 0 \tag{3.27}$$

$$\boldsymbol{c}^{k^T}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + d^k \leq 0 \qquad \text{primal constraints} \tag{3.28}$$

$$\lambda^* \geq 0 \qquad \text{dual constraints} \tag{3.29}$$

$$\lambda^*(\boldsymbol{c}^{k^T}(\boldsymbol{\theta}^* - \boldsymbol{\theta}^k) + d^k) = 0 \qquad \text{complementary slackness} \tag{3.30}$$

By Eq. (3.26), we have $\boldsymbol{\theta}^* = \boldsymbol{\theta}^k + \lambda^* \boldsymbol{L}^{-1}\boldsymbol{c}^k$. And by plugging Eq. (3.26) into Eq. (3.27) and Eq. (3.29), we have $\lambda^* = \max(0, \frac{\boldsymbol{c}^{k^T}(\boldsymbol{\theta}^{k+\frac{1}{3}} - \boldsymbol{\theta}^k) + d^k}{\boldsymbol{c}^k \boldsymbol{L}^{-1} \boldsymbol{c}^k})$. Hence we have a solution

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^* = \boldsymbol{\theta}^{k+\frac{1}{3}} - \max(0, \frac{\boldsymbol{c}^{k^T}(\boldsymbol{\theta}^{k+\frac{1}{3}} - \boldsymbol{\theta}^k) + d^k}{\boldsymbol{c}^{k^T} \boldsymbol{L}^{-1} \boldsymbol{c}^{k^T}}) \boldsymbol{L}^{-1} \boldsymbol{c}^k. \tag{3.31}$$

Hence by combining Eq. (3.19), Eq. (3.25) and Eq. (3.31), we have

$$\begin{aligned} \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \sqrt{\frac{2\delta}{\boldsymbol{g}^{k^T} \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k}} \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k - \max(0, \frac{\sqrt{\frac{2\delta}{\boldsymbol{g}^{k^T} \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k}} \boldsymbol{a}^{k^T} \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k + b^k}{\boldsymbol{a}^{k^T} \boldsymbol{L}^{-1} \boldsymbol{a}^k}) \boldsymbol{L}^{-1} \boldsymbol{a}^k \\ - \max(0, \frac{\sqrt{\frac{2\delta}{\boldsymbol{g}^{k^T} \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k}} \boldsymbol{c}^{k^T} \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k + d^k}{\boldsymbol{c}^{k^T} \boldsymbol{L}^{-1} \boldsymbol{c}^k}) \boldsymbol{L}^{-1} \boldsymbol{c}^k. \end{aligned}$$

□

Algorithm 2 shows the corresponding pseudocode.

3.4.2 Convergence Analysis of SPACE Update Rule

We consider the following simplified problem to provide a convergence guarantee of SPACE:

$$\min_{\boldsymbol{\theta} \in \mathcal{C}_1 \cap \mathcal{C}_2} f(\boldsymbol{\theta}), \tag{3.32}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function at every point in a open set $\mathcal{X} \subseteq \mathbb{R}^n$, and $\mathcal{C}_1 \subseteq \mathcal{X}$ and $\mathcal{C}_2 \subseteq \mathcal{X}$ are compact convex sets with $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$. The function f is the negative reward function of our CMDP, and the two constraint sets represent the cost constraint set and the region around the baseline policy π_B.

For a vector $\boldsymbol{x}$, let $\|\boldsymbol{x}\|$ denote the Euclidean norm. For a matrix $\boldsymbol{M}$ let $\|\boldsymbol{M}\|$ denote the induced matrix 2-norm, and $\sigma_i(\boldsymbol{M})$ denote the i-th largest singular value of $\boldsymbol{M}$.

Assumption 1. We assume:

(1.1) The gradient ∇f is L-Lipschitz continuous over a open set $\mathcal{X}$.

(1.2) For some constant G, $\|\nabla f(\boldsymbol{\theta})\| \leq G$.

(1.3) For a constant H, $\text{diam}(\mathcal{C}_1) \leq H$ and $\text{diam}(\mathcal{C}_2) \leq H$.

Assumptions (1.1) and (1.2) ensure that the gradient can not change too rapidly and the norm of the gradient can not be too large. (1.3) implies that for every iteration, the diameter of the region around π_B is bounded above by H.

We will need a concept of an ϵ-first order stationary point [115]. For $\epsilon > 0$, we say that $\boldsymbol{\theta}^* \in \mathcal{C}_1 \cap \mathcal{C}_2$ an ϵ-first order stationary point (ϵ-FOSP) of Problem (3.32) under KL-divergence projection if

$$\nabla f(\boldsymbol{\theta}^*)^T(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \geq -\epsilon, \quad \forall \boldsymbol{\theta} \in \mathcal{C}_1 \cap \mathcal{C}_2. \tag{3.33}$$

Similarly, under the 2-norm projection, $\boldsymbol{\theta}^* \in \mathcal{C}_1 \cap \mathcal{C}_2$ an ϵ-FOSP of (3.32) if

$$\nabla f(\boldsymbol{\theta}^*)^T \boldsymbol{F}^*(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \geq -\epsilon, \quad \forall \boldsymbol{\theta} \in \mathcal{C}_1 \cap \mathcal{C}_2, \tag{3.34}$$

where $\boldsymbol{F}^* \doteq \nabla^2_{\boldsymbol{\theta}} \mathbb{E}_{s \sim d^{\pi^*}} \left[D_{\mathrm{KL}}(\pi(s) \| \pi^*(s)) \right]$. Notice that SPACE converges to distinct stationary points under the two possible projections We now describe the reason for choosing two variants of ϵ-FOSP under two possible projections. Let η_R^k denote the step size for the reward, η_D^k denote the step size for the divergence cost, and η_C^k denote the step size for the constraint cost. Without loss of generality, under the KL-divergence projection, at step $k+1$ SPACE does

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \eta_R^k \boldsymbol{F}^{k^{-1}} \boldsymbol{g}^k - \eta_D^k \boldsymbol{F}^{k^{-1}} \boldsymbol{a}^k - \eta_C^k \boldsymbol{F}^{k^{-1}} \boldsymbol{c}^k.$$

Similarly, under the 2-norm projection, at step $k+1$ SPACE does

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \eta_R^k \boldsymbol{F}^k \boldsymbol{g}^k - \eta_D^k \boldsymbol{a}^k - \eta_C^k \boldsymbol{c}^k.$$

With this definition, we have the following Lemma.

Lemma 3.4.1 (**Stationary Points for SPACE**). *Under the KL-divergence projection, SPACE converges to a stationary point $\boldsymbol{\theta}^*$ satisfying*

$$\eta_R^* \boldsymbol{g}^* = \eta_D^* \boldsymbol{a}^* + \eta_C^* \boldsymbol{c}^*.$$

Under the 2-norm projection, SPACE converges to a stationary point $\boldsymbol{\theta}^*$ satisfying

$$\eta_R^* \boldsymbol{g}^* = \boldsymbol{F}^* (\eta_D^* \boldsymbol{a}^* + \eta_C^* \boldsymbol{c}^*).$$

Proof. Under the KL-divergence projection, by using the definition of a stationary

point we have

$$\begin{aligned} & \boldsymbol{\theta}^* = \boldsymbol{\theta}^* + \eta_R^* \boldsymbol{F}^{*-1} \boldsymbol{g}^* - \eta_D^* \boldsymbol{F}^{*-1} \boldsymbol{a}^* - \eta_C^* \boldsymbol{F}^{*-1} \boldsymbol{c}^* \\ \Rightarrow \quad & \eta_R^* \boldsymbol{F}^{*-1} \boldsymbol{g}^* = \eta_D^* \boldsymbol{F}^{*-1} \boldsymbol{a}^* + \eta_C^* \boldsymbol{F}^{*-1} \boldsymbol{c}^* \\ \Rightarrow \quad & \eta_R^* \boldsymbol{g}^* = \eta_D^* \boldsymbol{a}^* + \eta_C^* \boldsymbol{c}^*. \end{aligned}$$

Under the 2-norm projection, by using the definition of a stationary point we have

$$\begin{aligned} & \boldsymbol{\theta}^* = \boldsymbol{\theta}^* + \eta_R^* \boldsymbol{F}^{*-1} \boldsymbol{g}^* - \eta_D^* \boldsymbol{a}^* - \eta_C^* \boldsymbol{c}^* \\ \Rightarrow \quad & \eta_R^* \boldsymbol{F}^{*-1} \boldsymbol{g}^* = \eta_D^* \boldsymbol{a}^* + \eta_C^* \boldsymbol{c}^* \\ \Rightarrow \quad & \eta_R^* \boldsymbol{g}^* = \boldsymbol{F}^* (\eta_D^* \boldsymbol{a}^* + \eta_C^* \boldsymbol{c}^*). \end{aligned}$$

□

Hence Lemma 3.4.1 motivates the need for defining two variants of FOSP. With these assumptions, we have the following Theorem.

Theorem 3.4.2 (**Finite-Time Convergence Guarantee of SPACE**). *Under the KL-divergence projection, there exists a sequence* $\{\eta^k\}$ *such that SPACE converges to an* ϵ*-FOSP in at most* $\mathcal{O}(\epsilon^{-2})$ *iterations. Moreover, at step* $k+1$

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^k) - \frac{L\epsilon^2}{2(G + \frac{H\sigma_1(\boldsymbol{F}^k)}{\eta^k})^2}. \tag{3.35}$$

Similarly, under the 2-norm projection, there exists a sequence $\{\eta^k\}$ *such that SPACE converges to an* ϵ*-FOSP in at most* $\mathcal{O}(\epsilon^{-2})$ *iterations. Moreover, at step* $k+1$

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^k) - \frac{L\epsilon^2}{2(G\sigma_1(\boldsymbol{F}^{k^{-1}}) + \frac{H}{\eta^k})^2}. \tag{3.36}$$

Proof. **SPACE under the KL-divergence projection converges to an ϵ-FOSP.**

Based on Lemma 2.5.2 under the KL-divergence projection, and setting $\boldsymbol{\theta} = \boldsymbol{\theta}^k - \eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k)$, $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{k+\frac{2}{3}}$ and $\boldsymbol{\theta}' = \boldsymbol{\theta}^k$, we have

$$\begin{aligned} & (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T \boldsymbol{F}^k (\boldsymbol{\theta}^k - \eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k) - \boldsymbol{\theta}^{k+\frac{2}{3}}) \leq 0 \\ \Rightarrow \quad & \nabla f(\boldsymbol{\theta}^k)^T (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) \leq -\frac{1}{\eta^k} (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k). \end{aligned} \tag{3.37}$$

Based on the L-Lipschitz continuity of gradients and Eq. (3.37), we have

$$\begin{aligned} f(\boldsymbol{\theta}^{k+\frac{2}{3}}) &\leq f(\boldsymbol{\theta}^k) + \nabla f(\boldsymbol{\theta}^k)^T (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + \frac{L}{2} \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 \\ &\leq f(\boldsymbol{\theta}^k) - \frac{1}{\eta^k} (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + \frac{L}{2} \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 \\ &= f(\boldsymbol{\theta}^k) - \frac{L}{2} \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 - \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) - \frac{L}{2} \|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2, \end{aligned} \tag{3.38}$$

where the equality follows by setting δ (*i.e.,* the size of the trust region) such that

$$\eta^k = \frac{(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)}{L \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 + \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) + \frac{L}{2} \|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2}.$$

Again, based on Lemma 2.5.2, for $\boldsymbol{\theta} \in \mathcal{C}_2$ we have

$$\begin{aligned} & (\boldsymbol{\theta}^k - \eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k) - \boldsymbol{\theta}^{k+\frac{2}{3}}) \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \leq 0 \\ \Rightarrow \quad & (-\eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k))^T \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \leq -(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\ \Rightarrow \quad & \nabla f(\boldsymbol{\theta}^k)^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \geq \frac{1}{\eta^k} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\ \Rightarrow \quad & \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{\theta} \geq \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{\theta}^{k+\frac{2}{3}} + \frac{1}{\eta^k} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\ \Rightarrow \quad & f(\boldsymbol{\theta}^k)^T (\boldsymbol{\theta} - \boldsymbol{\theta}^k) \geq \nabla f(\boldsymbol{\theta}^k)^T (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + \frac{1}{\eta^k} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T \boldsymbol{F}^k (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\ & \geq -\|\nabla f(\boldsymbol{\theta}^k)\| \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\| - \frac{1}{\eta^k} \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\| \|\boldsymbol{F}^k\| \|\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}\| \\ & \geq -\big(G + \frac{D\sigma_1(\boldsymbol{F}^k)}{\eta^k}\big) \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|, \end{aligned} \tag{3.39}$$

where in the last two inequalities we use the property of the norm. Before reaching an ϵ-FOSP, Eq. (3.39) implies that

$$\begin{aligned} -\epsilon \geq \min_{\boldsymbol{\theta}\in\mathcal{C}_2} \nabla f(\boldsymbol{\theta}^k)^T(\boldsymbol{\theta}-\boldsymbol{\theta}^k) \geq -\big(G+\frac{D\sigma_1(\boldsymbol{F}^k)}{\eta^k}\big)\|\boldsymbol{\theta}^{k+\frac{2}{3}}-\boldsymbol{\theta}^k\| \\ \Rightarrow \quad \|\boldsymbol{\theta}^{k+\frac{2}{3}}-\boldsymbol{\theta}^k\| \geq \frac{\epsilon}{G+\frac{D\sigma_1(\boldsymbol{F}^k)}{\eta^k}}. \end{aligned} \tag{3.40}$$

Based on Eq. (3.38) and Eq. (3.40), we have

$$\begin{aligned} f(\boldsymbol{\theta}^{k+\frac{2}{3}}) &\leq f(\boldsymbol{\theta}^k) - \frac{L}{2}\|\boldsymbol{\theta}^{k+\frac{2}{3}}-\boldsymbol{\theta}^k\|^2 - \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T(\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k+\frac{2}{3}}) - \frac{L}{2}\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k+\frac{2}{3}}\|^2 \\ &\leq f(\boldsymbol{\theta}^k) - \frac{L\epsilon^2}{2(G+\frac{D\sigma_1(\boldsymbol{F}^k)}{\eta^k})^2} - \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T(\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k+\frac{2}{3}}) - \frac{L}{2}\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k+\frac{2}{3}}\|^2. \end{aligned} \tag{3.41}$$

Based on the L-Lipschitz continuity of gradients, for the projection to the constraint set $\mathcal{C}_1$ we have

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^{k+\frac{2}{3}}) + \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T(\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k+\frac{2}{3}}) + \frac{L}{2}\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k+\frac{2}{3}}\|^2. \tag{3.42}$$

Combining Eq. (3.41) with Eq. (3.42), we have

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^k) - \frac{L\epsilon^2}{2(G+\frac{D\sigma_1(\boldsymbol{F}^k)}{\eta^k})^2}. \tag{3.43}$$

Hence it takes $\mathcal{O}(\epsilon^{-2})$ iterations to reach an ϵ-FOSP.

SPACE under the 2-norm projection converges to an ϵ-FOSP. Based on Lemma 2.5.2 under the 2-norm projection, and setting $\boldsymbol{\theta} = \boldsymbol{\theta}^k - \eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k)$,

$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{k+\frac{2}{3}}$ and $\boldsymbol{\theta}' = \boldsymbol{\theta}^k$, we have

$$\begin{aligned}
&(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T(\boldsymbol{\theta}^k - \eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k) - \boldsymbol{\theta}^{k+\frac{2}{3}}) \leq 0 \\
\Rightarrow &(\boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k))^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) \leq -\frac{1}{\eta^k}(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k).
\end{aligned} \tag{3.44}$$

Based on the L-Lipschitz continuity of gradients and Eq. (3.44), we have

$$\begin{aligned}
f(\boldsymbol{\theta}^{k+\frac{2}{3}}) &\leq f(\boldsymbol{\theta}^k) + \nabla f(\boldsymbol{\theta}^k)^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + \frac{L}{2}\|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 \\
&\leq f(\boldsymbol{\theta}^k) + (\boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k))^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + Q + \frac{L}{2}\|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 \\
&\leq f(\boldsymbol{\theta}^k) - \frac{1}{\eta^k}(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + Q + \frac{L}{2}\|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 \\
&= f(\boldsymbol{\theta}^k) - \frac{L}{2}\|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 - \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) - \frac{L}{2}\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2,
\end{aligned} \tag{3.45}$$

where $Q := \nabla f(\boldsymbol{\theta}^k)^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) - (\boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k))^T(\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k)$, which represents the difference between the gradient and the nature gradient, and the equality follows by setting δ (*i.e.,* the size of the trust region) such that

$$\eta^k = \frac{\|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2}{L\|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 + Q + \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) + \frac{L}{2}\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2}.$$

Again, based on Lemma 2.5.2, for $\boldsymbol{\theta} \in \mathcal{C}_2$ we have

$$
\begin{aligned}
& (\boldsymbol{\theta}^k - \eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k) - \boldsymbol{\theta}^{k+\frac{2}{3}})(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \leq 0 \\
\Rightarrow \quad & (-\eta^k \boldsymbol{F}^{k^{-1}} \nabla f(\boldsymbol{\theta}^k))^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \leq -(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\
\Rightarrow \quad & \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{F}^{k^{-1}} (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \geq \frac{1}{\eta^k} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\
\Rightarrow \quad & \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{F}^{k^{-1}} \boldsymbol{\theta} \geq \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{F}^{k^{-1}} \boldsymbol{\theta}^{k+\frac{2}{3}} + \frac{1}{\eta^k} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\
\Rightarrow \quad & \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{F}^{k^{-1}} (\boldsymbol{\theta} - \boldsymbol{\theta}^k) \geq \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{F}^{k^{-1}} (\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k) + \frac{1}{\eta^k} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}) \\
& \geq -\|\nabla f(\boldsymbol{\theta}^k)\| \|\boldsymbol{F}^{k^{-1}}\| \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\| - \frac{1}{\eta^k} \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\| \|\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{2}{3}}\| \\
& \geq -\big(G\sigma_1(\boldsymbol{F}^{k^{-1}}) + \frac{D}{\eta^k}\big) \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|, \qquad (3.46)
\end{aligned}
$$

where in the last two inequalities we use the property of the norm. Before reaching an ϵ-FOSP, Eq. (3.46) implies that

$$
\begin{aligned}
& -\epsilon \geq \min_{\boldsymbol{\theta} \in \mathcal{C}_2} \nabla f(\boldsymbol{\theta}^k)^T \boldsymbol{F}^{k^{-1}} (\boldsymbol{\theta} - \boldsymbol{\theta}^k) \geq -\big(G\sigma_1(\boldsymbol{F}^{k^{-1}}) + \frac{D}{\eta^k}\big) \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\| \\
\Rightarrow \quad & \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\| \geq \frac{\epsilon}{\big(G\sigma_1(\boldsymbol{F}^{k^{-1}}) + \frac{D}{\eta^k}\big)}. \qquad (3.47)
\end{aligned}
$$

Based on Eq. (3.45) and Eq. (3.47), we have

$$
\begin{aligned}
f(\boldsymbol{\theta}^{k+\frac{2}{3}}) & \leq f(\boldsymbol{\theta}^k) - \frac{L}{2} \|\boldsymbol{\theta}^{k+\frac{2}{3}} - \boldsymbol{\theta}^k\|^2 - \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) - \frac{L}{2} \|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2 \\
& \leq f(\boldsymbol{\theta}^k) - \frac{L\epsilon^2}{2(G\sigma_1(\boldsymbol{F}^{k^{-1}}) + \frac{D}{\eta^k})^2} - \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) - \frac{L}{2} \|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2. \qquad (3.48)
\end{aligned}
$$

Based on the L-Lipschitz continuity of gradients, for the projection to the constraint set $\mathcal{C}_1$ we have

$$
f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^{k+\frac{2}{3}}) + \nabla f(\boldsymbol{\theta}^{k+\frac{2}{3}})^T (\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}) + \frac{L}{2} \|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^{k+\frac{2}{3}}\|^2. \qquad (3.49)
$$

Combining Eq. (3.48) with Eq. (3.49), we have

$$f(\boldsymbol{\theta}^{k+1}) \leq f(\boldsymbol{\theta}^k) - \frac{L\epsilon^2}{2(G\sigma_1(\boldsymbol{F}^{k^{-1}}) + \frac{D}{\eta^k})^2}. \tag{3.50}$$

Hence it takes $\mathcal{O}(\epsilon^{-2})$ iterations to reach an ϵ-FOSP. □

We now make several observations for Theorem 3.4.2.

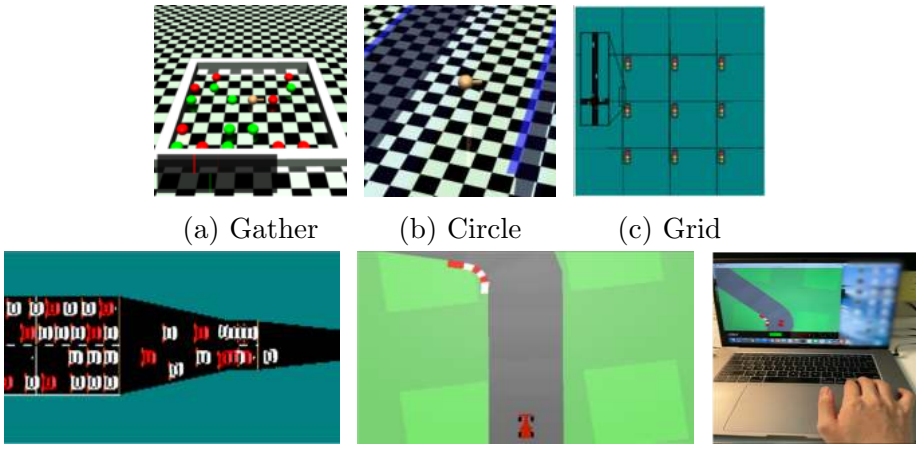
(1) The smaller H is, the greater the decrease in the objective. This observation supports the idea of starting with a small value for h_D and increasing it only when needed.

(2) Under the KL-divergence projection, the effect of $\sigma_1(\boldsymbol{F}^k)$ is negligible. This is because in this case, η^k is proportional to $\sigma_1(\boldsymbol{F}^k)$. Hence $\sigma_1(\boldsymbol{F}^k)$ does not play a major role in decreasing the objective value.

(3) Under the 2-norm projection, the smaller $\sigma_1(\boldsymbol{F}^{k^{-1}})$ (*i.e.,* larger $\sigma_n(\boldsymbol{F}^k)$) is, the greater the decrease in the objective. This is because a large $\sigma_n(\boldsymbol{F}^k)$ means a large curvature of f in all directions. This implies that the 2-norm distance between the pre-projection and post-projection points is small, leading to a small deviation from the reward improvement direction after doing projections.

3.5 Experiments

Our experiments study the following three questions: **(1)** How does SPACE perform compared to other baselines in behavior cloning and safe RL in terms of learning efficiency and constraint satisfaction? **(2)** How does SPACE trained with sub-optimal π_B perform (*e.g.,* human demonstration)? **(3)** How does step 2 in SPACE affect the performance?



(a) Gather (b) Circle (c) Grid

(d) Bottleneck (e) Car-racing (f) Demo.

Figure 3.3: **(a)** Gather: the agent is rewarded for gathering green apples, but is constrained to collect a limited number of red apples [6]. **(b)** Circle: the agent is rewarded for moving in a specified wide circle, but is constrained to stay within a safe region smaller than the radius of the circle [6]. **(c)** Grid: the agent controls the traffic lights in a grid road network and is rewarded for high throughput, but is constrained to let lights stay red for at most 7 consecutive seconds [161]. **(d)** Bottleneck: the agent controls a set of autonomous vehicles (shown in red) in a traffic merge situation and is rewarded for achieving high throughput, but constrained to ensure that human-driven vehicles (shown in white) have the low speed for no more than 10 seconds [161]. **(e)** Car-racing: the agent controls an autonomous vehicle on a race track and is rewarded for driving through as many tiles as possible, but is constrained to use the brakes at most 5 times to encourage a smooth ride [24]. **(f)** A human player plays car-racing with demonstration data logged. These tasks are to show the applicability of our approach to a diverse set of problems.

3.5.1 Setup

Tasks. We compare the proposed algorithm with existing approaches on five control tasks: three tasks with safety constraints ((a), (b) and (e) in Fig. 3.3), and two tasks with fairness constraints ((c) and (d) in Fig. 3.3). These tasks are briefly described in the caption of Fig. 3.3. We chose the traffic management tasks since a good control policy can benefit millions of drivers. In addition, we chose the car-racing task since a good algorithm should safely learn from baseline human policies. For all the algorithms, we use neural networks to represent Gaussian policies. We use the KL-divergence projection in the Mujoco and car-racing tasks, and the 2-norm projection in the traffic management task since it achieves better performance. We use a grid search to select the hyper-parameters.

Baseline Policies π_B. To test whether SPACE can safely and efficiently leverage the baseline policy, we consider *three variants* of the baseline policies.

(1) *Sub-optimal* π_B^{cost} with $J_C(\pi_B^{\text{cost}}) \approx 0$.

(2) *Sub-optimal* π_B^{reward} with $J_C(\pi_B^{\text{reward}}) > h_C$.

(3) π_B^{near} with $J_C(\pi_B^{\text{near}}) \approx h_C$ (*i.e.,* the baseline policy has the same cost constraint as the agent, but is not guaranteed to have an optimal reward performance).

These π_B have *different* degrees of constraint satisfaction. This is to examine whether SPACE can safely learn from sub-optimal π_B. In addition, in the car-racing task we pre-train π_B using an off-policy algorithm (DDPG [103]), which directly learns from *human demonstration data* (Fig. 3.3(f)). This is to demonstrate that π_B may come from a teacher or demonstration data. This *sub-optimal* human baseline policy is denoted by π_B^{human}.

For ease of computation, we update h_D using $v \cdot (J_C(\pi^k) - h_C)^2 + h_D^k$ from Lemma 3.3.1, with a constant $v > 0$. We found that the performance is not heavily affected by v since we will update h_D at a later iteration.

Baseline Algorithms. Our goal is to study how to *safely* and *efficiently* learn

from *sub-optimal* (possibly unsafe) baseline policies. We compare SPACE with five baseline methods that combine behavior cloning and safe RL algorithms.

(1) *Fixed-point Constrained Policy Optimization (f-CPO).* In f-CPO, we add the divergence objective in the reward function. The weight λ is fixed followed by a CPO update (optimize the reward and divergence cost w.r.t. the trust region and the cost constraints). The f-CPO policy update solves [6]:

$$\begin{aligned}\boldsymbol{\theta}^{k+1} = \arg\max_{\boldsymbol{\theta}} \ & (\boldsymbol{g}^k + \lambda \boldsymbol{a}^k)^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \\ \text{s.t. } & \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \leq \delta \\ & {\boldsymbol{c}^k}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + d^k \leq 0.\end{aligned}$$

(2) *Fixed-point PCPO (f-PCPO).* In f-PCPO, we add the divergence objective in the reward function. The weight λ is fixed followed by a PCPO update (two-step process: optimize the reward and divergence cost, followed by the projection to the safe set). The f-PCPO policy update solves:

$$\begin{aligned}&\boldsymbol{\theta}^{k+\frac{1}{2}} = \arg\max_{\boldsymbol{\theta}} \ (\boldsymbol{g}^k + \lambda \boldsymbol{a}^k)^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \\ &\text{s.t. } \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^k)^T \boldsymbol{F}^k(\boldsymbol{\theta} - \boldsymbol{\theta}^k) \leq \delta, \quad \text{(trust region)} \\ &\boldsymbol{\theta}^{k+1} = \arg\min_{\boldsymbol{\theta}} \ \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{2}})^T \boldsymbol{L}(\boldsymbol{\theta} - \boldsymbol{\theta}^{k+\frac{1}{2}}) \\ &\text{s.t. } {\boldsymbol{c}^k}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^k) + d^k \leq 0. \quad \text{(cost constraint)}\end{aligned}$$

(3) *Dynamic-point Constrained Policy Optimization (d-CPO).* The d-CPO update solves f-CPO problem with a stateful $\lambda^{k+1} = (\lambda^k)^\beta$, where $0 < \beta < 1$. This is inspired by [126], in which they have the same weight-scheduling method to adjust λ^k.

(4) *Dynamic-point PCPO (d-PCPO).* The d-PCPO update solves f-PCPO problem with a stateful $\lambda^{k+1} = (\lambda^k)^\beta$, where $0 < \beta < 1$.

For all the experiments and the algorithms, the weight is fixed and it is set to 1. Note that both d-CPO and d-PCPO regularize the standard RL objective with the distance w.r.t. the baseline policy and make the regularization parameter (*i.e.,* λ) fade over time. This is a common practice to learn from the baseline policy. In addition, in many real applications, you cannot have access to parameterized π_B (*e.g.,* neural network policies) or you want to design a policy with different architectures than π_B. Hence in our setting, we cannot directly initialize the learning policy with the baseline policy and then fine-tune it.

Experimental Details. For the gather and circle tasks we test two distinct agents: a point-mass ($S \subseteq \mathbb{R}^9, A \subseteq \mathbb{R}^2$), and an ant robot ($S \subseteq \mathbb{R}^{32}, A \subseteq \mathbb{R}^8$). The agent in the grid task is $S \subseteq \mathbb{R}^{156}, A \subseteq \mathbb{R}^4$, and the agent in the bottleneck task is $S \subseteq \mathbb{R}^{141}, A \subseteq \mathbb{R}^{20}$. Finally, the agent in the car-racing task is $S \subseteq \mathbb{R}^{96\times 96\times 3}, A \subseteq \mathbb{R}^3$.

For the simulations in the gather and circle tasks, we use a neural network with two hidden layers of size (64, 32) to represent Gaussian policies. And we use the KL-divergence projection. For the simulations in the grid and bottleneck tasks, we use a neural network with two hidden layers of size (16, 16) and (50, 25) to represent Gaussian policies, respectively. And we use the 2-norm projection. For the simulation in the car-racing task, we use a convolutional neural network with two convolutional operators of sizes 24 and 12 followed by a dense layer of size (32, 16) to represent a Gaussian policy. And we use the KL-divergence projection. The choice of the projections depends on the task itself, we report the best performance among the two projections. We use tanh as an activation function for all the neural network policies. In the experiments, since the step size is small, we reuse the Fisher information matrix of the reward improvement step in the KL-divergence projection step to reduce the computational cost.

We use GAE-λ approach [134] to estimate $A^\pi_R(s,a)$, $A^\pi_C(s,a)$, and $A^\pi_D(s)$. For the simulations in the gather, circle, and car-racing tasks, we use neural network

Parameter	PC	PG	AC	AG	Gr	BN	CR
Reward dis. factor γ	0.995	0.995	0.995	0.995	0.999	0.999	0.990
Constraint cost dis. factor γ_C	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Divergence cost dis. factor γ_D	1.0	1.0	1.0	1.0	1.0	1.0	1.0
step size δ	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	5×10^{-4}
λ_R^{GAE}	0.95	0.95	0.95	0.95	0.97	0.97	0.95
λ_C^{GAE}	1.0	1.0	0.5	0.5	0.5	1.0	1.0
λ_D^{GAE}	0.95	0.95	0.95	0.95	0.90	0.90	0.95
Batch size	50,000	50,000	100,000	100,000	10,000	25,000	10,000
Rollout length	50	15	500	500	400	500	1000
Constraint cost threshold h_C	5	0.5	5	0.2	0	0	5
Divergence cost threshold h_D^0	5	3	5	3	10	10	5
Number of policy updates	1,000	1,200	2,500	1,500	200	300	600

Table 3.1: Parameters used in all tasks. (PC: point circle, PG: point gather, AC: ant circle, AG: ant gather, Gr: grid, BN: bottleneck, and CR: car-racing tasks)

baselines with the same architecture and activation functions as the policy networks. For the simulations in the grid and bottleneck tasks, we use linear baselines. The hyperparameters of all algorithms and all tasks are in Table 3.1.

3.5.2 Experiment Results

Overall Performance. The learning curves of the discounted reward ($J_R(\pi)$), the undiscounted constraint cost ($J_C(\pi)$), and the undiscounted divergence cost ($J_D(\pi)$) over policy updates are shown for all tested algorithms and tasks in Fig. 3.4. We use π_B^{near} in bottleneck and grid tasks, and π_B^{human} in car-racing task. Note that π_B^{human} from human demonstration is *highly sub-optimal* to the agent (*i.e.,* $J_R(\pi_B^{\text{human}})$ is small). The value of the reward is only around 5 as shown in the plot. It does not solve the task at hand. Overall, we observe that **(1)** SPACE achieves at least 2 times faster cost constraint satisfaction in all cases even learning from π_B^{human}. **(2)** SPACE achieves at least 10% more reward in the bottleneck and car-racing tasks compared to the best baseline, and **(3)** SPACE is the only algorithm that satisfies the cost constraints in all cases. In contrast, even if f(d)-CPO and f(d)-PCPO (similar to behavior cloning) are provided with good baseline policies π_B^{near}, they do not learn efficiently due to the conflicting reward and cost objectives. In addition, PCPO is

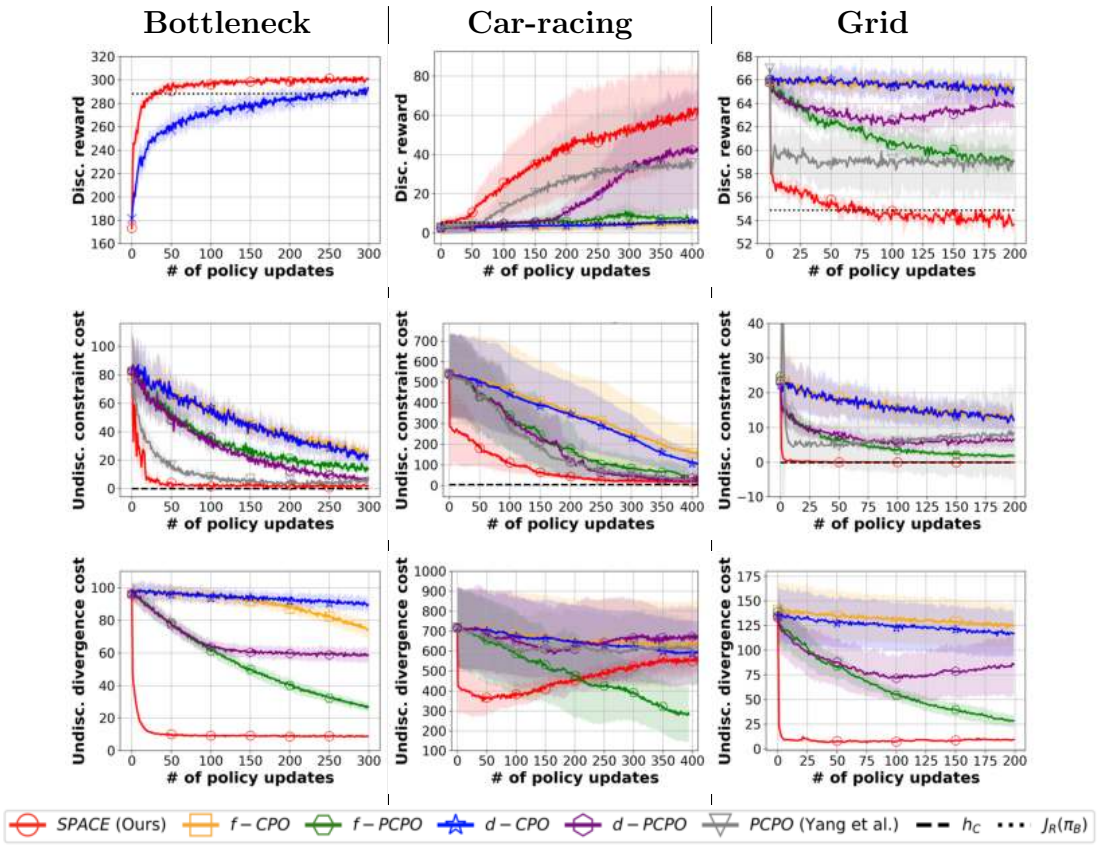


Figure 3.4: The discounted reward, the undiscounted constraint cost, and the undiscounted divergence cost over policy updates for the tested algorithms and tasks. The solid line is the mean and the shaded area is the standard deviation over 5 runs (random seed). The baseline policies in the grid and bottleneck tasks are π_B^{near}, and the baseline policy in the car-racing task is π_B^{human}. The black dashed line is the **cost constraint threshold** h_C. We observe that SPACE is the only algorithm that satisfies the constraints while achieving superior reward performance. Although π_B^{human} has substantially low reward, SPACE still can learn to improve the reward. (We show the results in these tasks as representative cases since they are more challenging. Best viewed in color.)

less sample-efficient, which shows the accelerated learning of SPACE.

For example, in the car-racing task we observe that $J_D(\pi)$ in SPACE decreases at the initial iteration, but increases in the end. This implies that the learned policy is guided by the baseline policy π_B^{human} in the beginning, but uses less supervision in the end. In addition, in the grid task, we observe that the final reward of SPACE is lower than the baseline algorithm. This is because that SPACE converges to a policy in the cost constraint set, whereas the baseline algorithms do not find constraint-satisfying policies. Furthermore, we observe that $J_D(\pi)$ in the traffic tasks decreases throughout the training. This implies that SPACE intelligently adjusts h_D^k w.r.t. the performance of π_B to achieve safe learning.

f-CPO and f-PCPO. f-CPO and f-PCPO fail to improve the reward and have more cost violations. Most likely this is due to persistent supervision from the baseline policies which need not satisfy the cost constraints nor have a high reward. For example, in the car-racing task, we observe that the value of the divergence cost decreases throughout the training. This implies that the learned policy overly evolves to the sub-optimal π_B and hence degrades the reward performance. **d-CPO and d-PCPO.** d-CPO and d-PCPO improve the reward slowly and have more cost violations. They do not use projection to quickly learn from π_B. For example, in the car-racing task, $J_D(\pi)$ in d-CPO and d-PCPO are high compared to SPACE throughout the training. This suggests that simply regularizing the RL objective with the faded weight is susceptible to a sub-optimal π_B. In contrast to this heuristic, we use Lemma 3.3.1 to update h_D when needed, allowing π_B to influence the learning of the agent at any iterations depending on the learning progress of the agent.

Importantly, in our setup, the agent does not have any prior knowledge about π_B. The agent has to stay close to π_B to verify its reward and cost performance. It is true that π_B may be constraint-violating, but it may also provide a useful signal for maximizing the reward. For example, in the grid task (Fig. 3.4), although π_B does

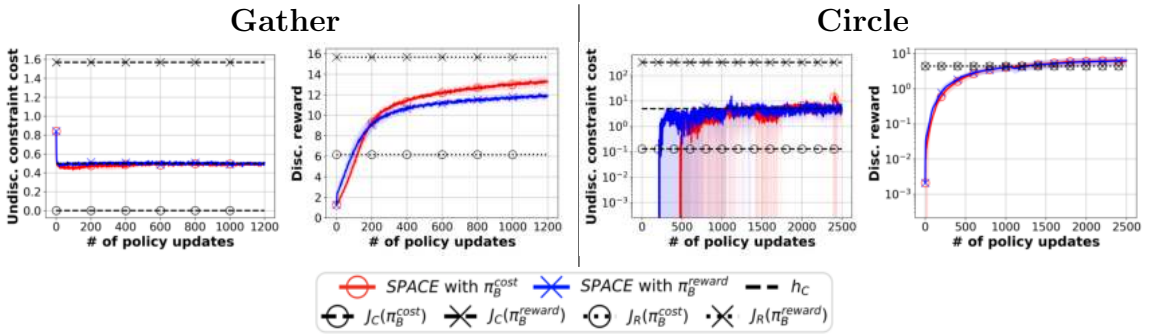


Figure 3.5: Learning from *sub-optimal* π_B. The undiscounted constraint cost and the discounted reward over policy updates for the gather and the circle tasks. The solid line is the mean and the shaded area is the standard deviation over 5 runs. The black dashed line is the cost constraint threshold h_C. We observe that SPACE satisfies the cost constraints even when learning from the sub-optimal π_B.

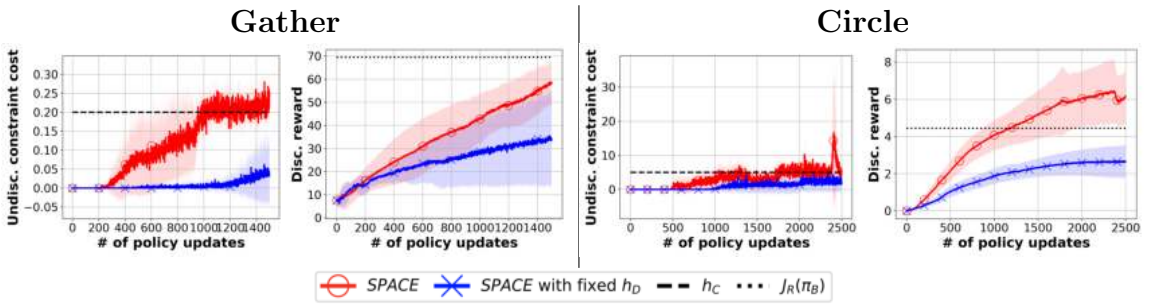


Figure 3.6: Ablation studies on the fixed h_D. The undiscounted constraint cost and the discounted reward over policy updates for the gather and the circle tasks. The solid line is the mean and the shaded area is the standard deviation over 5 runs. The black dashed line is the cost constraint threshold h_C. We observe that the update rule is critical for ensuring learning performance improvement.

not satisfy the cost constraint, it still helps the SPACE agent (by being close to π_B) to achieve faster cost satisfaction.

Having demonstrated the overall effectiveness of SPACE, our remaining experiments explore **(1)** SPACE's ability to safely learn from sub-optimal polices, and **(2)** the importance of the update method in Lemma 3.3.1. For compactness, we restrict our consideration to SPACE and the Mujoco tasks, which are widely used in the RL community.

Sub-optimal π_B^{cost} and π_B^{reward}. Next, we test whether SPACE can learn from sub-optimal π_B. The learning curves of $J_C(\pi)$ and $J_R(\pi)$ over policy updates are shown

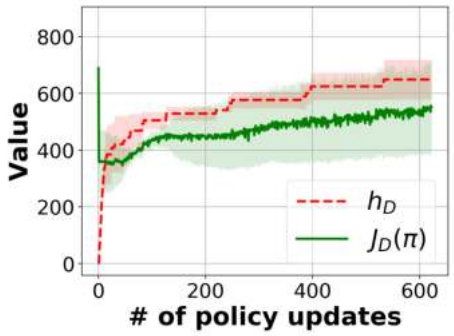


Figure 3.7: The divergence cost $J_D(\pi)$ and the value of h_D over the iterations in the car-racing task. We see that SPACE controls h_D to ensure divergence constraint satisfaction.

for the gather and circle tasks in Fig. 3.5. We use two *sub-optimal* π_B: π_B^{cost} and π_B^{reward}, and learning agent's h_C is set to 0.5 (*i.e.,* π_B do not solve the task at hand). We observe that SPACE robustly satisfies the cost constraints in all cases even when learning from π_B^{reward}. In addition, we observe that learning guided by π_B^{reward} achieves faster reward learning efficiency at the *initial* iteration. This is because $J_R(\pi_B^{\text{reward}}) > J_R(\pi_B^{\text{cost}})$ as seen in the reward plot. Furthermore, we observe that learning guided by π_B^{cost} achieves faster reward learning efficiency at the *later* iteration. This is because by starting in the interior of the cost constraint set (*i.e.,* $J_C(\pi_B^{\text{cost}}) \approx 0 \leq h_C$), the agent can safely exploit the baseline policy. The results show SPACE enables fast convergence to a constraint-satisfying policy, even if π_B does not meet the constraint or does not optimize the reward.

SPACE with a Fixed h_D. In our final experiments, we investigate the importance of updating h_D when learning from a sub-optimal π_B. The learning curves of the $J_C(\pi)$ and $J_R(\pi)$ over policy updates are shown for the gather and circle tasks in Fig. 3.6. We observe that SPACE with fixed h_D converges to less reward. For example, in the circle task SPACE with the dynamic h_D achieves 2.3 times more reward. This shows that π_B in this task is highly sub-optimal to the agent and the need of using stateful h_D^k.

Moreover, Fig. 3.7 shows the divergence cost $J_D(\pi)$ and the value of h_D over the

iterations in the car-racing task. We observe that SPACE gradually increases h_D to improve reward and cost satisfaction performance.

3.6 Discussion and Conclusion

In this chapter, we addressed the problem of learning constraint-satisfying policies given potentially sub-optimal baseline policies. We explicitly analyzed how to safely learn from the baseline policy, and hence proposed an iterative policy optimization algorithm that alternates between maximizing the expected return on the task, minimizing the distance to the baseline policy, and projecting the policy onto the constraint-satisfying set. Our algorithm efficiently learns from a baseline policy as well as human-provided demonstration data and achieves superior reward and cost performance compared with state-of-the-art approaches.

No algorithm is without limitations. Future work could improve SPACE in several ways. For instance, in Lemma 3.3.1, we do not guarantee that SPACE will increase h_D enough for the region around the baseline policy to contain the *optimal* policy. This is challenging since the optimization problem is non-convex. One possible solution is to rerun SPACE multiple times and reinitialize π_B with the previously learned policy each time. One evidence to support this method is that in the bottleneck task (Fig. 3.4), the agent trained with SPACE outperforms PCPO agent by achieving higher rewards and faster constraint satisfaction. The PCPO agent here can be seen as the SPACE agent trained without π_B. And then we train the SPACE agent with π_B from the learned PCPO agent. This shows that based on the learned policy, we can use SPACE to improve performance. In addition, it would be interesting to explore using other types of baseline policies such as rule-based policies, and see how they impact the learning dynamics of SPACE.

Acknowledgements. We thank Dr. Justinian Rosca for providing the fund-

ing and the helpful discussion, and Prof. Peter J. Ramadge and Prof. Karthik Narasimhan for the helpful discussion.

Chapter 4

Safe Reinforcement Learning for Quadrupedal Locomotion

4.1 Introduction

In Chapter 2 and 3, we investigate the problem of safe reinforcement learning, and verify the proposed algorithms (*i.e.,* PCPO and SPACE) in the simulated tasks. In this chapter, we extend the safe reinforcement learning algorithm in real-world hardware such as legged robots[1] to understand the resilience of the algorithm and show real-world applicability.

The promise of deep reinforcement learning (RL) in solving complex and high-dimensional problems autonomously has attracted much interest among robotics researchers. However, effectively training an RL policy requires exploring a large set of robot states and actions, including many that are not safe for the robot. This is especially true for systems that are inherently unstable such as legged robots. One way to leverage RL for robotics problems is to learn the policy in computer simulation and then deploy it in the real world [97, 8]. However, this requires addressing the

[1]We use "legged robots" and "quadruped" interchangeably in the thesis.

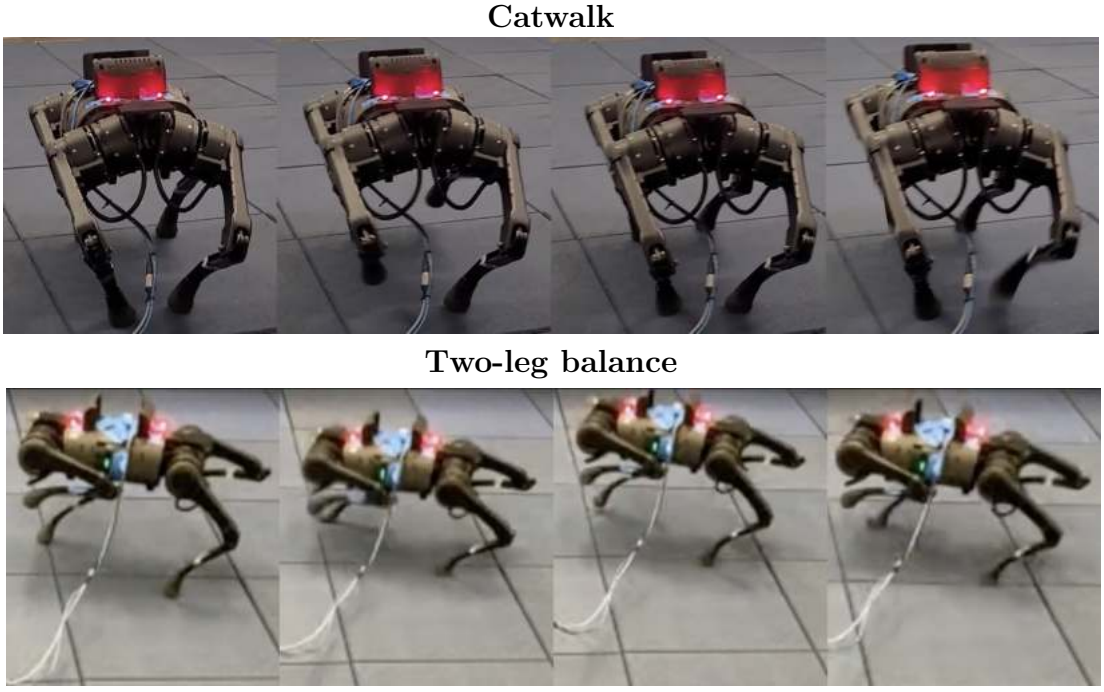


Figure 4.1: We evaluate our algorithm in quadrupedal locomotion tasks: catwalk and two-leg balance. The narrow feet placement and standing with two legs lead to falling easily during the learning process. Our algorithm overcomes these challenges–outperforming prior methods in terms of safety violations and learning efficiency both in simulation and the real world.

challenging sim-to-real gap. Another approach to tackle this issue is to directly learn or fine-tune a control policy in the real-world [63, 64], with the main challenge being ensuring safety during learning. Our work falls into this category by introducing a safe RL framework for learning quadrupedal locomotion while satisfying the safety constraints during training.

We formulate the problem of safe locomotion learning in the context of safe RL. Inspired by prior work [149, 178, 19], our learning framework adopts a two-policy structure: a *safe recovery policy* that recovers robots from near-unsafe states, and a *learner policy* that is optimized to perform the desired control task. Our safe learning framework switches between the safe recovery policy and the learner policy to prevent the learning agent from safety constraint violations (*e.g.,* falls).

Different from prior methods that learn a safety critic function which predicts the

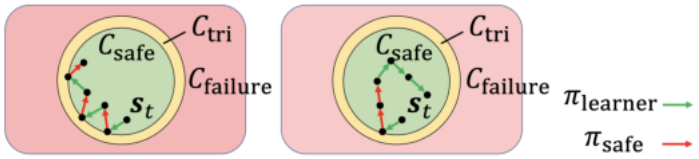


Figure 4.2: An illustration of safety trigger set $\mathcal{C}_{\text{tri}}$ (orange), safe set $\mathcal{C}_{\text{safe}}$ (green), and failure set $\mathcal{C}_{\text{failure}}$ (red) with two policies π_{learner}, and π_{safe}. On the left, without the reachability criteria, we encounter a frequent switch between the safe recovery policy π_{safe} and the learner policy π_{learner}. On the right, by applying the reachability criteria, we push the learning agent away from $\mathcal{C}_{\text{tri}}$, which reduces the risk of violating the safety constraints.

possibility of safe violations [149, 178, 19], we propose a model-based approach to determine when to switch between the two policies based on the knowledge about the system dynamics. In reality, it is often the case that the designer has some knowledge of the system dynamics at hand. Our goal is to *exploit this knowledge to design a safety mechanism without relying on a black-box approach* (*e.g.,* neural networks). More specifically, we first define a *safety trigger set* that includes states where the robot is close to violating safety constraints but can still be saved by a safe recovery policy. When the learner policy takes the robot to the safety trigger set, we switch to the safe recovery policy, which drives the robot back to safe states. We then determine when to switch back to the learner policy by leveraging an approximate dynamics model of the robot (*e.g.,* centroidal dynamics model for legged robots) to rollout the planned future robot trajectory: if the predicted future states are all in the safe states, we will hand the control back to learner policy, otherwise, we will keep using the safe recovery policy (see Fig. 4.2 for illustration). Such switch criteria allow the learning agent to explore near safety-violation regions while minimally intervening in the learning process. Notice that this dynamics model is a linear system, and it is derived based on the physics knowledge about the quadrupedal robot. The system parameters in this dynamics model need to be specified such as the friction coefficient. We assume the system parameters are known, and in the next Chapter 5, we will develop an approach to identify these parameters.

The contributions in this chapter are as follows. **(1)** We provide a theoretical analysis of the proposed framework and derive performance bounds under system dynamics error since perfect knowledge about the system dynamics is hard to obtain. Specifically, we construct a regret-type of bound that compares the learning outcome of the agent to the one with full knowledge of the system dynamics. This helps us to quantify the effect of dynamics error on learning performance. **(2)** To evaluate the performance of the framework, we compare the proposed safe learning framework with state-of-the-art safe RL algorithms on four simulated and real quadrupedal locomotion tasks: efficient gait, catwalk, two-leg balance, and pacing (see Fig. 4.1). In all cases, our algorithm achieves comparable or superior performance to prior approaches, averaging 48.6% fewer falls in simulation and zero or near-zero falls in the real world. In addition, we show that this framework can effectively reduce the number of uses of the safe recovery policy over prior approaches [149], which improves learning speed.

Prior Publications. Parts of this thesis have been published in [176].

Code. `https://sites.google.com/view/saferlleggedlocomotion/`

4.2 Problem Setup

We frame our problem as a constrained Markov Decision Process (CMDP) [10], defined as a tuple $< \mathcal{S}, \mathcal{A}, T, R, C >$. Here $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and T is the transition function that specifies the conditional probability $T(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ or the deterministic function $\boldsymbol{s}' = T(\boldsymbol{s}, \boldsymbol{a})$ that the next state is $\boldsymbol{s}'$ given the current state $\boldsymbol{s}$ and action $\boldsymbol{a}$. In addition, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and $C : \mathcal{S} \to \mathbb{R}$ is a constraint cost function that indicates the failure state (*e.g.,* falling down). The reward function encodes the benefit of using action $\boldsymbol{a}$ in state $\boldsymbol{s}$, while the cost function encodes the corresponding constraint violation penalty (we assume the value of

C is non-negative).

A policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ is a map from states to probability distributions on $\mathcal{A}$: $\boldsymbol{a} \sim \pi(\boldsymbol{a}|\boldsymbol{s})$. The state then transits from $\boldsymbol{s}$ to $\boldsymbol{s}'$ according to T. Finally, the agent receives a reward $R(\boldsymbol{s}, \boldsymbol{a})$ and incurs a cost $C(\boldsymbol{s})$.

Let $\gamma \in (0, 1)$ denote a discount factor, and τ denote the trajectory $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \cdots)$ from a policy π. We seek a policy π that maximizes a cumulative discounted reward: $J_R(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$, while keeping the cumulative discounted cost below a pre-defined threshold h : $J_C(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t)\right] \leq h$. Note that we refer to systems with $J_C(\pi) > h$ as *cost violations* or *failure*. One difference between conventional RL and safe RL is that we want the *entire* training process to satisfy the safety constraints, not just at the end of training.

4.3 Algorithm

In this section, we first provide the necessary definitions, followed by proposing the safe control approach using approximate knowledge about system dynamics. Finally, we conclude the section by outlining the proposed framework.

4.3.1 Setup

Definition 1. We define:

(1.1) Safety trigger set $\mathcal{C}_{\text{tri}} \subseteq \mathcal{S}$, a set of states that triggers π_{safe}.

(1.2) Safe set $\mathcal{C}_{\text{safe}} = \mathcal{C}_{\text{tri}} \setminus \mathcal{S}$ in which the state does not belong to the safety trigger set;

(1.3) Failure set $\mathcal{C}_{\text{failure}} = \{\boldsymbol{s} \in \mathcal{S} : C(\boldsymbol{s}) > 0\}$ in which an non-negative cost penalty (*i.e.,* falls) incurred for certain $\boldsymbol{s}$. Note that $\mathcal{C}_{\text{failure}} \subseteq \mathcal{C}_{\text{tri}}$.

Definition (1.3) defines unsafe states that we want to avoid. For example, when the height of a robot is too low (*i.e.,* indication of falls), the learning agent is in $\mathcal{C}_{\text{failure}}$. Fig. 4.2 shows these three sets. To ensure the safety of the learning agent, we will also need the following assumptions.

Assumption 2. An approximate knowledge of the true system dynamics T, denoted by $\hat{T}$, in which their difference is bounded by $\|T - \hat{T}\|_2 \leq \epsilon$.

Approximate dynamics models are often leveraged to achieve efficient planning of robot motions. For example, the centroidal dynamics model (CDM) or spring-loaded inverted pendulum model (SLIP) are commonly used to design locomotion controllers, while a unicycle model is popular for modeling wheeled robots. In this work, we utilize a centroidal dynamics model to predict the future state trajectory given the current state and π_{learner}'s control inputs to certify the learning agent's safety.

4.3.2 The Safe Control Approach

Based on these definitions and assumptions, the proposed algorithm uses two policies, a *learner policy* π_{learner}, which maximizes the task reward, and a *safe recovery policy* π_{safe}, which tries to bring the agent back to safe states when reaching the safety trigger set. A naive way to select which policy to query is as follows.

$$\boldsymbol{a}_{\text{naive}} = \begin{cases} \pi_{\text{safe}}(\cdot|\boldsymbol{s}), & \text{if } \boldsymbol{s} \in \mathcal{C}_{\text{tri}} \\ \pi_{\text{learner}}(\cdot|\boldsymbol{s}), & \text{otherwise.} \end{cases} \tag{4.1}$$

This control approach implies that whenever the state is in the safety trigger set, the safe recovery policy takes over the control from the learner, and returns the control back when the learner is in the safe set.

However, this poses a potential issue: while in principle the control approach in

Eq. (4.1) can ensure safety, it may cause frequent switches between π_{safe} and π_{learner}, which hinders policy exploration and impacts learning efficiency. Fig. 4.2 illustrates this observation. To address this issue, when the learning agent is in the safety trigger set, the safe recovery policy takes over the control and should return the control to the learning agent only when π_{learner} can guarantee that there would be *no further switches* for a specific future horizon. This reduces the potential use of the safe recovery policy. Formally, we define the following planning criteria for determining when should π_{safe} hand control back to π_{learner}.

Definition 2. Reachability Planning Criteria. $\exists\{\boldsymbol{a}_\tau\}_{\tau=t}^{i-1} \in \pi_{\text{safe}}$ with $\min_{i\in[t+1,t+w-1]} i$ such that $\{\boldsymbol{a}_\tau\}_{\tau=i}^{t+w-1} \in \pi_{\text{learner}}$ with $\boldsymbol{s}_\tau \notin \mathcal{C}_{\text{tri}}$ where $w \in \mathbb{R}^+$ is the planning step.

Definition (2) says that we want to find a minimal step i (*i.e.,* the minimal intervention) such that all the remaining actions are from π_{learner} after a minimal initial sequence of actions from π_{safe}. This ensures that when π_{safe} hands the control back to π_{learner}, π_{learner} can maximally reduce future use of the safe recovery policy. However, Definition (2) is computationally intensive and non-convex, which can limit the practicality of the proposed approach for real-time validation of safety in some applications such as robots with onboard processing. An alternative is to remove the planning component in reachability planning criteria and only verify whether future states are in the safety trigger set given all actions from π_{learner}. Formally, π_{safe} returns the control back to π_{learner} if the following criteria are satisfied.

Definition 3. Reachability Criteria. Check $\{\boldsymbol{a}_\tau\}_{\tau=t}^{w+t-1} \in \pi_{\text{learner}}$ such that $\{\boldsymbol{s}_\tau\}_{\tau=t+1}^{w+t} \notin \mathcal{C}_{\text{tri}}$.

The criteria say that at time t, we check whether the future states are in $\mathcal{C}_{\text{tri}}$ for

w steps. Hence, the final proposed safe control approach becomes

$$\boldsymbol{a}_t = \begin{cases} \pi_{\text{safe}}(\cdot|\boldsymbol{s}_t), & \underbrace{\text{if } (\boldsymbol{s}_t \in \mathcal{C}_{\text{tri}})}_{\pi_{\text{safe}} \text{ trigger criterion}} \quad \vee \\ & \big((\boldsymbol{a}_{t-1} \in \pi_{\text{safe}}) \ \wedge \\ & (\nexists \{\boldsymbol{a}_\tau\}_{\tau=t}^{w+t-1} \in \pi_{\text{learner}} \text{ s.t.} \\ & \quad \underbrace{\{\boldsymbol{s}_\tau\}_{\tau=t+1}^{w+t} \notin \mathcal{C}_{\text{tri}})\big)}_{\pi_{\text{safe}} \text{ handing control back criterion}} \\ \pi_{\text{learner}}(\cdot|\boldsymbol{s}), & \text{otherwise.} \end{cases} \tag{4.2}$$

There are two conditions that we use a safe recovery policy. The first condition (*i.e.,* safe recovery policy trigger criterion) is when the current state is in the safety trigger set (*i.e.,* $\boldsymbol{s}_t \in \mathcal{C}_{\text{tri}}$). The second condition (*i.e.,* safe recovery policy handing control back criterion) is that when the previous action is from safe recovery policy and there *does not* exist a sequence of actions produced by the learner policy such that the resulting states are not in the safety trigger set (*i.e.,* $(\boldsymbol{a}_{t-1} \in \pi_{\text{safe}}) \ \wedge \ (\nexists \{\boldsymbol{a}_\tau\}_{\tau=t}^{w+t-1} \in \pi_{\text{learner}} \text{ s.t. } \{\boldsymbol{s}_\tau\}_{\tau=t+1}^{w+t} \notin \mathcal{C}_{\text{tri}})$). Note that the reachability criteria (*i.e.,* the second condition) are only checked if the previous action is from π_{safe}, which determines when to *hand the control back* to the learner. This is different from Recovery RL [149], in which π_{safe} hands the control back to the learner whenever the prediction of future cost constraint violations are below a certain threshold at any state in the environment. Our approach allows the agent to have enough freedom to explore the states that are near $\mathcal{C}_{\text{tri}}$ while avoiding constant use of π_{safe}. π_{safe} is triggered only when needed. Finally, the approximate dynamics $\hat{T}$ is used to unroll the future states given the current state and π_{learner} for the reachability criteria.

Algorithm 3 Safe RL with Approximate System Dynamics

1: **Given: (1)** Knowledge about system dynamics $\boldsymbol{s}' = \hat{T}(\boldsymbol{s}, \boldsymbol{a})$ or $\boldsymbol{s}' = \hat{T}(\cdot|\boldsymbol{s}, \boldsymbol{a})$, **(2)** an initial learner policy $\pi^0_{\text{learner}} = \pi(\cdot|\boldsymbol{\theta}^0_{\text{learner}})$ parameterized by $\boldsymbol{\theta}^0_{\text{learner}}$, **(3)** a safe recovery policy $\pi_{\text{safe}}(\cdot|\boldsymbol{s})$, **(4)** safety trigger set $\mathcal{C}_{\text{tri}}$, and **(5)** a trajectory buffer $\mathcal{B}$
2: **Parameter: (1)** the number of policy updates K, **(2)** the number of interactions W, **(3)** planning horizon w
3: **for** $k = 0, 1, \cdots, K$ **do**
4: **for** $t = 1, \cdots, W$ **do**
5: Use control approach in Eq. (4.2)
6: Observe $\boldsymbol{s}_{t+1}, r_t = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$
7: Store data $(\boldsymbol{s}_t, \boldsymbol{a}_t^{\text{learner}}, r_t - z)$ in $\mathcal{B}$, where $z = 1$ if $\boldsymbol{a}_t \in \pi_{\text{safe}}$, and zero otherwise.
8: Update $\boldsymbol{\theta}^k_{\text{learner}}$ using $\mathcal{B}$

4.3.3 Reinforcement Learning with the Safe Control Approach

Algorithm 3 illustrates the proposed framework with the safe control approach. In addition, when storing the data trajectory in the replay buffer (line 7), we always use the action proposed by π_{learner} with the negative penalty for indicating the use of π_{safe}. This implies that we treat π_{safe} as *part of the environment*, and we want the learning agent to reduce the use of π_{safe} over the learning process to embed safety to the agent. Note that one can also terminate the episode when triggering π_{safe} to discourage the use of π_{safe}.

4.4 Theoretical Analysis for Dynamics Error

The proposed reachability criteria rely on the knowledge about the system dynamics T to predict the future state trajectory. Hence, we want to quantify the effect of the approximation error of the system dynamics on the performance of the task objective. Let $\{\boldsymbol{y}_t\} \in \mathcal{S}$ denote a target safe trajectory that the learning agent wants to track. Then, we consider the following simplified problem in which we want to find a sequence of actions that minimize the tracking errors for W steps between the safe

and the planned trajectories under the proposed control approach.

$$\underset{\boldsymbol{a}_1 \cdots \boldsymbol{a}_W}{\arg\min} \sum_{t=1}^{W} \underbrace{\left\| \hat{T}(\boldsymbol{x}_t, \boldsymbol{a}_t) - \boldsymbol{y}_{t+1} \right\|_2}_{\text{task objective}} \text{ s.t. control method in (4.2)} \tag{4.3}$$

Assumption 3. We assume:

(2.1) **Dynamics Parameterization**: The true dynamics $\boldsymbol{s}' = T(\boldsymbol{s}, \boldsymbol{a})$ is a time-invariant linear dynamics $\boldsymbol{s}' = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{B}\boldsymbol{a}$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are system matrices.

(2.2) **Approximate Dynamics**: The dynamics $\hat{T}(\boldsymbol{s}, \boldsymbol{a})$ is a time-invariant linear dynamics $\boldsymbol{s}' = \hat{\boldsymbol{A}}\boldsymbol{s} + \hat{\boldsymbol{B}}\boldsymbol{a}$.

(2.3) **Smoothness**: The system dynamics $T(\boldsymbol{s}, \boldsymbol{a})$ is α-Holder continuous in the second-order argument for $\alpha \in \mathbb{R}^+$, *i.e.,* for all $\boldsymbol{s}$, there exists $G \in \mathbb{R}^+$, such that

$$\|T(\boldsymbol{s}, \boldsymbol{a}_1) - T(\boldsymbol{s}, \boldsymbol{a}_2)\|_2 \leq G\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|_2^{\alpha}, \forall \boldsymbol{s}_1, \boldsymbol{s}_2.$$

G and α control the sensitivity of the system dynamics function to a perturbation in $\boldsymbol{a}$. This is also true for $\hat{T}$.

(2.4) **Size of the Control Space**: For a constant D, $\text{diam}(\mathcal{A}) \leq D$, where $\mathcal{A}$ is the action space and diam is the diameter of the set (*i.e.,* $\|\boldsymbol{a}_1 - \boldsymbol{a}_2\|_2 \leq D$, for every $\boldsymbol{a}_1, \boldsymbol{a}_2 \in \mathcal{A}$).

Asm. (2.1) and (2.2) define the structure of the system dynamics, which has been applied to a wide range of tasks such as quadrupedal locomotion [44]. In addition, Asm. (2.3) says that the function value does not change abruptly given two points. Asm. (2.4) implies that the size of $\mathcal{A}$ is bounded.

To provide a performance guarantee of Problem (4.3), we compare the value of the task objective of the proposed algorithm running *online* (denoted by $\text{cost}(\hat{T})$) against

the optimal *offline* algorithm, which makes the optimal decision with full knowledge of the system dynamics T (denoted by $\text{cost}(T)$). Hence, we have the following Theorem.

Theorem 4.4.1. ***(Performance Guarantee.)*** *We have*

$$\begin{aligned}\mathbb{E}[\text{cost}(\hat{T})] \leq & \mathbb{E}[\text{cost}(T)] + 2^{\frac{\alpha}{2}+1} GW \cdot |\sigma_1(\boldsymbol{A} - \hat{\boldsymbol{A}})|^{\alpha} \\ & \cdot \left(|\sigma_1(\boldsymbol{B} - \hat{\boldsymbol{B}})|^{\alpha} a_{\max}^{\alpha} \left(\frac{2}{\alpha + 2} \right) + \|\boldsymbol{s}_1\|_2^{\alpha} \right),\end{aligned}$$

where $\sigma_1(\cdot)$ *is the largest singular value of a matrix,* $\boldsymbol{s}_1$ *is the initial state, and scalar value* $a_{\max} := \max_{t \in \{1, \cdots, W\}} \|\boldsymbol{a}_t\|_2$.

Proof. Our proof follows the setup in [32]. Please read the paper for more details. Let modeled system dynamics be $\hat{\boldsymbol{x}}_{t+1} = (\boldsymbol{A} + \boldsymbol{A}_E)\hat{\boldsymbol{x}}_t + (\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_t$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the *true* system matrices and $\boldsymbol{A}_E$ and $\boldsymbol{B}_E$ represent the model error between the true dynamics and the modeled dynamics. In addition, we use this parameterization of the observation tracking error

$$\boldsymbol{y}_t - \boldsymbol{y}_{t|\tau} = \sum_{s=\tau+1}^{t} f(t-s)\boldsymbol{e}(s), \tag{4.4}$$

where $\boldsymbol{y}_t$ is the true target state at time t and $\boldsymbol{y}_{t|\tau}$ is the predicted state at time t with the initial state at time τ (*i.e.,* $t > \tau$). Here, we want to make some observations

about the model error over time:

$$
\begin{aligned}
\hat{\boldsymbol{x}}_1 &= (\boldsymbol{A} + \boldsymbol{A}_E)\boldsymbol{x}_0 + (\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_0, \\
\hat{\boldsymbol{x}}_2 &= (\boldsymbol{A} + \boldsymbol{A}_E)\boldsymbol{x}_1 + (\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_1 \\
&= (\boldsymbol{A} + \boldsymbol{A}_E)\big((\boldsymbol{A} + \boldsymbol{A}_E)\boldsymbol{x}_0 + (\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_0\big)\boldsymbol{x}_1 \\
&\quad + (\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_1 \\
&= (\boldsymbol{A} + \boldsymbol{A}_E)^2\boldsymbol{x}_0 + (\boldsymbol{A} + \boldsymbol{A}_E)(\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_0 \\
&\quad + (\boldsymbol{A} + \boldsymbol{A}_E)^0(\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_1
\end{aligned} \tag{4.5}
$$

Then we can define

$$
\boldsymbol{e}(s) = \begin{cases} (\boldsymbol{B} + \boldsymbol{B}_E)\boldsymbol{u}_{s-1}, & \text{if } s \geq 1 \\ \boldsymbol{x}_0, & \text{if } s = 0. \end{cases} \tag{4.6}
$$

and

$$
f(s) = (\boldsymbol{A} + \boldsymbol{A}_E)^s, \quad \text{if } s \geq 0. \tag{4.7}
$$

Then

$$
\begin{aligned}
&\mathbb{E}[\| \sum_{s=\tau-k}^{\tau} f(\tau-s)\boldsymbol{e}(s)\|_2^2] \\
&= \mathrm{tr}(\sum_{s_1,s_2=0}^{k} f(s_1)^T f(s_2)\mathbb{E}[\boldsymbol{e}(\tau-s_2)\boldsymbol{e}(\tau-s_1)^T]) \\
&= \mathrm{tr}\Big(\sum_{s=0}^{k} \big((\boldsymbol{A}+\boldsymbol{A}_E)^s\big)^T (\boldsymbol{A}+\boldsymbol{A}_E)^s \\
&\cdot \mathbb{E}[(\boldsymbol{B}+\boldsymbol{B}_E)\boldsymbol{u}_{k-s}\boldsymbol{u}_{k-s}^T(\boldsymbol{B}+\boldsymbol{B}_E)^T]\Big) \\
&+ \big((\boldsymbol{A}+\boldsymbol{A}_E)^{k+1}\boldsymbol{x}_0\big)^T\big((\boldsymbol{A}+\boldsymbol{A}_E)^{k+1}\boldsymbol{x}_0\big) \\
&= \mathrm{tr}\Big(\sum_{s=0}^{k} \big((\boldsymbol{A}+\boldsymbol{A}_E)^s(\boldsymbol{B}+\boldsymbol{B}_E)\boldsymbol{u}_{k-s}\big)^T \\
&\cdot \big((\boldsymbol{A}+\boldsymbol{A}_E)^s(\boldsymbol{B}+\boldsymbol{B}_E)\boldsymbol{u}_{k-s}\big)\Big) + \|(\boldsymbol{A}+\boldsymbol{A}_E)^{k+1}\boldsymbol{x}_0\|_2^2 \\
&= \sum_{s=0}^{k} \|(\boldsymbol{A}+\boldsymbol{A}_E)^s(\boldsymbol{B}+\boldsymbol{B}_E)\boldsymbol{u}_{k-s}\|_2^2 + \|(\boldsymbol{A}+\boldsymbol{A}_E)^{k+1}\boldsymbol{x}_0\|_2^2 \\
&\le \sum_{s=0}^{k} (\|(\boldsymbol{A}+\boldsymbol{A}_E)^s\|_2^2)(\|\boldsymbol{B}+\boldsymbol{B}_E\|_2^2)(\|\boldsymbol{u}_{k-s}\|_2^2) \\
&+ \|(\boldsymbol{A}+\boldsymbol{A}_E)^{k+1}\|_2^2\|\boldsymbol{x}_0\|_2^2 \\
&= \sum_{k=0}^{v-1}\Big(\sum_{s=0}^{k-1} \mathbb{E}[\|\boldsymbol{A}^{s+1}-(\boldsymbol{A}+\boldsymbol{E})^{s+1}\|_2^2\|\boldsymbol{u}_{k-s-1}\|_2^2] \\
&+ \mathbb{E}[\|\boldsymbol{A}^{k+1}-(\boldsymbol{A}+\boldsymbol{E})^{k+1}\|_2^2\|\boldsymbol{B}^{k+1}-(\boldsymbol{B}+\boldsymbol{E})^{k+1}\|_2^2\|\boldsymbol{x}_0\|_2^2]\Big)^{\frac{\alpha}{2}} \\
&\le \sum_{k=0}^{v-1}\Big(k\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^2\sigma_1(\boldsymbol{B}-\hat{\boldsymbol{B}})^2 a_{\max}^2 + \sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^2\|\boldsymbol{x}_0\|_2^2\Big)^{\frac{\alpha}{2}} \\
&\le 2^{\frac{\alpha}{2}} \sum_{k=0}^{v-1}\Big(k^{\frac{\alpha}{2}}\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha\sigma_1(\boldsymbol{B}-\hat{\boldsymbol{B}})^\alpha + \sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha\|\boldsymbol{x}_0\|_2^\alpha\Big) \\
=&2^{\frac{\alpha}{2}}\Big(\sigma_1(\boldsymbol{B}-\hat{\boldsymbol{B}})^\alpha\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha a_{\max}^\alpha \sum_{k=0}^{v-1} k^{\frac{\alpha}{2}} + v\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha\|\boldsymbol{x}_0\|_2^2\Big) \\
\le&2^{\frac{\alpha}{2}}\Big(\sigma_1(\boldsymbol{B}-\hat{\boldsymbol{B}})^\alpha\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha a_{\max}^\alpha \frac{2v^{\frac{\alpha}{2}+1}}{\alpha+2} + v\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha\|\boldsymbol{x}_0\|_2^\alpha\Big) \\
&= 2^{\frac{\alpha}{2}}\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^\alpha(\sigma_1(\boldsymbol{B}-\hat{\boldsymbol{B}})^\alpha\frac{2}{\alpha+2} + \|\boldsymbol{x}_0\|_2^\alpha)
\end{aligned} \tag{4.8}
$$

Hence by

$$\mathbb{E}[\text{cost}(\hat{T})] \leq \mathbb{E}[\text{cost}(T)] + 2GW \sum_{k=0}^{v-1} \|f_k\|^{\alpha}, \tag{4.9}$$

we have

$$\begin{aligned} &\mathbb{E}[\text{cost}(\hat{T})] \leq \mathbb{E}[\text{cost}(T)] \\ &+ 2^{\frac{\alpha}{2}+1} GW\sigma_1(\boldsymbol{A}-\hat{\boldsymbol{A}})^{\alpha}(\sigma_1(\boldsymbol{B}-\hat{\boldsymbol{B}})^{\alpha}\frac{2}{\alpha+2} + \|\boldsymbol{x}_0\|_2^{\alpha}), \end{aligned} \tag{4.10}$$

which we complete the proof. □

We now make several observations for Theorem 4.4.1.

(1) If there is no dynamics error, *i.e.,* $\boldsymbol{A} = \hat{\boldsymbol{A}}, \boldsymbol{B} = \hat{\boldsymbol{B}}$, then $\mathbb{E}[\text{cost}(\hat{T})] \leq \mathbb{E}[\text{cost}(T)]$, which implies that the online decision planning performs as good as the offline algorithm. However, when there is a dynamics error, the leading singular value of the system matrices difference will potentially lower the performance of the task objective with the cost violations.

(2) If the system dynamics is less smooth, *i.e.,* α is large, then the performance of the proposed algorithm degrades. This implies that when the system is unstable and unpredictable, we have a large tracking error.

(3) It shows the performance drop to the offline algorithm grows linearly w.r.t the horizon W. In addition, the smoothness value G represents how *predicable* the dynamics is–the smaller the value is, the more predicable the dynamics are.

(4) In practice for the locomotion tasks in the paper, their system dynamics are non-linear. However, we can treat it *locally* linear when the time step is small enough. We leave the non-linear version of the bound as the future work. In summary, Thm 4.4.1 provides an intuition about the effect of horizon W and model errors.

4.5 Experiments

We study the following questions: **(1)** How does our algorithm perform compared to other baselines in terms of constraint satisfaction and final reward? **(2)** What is the effect of the planning horizon and the safety trigger set?

4.5.1 Setup

Robots. We evaluate our approach on a simulated Unitree Laikago [2], a quadrupedal robot that weighs 24kg and has 12 actuated joints. For the real-world experiment, we use a Unitree A1 [1], a quadrupedal robot that weighs 12.7kg with better hardware reliability and agility.

Hierarchical Policies. We use a hierarchical policy framework that combines RL and optimal control for π_{learner} and π_{safe}. This framework consists of a high-level RL policy that produces gait parameters and feet placements, and a low-level model predictive control (MPC) controller [44] that takes in these parameters to compute desired torque for each motor in the robot. Instead of directly commanding the motor's angle, this approach offers stable operation and streamlines the policy training due to a smaller action space and a robust MPC controller. The gait parameters include stepping frequency ($\Omega \in \mathbb{R}$, the frequency of completing one swing-stance cycle of each leg), swing ratio ($p_{\text{swing}} \in \mathbb{R}$, the portion of swing duration), and phase offsets relative to the front-right leg ($\theta_1, \theta_2, \theta_3 \in \mathbb{R}$, the angle added to the front-right leg's phase angle for the other three leg's phase angles). The feet placements include $p_{\text{FR},y}, p_{\text{FL},y}, p_{\text{RR},y}$ and $p_{\text{RL},y} \in \mathbb{R}$, which are the desired feet positions in the side-way (y) direction (FR: front-right; FL: front-left; RR: rear-right; RL: rear-left). Such policy parameterization allows us to take advantage of the MPC controller while being expressive enough to produce a diverse set of gaits.

The MPC controller uses a *centroidal dynamics model* (CDM) proposed in [44]

to serve as $\hat{T}$. The dynamics can be expressed as a linear dynamic model $\boldsymbol{s}' = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{B}\boldsymbol{a}$, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are state and control matrices that depend on the gait parameters and feet positions. When rolling out the states using $\hat{T}$, we use the action from π_{learner} and keep the other MPC parameters fixed. The state in CDM is $\boldsymbol{s}_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$, where $x, y, z \in \mathbb{R}$ are the robot's position, $\dot{x}, \dot{y}, \dot{z} \in \mathbb{R}$ are the velocity, ϕ, θ, ψ are the roll, pitch, yaw, and $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are the angular velocity.

Tasks. We compare our algorithm with existing approaches on four quadrupedal locomotion tasks shown in Fig. 4.1 and Fig. 4.3.

(1) Efficient Gait. The robot learns how to walk with low energy consumption. The robot is rewarded for consuming less energy. The actions of the policy network produce the *delta change* of the gait parameters at each step, including the delta change of the stepping frequency Ω, the swing ratio p_{swing}, and the phase offsets $\theta_1, \theta_2, \theta_3$.

(2) Catwalk. The robot learns a catwalk gait pattern, in which the left and right two feet are close to each other. This is challenging because by narrowing the support polygon, the robot becomes less stable. The robot is rewarded for using narrow foot placement: $R := e - (p_{\text{FR},y} - p_{\text{FL},y})^2 - (p_{\text{RR},y} - p_{\text{RL},y})^2$, where $e \in \mathbb{R}^+$ is a positive survival bonus to make the reward non-negative. The actions of the policy network produce the gait parameters in the efficient gait task and the feet placement $p_{\text{FR},y}, p_{\text{FL},y}, p_{\text{RR},y}$ and $p_{\text{RL},y}$.

(3) Two-leg Balance. The robot learns a two-leg balance policy, in which the front-right and rear-left feet are in stance, and the other two are lifted. The robot can easily fall without delicate balance control because the contact polygon degenerates into a line segment. The robot can only control these two stance feet, and is rewarded for staying at a target height of 0.45m: $R := e - (z - 0.45)^2$. The action includes the delta change of the stepping frequency Ω, the swing ratio p_{swing}, and the phase offset

θ_3.

(4) Pacing. We want to produce the desired stepping frequency and the swing ratio for performing the pacing behavior under different desired speeds. The robot is rewarded for matching target speed: $R := e - \|[\dot{x}, \dot{y}, \dot{z}]^T - [\dot{x}_{\text{target}}, \dot{y}_{\text{target}}, \dot{z}_{\text{target}}]^T\|_2^2$, where $\dot{x}_{\text{target}}, \dot{y}_{\text{target}}, \dot{z}_{\text{target}}$ are the target velocities. The action includes the delta change of the stepping frequency Ω, and the swing ratio p_{swing}. Note that here we fixed the phase offsets to be $\theta_1 = \pi, \theta_2 = 0, \theta_3 = \pi$ to enforce the pacing gait: legs on the same side move in sync.

Baselines. The goal is to demonstrate that the proposed approach can enable safe training using the knowledge of the system dynamics to determine switching timing. In addition, we position our paper in the field of RL. Hence we compare our method with the following state-of-the-art RL or safe RL baselines.

(1) TRPO. Trust region policy optimization (TRPO [133]) only optimizes the reward objective. This will serve as a cost constraint performance *lower* bound.

(2) PCPO. Projection-based constrained policy optimization (PCPO, [173]) optimizes the reward while using projection to satisfy the cost constraint. In addition, PCPO is the state-of-the-art model-free approach over CPO [6].

(3) TRPO w/ negative penalty (TRPO-N). It optimizes the reward objective with an added penalty for entering $\mathcal{C}_{\text{failure}}$: $R(s, a) - C(s)$. This is to show that the penalty-based approach would fail when the dynamics are complex.

(4) Recovery RL. We are inspired by Recovery RL [149], which also consists of two policies: a safe recovery policy and a learner policy. This is a direct comparison of the proposed switch criteria with the value function-based method. We use hierarchical policies for all the baselines.

Implementation Details. We implement these tasks using Pybullet [40] simulator. We use a two-layer multi-layer perceptron (MLP)-based policy network with a tanh activation function for π_{learner} and π_{safe}. The low-level MPC controller runs at the

frequency of 250Hz, and the policy network predicts the gait parameters with 125Hz.

(1) Observation Space. The observation space of the policy network includes the previous gait parameters, the height of the robot, base orientation, linear and angular velocities.

(2) Safe Recovery Policy π_{safe}. We want to learn a policy that can safely bring the agent to a balanced and stationary state. We use the above hierarchical approach to train the policy, and add random perturbation to the robot to simulate the unstable and unsafe behavior encountered during the learning process. This allows us to train a policy that can stabilize the robots robustly. The robot is rewarded for being stable with zero velocity and maintaining the desired height: $R := e - \|[z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T - [0.45, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T\|_2^2$. The action includes the gait parameters used in the efficient gait. Note that all π_{safe} in the simulation are learned first before learning π_{learner} in the proposed method and Recovery RL. We stop training π_{safe} during learning π_{learner}. For the real-world experiments, we compare two types of π_{safe}: a learned MLP-based π_{safe} and a simulation-tuned MPC controller. We find that the optimized MPC controller works better in the real world, possibly because the smaller number of tunable parameters leads to a smaller sim-to-real gap. As such, we use this controller in our real-world experiments.

(3) Constraint Cost Function C. For all the tasks, $C(\cdot)$ outputs 1 when the height is below 0.1m and zero otherwise.

(4) Safety Trigger Set $\mathcal{C}_{\text{tri}}$. To ensure safety, we want the robot to maintain a certain height, stay upright with smaller side-tilting velocities. Hence we use the following safety trigger set for Laikago: $\mathcal{C}_{\text{tri}} = \{\boldsymbol{s} \in \mathcal{S} : z < 0.4 \vee z > 0.55 \vee |\phi| > 0.26 \vee |\theta| > 0.26 \vee |\dot{y}| > 0.5 \vee |\dot{\phi}| > 0.5\}$, where the unit for angle is in radian. For A1 robot, we use the same safety trigger set except for the height bounds to be $z < 0.2$ and $z > 0.3$ since the robot is smaller. We design our $\mathcal{C}_{\text{tri}}$ inspired by the capturability theory [30]. In particular, we choose an initial tight $\mathcal{C}_{\text{tri}}$ such that the corresponding

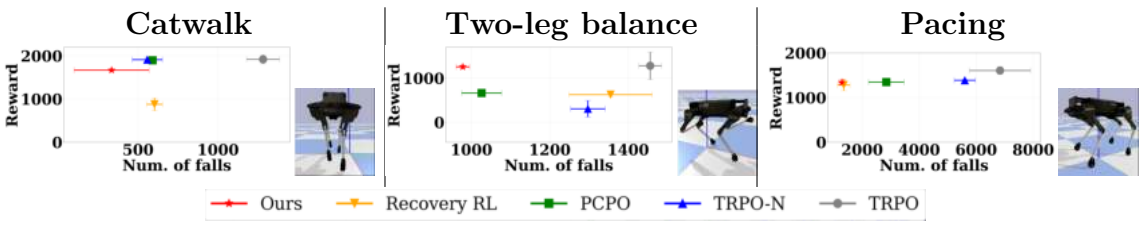


Figure 4.3: **Simulation Results.** We report the total number of falls versus final rewards for the tested algorithms and task pairs over five runs. We observe that the proposed approach achieves the fewest number of falls while having comparable reward performance (*i.e.,* in the top-left corner) (best viewed in color).

capture point for the robot does not exceed the task space (0.15m for Laikago and 0.1m for A1). We then fine-tune $\mathcal{C}_{\text{tri}}$ on the real A1 robot with a random policy to identify $\mathcal{C}_{\text{tri}}$ that can prevent the robot from falling. In addition, we also conduct an ablation on the effect of the safety trigger set where we use a *stricter* criterion: $\mathcal{C}^2_{\text{tri}} = \{\boldsymbol{s} \in \mathcal{S} : z < 0.4, z > 0.55, |\phi| > 0.26, |\theta| > 0.26, |\dot{y}| > 0.375, |\dot{\phi}| > 0.375\}$, and $\mathcal{C}^3_{\text{tri}} = \{\boldsymbol{s} \in \mathcal{S} : z < 0.4, z > 0.55, |\phi| > 0.26, |\theta| > 0.26, |\dot{y}| > 0.25, |\dot{\phi}| > 0.25\}$. Note that $\mathcal{C}^1_{\text{tri}} := \mathcal{C}_{\text{tri}}$ (default criteria).

(5) Planning Horizon w**.** We also ablate the planning horizon w (one step is 0.016 seconds) to see its effect on the number of uses of the safe recovery policy and falling events. Specifically, we use $w = 0, 5, 10, 15, 20$. Other than explicitly specified, we use $w = 10$ as the default in simulation. For the real-world experiment, we use $w = 0$ for π_{safe} that lasts 1 second due to computation budget in planning and we find this approach empirically works well.

4.5.2 Experiment Results

Simulation Results. The total number of falls versus the final reward value is shown for all tested algorithms and tasks in Fig. 4.3. The learning performance for baseline oracle, TRPO, indicates the reward and the constraint performance when the safety constraint is *ignored*. Ideally, we want the algorithm to be in the *top-left* corner (more rewards and fewer falls). Overall, we find that our algorithm is

(a)

	Num. of falls	Num. of π_{safe}	Reward
Ours $w = 0$	551 ± 84	106,280 ± 4,921	1517 ± 282
Ours $w = 5$	548 ± 87	**104,532 ± 15,496**	**1677 ± 32**
Ours $w = 10$	335 ± 34	116,155 ± 55,715	1575 ± 275
Ours $w = 15$	299 ± 86	157,285 ± 5,085	1506 ± 229
Ours $w = 20$	**204 ± 20**	194,198 ± 5,114	1433 ± 307
Recovery RL	603 ± 50	485,633 ± 325,133	894 ± 163

(b)

	Num. of falls	Num. of π_{safe}	Reward
Ours $\mathcal{C}^1_{\text{tri}}$	335 ± 234	**116,155 ± 55,715**	**1575 ± 275**
Ours $\mathcal{C}^2_{\text{tri}}$	313 ± 32	141,488 ± 16,536	1525 ± 180
Ours $\mathcal{C}^3_{\text{tri}}$	**275 ± 137**	246,651 ± 25,982	677 ± 154
Recovery RL	603 ± 50	485,633 ± 325,133	894 ± 163

Figure 4.4: **Simulation Results. (a)** Ablations on the planning horizon w in the catwalk task in terms of the number of falls, the number of uses of π_{safe}, and the reward performance. Results suggest that with an increasing number of w, one can reduce the number of falls with more safety trigger events. **(b)** Ablations on the design of the safety trigger set $\mathcal{C}_{\text{tri}}$ in the catwalk task. Results suggest that the looser $\mathcal{C}_{\text{tri}}$ is, the fewer number of falls are. The best numbers are bold.

able to improve the reward while achieving the fewest number of falls in all tasks (in the top-left corner). In addition, we observe that **(1)** Recovery RL has more falls and cannot improve reward effectively in the catwalk tasks. This is because the safety critic cannot predict the cost violation well; **(2)** TRPO-N requires a significant effort to select a good value of C to balance between the reward improvement and the constraint satisfaction; **(3)** PCPO has a moderate reward and constraint performance due to the projection step.

We also observe that the learning algorithm in the two-leg balance (7.19% for our algorithm of the total of 13,600 training trajectories) and pacing tasks (9.50%) tend to have more falls than that of the catwalk task (2.46%). This is because the robot with only two stance legs is highly unstable, and the pacing gait is different from the gait used in the safe recovery policy, which creates a sudden change of the gait and hence causes a wiggling movement.

Ablation Studies. To demonstrate the efficacy of reachability criteria in the proposed framework, we ablate the planning horizon w. Fig. 4.4(a) shows the results with $w = 0, 5, 10, 15, 20$ in the catwalk task in terms of the total number of falls, the number of uses of π_{safe}, and the reward. Note that when $w = 0$, it is equivalent to the removal of the reachability criteria in our algorithm, and similar to a shielding-based MPC approach in [16]. In addition, $w = 0$ does not mean our method is similar to recovery RL since it can trigger π_{safe} before reaching the safety boundary. Overall, we observe that our approach outperforms Recovery RL in terms of the number of uses of π_{safe} and the reward, no matter how long the horizon is. This is because the safety critic in Recovery RL is too conservative in predicting the cost violation, which hinders the exploration of the agent (reducing the reward). In addition, the safety critic is pre-trained based on the worst-case policy that maximizes the cost violation. This leads to a conservative learning strategy. Furthermore, as we keep increasing w to 20, the number of falls reduces with more uses of π_{safe} and fewer rewards. This is expected since π_{safe} intervenes in the learning agent more often, which reduces the reward performance. This shows a trade-off between safety and reward performance, and the horizon w gives users a knob to tune based on the problems.

Furthermore, Fig. 4.4(b) ablates the design of the safety trigger set $\mathcal{C}_{\text{tri}}$ in the catwalk task. We see that the stricter $\mathcal{C}_{\text{tri}}$ is, the fewer the falls and the more uses of π_{safe} are. This observation implies one needs to design a good $\mathcal{C}_{\text{tri}}$ based on the recovery capability of π_{safe}.

Real-world Experiments. Fig. 4.5 shows the reward and the percentage of uses of π_{safe} among the trajectory steps collected for one policy update on the efficient gait, catwalk, and two-leg balance tasks. For the efficient gait and the catwalk tasks, one policy update corresponds to 10 trajectories of a total of 4,000 steps, and for the two-leg balance task, it corresponds to 5 trajectories of a total of 2,000 steps. Here, all the policies are trained from scratch, except for the two-leg balance task with the

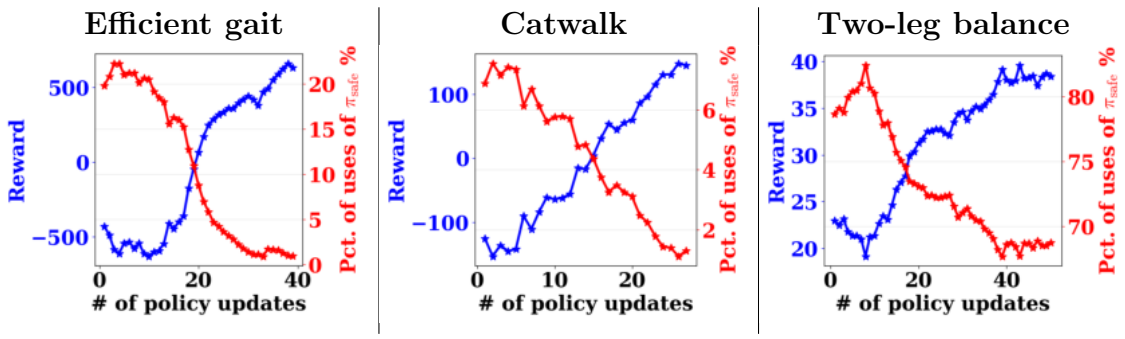


Figure 4.5: **Real-world Experiments.** We report the real-world experiment results of reward learning curves and the percentage of uses of π_{safe} on the efficient gait, catwalk, and two-leg balance tasks. We observe that our algorithm is able to improve the reward while avoiding triggering π_{safe} over the learning process. The learned policies are shown in Fig. 4.1.

policy pre-trained in the simulation since it requires more samples to train. Note that we do not include the other baselines since they cannot even collect 10 trajectories of data without falling to perform one policy gradient update. And Recovery RL requires much more steps to learn since its safety critic, pre-trained with the objective function to maximize the cost values in simulation, is too conservative in predicting the cost violations in the real world. For instance, in the catwalk task, our approach achieves *zero* falls and uses π_{safe} 6.8% of total steps among the first policy update, compared to Recovery RL with zero falls but 75.6% of total steps of use of π_{safe}. This makes these baselines laborious to train in practice.

Overall, we see that on these tasks, the reward increases, and the percentage of uses of π_{safe} decreases over policy updates. For instance, the percentage of uses of π_{safe} decreases from 20% to near 0% in the efficient gait task. For the two-leg balance task, the percentage drops from near 82.5% to 67.5%, suggesting that the two-leg balance is substantially harder than the previous two tasks. Still, the policy does improve the reward. This observation implies that the learner can gradually learn the task while avoiding triggering π_{safe}. In addition, this suggests that we can design a $\mathcal{C}_{\text{tri}}$ and π_{safe} that do not hinder the exploration of the policy as the performance increases.

Finally, Fig. 4.1 shows the results of learned locomotion skills (please visit our

project website for more videos). First, in the efficient gait task, the robot learns to use a smaller stepping frequency and achieves 34% less energy than the nominal trotting gait. Second, in the catwalk task, the distance between two sides of the legs is 0.09m, which is 40.9% smaller than the nominal distance. Third, in the two-leg balance task, the robot can maintain balance by jumping up to four times via two legs, compared to two jumps from the policy pre-trained from simulation. Without π_{safe}, learning such locomotion skills would damage the robot and require manually re-positioning the robot when falling. Note that there is *no single fall* nor *a manual reset* during the entire learning in the efficient gait (45mins of real-world training data collection, excluding automatic position reset or battery replacement) and catwalk tasks (29mins), and less than 5 falls in the two-leg balance task (28mins). Our results suggest that learning quadrupedal locomotion skills autonomously is possible in the real world.

4.6 Discussion and Conclusion

We studied the problem of safe reinforcement learning for acquiring locomotion skills for quadruped robots by combining a safe recovery policy and a learner policy. The safe recovery policy takes over the control when the robot is close to a safety violation, and returns the control back when the robot stays safe in the estimated near future. We provided a theoretical analysis of the algorithm and quantified the effect of the model errors on the learning performance. We evaluate our algorithm on both simulated and real quadruped robots for a variety of challenging locomotion tasks and demonstrate the effectiveness of the proposed framework compared to baseline methods.

No model is without limitation. Although uncommon, our current π_{safe} can still fail to save the robot from unsafe states (see supplementary video for failure case)

due to imperfections in π_{safe} and the safety trigger set $\mathcal{C}_{\text{tri}}$. Further investigations in improving these would enable the learner policy to explore the environment more effectively and thus improve the learning efficiency. In addition, we currently do not consider the model uncertainty from the environment and non-linear dynamics in our theoretical analysis. Including these would further improve the generality of our approach. Finally, designing an appropriate reward when incorporating the π_{safe} can impact our learning performance. We use a penalty-based approach that obtained reasonable results in our experiments. We plan to investigate this in future work to further improve the learning performance.

Acknowledgements. We thank Dr. Tingnan Zhang, Linda Luu, Dr. Sehoon Ha, Dr. Jie Tan, and Dr. Wenhao Yu for the helpful discussion and for setting up the experiments.

Chapter 5

Learning Informed by Prior Physics Models

5.1 Introduction

In Chapter 2 and Chapter 3, we assume that we know the system dynamics to get a simulator in which control policies are trained. Then in Chapter 4, the linear model based on the knowledge about the system dynamics is used to verify whether the state trajectory is safe or not. In addition, many RL algorithms require simulators or models to faithfully simulate the true system dynamics in the real world. Without an accurate simulator, the policy trained in simulation could have low reward task performance in the real world due to environment distribution shift. This can cause the policy to have unseen or undesirable behaviors, leading to unsafe events when deploying in the real world. Furthermore, in many applications system designers do know the dynamic equations of the system (*i.e.,* physics) but do not know its state and the system parameters. For example, the linear model used in quadrupedal robots in Chapter 4 requires specifying system parameters such as friction coefficient. And the friction coefficient can change over time due to motor deterioration or even

it is unknown to system designers. Hence in this chapter, we focus on the setting in which the system dynamics are known but its system parameters are unknown. We propose an approach to learning those dynamics for learning control policies in simulation or tracking the evolution of the system parameters. Knowing the system parameters could enhance the safety of the systems since we can train a better control policy in simulation that is useful for the real world deployment and it can also help us to schedule a repair when the system parameters start to degrade. Specifically, we consider the problem of estimating states (*e.g.,* position and velocity) and physical parameters (*e.g.,* friction, elasticity) from a sequence of observations when provided a dynamic equation that describes the behavior of the system.

The dynamic equation can arise from first principles (*e.g.,* Newton's laws for self-driving cars) and provide useful cues for learning, but its physical parameters are unknown. In addition, neural networks have become a core computational component in domains such as computer vision [67], natural language processing [43], and deep reinforcement learning [113]. Recent work has shown that neural networks can exhibit an inductive bias that is often introduced via designing specific structures [20]. This bias can be used to encode prior task knowledge that helps the network generalize to unseen data. For example, convolution neural networks capture the translation invariance of key image features. In this spirit, we develop a structured neural network model that leverages a dynamic equation to estimate both the state of a dynamical system (*e.g.,* position and velocity) and its physical parameters (*e.g.,* friction constants) from a sequence of partial observations (*e.g.,* images). Knowing the state and parameters of the system is useful for designing the control policy. For instance, for a self-driving car, we want to learn a neural network that estimates the vehicle's position and velocity from a sequence of egocentric camera images. The estimated state can then be used in a control policy. In addition, we want to track physical parameters over time, *e.g.,* friction coefficients. This is useful for vehicle maintenance and

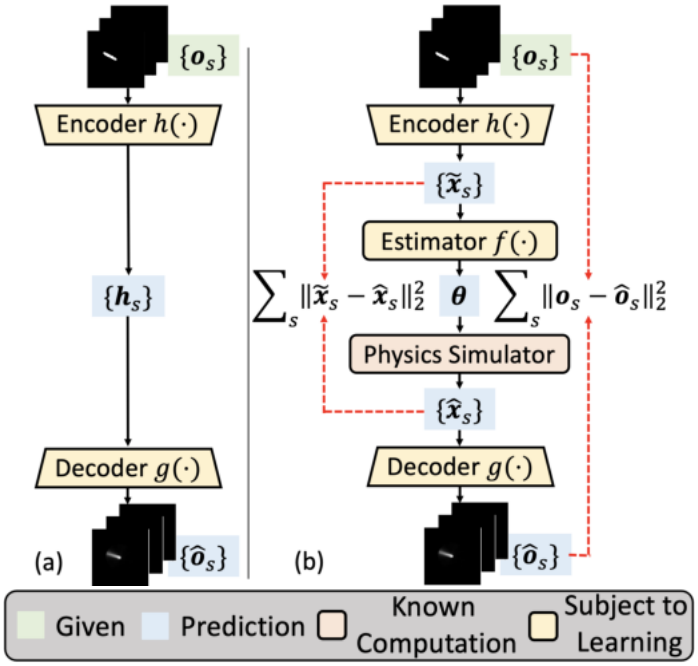


Figure 5.1: **(a)** An autoencoder learns a latent representation from observations. The observations here could be images or raw sensor measurements depending on the task. It shows an example with images of a pendulum. **(b)** Combining a dynamic equation and parameter estimator in (a). Our contribution is that the model is able to identify the system parameters from multiple observations without re-training the model. In contrast, the traditional grey-box system identification approach requires rerunning the identification procedure, which is time-consuming for run-time embedded systems. (best viewed in color)

safety. We expect that exploiting a neural network, regularized to follow a dynamic equation, will streamline the required data-intensive operations and yield improved performance.

Observations can be direct measurements of some system variables, or take the form of images. We group such observations over specified time windows and also refer to these groupings as observations. We obtain a compact representation for these observations that permits observation reconstruction using an autoencoder (see Fig. 5.1(a)). $\{\boldsymbol{h}_s\}$ in Fig. 5.1(a) is a representation of an observation sequence $\{\boldsymbol{o}_s\}$ over a specified time window, and $\{\hat{\boldsymbol{o}}_s\}$ is a reconstructed observation sequence. Learning the autoencoder is both data and computationally intensive. In addition, $\boldsymbol{h}_s$ may not be physically interpretable and is not guaranteed to have the Markov property, *i.e.,* be a "system state."

Motivated by these observations, we assume a simulator is provided that specifies the physical laws of the system. This model can arise from first principles (*e.g.,* Newton's laws), but its free parameters $\boldsymbol{\theta}$ (*e.g.,* masses, lengths) remain to be specified. In addition, we also assume that the model is provided with the observations from multiple sets of the system parameters. For example, engineers can gather the observations from a fleet of trains with different conditions of system components. This creates a dataset for the estimator to learn to predict the system parameters given the observations. Given an initial state and $\boldsymbol{\theta}$, the physics simulator generates a state trajectory $\{\hat{\boldsymbol{x}}_s\}$ consistent with the laws of physics. To leverage this model, we require an estimator $f(\cdot)$ that maps a sequence of states $\{\tilde{\boldsymbol{x}}_s\}$ to an estimate of $\boldsymbol{\theta}$. We then couple the estimator f and the physics simulator with the autoencoder as shown in Fig. 5.1(b). We train the autoencoder and $f(\cdot)$ to minimize the observation reconstruction loss $\sum_s \|\boldsymbol{o}_s - \hat{\boldsymbol{o}}_s\|_2^2$. Within this process, we train the encoder $h(\cdot)$ to minimize the sum of squared state errors: $\sum_s \|\tilde{\boldsymbol{x}}_s - \hat{\boldsymbol{x}}_s\|_2^2$. The complete model (Fig. 5.1(b)) is called *Autoencoder with Latent Physics* (ALPS).

This chapter's contributions are three-fold. **(1)** We propose and explore a model that integrates physics into an autoencoder to perform unsupervised state and physical parameter predictions. Compared to the prior work in system identification, our approach only requires a single pass to identify the system parameters without re-running the simulator. **(2)** We show that one can learn periodic or vibrational behavior in this setting using a Fourier feature of states. **(3)** We evaluate the proposed model in five tasks: (**a**) pendulum with friction observed through images, (**b**) mass-spring-damper system with stiffness observed through images, (**c**) two-body systems with potential energy observed through images, and (**d**) real-world vehicle wheel suspension system with stiffness observed through sensor measurements. (**e**) real-world full-scale vehicle wheel suspension system with stiffness and damper coefficient observed through sensor measurements. We first show the basic form of ALPS without the encoder, estimator, and decoder can identify the system parameters in the fully and partially observable cases. We then extend the results by testing the full model of ALPS. The proposed approach can achieve up to 4.8x and 6.3x better physical parameter and state prediction accuracy, respectively, over prior approaches. Finally, we conclude this chapter by outlining several potential research directions including combing safety analysis with ALPS for dealing with changing system dynamics.

5.2 Related Work

5.2.1 System Identification

The approaches that learn system dynamics can be grouped into three categories: black-box methods, grey-box methods, and white-box methods. First, black-box methods [80, 171] do not get access to the system dynamics. This approach usually trains a neural network that directly predicts the states given the input of observations and actions. However, this method lacks interpretability due to the absence of

constraints on the latent representations. In addition, it is unclear whether the model learns to generalize to the unseen data during training. Second, grey-box methods [41] exploit partial knowledge about the system dynamics and assume some parameters inside the system are unknown. This approach offers great flexibility in predicting system parameters and improves the interpretability of the model. Third, white box methods [122, 102] impose prior knowledge by using explicit dynamics models. Such an approach reduces the search space of neural networks but requires more knowledge about the system dynamics. Our work is closely related to the grey-box method. In addition, some works from adaptive system identification literature [61, 117] also estimate the system parameters given the recent observations collected online from the system. However, compared to the prior grey-box methods and adaptive system identification literature [93, 14], the estimator in ALPS provides a direct estimate of the system parameters via a single pass of the neural network without running the simulation, which can be costly during test time. Our approach improves the runtime estimate of the system parameters over the prior approach in classical system identification literature.

5.2.2 Physics-informed Neural Networks

There is growing interest in including a physics prior or algebraic and logical constraints into neural networks [140, 169, 187, 87, 166, 172, 167, 27, 17, 131, 170, 125, 25, 185, 104]. For example, [107, 62, 41, 59, 168, 183, 154, 72, 123, 184, 99] exploit Lagrangian or Hamiltonian mechanics to learn an energy-conserving system based on position, momentum, and the derivatives thereof along trajectories. These works assume the physical parameters of the system are constant and need not be estimated. In contrast, we estimate the physical parameters. This is important for fault detection and localization, and for safety. [130, 157, 68, 84] learn a general physical simulation from data, but their model is required to have *state* information or

a *known* forward rendering engine to map states to observations. In contrast, we learn a physics-based autoencoder to estimate states in an unsupervised manner. [82] also uses an autoencoder with physics to predict parameters. However, we show that learning state sequences as in [82] fails to generalize to unseen parameters when the system exhibits high-frequency behavior.

5.2.3 Fourier Features for High-frequency Data

Fourier features have been used to represent the positional information for the words within the sentence in the language model [158]. It is shown that Fourier features can improve the performance of the downstream natural language processing tasks. [144] further shows that using Fourier features helps neural networks learn high-frequency content in image regression tasks. They show that Fourier features increase the eigenvalues of neural networks, making neural networks converge faster. Our work is also inspired by [144]. However, there is a key difference in the use of Fourier features. Their approach requires specifying the Fourier series coefficients and the basis frequencies. In contrast, the Fourier features in our model are computed from the states, which allows us to predict physical parameters. In addition, the concurrent work [100] replaces the self-attention sublayers [158] with a Fourier transformation of the input word token in natural language processing. They show that this method is sufficient to capture semantic relationships in several text classification tasks. In this section, we provide a theoretical justification for using Fourier features to learn system dynamics. Next, we will formulate the problem.

5.3 Problem Setup

We consider the continuous-time system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}; \quad \boldsymbol{o} = g(\boldsymbol{x}), \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state, $\boldsymbol{u} \in \mathbb{R}^p$ is the input, $\boldsymbol{o} \in \mathbb{R}^q$ is the observation, and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times p}$. The function g provides a partial observation $\boldsymbol{o}$ of $\boldsymbol{x}$. In most situations, we have partial knowledge of $\boldsymbol{A}$ and $\boldsymbol{B}$ from physics. This is true and reasonable in many large-scale industrial applications ranging from wind turbines to aircraft, where system designers use physics knowledge to design machines. Hence we assume mappings $A(\cdot) : \Theta \to \mathbb{R}^{n\times n}$ and $B(\cdot) : \Theta \to \mathbb{R}^{n\times p}$, from physical parameters $\boldsymbol{\theta} \in \Theta$ to the system matrices $A(\boldsymbol{\theta}), B(\boldsymbol{\theta})$, are given.

In practice, we only have observations at discrete points in time. For simplicity, we assume these are equally spaced at times $t = 0, 1, \ldots$. At sample time t, we have the window of observations $\{\boldsymbol{o}_s\}_{s=t-\tau+1}^{t}$, where τ is the window length. We assume the sampling rate satisfies the Nyquist rate.

Our problem can now be stated as follows. Given functions $A(\cdot) : \Theta \to \mathbb{R}^{n\times n}$, $B(\cdot) : \Theta \to \mathbb{R}^{n\times p}$, we seek to learn a network that estimates a state sequence $\{\boldsymbol{x}_s\}_{s=t-\tau+1}^{t}$, physical parameters $\boldsymbol{\theta}$, and a mapping $g(\cdot)$ from a finite sequence of past observations $\{\boldsymbol{o}_s\}_{s=t-\tau+1}^{t}$ and past known inputs $\{\boldsymbol{u}_s\}_{s=t-\tau+1}^{t}$. Once trained, the network can predict states and has learned to adapt physical parameters to the context without re-training. Note that even though the network operates with time sampled variables, the physics simulator can be used to predict states at any time.

This problem setup is different from that of HNN [59], HGN [154], or Symplectic ODE-Net [183]. In their setups, the model is only trained on data from a *single* physical parameter $\boldsymbol{\theta}$, and hence they need to re-train networks for every new set of physical parameters.

To make the learning problem well-defined, we make the following assumption.

Assumption 4. (Identifiability [58]): For a sufficiently large τ, the sequence of observations and inputs $\{(\boldsymbol{x}_s, \boldsymbol{u}_s)\}_{s=t-\tau+1}^{t}$ uniquely specifies the parameters $\boldsymbol{\theta}$.

Parameter identifiability is a well-studied problem in system identification [58, 120, 109]. For example, the (continuous time) system $\dot{\boldsymbol{x}} = \begin{bmatrix} -(a+b) & b \\ b & -c \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \boldsymbol{u}$ with $\boldsymbol{o} = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x}$, where $\boldsymbol{\theta} = [a, b, c]^T$ and $g(\cdot)$ being a linear function, satisfies Assumption 4 when $b \neq 0$.

5.4 Network Architecture

The network in Fig. 5.1(b) is expanded in Fig. 5.2 into its four parts: an encoder network, a parameter estimator network, a physics simulator, and a decoder network.

The Encoder Network $h(\cdot)$ (from $\{\boldsymbol{o}_s\}$ to $\{\tilde{\boldsymbol{x}}_s\}$). We consider two types of observations: **(1)** pixel images, or **(2)** direct measurements of some system variables. For the first case, we use a convolution neural network to compress a pixel observation $\boldsymbol{o}$ into a compact vector embedding $\boldsymbol{z}' \in \mathbb{R}^d$, which preserves image features. For the second case, we use a feedforward network to project an observation $\boldsymbol{o}$ into some higher dimensional spaces with a vector embedding $\boldsymbol{z}'$. In addition, to estimate states from these vector embeddings, we need to aggregate $\{\boldsymbol{z}'_s\}$ to extract the local and global context of the dynamics. One approach is to use recurrent neural networks (RNN) (*e.g.,* Dreamer [65]). However, RNN suffers from gradient vanishing problems and slow computation when processing long-term sequences. Hence we use a self-attention network [158] to attend to $\boldsymbol{z}'$ of greatest importance to predict states and improve efficiency.

Furthermore, to inject a position signal of observations in the sequence, we add a positional encoding $\boldsymbol{p} \in \mathbb{R}^d$ to $\boldsymbol{z}'$ ($\boldsymbol{z} := \boldsymbol{z}' + \boldsymbol{p}$), where $\boldsymbol{p}$ are sine and cosine functions

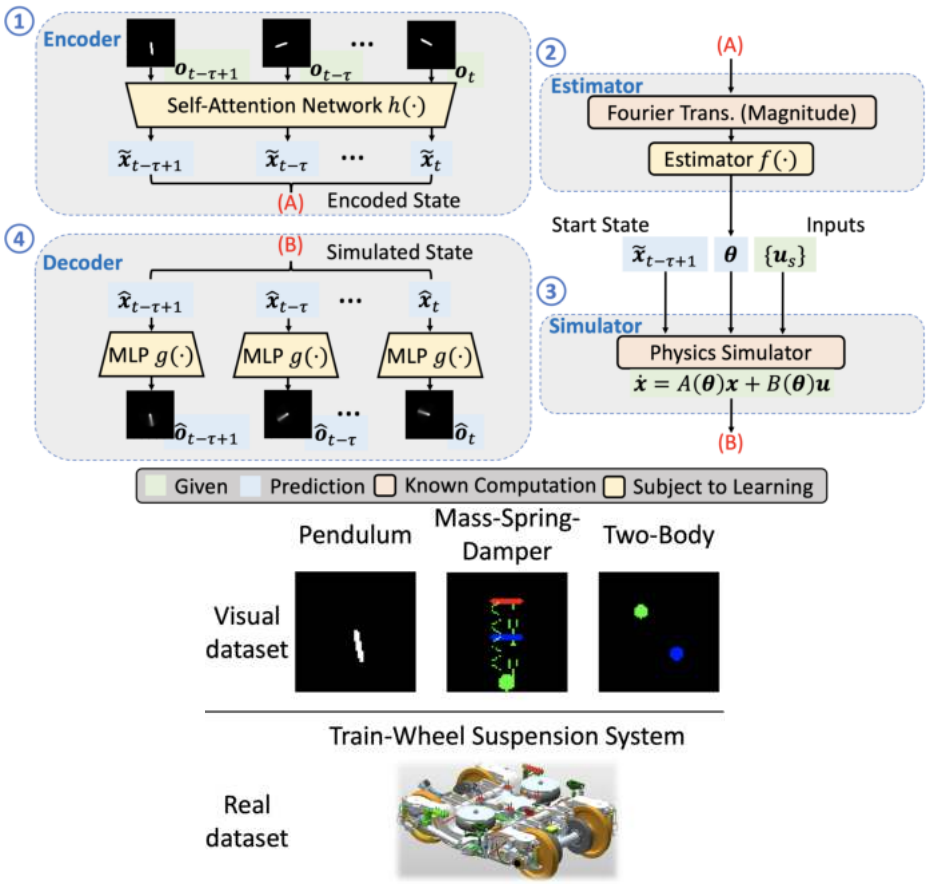


Figure 5.2: **ALPS and the Tasks.** ALPS consists of four parts: **(1)** an encoder network that estimates states from observations, **(2)** a parameter estimator network that predicts physical parameters from a state sequence, **(3)** a physics simulator generates a state trajectory provided with an initial state and values for the physical parameters, and **(4)** a decoder network reconstructs observations from states.

of different frequencies (see [158]). Finally, we stack embeddings over τ steps to form a matrix $\boldsymbol{Z} \in \mathbb{R}^{\tau \times d}$.

The self-attention module can be formulated as querying a dictionary with key-value pairs associated with learnable weight matrices $\boldsymbol{W}^Q \in \mathbb{R}^{d \times d_Q}$, $\boldsymbol{W}^K \in \mathbb{R}^{d \times d_K}$, and $\boldsymbol{W}^V \in \mathbb{R}^{d \times d_V}$ ($d_Q = d_K$ here): $\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V}$, where $\boldsymbol{Q} := \boldsymbol{Z}\boldsymbol{W}^Q$, $\boldsymbol{K} := \boldsymbol{Z}\boldsymbol{W}^K$, $\boldsymbol{V} := \boldsymbol{Z}\boldsymbol{W}^V$, and the softmax is taken over the sequence length τ. To provide multiview of the embedding subspace, we use the multihead variant of the attention module by concatenating each attention head along the sequence axis

$$\begin{aligned} \text{Multihead} &:= \text{Concat}(\text{head}_1, \ldots, \text{head}_i, \ldots, \text{head}_I)\boldsymbol{W}^O, \\ \text{head}_i &:= \text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}), \end{aligned}$$

where $\boldsymbol{W}^O \in \mathbb{R}^{Id_V \times d}$ are learnable matrices, I is the number of heads, and each attention head i has its own learnable weight matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d \times d_Q}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d \times d_K}$, and $\boldsymbol{W}_i^V \in \mathbb{R}^{d \times d_V}$.

Finally, a feedforward network takes in the Multihead embedding (of size $\mathbb{R}^{\tau \times d}$) and produces the parameters of the distribution for each state in the sequence. The parameters are used to define a posterior distribution over the encoded state $\tilde{\boldsymbol{x}}_s \sim Q(\cdot|\tilde{\boldsymbol{o}}_s)$ with the prior $P(\tilde{\boldsymbol{x}}_s)$. For a translational coordinate, the posterior distribution is a Gaussian distribution with a unit Gaussian prior. Hence the network predicts a mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and a standard deviation $\boldsymbol{\sigma} \in \mathbb{R}^n$ of a Gaussian distribution. In addition, for a rotational coordinate, the posterior distribution is a von Mises (vM) distribution with a unit vM prior. Similar to a Gaussian distribution, a vM distribution is defined by two parameters: a mean $\boldsymbol{\mu} \in \mathbb{R}^2$, $\|\boldsymbol{\mu}\|_2 = 1$ (the angular position $(\cos\varphi, \sin\varphi)$), and a concentration $\kappa \in \mathbb{R}^+$ around $\boldsymbol{\mu}$. Such parameterization is found useful in practice, *e.g.,* [186].

The Parameter Estimator Network $f(\cdot)$ (from $\{\tilde{\boldsymbol{x}}_s\}$ to $\boldsymbol{\theta}$). The parameter estimator network predicts physical parameters from state sequences $\{\tilde{\boldsymbol{x}}_s\}$. However, for systems that involve periodic or vibrational behavior, prior work [144] has shown that neural networks fail to capture the high-frequency content in the data. To solve this, we do a Fourier transform on each component j of state trajectories $\{\tilde{\boldsymbol{x}}_s(j)\}_{s=t-\tau+1}^{t}$ to get $\{\tilde{X}_\omega(j)\}_{\omega=t-\tau+1}^{t} : \tilde{X}_\omega(j) := \sum_{k=t-\tau+1}^{t} \tilde{\boldsymbol{x}}_k(j)\Big[\cos\Big(\frac{2\pi}{\tau}\omega k\Big) - i \cdot \sin\Big(\frac{2\pi}{\tau}\omega k\Big)\Big]$.

One may use $\{\tilde{X}_\omega(j)\}$ as features for the parameter estimator network to predict physical parameters. However, in the following section, we will show that by using the neural tangent kernel (NTK) theory [77], which treats neural networks as a kernel regression, the resulting kernel matrix of $\{\tilde{X}_\omega(j)\}$ does not preserve high-frequency components in the data. To solve this, we will show that using the *magnitude* of the Fourier features $\{|\tilde{X}_\omega(j)|\}$ alleviates the issue. Hence the parameter estimator network takes in a concatenation of $\{|\tilde{X}_\omega(j)|\}$ from each component of the state and predicts physical parameters $\boldsymbol{\theta}$.

The Physics Simulator (from $\tilde{\boldsymbol{x}}_{t-\tau+1}$ and $\boldsymbol{\theta}$ to $\{\hat{\boldsymbol{x}}_s\}$). Given a start state $\tilde{\boldsymbol{x}}_{t-\tau+1}$ (from the encoder's first state prediction) and values for the physical parameters $\boldsymbol{\theta}$, we use the neural ordinary differential equation (ODE) [33], a differential ODE solver, to generate a simulated state trajectory $\{\hat{\boldsymbol{x}}_s\}_{s=t-\tau+1}^{t} : \hat{\boldsymbol{x}}_{t-\tau+1}, \ldots, \hat{\boldsymbol{x}}_{t-1}, \hat{\boldsymbol{x}}_t = \text{ODESolver}(\tilde{\boldsymbol{x}}_{t-\tau+1}, \dot{\boldsymbol{x}} = A(\boldsymbol{\theta})\boldsymbol{x} + B(\boldsymbol{\theta})\boldsymbol{u}, \tau, \Delta)$, where ODESolver takes in a start state, an ODE, a window length, and a sampling time interval Δ. Note that $\hat{\boldsymbol{x}}_{t-\tau+1} = \tilde{\boldsymbol{x}}_{t-\tau+1}$. In addition, using an ODE solver allows us to generate an accurate state trajectory compared to that of RNN as in [65].

The Decoder Network $g(\cdot)$ (from $\hat{\boldsymbol{x}}_s$ to $\hat{\boldsymbol{o}}_s$). Finally, the decoder network is either a deconvolutional network (for image observations) or a feedforward network (for sensor measurements) that takes in each individual ODE-simulated state $\hat{\boldsymbol{x}}_s$ and generates a reconstructed observation $\hat{\boldsymbol{o}}_s$.

The Loss Function. Given a sequence of τ observations, we minimize the following loss function:

$$\mathcal{L} = \underbrace{\sum_{s=t-\tau+1}^{t} D_{\mathrm{KL}}(Q(\tilde{\boldsymbol{x}}_s|\boldsymbol{o}_s)||P(\tilde{\boldsymbol{x}}_s))}_{\text{VAE loss for } h, f, \text{ and } g} + \underbrace{\sum_{s=t-\tau+1}^{t} \|\boldsymbol{o}_s - \hat{\boldsymbol{o}}_s\|_2^2}_{\text{Obs. recons. loss for } h, f, \text{ and } g} + \underbrace{\sum_{s=t-\tau+1}^{t} \|\tilde{\boldsymbol{x}}_s - \hat{\boldsymbol{x}}_s\|_2^2}_{\text{State recons. loss for } f \text{ and } h} .$$

The variational autoencoder (VAE) [90] loss is a variational bound on the marginal log-likelihood of the data. It is used to train the encoder h, the estimator f, and the decoder g. Using VAEs avoids learning degenerated solutions and provides stable training of the network over a deterministic network In addition, the observation reconstruction loss encourages reconstructed observations $\{\hat{\boldsymbol{o}}_s\}$ to match true observations $\{\boldsymbol{o}_s\}$, and the state reconstruction loss constrains encoded states $\{\tilde{\boldsymbol{x}}_s\}$ to follow simulated states $\{\hat{\boldsymbol{x}}_s\}$ generated by physics. The former is used to train h, f, and g, and the latter is used to train f and h. Both observation and state reconstruction losses are important for training. We find that removing the state reconstruction term reduces the state prediction performance of the encoder since the network can predict arbitrary sequences without constraints. In addition, removing the observation reconstruction term impedes the image reconstruction quality, which is vital for training the whole model. In practice, we find that using the same weight for each loss term works well.

5.5 Fourier Feature Mappings

Prior work [144] has found that using Fourier features improves the generalization of deep neural networks. This is due to the fact that Fourier features are able to adjust

the spectral bias of neural networks into one that is easy to optimize [144]. For this reason, we want to incorporate Fourier features into the design of ALPS to take advantage of this observation. Moreover, we will show that it helps in this section. To lay the foundation for the justification of using Fourier features for predicting system parameters, in Section 5.5.1 we first review the recent work that uses NTK theory [77, 144]. This allows us to control the training of our parameter estimator network as fully-connected networks. Then in Section 5.5.2 we use these tools to analyze the effects of using Fourier features, their magnitudes, and their phases for predicting system parameters.

5.5.1 Deep Networks as a Kernel Regression

Consider a set of labelled training data $\{(\boldsymbol{v}_i, y_i)\}_i^m$ with $\boldsymbol{v}_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$, and $i \in [1 : m]$. Set $\boldsymbol{y} = [y_1, \ldots, y_m]^T \in \mathbb{R}^m$. Now bring in a feature map $\phi : \mathbb{R}^n \to \mathbb{R}^r$ with kernel $k(\boldsymbol{v}_i, \boldsymbol{v}_j) = \phi(\boldsymbol{v}_i)^T \phi(\boldsymbol{v}_j)$. Let $\boldsymbol{K} = [k(\boldsymbol{v}_i, \boldsymbol{v}_j)] \in \mathbb{R}^{m \times m}$ denote the kernel matrix for the training examples and $k(\boldsymbol{v}) = [k(\boldsymbol{v}_i, \boldsymbol{v})] \in \mathbb{R}^m$ denote the vector of kernel evaluations $k(\boldsymbol{v}_i, \boldsymbol{v}), i \in [1, m]$, for a test sample $\boldsymbol{v} \in \mathbb{R}^n$. The resulting kernel regression predictor is $\hat{y}(\boldsymbol{v}) = \boldsymbol{y}^T \boldsymbol{K}^{-1} k(\boldsymbol{v})$.

Now bring the concept of NTK proposed by [77]. The theory in [77] says that when the width of the layers of fully-connected deep networks with weights $\boldsymbol{w}$ initialized from a Gaussian distribution $\mathcal{N}$ tends to infinity, and the learning rate for stochastic gradient descent tends to zero, the neural network estimator $\hat{y}(\boldsymbol{v}; \boldsymbol{w})$ converges to the kernel regression solution using NTK. The NTK is defined as

$$k_{\text{NTK}}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \mathbb{E}_{\boldsymbol{w} \sim \mathcal{N}} \Big[\Big(\frac{\partial \hat{y}(\boldsymbol{v}_i; \boldsymbol{w})}{\partial \boldsymbol{w}} \Big)^T \Big(\frac{\partial \hat{y}(\boldsymbol{v}_j; \boldsymbol{w})}{\partial \boldsymbol{w}} \Big) \Big].$$

Under asymptotic conditions, a neural network's output after t updates can be ap-

proximated as

$$\hat{y}^{(t)}(\boldsymbol{v};\boldsymbol{w}) \approx \boldsymbol{y}^T(\boldsymbol{I} - e^{-\eta \boldsymbol{K} t})\boldsymbol{K}^{-1}k(\boldsymbol{v}),$$

where $e^{\boldsymbol{M}}$ is the n by n matrix $\boldsymbol{M}$ given by the power series $e^{\boldsymbol{M}} = \sum_{i=0}^{\infty} \frac{1}{i!}\boldsymbol{M}^i$ with $\boldsymbol{M}^0 = \boldsymbol{I}$.

Spectral Bias in Neural Networks. Now we want to compute the training error of a neural network after t times update. Let $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ denote the eigen-decomposition of the kernel matrix $\boldsymbol{K}$ which must be positive semidefinite (PSD). Here $\boldsymbol{U}$ is an orthogonal matrix and $\boldsymbol{\Sigma}$ is a diagonal matrix whose entries are the nonnegative eigenvalues ordered by magnitude: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$. So the training error in terms of L_2 norm $\|\cdot\|_2^2$ is

$$\begin{aligned}
\left\|\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}\right\|_2^2 &= \left\| \begin{bmatrix} \boldsymbol{y}^T(\boldsymbol{I} - e^{-\eta \boldsymbol{K} t})\boldsymbol{K}^{-1}k(\boldsymbol{v}_1) \\ \vdots \\ \boldsymbol{y}^T(\boldsymbol{I} - e^{-\eta \boldsymbol{K} t})\boldsymbol{K}^{-1}k(\boldsymbol{v}_m) \end{bmatrix}_{m\times 1} - \boldsymbol{y} \right\|_2^2 \\
&= \left\|\boldsymbol{K}\boldsymbol{K}^{-1}(\boldsymbol{I} - e^{-\eta \boldsymbol{K} t})\boldsymbol{y} - \boldsymbol{y}\right\|_2^2 \\
&= \left\|e^{-\eta \boldsymbol{K} t}\boldsymbol{y}\right\|_2^2 \\
&= \left\|\boldsymbol{U}e^{-\eta \boldsymbol{\Sigma} t}\boldsymbol{U}^T\boldsymbol{y}\right\|_2^2 \\
&= \left\|\boldsymbol{U}\mathrm{diag}([e^{-\eta\lambda_1 t}, \ldots, e^{-\eta\lambda_m t}]^T)\boldsymbol{U}^T\boldsymbol{y}\right\|_2^2. \qquad (5.2)
\end{aligned}$$

Eq. (5.2) shows the training convergence will decay exponentially at the rate $\eta\lambda_i$. Hence the components in $\boldsymbol{y}$ will be learned faster if their corresponding eigenvalue is larger. In Section 5.5.2 we will show that for a sequence of states without doing Fourier transform, the resulting NTK will have smaller eigenvalues at the high-frequency components. This leads to a slower convergence in high-frequency components of data which are essential to identify parameters.

5.5.2 The Effect of Fourier Feature Mapping

To understand the effect of the Fourier feature mapping, we first derive the kernel function of Fourier features, their magnitudes, and their phases. Then we compare the spatial bias of the kernel matrices from these three kernels. Finally, we show a one-dimensional example.

Kernels of Fourier Feature Mapping. Consider a state trajectory $\boldsymbol{v} = [x_0, x_1, \ldots, x_{\tau-1}]^T$, where here we consider a 1D case for a scalar state $x \in \mathbb{R}$ although it can be extended to a vector state $\boldsymbol{x} \in \mathbb{R}^n$. The feature maps of the Fourier feature mapping, their magnitudes, and their phases are

$$\textbf{(1)} \quad \phi_{\text{DFT}}(\boldsymbol{v}) = [X_0, \ldots, X_\omega, \ldots, X_{\tau-1}]^T \in \mathbb{R}^\tau;$$
$$\textbf{(2)} \quad \phi_{\text{MAG}}(\boldsymbol{v}) = [|X_0|, \ldots, |X_{\tau-1}|]^T \in \mathbb{R}^\tau;$$
$$\textbf{(3)} \quad \phi_{\text{PHA}}(\boldsymbol{v}) = [\arg(X_0), \ldots, \arg(X_{\tau-1})]^T \in \mathbb{R}^\tau,$$

where $X_\omega = \sum_{j=0}^{\tau-1} x_j \Big[\cos\left(\frac{2\pi}{\tau} kj\right) - i \sin\left(\frac{2\pi}{\tau} kj\right) \Big]$. Set $\boldsymbol{C}_k = \big[\cos\left(\frac{2\pi}{\tau} ki\right) - \sin\left(\frac{2\pi}{\tau} kj\right) \big] \in \mathbb{R}^{\tau \times \tau}$, then the kernel functions of these mappings are

$$\textbf{(1)} \quad k_{\text{DFT}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \sum_{k=0}^{\tau-1} \boldsymbol{v}_1^T \boldsymbol{C}_k \boldsymbol{v}_2;$$
$$\textbf{(2)} \quad k_{\text{MAG}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \sum_{k=0}^{\tau-1} \sqrt{\boldsymbol{v}_1^T \boldsymbol{C}_k \boldsymbol{v}_1 \boldsymbol{v}_2^T \boldsymbol{C}_k \boldsymbol{v}_2};$$
$$\textbf{(3)} \quad k_{\text{PHA}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \phi_{\text{PHA}}(\boldsymbol{v}_1)^T \phi_{\text{PHA}}(\boldsymbol{v}_2).$$

Proof. **(1)** $k_{\mathrm{DFT}}(\boldsymbol{v}_1, \boldsymbol{v}_2)$: Let $\boldsymbol{v}_1 = \begin{bmatrix} x_{1,1}, \\ \vdots, \\ x_{\tau-1,1} \end{bmatrix}$ and $\boldsymbol{v}_2 = \begin{bmatrix} x_{1,2}, \\ \vdots, \\ x_{\tau-1,2} \end{bmatrix}$. Then

$$
\begin{aligned}
&k_{\mathrm{DFT}}(\boldsymbol{v}_1, \boldsymbol{v}_2) \\
&\quad = \phi_{\mathrm{DFT}}(\boldsymbol{v}_1)^T \phi_{\mathrm{DFT}}(\boldsymbol{v}_2) \\
&\quad = \left[\textstyle\sum_{j=0}^{\tau-1} x_{j,1} \cos\left(\frac{2\pi}{\tau} j \cdot k\right), \sum_{j=0}^{\tau-1} x_{j,1} \sin\left(\frac{2\pi}{\tau} j \cdot k\right) \right]_k \begin{bmatrix} \sum_{j=0}^{\tau-1} x_{j,2} \cos\left(\frac{2\pi}{\tau} j \cdot k\right) \\ \sum_{j=0}^{\tau-1} x_{j,2} \sin\left(\frac{2\pi}{\tau} j \cdot k\right) \end{bmatrix}_k \\
&\quad = \sum_{k=0}^{\tau-1} \Bigg(\boldsymbol{v}_1^T \begin{bmatrix} \cos\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right) \\ \vdots \\ \cos\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \end{bmatrix} \boldsymbol{v}_2^T \begin{bmatrix} \cos\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right) \\ \vdots \\ \cos\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \end{bmatrix} \\
&\qquad\qquad + \boldsymbol{v}_1^T \begin{bmatrix} \sin\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right) \\ \vdots \\ \sin\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \end{bmatrix} \boldsymbol{v}_2^T \begin{bmatrix} \sin\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right) \\ \vdots \\ \sin\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \end{bmatrix} \Bigg) \\
&\quad = \sum_{k=0}^{\tau-1} \boldsymbol{v}_1^T \Bigg(\begin{bmatrix} \cos\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right) \\ \vdots \\ \cos\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \end{bmatrix} \left[\cos\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right), \ldots, \cos\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \right] \\
&\qquad\qquad + \begin{bmatrix} \sin\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right) \\ \vdots \\ \sin\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \end{bmatrix} \left[\sin\left(\frac{2\pi}{\tau} \cdot 1 \cdot k\right), \ldots, \sin\left(\frac{2\pi}{\tau} \cdot (\tau-1) \cdot k\right) \right] \Bigg) \boldsymbol{v}_2 \\
&\quad = \sum_{k=0}^{\tau-1} \boldsymbol{v}_1^T \boldsymbol{C}_k \boldsymbol{v}_2,
\end{aligned}
$$

which completes the proof of $k_{\mathrm{DFT}}(\boldsymbol{v}_1, \boldsymbol{v}_2)$.

(2) $k_{\mathrm{MAG}}(\boldsymbol{v}_1, \boldsymbol{v}_2)$: Let $\boldsymbol{c}_k = \begin{bmatrix} \cos\left(\frac{2\pi}{\tau} \cdot k \cdot 0\right) \\ \vdots \\ \cos\left(\frac{2\pi}{\tau} \cdot k \cdot (\tau - 1)\right) \end{bmatrix}$ and $\boldsymbol{s}_k = \begin{bmatrix} \sin\left(\frac{2\pi}{\tau} \cdot k \cdot 0\right) \\ \vdots \\ \sin\left(\frac{2\pi}{\tau} \cdot k \cdot (\tau - 1)\right) \end{bmatrix}$.

Then

$$
\begin{aligned}
&k_{\mathrm{MAG}}(\boldsymbol{v}_1, \boldsymbol{v}_2) \\
&\quad = \phi_{\mathrm{MAG}}(\boldsymbol{v}_1)^T \phi_{\mathrm{MAG}}(\boldsymbol{v}_2) \\
&\quad = \left[\sqrt{\left(\sum_{j=0}^{\tau-1} x_{j,1} \cos\left(\tfrac{2\pi}{\tau} j \cdot k\right)\right)^2 + \left(\sum_{j=0}^{\tau-1} x_{j,1} \sin\left(\tfrac{2\pi}{\tau} j \cdot k\right)\right)^2} \right]_k^T \\
&\qquad\qquad \left[\sqrt{\left(\sum_{j=0}^{\tau-1} x_{j,2} \cos\left(\tfrac{2\pi}{\tau} j \cdot k\right)\right)^2 + \left(\sum_{j=0}^{\tau-1} x_{j,2} \sin\left(\tfrac{2\pi}{\tau} j \cdot k\right)\right)^2} \right]_k \\
&\quad = \sum_{k=0}^{\tau-1} \left(\sqrt{\left(\boldsymbol{v}_1^T \boldsymbol{c}_k\right)^2 + \left(\boldsymbol{v}_1^T \boldsymbol{s}_k\right)^2} \sqrt{\left(\boldsymbol{v}_2^T \boldsymbol{c}_k\right)^2 + \left(\boldsymbol{v}_2^T \boldsymbol{s}_k\right)^2} \right) \\
&\quad = \sum_{k=0}^{\tau-1} \sqrt{\left(\boldsymbol{v}_1^T \boldsymbol{s}_k \boldsymbol{c}_k^T \boldsymbol{v}_2\right)^2 + \left(\boldsymbol{v}_1^T \boldsymbol{c}_k \boldsymbol{s}_k^T \boldsymbol{v}_2\right)^2 + \left(\boldsymbol{v}_1^T \boldsymbol{c}_k \boldsymbol{c}_k^T \boldsymbol{v}_2\right)^2 + \left(\boldsymbol{v}_1^T \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{s}_2\right)^2}. \qquad (5.3)
\end{aligned}
$$

Set $a_k = \boldsymbol{v}_1^T \boldsymbol{s}_k, b_k = \boldsymbol{v}_2^T \boldsymbol{c}_k, c_k = \boldsymbol{v}_1^T \boldsymbol{c}_k$, and $d_k = \boldsymbol{v}_2^T \boldsymbol{s}_k$. Then

$$
\begin{aligned}
(5.3) &= \sum_{k=0}^{\tau-1} \sqrt{\left(a_k b_k\right)^2 + \left(c_k d_k\right)^2 + \left(b_k c_k\right)^2 + \left(a_k d_k\right)^2} \\
&= \sum_{k=0}^{\tau-1} \sqrt{(a_k^2 + c_k^2)(b_k^2 + d_k^2)} \\
&= \sum_{k=0}^{\tau-1} \sqrt{\left(\boldsymbol{v}_1^T \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{v}_1 + \boldsymbol{v}_1^T \boldsymbol{c}_k \boldsymbol{c}_k^T \boldsymbol{v}_1\right)^2 + \left(\boldsymbol{v}_2^T \boldsymbol{c}_k \boldsymbol{c}_k^T \boldsymbol{v}_2 + \boldsymbol{v}_2^T \boldsymbol{s}_k \boldsymbol{s}_k^T \boldsymbol{v}_2\right)^2} \\
&= \sum_{k=0}^{\tau-1} \sqrt{\left(\boldsymbol{v}_1^T (\boldsymbol{s}_k \boldsymbol{s}_k^T + \boldsymbol{c}_k \boldsymbol{c}_k^T) \boldsymbol{v}_1\right)^2 + \left(\boldsymbol{v}_2^T (\boldsymbol{c}_k \boldsymbol{c}_k^T + \boldsymbol{s}_k \boldsymbol{s}_k^T) \boldsymbol{v}_2\right)^2} \\
&= \sum_{k=0}^{\tau-1} \sqrt{\boldsymbol{v}_1^T \boldsymbol{C}_k \boldsymbol{v}_1 \boldsymbol{v}_2^T \boldsymbol{C}_k \boldsymbol{v}_2},
\end{aligned}
$$

which completes the proof of $k_{\mathrm{MAG}}(\boldsymbol{v}_1, \boldsymbol{v}_2)$. □

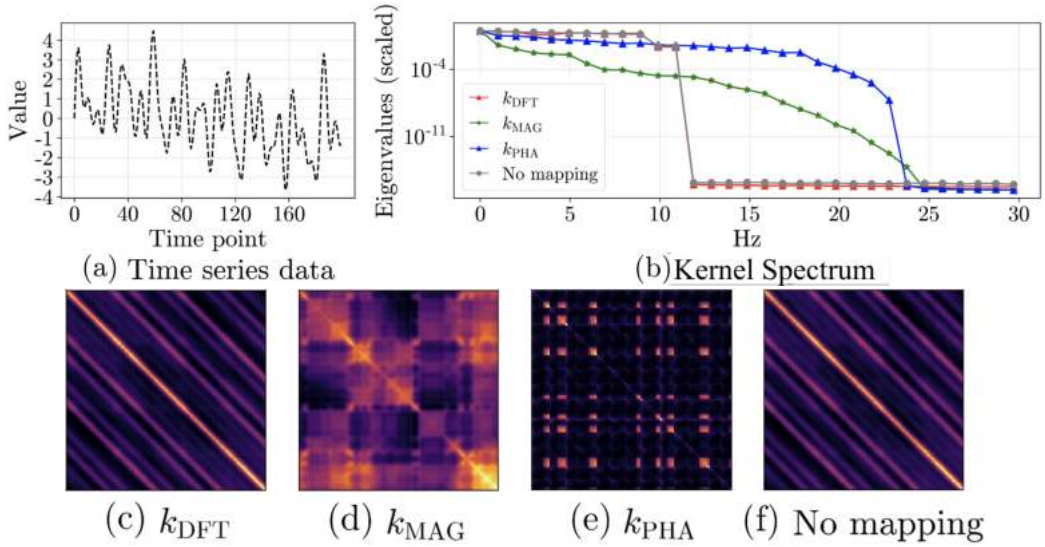


Figure 5.3: Using a Fourier feature mapping of k_{MAG} and k_{PHA} results in a wider spectrum, which lets neural networks learn a wide frequency content. Note that k_{DFT} and No mapping have the same kernel matrices due to the fact that k_{DFT} is a linear orthogonal transmutation of the original data. **(a)** The time series data with the sine basis frequency of $25, 17.5, 11, 7.7, 2, 1$Hz associated with the amplitude of $1, 1.2, 1, 1, 0.4, 1$. **(b)** The kernel spectrum of the kernel matrices. **(c-f)** The kernel matrices of the composed NTK. (best viewed in color)

After mapping the input points into the Fourier features, we feed them into a neural network to obtain $\hat{y}(\phi(\boldsymbol{v}); \boldsymbol{w})$. Hence for DFT kernel function the resulting composed kernel of the neural network is $k_{\text{NTK}}(\phi_{\text{DFT}}(\boldsymbol{v}_1), \phi_{\text{DFT}}(\boldsymbol{v}_2))$. Similarly, we have the kernels for k_{MAG} and k_{PHA}.

Visualizing the Composed NTK. We generate time series data composed of different frequencies and magnitude of sine waves. The length of the data is 200 samples, and we set $\tau = 100$ (*i.e.,* $\boldsymbol{v} \in \mathbb{R}^{100}$) with the sampling rate being 100Hz. We slide a window and hence have 101 instances in total. Fig. 5.3 shows the time series data, the effects of each kernel, and its spatial plot. By construction, k_{MAG} and k_{PHA} have a slower decay in the high-frequency domain. In addition, k_{DFT} and no mapping have the same kernel matrix and a narrower kernel spectrum. This is because that k_{DFT} is a linear orthogonal transmutation of the original data. This observation supports the idea of not using X_ω.

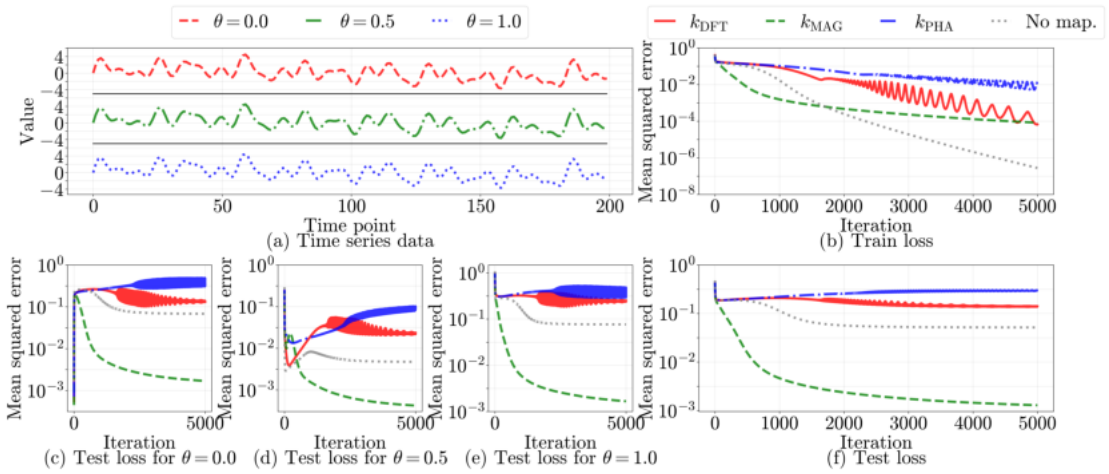


Figure 5.4: Mapping the time series data into the magnitude of the Fourier features (*i.e.,* k_{MAG}) achieves the lowest test error over the other mappings. The neural network is regressed to predict the target physical parameter ($\theta \in \mathbb{R}$) from the input features (k_{DFT}, k_{MAG}, k_{PHA}, and No mapping) with a mean squared error loss. **(a)** Three values of time series data. We generate the training and test data by combining the different weighted frequencies of sine waves, and assigning a target value to each of them. For each value, we slice the data and set $\tau = 100$. **(b)(f)** We report the mean squared error in the training and test dataset. Using the mapping of k_{MAG} achieves the lowest training and test error, preventing from being affected by the shift of the data while learning from high-frequency content. In contrast, the other mappings result in memorizing the training data (*i.e.,* overfitting). **(c)(d)(e)** We report the mean squared error in the test dataset for each class. We see that the mapping of k_{PHA} suffers from the shift of the data while that of k_{MAG} achieves better performance.

Using Fourier Feature Mappings to Predict Physical Parameters. One may think that using k_{PHA} also lets neural networks identify physical parameters. However, a shift of samples in the time domain lets magnitudes unchanged, but adds a linear term to phases for the observations from the same physical parameters. In Fig. 5.4, we train a feedforward network (4 layers, each layer has 1024 neurons, relu activations) to predict the physical parameters from time series data using the Fourier feature mappings. There are three true parameter values ($\theta = 0.0, 0.5, 1.0$) that are used to generate the time series data, *i.e.,* $\dot{\boldsymbol{x}} = \boldsymbol{A}(\theta)\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$. We excite the system with the input control $\boldsymbol{u}$ that contains high frequency components. To do well in this task, the model needs to distinguish the high-frequency component in the data. See more details in the caption. We see that using the mapping of k_{MAG} allows the neural network to robustly identify the physical parameters from the data, whereas the other mappings overfit the data without separating the high-frequency component for each class. The individual test error for each class of θ in Fig. 5.4(c)(d)(e) implies that the mapping of the phase fails to identify the physical parameter within the same class because of the shift of the data. In addition, this observation also implies that although k_{DFT} contains the same information as a joint combination of k_{MAG} and k_{PHA}, their resulting representations are different, leading to learning differently. Finally, due to time constraints, we do not test the result with features from both k_{MAG} and k_{PHA}. We expect two outcomes: neural networks may learn to disregard the feature from k_{PHA} and perform well, or neural networks may be distracted by the feature from k_{PHA} and perform poorly. We leave this a future work.

Comparison to [144]. Our work is inspired by [144], which also uses Fourier feature mappings. However, there are a few key differences. **(1) Setup.** The Fourier feature mapping in [144] transforms the low-dimensional x-y coordinates into high-dimensional Fourier features, which *projects* data into a high-dimensional space. In contrast, we use Fourier feature mapping of raw time series data, which *compresses*

data into more compact representations. **(2) Fourier features**. In [144], the basis and coefficients of Fourier features are hyper-parameters, which are tuned depending on the task. In contrast, in our work, the basis and coefficients of Fourier features are directly transformed from data, which are useful for predicting physical parameters. **(3) Analysis**. We discuss the difference between Fourier feature mappings, their magnitudes, and their phases to understand the effect of each mapping for predicting system parameters. In contrast, [144] does not have such analysis.

5.6 Experiments

We study the following four questions: **(1)** How does the basic form of ALPS (*i.e.,* without the encoder, parameter estimator, and decoder) perform? The result will provide useful insight to understand whether differential physics work or not. In addition, it allows us to examine the correctness of ALPS. **(2)** How does ALPS perform compared to other baseline methods without the Fourier feature mapping and physics-in-the-loop? **(3)** What is the effect of self-attention networks in ALPS? **(4)** How does ALPS perform in real-world time series data?

5.6.1 Setup

Visual Dataset. We generate three visual datasets: pendulum, mass-spring-damper (MSD), and a two-body system to compare the performance of ALPS to that of the baseline in the literature. We first randomly sample an initial state and physical parameters, and then generate a 125 step rollout following the true system dynamics, and render corresponding 64 by 64 by 3 pixel observation snapshots. The sampling rate is 20Hz in the pendulum, 100Hz in MSD, and 6Hz in two-body systems. The observation length τ is 100. In total, we generate 500 training and 500 test trajectories, resulting in 13,000 training and test sequences.

(1) Pendulum. The dynamics is $\frac{d}{dt}\begin{bmatrix}\varphi\\ \dot{\varphi}\end{bmatrix} = \begin{bmatrix}\dot{\varphi}\\ -\beta\dot{\varphi} - \frac{G}{L}\sin(\varphi)\end{bmatrix}$, where φ is an angle, $\dot{\varphi}$ is an angular velocity, β is a friction coefficient, G is a gravitational constant, and L is the length of the pendulum. We fix $G = 10$, $L = 1$, then sample β from a uniform distribution $\beta \sim \mathbb{U}(0.1, 1)$, an initial angle from a uniform distribution $\varphi_0 \sim \mathbb{U}(-\pi, +\pi)$, and an initial angular velocity from a uniform distribution $\dot{\varphi}_0 \sim \mathbb{U}(0.5, 4)$. We predict $\boldsymbol{\theta} = [\beta]$ in this task.

(2) Mass-spring-damper. The dynamics of the double mass-spring-damper system is $\frac{d}{dt}\begin{bmatrix}x(1)\\ x(2)\\ x(3)\\ x(4)\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ \frac{-\alpha(1)-\alpha(2)}{m(1)} & \frac{-\alpha(3)-\alpha(4)}{m(1)} & \frac{\alpha(1)}{m(1)} & \frac{\alpha(3)}{m(1)}\\ 0 & 0 & 0 & 1\\ \frac{\alpha(1)}{m(2)} & \frac{\alpha(3)}{m(2)} & -\frac{\alpha(1)}{m(2)} & \frac{\alpha(3)}{m(2)}\end{bmatrix}\begin{bmatrix}x(1)\\ x(2)\\ x(3)\\ x(4)\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{\alpha(2)}{m(1)} & \frac{\alpha(4)}{m(1)}\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}u(1)\\ u(2)\end{bmatrix}$, where $x(1), x(3)$ are the displacement and velocity of the primary spring; $x(2), x(4)$ are the displacement and velocity of the secondary spring; $u(1), u(2)$ are the inputs; $\alpha(2), \alpha(4)$ are the stiffness and damping ratio of the primary spring and damper; $\alpha(1), \alpha(3)$ are the stiffness and damping ratio of the secondary spring and damper, and $m(1), m(2)$ are the mass of the primary and secondary spring. We fix all the parameters and the initial state except for $\alpha(2)$, which is sampled from a uniform distribution $\alpha(2) \sim \mathbb{U}(4 \times 10^3, 4 \times 10^6)$. The input excitations are sampled from a unit Gaussian distribution $\boldsymbol{u} \sim \mathcal{N}(0, 0.05) \times \mathcal{N}(0, 0.05)$ (*i.e.,* white noise). We predict $\boldsymbol{\theta} = [\alpha(2)]$ in this task.

(3) Two-body. In this system, two particles interact with each other via an attractive force. The dynamics is a Hamiltonian $\mathcal{H} = \frac{\|p(1)\|_2^2}{2m(1)} + \frac{\|p(2)\|_2^2}{2m(2)} + \frac{Gm(1)m(2)}{\|q(1)-q(2)\|_2}$, where $p(1), p(2)$ are the positions of the particles; $q(1), q(2)$ are the momentum of the particles; $m(1), m(2)$ are the masses of the particles, and G is a gravitational constant. We fix $G = 10, m(1) = m(2) = 1$, then sample $\mathcal{H}$ from a uniform distribution $\mathcal{H} \sim \mathbb{U}(0.4, 1)$. We predict $\boldsymbol{\theta} = [\mathcal{H}]$ in this task.

Time Series Dataset. In addition, we obtain an MSD system dataset with state

measurements.

(4) Real MSD Time Series Data (Wheel Suspension System). To show the applicability of ALPS, we conduct *real-world* experiments with the data obtained from sensors installed in the train wheel suspension system by a team of engineers. The dynamics are the same as that of pixel MSD. We want to predict the stiffnesses of the primary and secondary springs $\alpha(2), \alpha(1)$ for predictive maintenance towards safe operation: $\boldsymbol{\theta} = [\alpha(2), \alpha(1)]^T$. The dataset contains 20 trajectories with 500 steps sampled by 100Hz with larger health values of $\alpha(1) = 1 \times 10^6, \alpha(2) = 4 \times 10^6$, and smaller faulty values of $\alpha(1) = 1 \times 10^3, \alpha(2) = 4 \times 10^3$. We use a 50-50 split to get the training and test datasets with $\tau = 100$. Here we *do not* use the self-attention and decoder network in ALPS–the estimator takes in state measurements directly.

(5) Full-scale MSD Time Series Data (Wheel Suspension System). To test the correctness of ALPS, we further extend the previous dataset (*i.e.,* dataset 4) to a full-scale wheel suspension dynamics with 18 state elements, 12 observations, 16 input excitations, and 24 system parameters. We want to identify the stiffness and damping ratio of the 12 spring-dampers. The dynamics are complex due to the interactions of multiple springs and dampers. This system consists of the primary level (*i.e.,* closed to the track) and secondary level (*i.e.,* closed to the car body) of the spring-damper. The states here are the displacement and the velocity of the spring-damper and the observations are the acceleration of the spring-damper. In the experimental section, we will use this dataset to test the performance of the vanilla version of ALPS.

Baselines. We consider the following baselines.

(1) Context-aware Dynamics Model (CDM) [98]. CDM is the state-of-the-art method to learn the system dynamics into two stages: it first learns a context vector that captures the local dynamics, and then predicts the next state based on the context vector and the current state and input. CDM *does not* consider physics

prior *nor* Fourier feature mapping. This is to show that using physics improves performance and benchmarks the results.

(2) Autoencoder. We remove the estimator and the physics simulator to benchmark the mismatch between the true state and the latent representation (equal dimension) of the autoencoder in Fig. 5.1(a) to show the interpretability of ALPS.

(3) ALPS w/o the Fourier Feature Mapping. We consider a variant by replacing the Fourier feature mapping with raw encoded state trajectories $\{\tilde{\boldsymbol{x}}_s\}$. This method is *similar* to [82], which also uses time series data to learn system dynamics.

(4) ALPS w/o Self-attention Networks. We consider a variant by replacing the self-attention network with a simple MLP to predict the position of the system, followed by a first-order finite-difference estimator to estimate the velocity. This uses the same approach as in [186] to estimate states. Finally, we ensure that all baselines are comparable in terms of representation power and the number of parameters.

Evaluation Metrics. We use a mean squared error (MSE) between simulated $\{\hat{\boldsymbol{x}}_s\}$ and true states $\{\boldsymbol{x}_s\}$ to evaluate the performance: $\text{SE} := \frac{1}{N}\sum_{i=1}^{N}\sum_{s=t-\tau+1}^{t}\|\hat{\boldsymbol{x}}_s - \boldsymbol{x}_s\|_2^2$, where N is the number of test data. We also compute MSE between reconstructed $\{\hat{\boldsymbol{o}}_t\}$ and true observations $\{\boldsymbol{o}_t\}$: $\text{OE} := \frac{1}{N}\sum_{i=1}^{N}\sum_{s=t-\tau+1}^{t}\|\hat{\boldsymbol{o}}_s - \boldsymbol{o}_s\|_2^2$. Moreover, we compute the absolute value difference between estimated $\hat{\boldsymbol{\theta}}$ and true parameters $\boldsymbol{\theta}$: $\text{PE} := \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|_1$.

Training Setup. For all the tasks we train our model and the baselines using Adam [89] with a maximum of 10,000 epochs and a learning rate of 3×10^{-3}. The batch size is 32 (*i.e.,* 32 rollouts), and we use a gradient clipping of 1.0 in terms of 2-norm to stabilize training. For the physics simulator, we use the four-step Runge-Kutta (RK4) as the numerical integration scheme in neural ODE. For computational resources, we train our model and the baselines in two machines: machine A has an Intel i9-9980HK CPU, and machine B has an Intel i7-6850K CPU with 4 NVIDIA GTX 1080 GPUs. In general, it takes about 8 hours to finish training. Fig. 5.5 plots

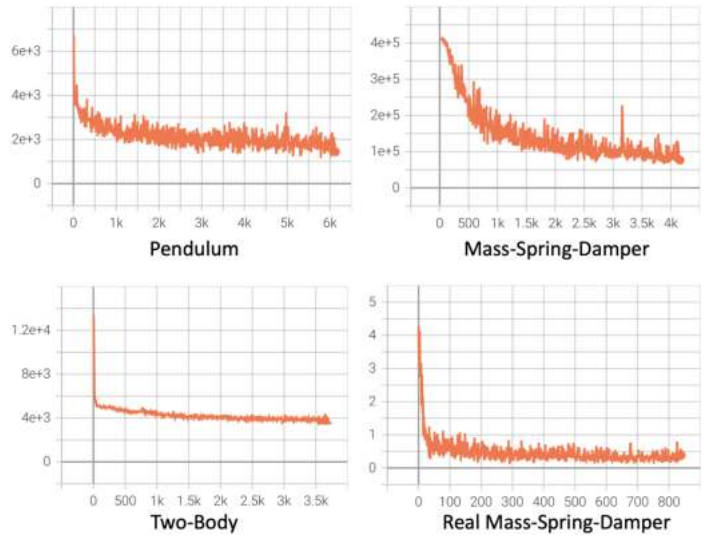


Figure 5.5: Learning curves during training of ALPS in the four tasks.

the training curves of the four tasks.

Implementation and Model Architectures. We use the same architecture over our model in the visual tasks. A grid search is conducted to find the hyper-parameters of the model (*i.e.,* size of the networks, types of the activation function). We now describe each component in ALPS.

The Encoder. To process pixel observations, the encoder uses a convolution neural network (CNN) with two convolutions and 2 by 2 max-poolings–the first convolution has a kernel size of 3 and a channel size of 3, and the second one has a kernel size of 4 and a channel size of 4, followed by vectorization to get a vector representation of an image. To provide the position information to this image representation, we obtain a positional embedding with the same size as the image representation using sine and cosine functions. We then concatenate these two vectors and feed them into a self-attention network, where the network has 10 attention heads and a relu activation function. Finally, the network outputs state estimations along the time horizon. Specifically, we only use the encoder part of the Transformer [158] without a dropout operation. We find that removing a dropout operation and using a relu activation function achieve the best result.

The Estimator. For each element of the state predicted from the encoder, we construct a Fourier feature mapping. For example, the state of the pendulum consists of an angle and an angular velocity–we will have two vectors of the Fourier feature mapping here. Then, for each Fourier feature mapping, we use a residual network [67] with two residual blocks and a tanh activation function to get a representation. Finally, we concatenate these representations and feed them into an MLP to predict a system parameter. We find that using residual networks and a tanh activation function makes training faster and converges to a good result.

The Simulator. The ODE solver takes in the predicted system parameter, and then rollouts state trajectories given the initial state predicted from the encoder.

The Decoder. We use a two-layer MLP with a relu activation function and the size of 400, 400 to take in the simulated states and reconstruct the observations.

For the real MSD task, we remove the encoder and decoder in ALPS since we get noisy measurements of states in train wheel systems. For ALPS without Fourier features, we use the same architecture but the estimator becomes a single residual network that takes in an entire state trajectory in the time domain. For ALPS without self-attention networks, we use the same architecture but the encoder becomes a two-layer MLP with a relu activation that takes in image representations and predicts the position of an object. To estimate the velocity, we follow the procedure in [186], which uses a finite difference method.

Finally, for the baseline model CDM, we use the same model in [98], which consists of a context network (an MLP with the size of 400, 400), a forward network (an MLP with the size of 20, 20), and a backward network (an MLP with the size of 20, 20).

Note that CDM is trained with the loss function used in [98]; Autoencoder is trained with the VAE loss and observation reconstruction loss terms; the ablations of ALPS are trained with the same loss functions as the full model.

In the following sections, to verify the correctness of ALPS and guide the reader

to understand the basic structure of ALPS, we first show the results of the vanilla version of ALPS (*i.e.,* without the encoder, parameter estimator, and decoder). Then, we show the results of the full version of ALPS.

5.6.2 Experiment Results for the vanilla version of ALPS

In this section, to guide readers to understand the basics of ALPS, we first introduce the vanilla version of ALPS (*i.e.,* without the encoder, estimator, and decoder), followed by elaborating on training details. Finally, we conclude this section by showing the results of fully and partially observable cases.

Vanilla version of ALPS. In the vanilla version of ALPS, we remove the encoder, estimator, and decoder. This ablation allows us to investigate whether ALPS can identify system parameters from data using the physics simulator without learning the encoder, estimator, and decoder. In addition, since we remove these three components, we do not have a notion of training and test datasets–ALPS now only needs to identify one set of system parameters from a single observation trajectory (*i.e.,* the dataset only contains a trajectory from a single set of system parameters). This setup is the same as the prior works [59, 183], which identify one set of system parameters one at a time. We now begin to elaborate on the experimental details.

Experimental Details.

(1) Setup. We consider two task setups in this section: fully and partially observable cases to gradually increase the difficulty of the tasks. Fig. 5.6 shows the vanilla version of ALPS in the fully and partially observable cases. In the first case, ALPS is provided with a sequence of true state trajectories $\{\boldsymbol{x}_s\}$ and the input excitations $\{\boldsymbol{u}_s\}$. The goal is to identify the system parameters $\boldsymbol{\theta}$ from this data. Next, in the second case, ALPS is provided with a sequence of observation trajectories $\{\boldsymbol{o}_s\}$ and the input excitations $\{\boldsymbol{u}_s\}$. In addition, we assume that the analytical form of the decoder is given (*i.e.,* $\boldsymbol{o} = g(\boldsymbol{x})$ is known), but the initial state $\boldsymbol{x}_0$ is not given. This

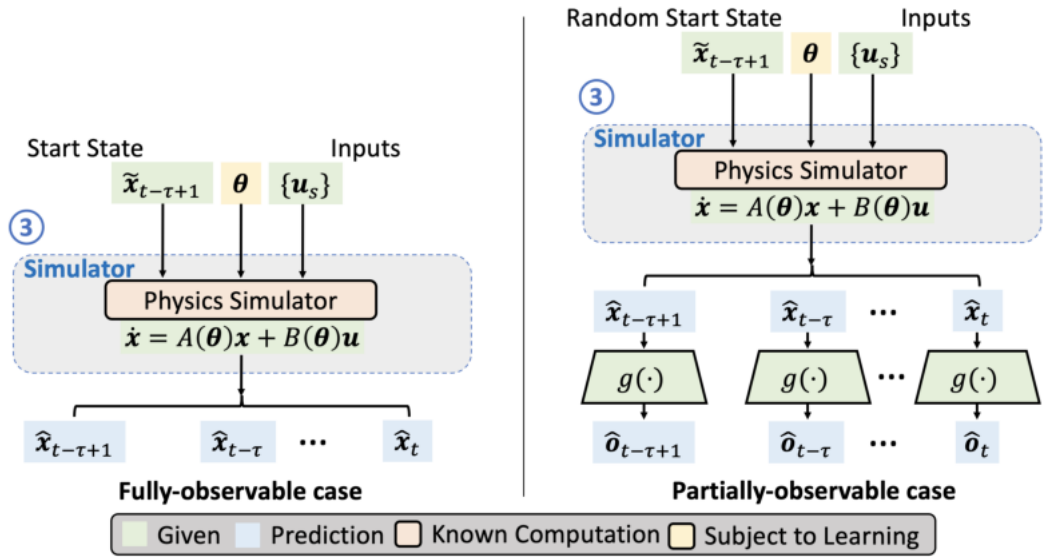


Figure 5.6: The vanilla version of ALPS in the fully and partially observable cases. In the fully-observable case, we are given the state sequence, input excitations, and system dynamics. We update the system parameters $\boldsymbol{\theta}$ based on the loss signal between the true states and the predicted states. In addition, in the partially-observable case, we are given the observation sequence, input excitations, the system dynamics, and the function mapping from states to observations. Since we do not know the initial state and the degradation effect of the initial state, we choose the random initial state. We update the system parameters $\boldsymbol{\theta}$ based on the loss signal between the true observations and the predicted observations.

case is aligned with many real-world applications where system designers cannot fully observe the state of the system. However, without the input of $\boldsymbol{x}_0$, we cannot run the physics simulator. Fortunately, since the train wheel system is a linear system and the system matrix $\boldsymbol{A}$ has non-positive eigenvalues in the most configurations, the effect of the initial states would die out exponentially fast. To see why this is true, mathematically, given the system equations

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

$$\boldsymbol{o} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}$$

, where $\boldsymbol{C}$ and $\boldsymbol{D}$ are the system matrices. The solution is given by the convolution equation[1]:

$$\boldsymbol{o}(t) = \boldsymbol{C}e^{\boldsymbol{A}t}\boldsymbol{x}(0) + \int_0^t \boldsymbol{C}e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau + \boldsymbol{D}\boldsymbol{u}(t).$$

We observe that the initial state only appears in the first part of the equation. In addition, when the eigenvalues of $\boldsymbol{A}$ are non-positive, $\boldsymbol{o}(t)$ would behave without the effect of $\boldsymbol{x}(0)$ as time goes long. This observation allows us to remove the effect of the initial states and update the estimation of the system parameters correctly. In practice, we initialize the initial states with all zeros as shown in Fig. 5.6 and let the simulation run additional steps.

(2) Loss Signals. For both fully and partially observable cases, we only consider the states and observations from the secondary level of the spring in the full-scale MSD time series data when computing the loss function. This is because the secondary level signals have more characteristics and the primary level signals contain the signals that are mostly from the input excitations, not from the springs and dampers themselves. In addition, empirically we do find that exclusion of the primary level sensors substantially improves the prediction quality. Specifically, for the

[1]We use a continuous version of notations here. For example, $\boldsymbol{x}(t)$ is the state at time t

vanilla version of ALPS in the fully observable case, the loss function is

$$\mathcal{L} = \sum_{s=t-\tau+1}^{t} \|\boldsymbol{s}_s - \hat{\boldsymbol{s}}_s\|_2^2.$$

In addition, for the partially observable case, the loss function is

$$\mathcal{L} = \sum_{s=t-\tau+1}^{t} \|\boldsymbol{o}_s - \hat{\boldsymbol{o}}_s\|_2^2,$$

where τ here is chosen carefully to avoid the effect of the initial state.

(3) Gradient Descent for the Parameters. Since the vanilla version of ALPS does not have the estimator, we conduct the gradient descent directly on the parameters

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \alpha\mathcal{L}(\boldsymbol{\theta}).$$

Result of the Fully Observable Case. Figure 5.7 shows the result of identifying 24 system parameters and the learning curve in the fully observable case. We can see that ALPS is able to accurately identify the system parameters with an average of 0.42% error rate.

Result of the Partially Observable Case. Figure 5.8 shows the result of identifying 24 system parameters and the learning curve in the partially observable case. We can see that ALPS is able to accurately identify the system parameters. Noticeably, we can see that when the system parameter dPS12 has a low value, ALPS is able to assign dPS12 to a small value. In the real-world application, when the value of the system parameter is lower than 20% of its normal value (*e.g.,* the normal value for dPS12 is 10000), the system component is due to replace. Hence, as long as the estimation value is below 20% of its normal value, this information is enough to notify the replacement of the component. In addition, this shows that the

	Full-scale MSD Time Series Data		
	True $\boldsymbol{\theta}$	Pred. $\hat{\boldsymbol{\theta}}$	$\lvert\frac{\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}}{\boldsymbol{\theta}}\rvert \times 100$ (%)
cPS11	900000	906036	0.67
cPS12	900000	899450	0.06
cPS21	900000	904543	0.50
cPS22	900000	895553	0.49
cPS31	900000	899453	0.06
cPS32	900000	901255	0.13
cPS41	900000	898952	0.11
cPS42	900000	901981	0.22
cSS11	500000	496639	0.67
cSS12	500000	503787	0.75
cSS21	500000	506055	1.21
cSS22	500000	491697	1.66
dPS11	10000	9947	0.53
dPS12	10000	10023	0.23
dPS21	10000	10009	0.09
dPS22	10000	9982	0.18
dPS31	10000	9980	0.20
dPS32	10000	10051	0.51
dPS41	10000	10038	0.38
dPS42	10000	9900	1.00
dSS11	21700	21664	0.16
dSS12	21700	21737	0.17
dSS21	21700	21736	0.16
dSS22	21700	21713	0.05

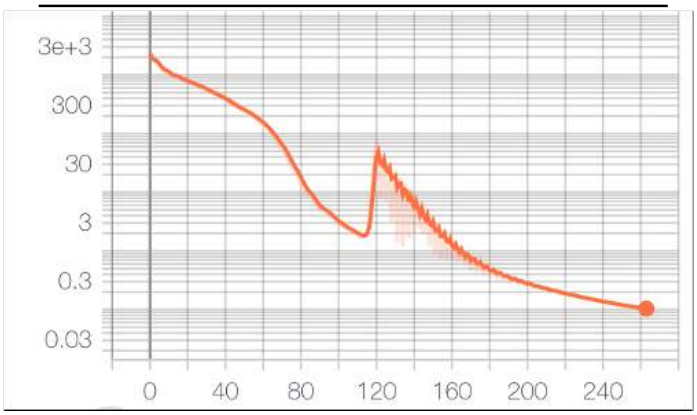


Figure 5.7: The prediction result and the learning curve in the fully observable case. cPS: the stiffness of the spring in the primary level (closed to the track); cSS: the stiffness of the spring in the secondary level (closed to the car); dPS: the damping coefficient of the damper in the primary level; dSS: the damping coefficient of the damper in the secondary level. The number after the name of the component indicates its position in the system. We see that ALPS can accurately identify the parameters.

	Full-scale MSD Time Series Data		
	True $\boldsymbol{\theta}$	Pred. $\hat{\boldsymbol{\theta}}$	$\lvert\frac{\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}}{\boldsymbol{\theta}}\rvert \times 100$ (%)
cPS11	900000	1015200	12.8
cPS12	900000	899710	0.03
cPS21	900000	857570	4.71
cPS22	900000	921980	2.44
cPS31	900000	935060	3.89
cPS32	900000	927620	3.06
cPS41	900000	910170	1.13
cPS42	900000	913290	1.47
cSS11	500000	670540	34.10
cSS12	500000	655230	31.04
cSS21	500000	676670	35.33
cSS22	500000	695170	39.03
dPS11	10000	8308	16.92
dPS12	0	2042	–
dPS21	10000	8163	18.37
dPS22	10000	10322	3.22
dPS31	10000	9639	3.61
dPS32	10000	9855	1.45
dPS41	10000	9220	7.80
dPS42	10000	9409	5.91
dSS11	21700	29357	35.28
dSS12	21700	26506	22.14
dSS21	21700	24480	12.81
dSS22	21700	23218	6.99

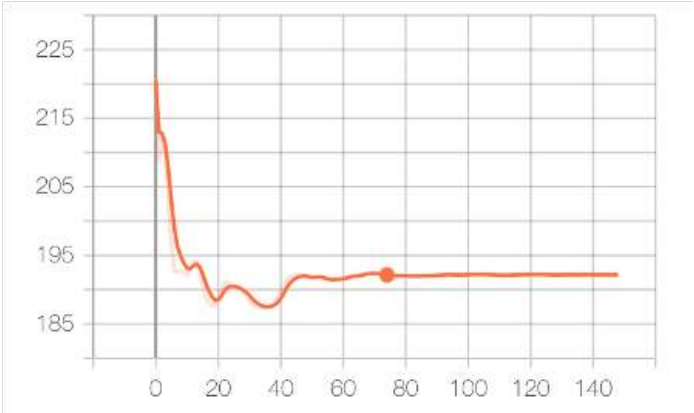


Figure 5.8: The prediction result and the learning curve in the partially observable case. cPS: the stiffness of the spring in the primary level (closed to the track); cSS: the stiffness of the spring in the secondary level (closed to the car); dPS: the damping coefficient of the damper in the primary level; dSS: the damping coefficient of the damper in the secondary level. The number after the name of the component indicates its position in the system. We see that ALPS can accurately identify the parameters.

	Pendulum			Mass-Spring-Damper			Two-body		
	SE	OE	PE	SE	OE	PE	SE	OE	PE
CDM [98]	345.22	1766.70	–	0.99	$\mathbf{5.69\times10^8}$	–	50.23	$\mathbf{2.03\times10^8}$	–
Autoencoder	3041.12	**600.84**	–	7.63	7.42×10^8	–	95.80	2.71×10^8	–
ALPS (ours)	**86.48**	1696.91	**0.06**	**0.29**	7.43×10^8	$\mathbf{0.60\times10^6}$	**0.45**	2.59×10^8	**0.02**
w/o Fourier feat. [82]	90.82	1773.39	0.29	0.93	7.44×10^8	2.28×10^6	1.86	2.56×10^8	**0.02**
w/o self-attention [186]	181.22	1950.77	**0.06**	1.85	7.44×10^8	1.87×10^6	511.49	2.72×10^8	0.27

Table 5.1: Evaluation of the tested networks in the visual tasks. SE: state prediction error; OE: observation prediction error; PE: parameter prediction error. ALPS achieves competitive performance in predicting physical parameters and states.

approach of isolating the effect of the initial state is effective. However, compared to the fully observable case, we can still see that the error rate is higher for the partially observable case. The performance can be improved by running the system longer to truly minimize the effect of the initial state or simultaneously estimating the initial states. We leave this as a future work

In this section, we have shown that ALPS can work in the vanilla setting. We now switch our focus to the full model of ALPS.

5.6.3 Experiment Results for ALPS

We now test the performance of the full models of ALPS (*e.g.,* with the encoder, estimation, and decoder). Note that we use a self-attention network for the encoder due to the fact that it is easy to optimize and can capture long-term dependency from the data.

Results in the Visual Tasks. Table 5.1 shows the results. We see that **(1)** ALPS achieves the best performance in predicting physical parameters in all cases, with at the most 4.8x lower error in the pendulum task. On the other hand, the Fourier feature mapping does not have much effect in predicting the physical parameter in the two-body system.

This verifies the analysis of the Fourier feature mapping: for the task with a wider frequency spectrum (*e.g.,* 100Hz in the MSD), the Fourier feature improves

	SE	PE
CDM [98]	56.04	–
ALPS (ours)	**3.02**	**0.44×10^6**
w/o Fourier feat. [82]	113.07	4.32×10^6

Table 5.2: Results in the MSD system for predicting two physical parameters from time series data.

the prediction due to higher eigenvalues in the high-frequency content, whereas for the task with only a low-frequency spectrum (*e.g.,* 6Hz in the two-body), raw state sequences can already capture low-frequency content. This supports the idea of using Fourier feature mappings for high-frequency data. **(2)** ALPS achieves competitive results in predicting the states, with at the most 6.3x lower error in the MSD task. Without self-attention networks, the network has a substantial SE due to a large error in computing the velocity, as in the two-body task. The smaller the sampling rate is, the greater the velocity estimation error is. In addition, the high SE in autoencoder suggests that its latent representation is uninterpretable. This verifies the idea of using physics to constrain the latent representation of the autoencoder to estimate states. **(3)** ALPS achieves comtextbfle results in reconstructing observations.

The low error for CDM in the MSD and two-body tasks is due to degenerated solutions, in which CDM reconstructs blur images. This also implies that using physics stabilizes the training and improves the reconstruction of observations. **(4)** Finally, the autoencoder baseline has higher reconstruction loss in some tasks. This is because that it learns a degenerated solution, producing blur images due to high-frequency movements of objects on the scene.

Our remaining simulations explore ALPS's ability in predicting physical parameters from raw state measurements.

Results in Real MSD Time Series Data. Table 5.2 shows the results. Here ALPS does not use the encoder or decoder network–the estimator takes in state measurements directly, as the system is fully-observable. Overall we see that **(1)** ALPS

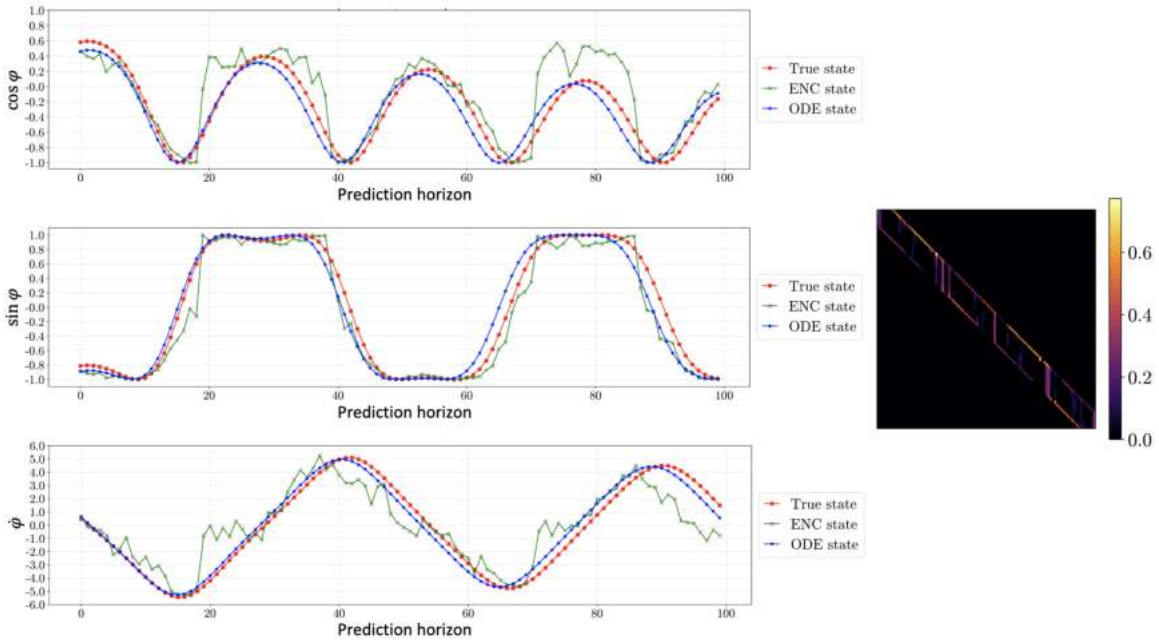


Figure 5.9: The state trajectories and the attention maps of the self-attention network in the pendulum task. We see that the self-attention network is able to infer states based on observations.

achieves the best performance. Without using the Fourier feature mapping, the network cannot learn to identify physical parameters. **(2)** CDM has worse SE since an MLP cannot learn well from data with high-frequency content. This also implies that by providing physics to the network, we can improve SE. These observations show ALPS can robustly identify two parameters simultaneously from state measurements, and support the theory of using Fourier features to learn dynamics. Moreover, this result shows ALPS can be deployed in real prognostic applications for tracking physical parameters to enhance railroad safety.

5.6.4 Additional Analysis

Visualization of Encoded States Prediction and Attention Map of the Self-attention Network. An important feature of ALPS is that it uses self-attention networks to infer states from pixel observations. Fig. 5.9 shows the encoded states

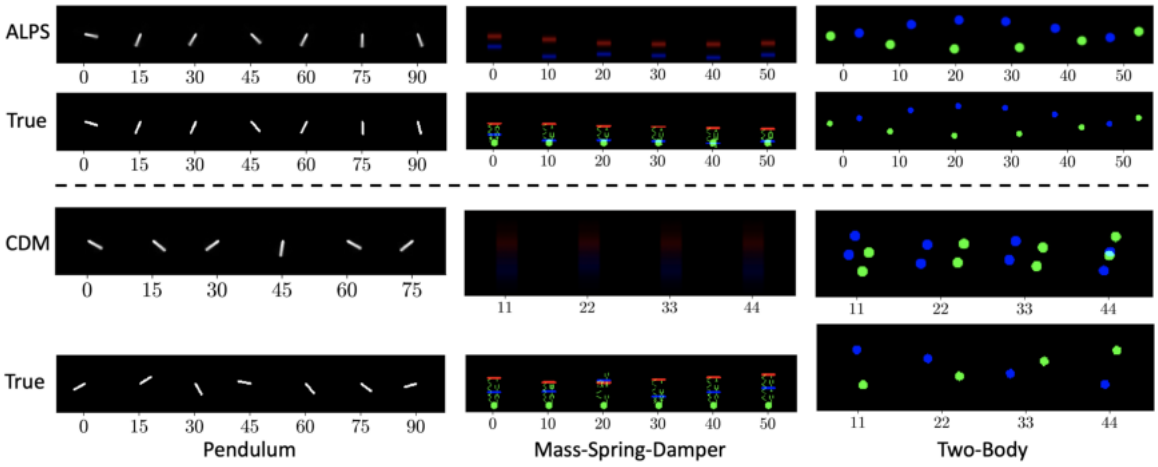


Figure 5.10: Examples of reconstructed and true observation sequences of ALPS and CDM over three visual tasks. We see that ALPS is able to robustly generate the observations.

$\{\tilde{\boldsymbol{x}}_s\}$, the ODE-simulated states $\{\hat{\boldsymbol{x}}_s\}$, and the true states $\{\boldsymbol{x}_s\}$ over time horizons τ as well as its attention map in the pendulum task. We see that the self-attention network is able to track the state evolution of the system from pixel observations. In addition, we observe that there is a small delay between the true states and the ODE-simulated states. This is due to a small prediction error in the initial state and the system parameter, which then accumulates over the time horizon.

For the attention map, the summation of each row is equal to one. This map shows that for each time step how much information of each observation in the entire trajectory is needed to predict the state at that time step. Here, we use a mask with the size of 15 to limit the attention width of the network. We see that the network mostly uses the observation at the beginning and the end of the attention mask to infer states. This suggests that the network learns to predict an *average* velocity. In addition, we see that there are vertical attention patterns, which happen to be at the peak of the cosine wave. We suspect that the network uses this information as an anchor to capture the periodic behavior of the pendulum. We leave this as future work to understand the network.

Visualization of Reconstructed Observations and Prediction of System

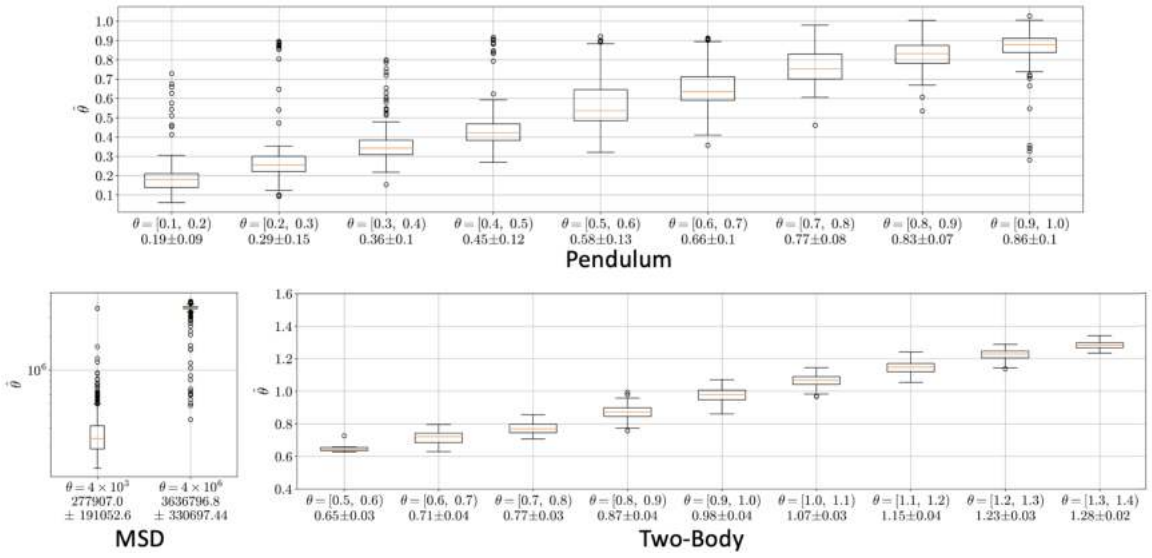
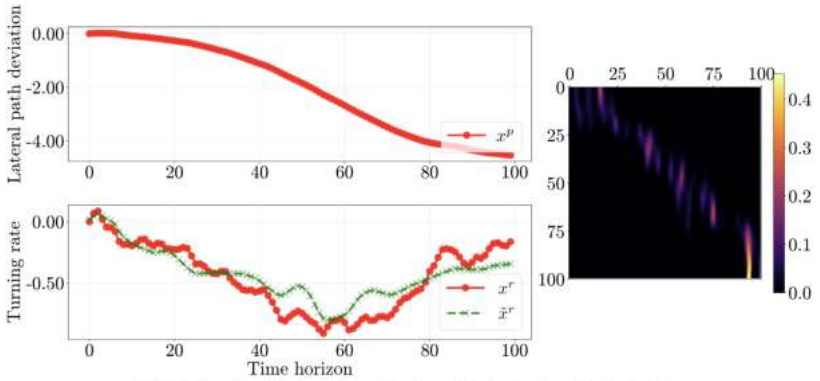


Figure 5.11: System parameter prediction performance of ALPS in the visual tasks. We categorize the value of the true system parameters, and show the box plot of the corresponding prediction values. The values at the bottom show the prediction mean and standard deviation of each group. We see that ALPS can identify the system parameters well.

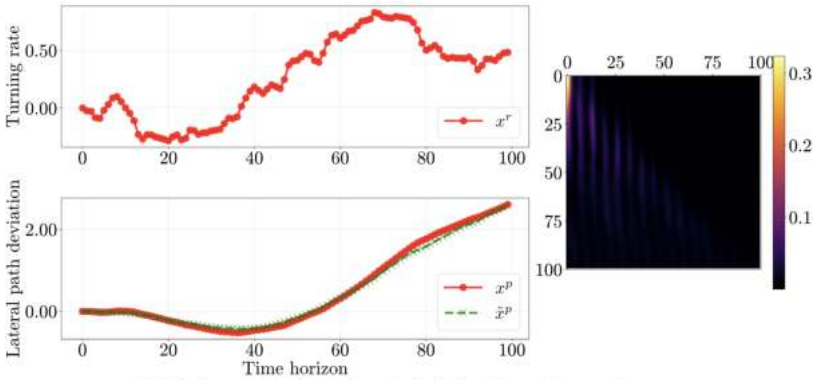
Parameters. ALPS uses an autoencoder to learn system dynamics. Fig. 5.10 shows reconstructed observations of ALPS and the baseline in three visual tasks. We see that ALPS is able to accurately reconstruct the observations, whereas CDM fails to predict observations with duplicated particles in the Two-Body system and blur images in the MSD system. This implies that using physics in the loop can help to converge to a good solution.

In addition, Fig. 5.11 shows the system parameter prediction performance of ALPS over three visual tasks. Please see the caption for more details about the figure. We see that the model is able to reliably predict the system parameters in most cases. However, we also see there are some outliers for each group. We leave this as a future improvement.

Analysis of self-attention networks An important feature of ALPS is the use of self-attention networks for estimating states from observations. To provide intuition about the mechanism of the self-attention network, we regress the self-



(a) Estimates the turning rate from the lateral path deviation



(b) Estimates the lateral path deviation from the turning rate

Figure 5.12: The patterns of the attention weight in the self-attention networks show the network attends to the local context (diagonal) to compute the gradient, and use the global context (lower triangular) to obtain the integral. **(a)** The case where the network estimates the turning rate x^r from the lateral path deviation x^p and its attention weight. **(b)** The case where the network estimates the lateral path deviation x^p from the turning rate x^r and its attention weight. The green lines $\tilde{x}^r$ and $\tilde{x}^p$ are the predictions and the red lines are the true values (turn the screen brighter to see the patterns).

attention network with true states of the system and visualize its attention weights on simple vehicle steering dynamics:

$$\frac{d\boldsymbol{x}}{dt} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0.1 \\ 1 \end{bmatrix} u,$$

where the first element in $\boldsymbol{x} = [x^p, x^r]^T$ represents the lateral path deviation (*i.e.,* the distance between the vehicle and the center of the road), the second element represents the turning rate, and u is the input force to the steering wheel. In this example, we want to estimate either x^p or x^r by observing x^r or x^p, respectively. Intuitively, in the first case (estimate x^p from x^r), a self-attention network should perform *derivative* operation, whereas in the second case (estimate x^r from x^p), a self-attention network should perform *integration* operation. Fig. 5.12 shows the result of the two cases. Here we use a single head self-attention network with a tanh activation function to regress the supervised data, and test on the testing data. In the first case, the network attends to the neighboring observations at the current time (*i.e.,* diagonal pattern), whereas in the second case the network attends to the observations from the beginning to the current time (*i.e.,* lower triangular pattern, global context). This observation suggests the network does exploit the local context to compute the gradient (*i.e.,* $x_t^r \approx \frac{x_{t+1}^p - x_t^p}{\Delta t}$), and use the global context to obtain the integral (*i.e.,* $x_t^p \approx \sum_{t'=0}^{t-1} x_{t'}^r$).

5.7 Discussion and Conclusion

In this chapter, we are motivated by the situation in Chapter 4 where some of the critical system parameters are unknown. We thus addressed the problem of predicting the states and physical parameters of a system from observations with dynamic equations. We showed that the latent representation of the autoencoder is uninter-

pretable. We then use dynamic equations to constrain the latent representation of the autoencoder to be consistent with the laws of physics. We analyzed the effect of Fourier features for estimating physical parameters. The first results showed that the basic form ALPS is able to identify the system parameters using gradient descent. The final results showed that ALPS achieves competitive performance in the visual tasks and the real raw time series dataset. This enables downstream applications.

No model is without limitations. Future work could improve ALPS in several ways. For instance, the understanding of underlying mechanisms in self-attention networks for estimating states from observations can be advanced. In addition, we require to have dynamic equations and assume that the system is linear and exhibits periodic or vibrational behaviors. These make it challenging to generalize to more complicated settings such as contact dynamics. One solution is to combine known physics with neural models (*e.g.,* interaction networks [17]) to compensate for modeling error, and use time-domain and frequency-domain features together to predict physical parameters. Moreover, it would also be interesting to explore other more complex applications that involve identifying physical parameters such as analysis of transient behaviors in electrical circuits. Last but not least, incorporating ALPS into safe reinforcement learning for conducting safety analysis (*e.g.,* similar to verifying the cost violation in the future in Chapter 4) would be an interesting direction for the safe RL problem in which the system dynamics are changing over time due to component deterioration or noise.

Acknowledgements. We thank Dr. Justinian Rosca for providing the funding and the helpful discussion, and Prof. Peter J. Ramadge and Prof. Karthik Narasimhan for the helpful discussion.

Chapter 6

Safe Reinforcement Learning with Natural Language Constraints

6.1 Introduction

In previous Chapters, we focus on the problem of safe reinforcement learning (Chapter 2), exploiting baseline policies (Chapter 3), applying the algorithm in real hardware (Chapter 4), and proposing unsupervised system identification approach (Chapter 5). While safe reinforcement learning (RL) holds great promise for many practical applications like robotics or autonomous cars, current approaches require specifying constraints in mathematical form. In addition, current autonomous agents do not have the capability to receive human feedback to correct their unsafe behavior after being programmed or designed in the factory. Such specifications demand domain expertise, limiting the adoption of safe RL. In addition, human learns to be safe by using verbal communication (*i.e.,* natural language) or body language to signal unsafe events. We expect a robot with such capability can greatly improve safety when deployed in the real world. In this chapter, we investigate the possibility of using natural language to specify the safety constraints for facilitating safe human-robot

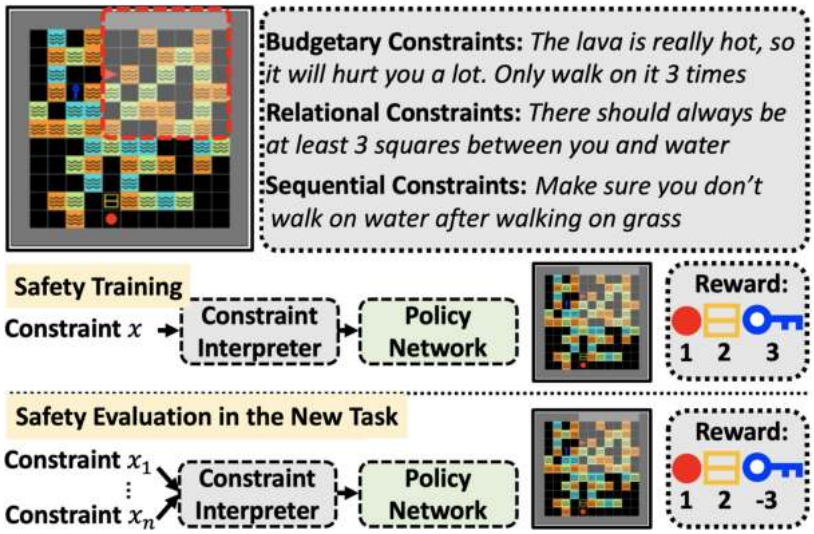


Figure 6.1: Learning to navigate with language constraints. The figure shows **(1)** a third-person view of the environment (red dotted square box), **(2)** three types of language constraints, **(3)** items which provide rewards when collected. During safety training, the agent learns to interpret textual constraints while learning the task (*i.e.,* collect rewards). During safety evaluation, the agent learns a new task with different rewards while following the constraints and minimizing violations.

interaction, which is important for the automated society in the future.

Although RL has shown promise in several simulated domains such as games [114, 137, 23] and autonomous navigation [11, 110], deploying RL in real-world scenarios remains challenging [47]. In particular, real-world RL requires ensuring the safety of the agent and its surroundings, which means accounting for *constraints* during training that are orthogonal to maximizing rewards. For example, a cleaning robot must be careful to not knock the television over, even if the television lies on the optimal path to cleaning the house.

Safe RL tackles these challenges with algorithms that maximize rewards while simultaneously minimizing constraint violations during exploration [6, 39, 173, 175, 5, 38, 18, 48, 156, 149]. However, these algorithms have two key limitations that prevent their widespread use. First, they require us to provide constraints in mathematical or logical forms, which calls for specific domain expertise. Second, a policy trained with a specific set of constraints cannot be transferred easily to learn new tasks with the

same set of constraints, since current approaches do not maintain an explicit notion of constraints separate from reward-maximizing policies. This means one would have to retrain the policy (with constraints) from scratch.

We consider the use of *natural language* to specify constraints (which are orthogonal to rewards) on learning. Human languages provide an intuitive and easily-accessible medium for describing constraints–not just for machine learning experts or system developers, but also for potential end users interacting with agents such as household robots. Consider the environment in Fig. 6.1 for example. Instead of expressing a constraint as $\sum_{t=0}^{T} \mathbf{1}_{s_t \in \text{lava}} \cdot \mathbf{1}_{\nexists s_{t'} \in \text{water},\ t' \in [0,1,\ldots,t-1]} = 0$, one could simply say "*Do not visit the lava before visiting the water*". Such an approach offers great flexibility for humans to specify the safety constraint for correcting robot behavior. In addition, it facilitates safe human-robot interaction. The challenge, of course, lies in training the RL agent to accurately interpret and adhere to the textual constraints as it learns a policy for the task.

To study this problem, we first create HazardWorld, a collection of grid-world and robotics environments for safe RL with textual constraints (Fig. 6.1). HazardWorld consists of separate '*safety training*' and '*safety evaluation*' sets, with disjoint sets of reward functions and textual constraints between training and evaluation. To do well on HazardWorld, an agent has to learn to interpret textual constraints during safety training and safely adhere to any provided constraints while picking up new tasks during the safety evaluation phase. Built on existing RL software frameworks [35, 127], HazardWorld consists of navigation and object collection tasks with diverse, crowdsourced, free-form text specifying three kinds of constraints: **(1)** *budgetary* constraints that limit the frequency of being in unsafe states, **(2)** *relational* constraints that specify unsafe states in relation to surrounding entities, and **(3)** *sequential* constraints that activate certain states to be unsafe based on past events (*e.g.,* "*Make sure you don't walk on water after walking on grass*"). Our setup differs

from instruction following [108, 28, 13, 111, 70, 66] in two ways. First, instructions specify what to do, while textual constraints only inform the agent on what *not to do*, independent of maximizing rewards. Second, learning textual constraints is a means for ensuring safe exploration while adapting to a new reward function.

In order to demonstrate learning in this setting, we develop ***P**olicy **O**ptimization with **L**anguage **CO**nstraints* (POLCO), where we disentangle the representation learning for textual constraints from policy learning. Our model first uses a *constraint interpreter* to encode language constraints into representations of forbidden states. Next, a *policy network* operates on these representations and state observations to produce actions. Factorizing the model in this manner allows the agent to retain its constraint comprehension capabilities while modifying its policy network to learn new tasks.

Experiments demonstrate that our approach achieves higher rewards (up to 11x) while maintaining lower constraint violations (up to 1.8x) compared to several baselines on two different domains within HazardWorld. Nevertheless, HazardWorld remains far from being solved, especially in tasks with high-dimensional observations, complex textual constraints, and those requiring high-level planning or memory-based systems.

Prior Publications. Parts of this thesis have been published in [174].

Code. `https://github.com/princeton-nlp/SRL-NLC`

6.2 Related Work

6.2.1 Instruction Following

Our work closely relates to the paradigm of instruction following in RL, which has previously been explored in several environments [108, 162, 28, 147, 13, 88, 12, 153, 106, 146, 165]. Prior work has also focused on creating realistic vision-language navigation

datasets [21, 31, 11, 42] and proposed computational models to learn multi-modal representations that fuse images with goal instructions [81, 22, 52, 105, 78, 54, 75, 53, 160]. Our work differs from the traditional instruction following setup in two ways: **(1)** Instruction following seeks to (roughly) 'translate' an instruction directly into an action policy. This does not apply to our setting since the textual constraints only tell an agent what *not to do*. To actually obtain rewards, the agent has to explore and figure out optimal policies on its own. **(2)** Since constraints are decoupled from rewards and policies, agents trained to understand certain constraints can transfer their understanding to respect these constraints in new tasks, even when the new optimal policy is drastically different. Therefore, we view this work as orthogonal to traditional instruction following–one could of course combine both instructions and textual constraints to simultaneously advise an agent on what to do and what not to do.

6.2.2 Connection to Seldonian Algorithms

POLCO can also be interpreted as a Seldonian algorithm [152]. Seldonian algorithms ensure ML safety through three steps: (1) defining a goal, (2) defining an interface for users to provide constraints, and (3) creating an algorithm that satisfies the goal and constraints. Here, we use natural language as the interface for end users and map natural language into optimization constants and vector representations. Thus, POLCO is also a potential step towards widely developing and deploying Seldonian algorithms.

6.2.3 Constraints in Natural Language

Our notion of 'constraints' in this paper differs from prior work that uses instructions to induce planning constraints [147, 74, 165]–these works again provide instructions for the agent on how to perform the task. Perhaps closest to this paper is the work of

Misra et al. [110], which proposes datasets to study spatial and temporal reasoning, containing a subset focusing on *trajectory constraints* (*e.g.,* "*go past the house by the right side of the apple*"). However, they do not disentangle the rewards from the constraints, which may be orthogonal to each other. Prakash et al. [121] train a constraint checker to identify whether a constraint (specified in text) has been violated in a trajectory. While their motivation is similar, they ultimately convert constraints to negative rewards, whereas we use a modular approach that allows disentangling reward maximization from minimizing constraint violations and is compatible with modern algorithms for safe RL.

6.3 Problem Setup

Problem Formulation. Our learning problem can be viewed as a partially observable constrained Markov decision process [10], which is defined by the tuple $< \mathcal{S}, \mathcal{O}, \mathcal{A}, T, Z, \mathcal{X}, R, C >$. Here $\mathcal{S}$ is the set of states, $\mathcal{O}$ is the set of observations, $\mathcal{A}$ is the set of actions, T is the conditional probability $T(s'|s,a)$ of the next state s' given the current state s and the action a, and Z is the conditional probability $Z(o|s)$ of the observation o given the state s. In addition, $\mathcal{X}$ is the set of textual constraint specifications, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which encodes the immediate reward provided when the agent takes an action a in state s, and $C : \mathcal{S} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ is the true underlying constraint function described by $x \in \mathcal{X}$, which specifies positive penalties for constraint violations due to an action a in a state s. Finally, we assume each $x \in \mathcal{X}$ corresponds to a specific cost function C.

RL with Constraints. The goal of the learning agent is to acquire a good control policy that maximizes rewards, while adhering to the specified constraints as much as possible during the learning process. Thus, the agent learns a policy $\pi : \mathcal{O} \times \mathcal{X} \to \mathcal{P}(\mathcal{A})$, which is a mapping from the observation space $\mathcal{O}$ and constraint

specification $\mathcal{X}$ to the distributions over actions $\mathcal{A}$. Let $\gamma \in (0, 1)$ denote a discount factor, $\mu(\mathcal{S})$ denote the initial state distribution, and τ denote a trajectory sequence of observations and actions induced by a policy π, *i.e.,* $\tau = (o_0, a_0, o_1, \cdots)$. For any given x, we seek a policy π that maximizes the cumulative discounted reward J_R while keeping the cumulative discounted cost J_C below a specified cost constraint threshold $h_C(x)$:

$$\max_{\pi} \quad J_R(\pi) \doteq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \quad \text{s.t.} \quad J_C(\pi) \doteq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t, x)\right] \leq h_C(x),$$

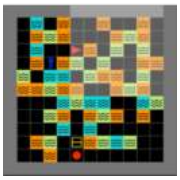
where $\tau \sim \pi$ is shorthand for indicating that the distribution over trajectories depends on $\pi : s_0 \sim \mu, o_t \sim Z(\cdot|s_t), a_t \sim \pi(\cdot|o_t, x), s_{t+1} \sim T(\cdot|s_t, a_t)$. We use $C(s_t, a_t, x)$ and $h_C(x)$ here to emphasize that both functions depend on the particular constraint specification x.

Task Setup. Our goal is to show that constraints specified in natural language allow for generalization to new tasks that require similar constraints during learning. With this in mind, we consider the following safety training and safety evaluation setup:

(1) Safety training: During training, we generate random environment layouts and starting states s_0 while keeping the reward function R fixed. For each episode, we randomly generate a constraint function C and limit h_C. We then sample a constraint text x that describes C and h_C from the training set of texts. The constraint text x is an input to the agent's policy. Whenever the agent violates a constraint (at any step), it is provided with a scalar cost penalty learned by the model from $C(s, a, x)$. The agent, therefore, sees a variety of different task layouts and constraints, and learns a policy with respect to the constraints for this task as well as how to interpret textual constraints.

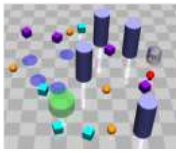
(2) Safety evaluation: During the evaluation, we place the agent in new envi-

Constraint	Examples
Budgetary	*Lava hurts a lot, but you have special shoes that you can use to walk on it, but only up to 5 times, remember!*
Relational	*Water will hurt you if you are two steps or less from them.*
Sequential	*Make sure you don't walk on water after walking on grass.*



(a) HazardWorld-grid

Constraint	Examples
Budgetary	*Do not enter the blue square. It is safe to never cross at all.*
Relational	*Three feet is the minimum distance to all the dark blue circles.*
Sequential	*Once a purple box gets touched dark blue circles are disallowed.*



(b) HazardWorld-robot

Table 6.1: Examples of textual constraints for HazardWorld-grid and HazardWorld-robot. **(a)** An agent (red triangle) seeks to collect the reward entity (ball, box, key) while avoiding the cost entity (lava, water, grass). **(b)** An agent (red point) aims to reach a goal position (green area) while avoiding the obstacles (vases, pillars, cubes, *etc.*). Please see the supplementary material for more details.

ronments with randomly generated layouts, with a different reward function R'. The set of possible constraints C is the same as seen in training, but the corresponding constraint texts are from an unseen test set. During this phase, the agent is not provided any cost penalties for the task. This setup allows us to measure two things: **(1)** how well an agent can learn new tasks while following previously learned textual constraints, and **(2)** the applicability of our method when using textual constraints unseen in training.

6.4 HazardWorld

To our knowledge, there do not currently exist datasets for evaluating RL agents that obey textual constraints.[1] Thus, we design a new benchmark called HazardWorld in which the agent starts each episode at a random location within a procedurally generated environment and receives a textual constraint x, sampled from a pool of available constraints. The agent's goal is to collect all the reward-providing entities while adhering to the specified constraint. Other than the constraint specified, the agent has complete freedom and is not told about how to reach reward-providing states.

HazardWorld contains three types of constraints–**(1)** *budgetary constraints*, which impose a limit on the number of times a set of states can be visited, **(2)** *relational constraints*, which define a minimal distance that must be maintained between the agent and a set of entities, and **(3)** *sequential constraints*, which are constraints that activate unsafe states when a specific condition has been met. In total, we collect 984 textual constraints for HazardWorld-grid (GridWorld environment) and 2,381 textual constraints for HazardWorld-robot (robotic tasks). Table 6.1 provides examples.

HazardWorld-grid. We implement HazardWorld-grid (Table 6.1(a)) atop the 2D GridWorld layout of BabyAI [35, 36]. We randomly place three *reward entities* on the map: '*ball*,' '*box*,' and '*key*,' with rewards of 1, 2, and 3, respectively. We also randomly place several *cost entities* on the map: '*lava*,' '*water*,' and '*grass*'. We give a cost penalty of 1 when agents step onto any cost entities, which are specified using a textual constraint x. The entire state s_t is a grid of size 13×13, including the walls, and the agent's observation o_t is a 7×7 grid of its local surroundings. There are 4 actions–moving up, down, left, and right. We use the deterministic transition here.

[1] Even though there are several instruction following tasks, our task setup is different, as mentioned previously.

Train-test Split. We generate two disjoint training and evaluation datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{eval}}$. $\mathcal{D}_{\text{train}}$ consists of 10,000 randomly generated maps paired with 80% of the textual constraints (787 constraints overall), *i.e.,* on average each constraint is paired with 12.70 different maps. $\mathcal{D}_{\text{eval}}$ consists of 5,000 randomly generated maps paired with the remaining 20% of the textual constraints (197 constraints), *i.e.,* on average one constraint is paired with 25.38 maps. In $\mathcal{D}_{\text{eval}}$ we change the rewards for a ball, box, and key to 1, 2, and -3, respectively. Therefore, in $\mathcal{D}_{\text{eval}}$, the agent has to avoid collecting the key to maximize the reward.

HazardWorld-robot. We build HazardWorld-robot (Table. 6.1(b)) atop the Safety Gym environment [127] to show the applicability of our model to tasks involving high-dimensional continuous observations. In this environment, there are five constraint entities paired with textual constraints: *hazards* (dark blue puddles), *vases* (stationary but movable teal cubes), *pillars* (immovable cylinders), *buttons* (touchable orange spheres), and *gremlins* (moving purple cubes). This task is more challenging than the 2D case since some obstacles are constantly moving. The agent receives a reward of 4 for reaching a goal position and a cost penalty of 1 for bumping into any constraint entities. The observation o_t is a vector of size 109, including coordinate location, the velocity of the agent, and observations from lidar rays that detect the distance to entities. The agent has two actions–control signals applied to the actuators to make it move forward or rotate. The transitions are all deterministic.

Train-test Split. We follow the same process for obtaining a train-test split as in HazardWorld-grid. $\mathcal{D}_{\text{train}}$ consists of 10,000 randomly generated maps paired with 80% of textual constraints (1,905 constraints), *i.e.,* on average one constraint is paired with 5.25 maps. $\mathcal{D}_{\text{eval}}$ consists of 1,000 randomly generated maps paired with the remaining 20% of textual constraints (476 constraints), *i.e.,* on average one constraint is paired with 2.10 maps. In $\mathcal{D}_{\text{eval}}$ we add four additional goal locations to each map (*i.e.,* the maximum reward is 20). The agent has to learn to navigate to

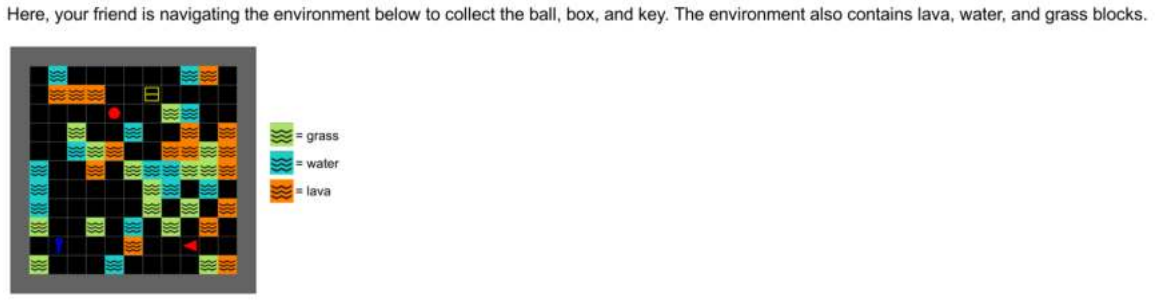
these new locations.

Data Collection. For the textual constraints in both environments, we collected free-form text in English using Amazon Mechanical Turk (AMT) [26]. To generate a constraint for HazardWorld, we provided workers with a description and picture of the environment, the cost entity to be avoided, and one of the following: **(a)** the cost budget (budgetary), **(b)** the minimum safe distance (relational), or **(c)** the other cost entity impacted by past events (sequential). We then cleaned the collected text by writing a keyword matching script followed by manual verification to ensure the constraints are valid.

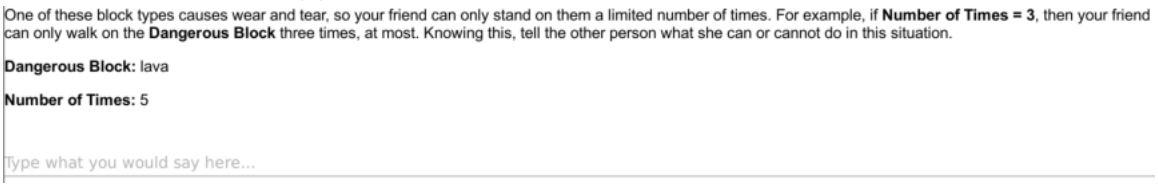
Dataset Details. At a high level, HazardWorld applies the instruction following paradigm to safe reinforcement learning. Concretely, this means that safety constraints in our environment are specified via language. Our dataset is thus comprised of two components: the environment, made up of the objects that the agent interacts with, and the constraint, which imposes a restriction on which environmental states can be visited.

The environment is procedurally generated. For each episode, HazardWorld places the agent at a randomized start location and fills the environment with objects. HazardWorld then randomly samples one constraint out of all possible constraints and assigns this constraint to the environment.

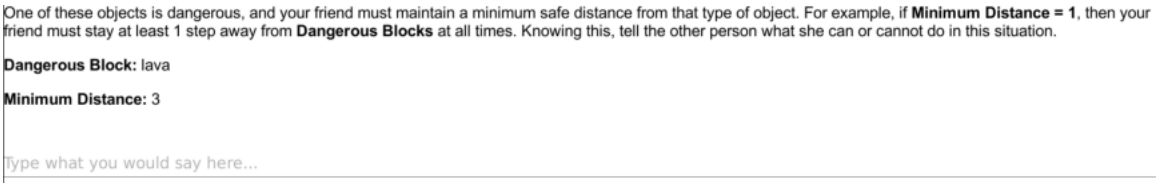
We collected natural language constraints in a two-step process. In the first step, or the data generation step, we prompted workers on Amazon Mechanical Turk with scenarios shown in Fig. 6.2. Workers are provided the minimum necessary information to define the constraint and asked to describe the situation to another person. For example, to generate a so-called budgetary constraint, workers are given the cost entity to avoid (*'lava', 'grass'* or *'water'*) and the budget (*i.e.,* h_C, a number 0 through 5). The workers use this information to write instructions for another person. This allows us to ensure that the texts we collected are free-form. These generations form



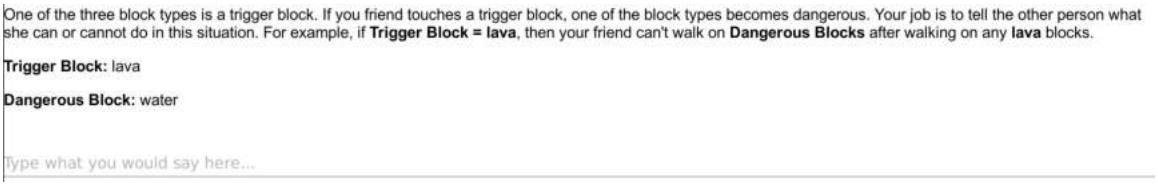
(a) General prompt for all constraint classes.



(b) Budgetary prompt.



(c) Relational prompt.



(d) Sequential prompt.

Figure 6.2: AMT workers receive the general prompt and one of the three specific prompts. They are then asked to instruct another person for the given situation. This ensures that the texts we collected are free-form.

Constraint Type	Examples
Budgetary	The water should only be stepped on a max of 5 times.
	Lava hurts a lot, but you have special shoes that you can use to walk on it, but only up to 5 times, remember!
	You can get in lava, but only once.
	Four is the most number of times you can touch water
	You cannot step on the lava block at all. You will die otherwise.
Relational	Water will hurt you if you are two steps or less from them.
	Always stay 1 step away from lava
	Any block within one unit of a grass cannot be touched.
	The explosion radius of grass is three, so stay at least that distance away from grass.
	Waters are dangerous, so do not cross them.
Sequential	Make sure you don't walk on water after walking on grass.
	Do not touch the water or water will become risky.
	You may touch the water first, but the lava is dangerous so do not touch it after.
	Avoid lava since you can only walk on it once. After that the lava will hurt you.
	Water will trigger grass to become dangerous.
Invalid	good
	move foreward
	Just avoid the perimeter when collecting the objects, and you'll be safe.
	Your directions are as follows: if you're facing a block with a water block in front of it, walk five blocks ahead ... (81 more words)
	asdf

Table 6.2: Examples from the various constraint classes. When a constraint does not fully describe all forbidden states in the environment, we classify it as invalid.

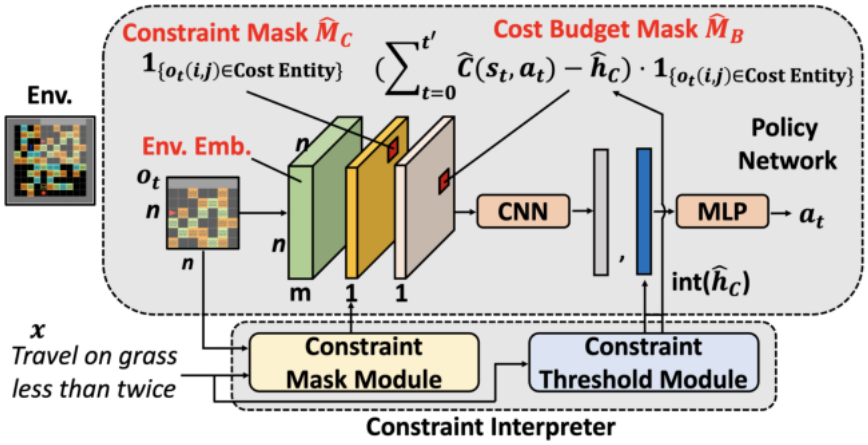


Figure 6.3: **Model Overview.** Our model consists of two parts: **(1)** the *constraint interpreter* produces a constraint mask and cost constraint threshold prediction from a textual constraint and an observation, **(2)** a *policy network* takes in these presentations and produces a constraint-satisfying policy. Best viewed in color.

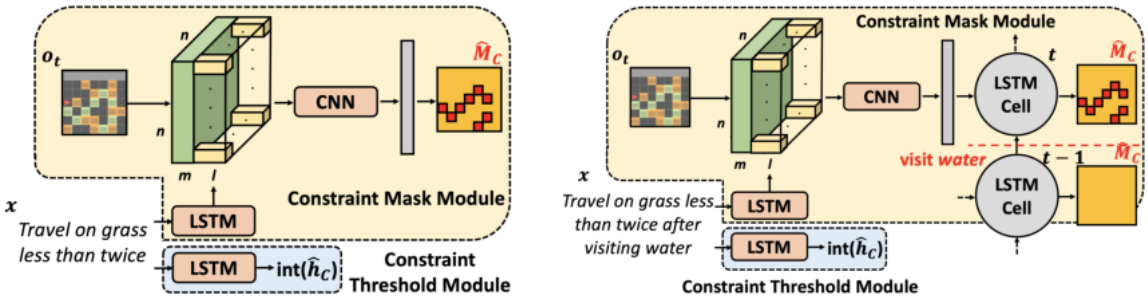
our language constraints.

In the second step or the data validation step, we employed an undergraduate student to remove invalid constraints. We define a constraint as invalid if (a) the constraint is off-topic or (b) the constraint does not clearly describe states that should be avoided. Examples of valid and invalid constraints are included in Table 6.2. Finally, we randomly split the dataset into 80% training and 20% test sets. In total, we spent about $ 1500 for constructing HazardWorld.

In HazardWorld and Lawawall, the agent has 4 actions in total: $a \in \mathcal{A} = \{\text{right}, \text{left}, \text{up}, \text{down}\}$. The transition dynamics T are deterministic.

6.5 Learning to Interpret Textual Constraints

We seek to train agents that can adhere to textual constraints even when learning policies for new tasks with different reward structures. We now describe our model and training and evaluation procedures.



(a) For budgetary and relational constraints

(b) For sequential constraints

Figure 6.4: **Constraint interpreter. (a)** For the budgetary and relational constraints, a constraint mask module takes the environment embedding and text vector representation as inputs and predicts $\hat{M}_C$. **(b)** For the sequential constraints, we use an LSTM to store the information of the past visited states. For these three types of constraints, we use another LSTM given x to predict $\hat{h}_C$.

6.5.1 Network Architecture

We design the RL agent as a deep neural network that consists of two parts (Fig. 6.3)– **(1)** a *constraint interpreter* which processes the text into structured safety criteria (a constraint mask and threshold) and **(2)** a *policy network* which uses the output of the interpreter along with observations to produce an action. For simplicity, in the following descriptions, we assume state s and observation o to be 2D matrices, although the model can easily be extended to other input representations.

(1) Constraint Interpreter (Fig. 6.4). We concatenate an observation embedding of size $n \times n \times m$ from observations o of size $n \times n$ with the embedding of the textual constraints x of size l from a long-short-term-memory (LSTM), followed by using a convolutional neural network (CNN) to get an embedding vector. We use this vector to produce a constraint mask $\hat{M}_C$, a binary matrix with the same dimension as o–each cell of the matrix is 0/1 depending on whether the model believes the absence or presence of a constraint-related entity (*e.g.,* '*lava*') in the corresponding cell of the observation o. In addition, we feed the textual constraints into an LSTM to produce $\hat{h}_C$, a real-valued scalar that predicts the constraint threshold, *i.e.,* the number of times an unsafe state is allowed.

For the case of sequential constraints with a long-term dependency of the past events, $\hat{M}_C$ will depend on the past states visited by the agent. For example, in Fig. 6.4(b), after the agent visits '*water*', $\hat{M}_C$ starts to locate the cost entity ('*grass*'). Thus, for sequential constraints, we modify the interpreter by adding an LSTM layer before computing $\hat{M}_C$ to take the state history into account. Using $\hat{M}_C$ and $\hat{h}_C$ allows us to embed textual constraints in the policy network.

(2) Policy Network. The policy network produces an action using the state observation o_t and the safety criteria produced by the constraint interpreter. The environment embedding is concatenated with the constraint mask $\hat{M}_C$ (predicted by the constraint interpreter) and a *cost budget mask*, denoted by $\hat{M}_B$. The cost budget mask is derived from $\hat{h}_C$ (also predicted by the constraint interpreter) and keeps track of the number of constraint violations that the agent has made in the past over the threshold. $\hat{M}_B$ is an $n \times n$ matrix where each element takes the value of $\sum_{t=0}^{t'} \hat{C}(s_t, a_t; x) - \hat{h}_C$ (*i.e.,* the value of constraint violations past the budget until t'th step) if there is a cost entity in $o_t(i, j)$, or zero otherwise. During the safety evaluation phase, we estimate the cumulative cost $\sum_{t=0}^{t'} \hat{C}(s_t, a_t; x)$ using the predicted $\hat{M}_C$ and the agent's current location at time t. After concatenating both the constraint mask $\hat{M}_C$ and cost budget mask $\hat{M}_B$ to the observation embedding, we then feed the resulting tensor into CNN to obtain a vector (grey in Fig. 6.3). This vector is concatenated with a vectorized $\text{int}(\hat{h}_C)$ (*i.e.,* $\hat{h}_C$ rounded down) and fed into an MLP to produce an action.

POLCO in HazardWorld-robot. To apply POLCO in this environment, the constraint interpreter predicts the cost entity given the textual constraints. We then map the cost entity to the pre-defined embedding vector (*i.e.,* one-hot encoding). We then concatenate the embedding vector, the embeddings of the predicted $\hat{h}_C$, and the value of the cost budget (rounded down) to the observation vector. Finally, the policy network takes in this concatenated observation and produces a safe action.

Advantages of the Design. The design of POLCO tightly incorporates textual constraints into the policy network. Our model factorization–into **(1)** a constraint interpreter and **(2)** a policy network–allows us to design specific constraint interpreters for different types of constraints.[2] Furthermore, our approach scales gracefully to handling multiple constraints. While existing safe RL algorithms require retraining the policy for each unique combination of constraints, we can simply add together the $\hat{M}_C$ of each constraint to handle multiple constraints imposed simultaneously.

6.5.2 Safety Training

We first train the constraint interpreter using a random policy to collect trajectories, and then we use the trained interpreter to predict constraints while training the policy network.

Stage 1: Interpreter Learning. We use a random policy to explore the environment, and obtain trajectories consisting of observations o_t, along with the corresponding textual constraint x. Using the constraint violations encountered in the trajectory and the cost specification C, we obtain a target M_C for training the constraint interpreter. In addition, we also obtain the ground-truth value of h_C for learning the constraint threshold module.

We train the constraint mask module of the constraint interpreter by minimizing the following binary cross-entropy loss over these trajectories:

$$\mathcal{L}(\Theta_1) = -\mathbb{E}_{(o_t,x)\sim\mathcal{D}_{\text{train}}}\Big[\frac{1}{|M_C|}\sum_{i,j=1}^{n} y\log\hat{y} + (1-y)\log(1-\hat{y})\Big],$$

where y is the target $M_C(i,j;o_t,x)$, which denotes the target (binary) mask label in ith row and jth column of the $n\times n$ observation o_t, $\hat{y}$ is the predicted $\hat{M}_C(i,j;o_t,x)$,

[2] $\hat{M}_B$ equates to a scaled-up version of $\hat{M}_C$ since we assume only one constraint specification per episode, but this is not necessary in general since we may have multiple constraints over different cost entities. In that case, $\hat{M}_B$ may have different cost budgets for different cells (entities).

i.e., the probability prediction of constraint mask, and Θ_1 are the parameters of the constraint mask module.

For the constraint threshold module, we minimize the following loss: $\mathcal{L}(\Theta_2) = \mathbb{E}_{(o_t,x)\sim\mathcal{D}_{\text{train}}}\left[(h_C(x) - \hat{h}_C(x))^2\right]$, where Θ_2 are the parameters of the constraint threshold module.

This approach ensures cost satisfaction *during both policy learning and safety evaluation*, an important feature of safe RL. If we train both the policy and the interpreter simultaneously, then we risk optimizing according to inaccurate $\hat{M}_C$ and $\hat{h}_C$ values, as observed in our experiments.

Stage 2: Policy Learning. We use a safe RL algorithm called projection-based constrained policy optimization (PCPO) [173] to train the policy network. During training, the agent interacts with the environment to obtain rewards, and penalty costs ($\hat{M}_C$) are provided from the trained constraint interpreter for computing $J_R(\pi)$ and $J_C(\pi)$ (ground-truth C is not used). PCPO is an iterative method that performs two key steps in each iteration[3]–optimize the policy according to reward and project the policy to a set of policies that satisfy the constraint. During safety evaluation, we evaluate our model in the new task with the new reward function and the textual constraints from $\mathcal{D}_{\text{eval}}$.

6.5.3 Safety Evaluation

(1) Transfer to new tasks: We take the policy trained in $\mathcal{D}_{\text{train}}$ and fine-tune it on tasks having new reward functions with textual constraints from $\mathcal{D}_{\text{eval}}$. We *do not* retrain the constraint interpreter on $\mathcal{D}_{\text{eval}}$. The policy is fine-tuned to complete the new tasks without the penalty signals from the cost function C. In HazardWorld-robot, we optimize the policy using CPO [6].

(2) Handling multiple textual constraints: We also test the ability of our

[3]One can use other safe RL algorithms such as Constrained Policy Optimization (CPO) [6]

model to handle multiple constraints imposed simultaneously (from $\mathcal{D}_{\text{eval}}$), by adding the cost constraint masks $\hat{M}_C$ of each constraint together when given multiple constraints. During safety training, the policy is still trained with a single constraint. No fine-tuning is performed and the reward function is maintained the same across training and evaluation in this case.

6.6 Experiments

Our experiments aim to study the following questions: **(1)** Does the policy network, using representations from the constraint interpreter, achieve fewer constraint violations in new tasks with different reward functions? **(2)** How does each component in POLCO affect its performance?

6.6.1 Setup

Baselines. We consider the following baselines:

(1) Constraint-Fusion (CF) with PCPO: This model [164] takes a concatenation of the observations and text representations as inputs (without M_C, M_B and h_C) and produces an action, trained with an end-to-end approach using PCPO. This model jointly processes the observations and the constraints.

(2) CF with TRPO: We train CF using trust region policy optimization (TRPO) [133], which *ignores* all constraints and only optimizes the reward. This is to demonstrate that the agent will have substantial constraint violations when ignoring constraints.

(3) Random Walk (RW): We also include a random walk (RW) baseline, where the agent samples actions uniformly at random.

Evaluation Metrics. To evaluate models, we use **(1)** the average value of the reward $J_R(\pi)$, and **(2)** the average constraint violations $\Delta_C := \max(0, J_C(\pi) - h_C)$.

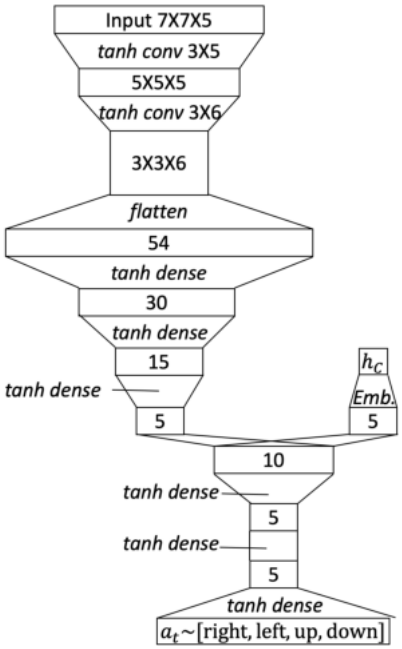


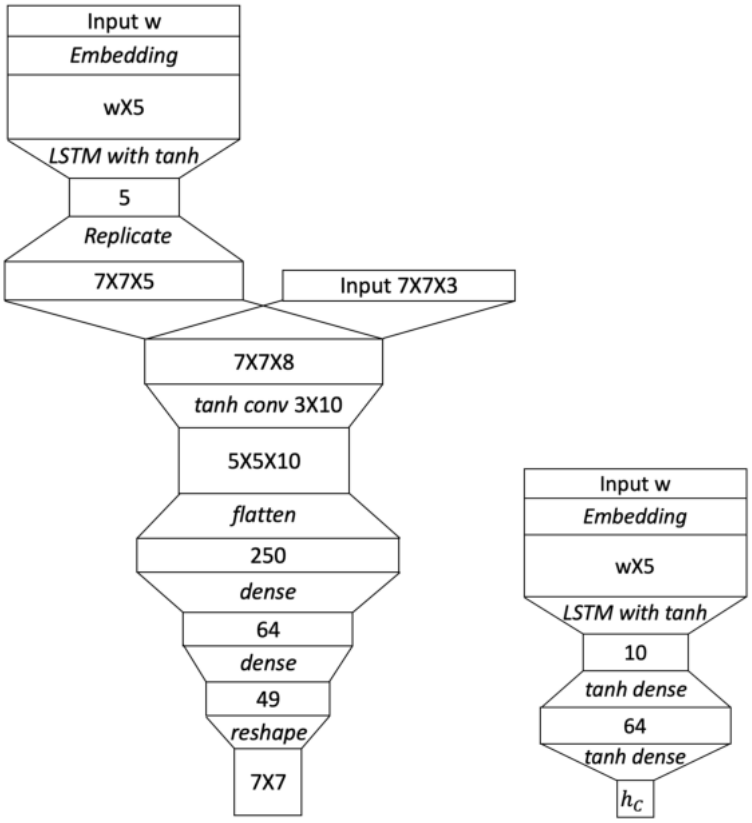
Figure 6.5: Description of the policy network in POLCO.

Good models should have a small Δ_C (*i.e.,* close to zero) while maximizing $J_R(\pi)$.

6.6.2 Architectures, Parameters, and Training Details

Policy Network in POLCO. The architecture of the policy network is shown in Fig. 6.5. The environment embedding for the observation o_t is of the size 7×7×3. This embedding is further concatenated with the cost constraint mask M_C and the cost budget mask M_B. This forms the input with the size 7×7×5. We then use convolutions, followed by dense layers to get a vector with the size 5. This vector is further concatenated with the h_C embedding. Finally, we use dense layers to the categorical distribution with four classes (*i.e.,* turn right, left, up, or down in HazardWorld). We then sample an action from this distribution.

Constraint Interpreter in POLCO. The architecture of the constraint interpreter is shown in Fig. 6.6. For the constraint mask module, the input is the text with w words. We then use an embedding network, followed by an LSTM to obtain the text embedding with the size 5. The text embedding is duplicated to get a tensor with



(a) Constraint mask module (b) Constraint threshold module

Figure 6.6: Description of the constraint interpreter.

Parameter	
Reward dis. factor γ	0.99
Constraint cost dis. factor γ_C	1.0
step size δ	10^{-3}
λ_R^{GAE}	0.95
λ_C^{GAE}	0.9
Batch size	10,000
Rollout length	200
Number of policy updates	2,500

Table 6.3: Parameters used in POLCO.

the size 7×7×5. This tensor is concatenated with the observation of size $7 \times 7 \times 3$, creating a tensor with the size 7×7×8. In addition, we use a convolution, followed by dense layers and a reshaping to get the cost constraint mask M_C.

Next, we use a heuristic to compute $\hat{C}_{tot} := \sum_{t=0}^{t'} C(s_t, a_t; x)$ from M_C. At execution time, we give our constraint interpreter access to the agent's actions. We initialize $\hat{C}_{tot} = 0$. Per timestep, our agent either turns or moves forward. If the agent moves forward and the square in front of the agent contains a cost entity according to M_C, we increment $\hat{C}_{tot}$.

For the constraint threshold module, we use the same architecture to get the text embedding. We then use dense layers to predict the value of h_C.

Details of the Algorithm–PCPO. We use a KL divergence projection in PCPO to project the policy onto the cost constraint set since it has a better performance than L_2 norm projection. We use GAE-λ approach [134] to estimate $A_R^\pi(s, a)$ and $A_C^\pi(s, a)$. We use neural network baselines with the same architecture and activation functions as the policy networks. The hyperparameters of training POLCO are in Table 6.3. We conduct the experiments on the machine with an Intel Core i7-4770HQ CPU. The experiments are implemented in rllab [46], a tool for developing RL algorithms.

Baseline Model–Constraint Fusion (CF). The model is illustrated in Fig. 6.7. An LSTM takes the text x as input and produces a vector representation. The CNN

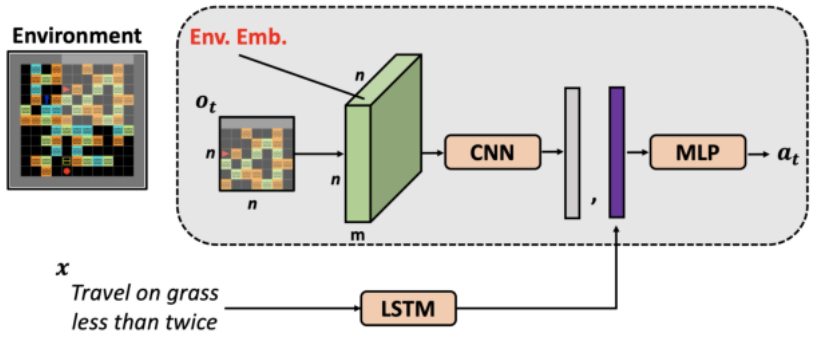


Figure 6.7: Baseline model–Constraint Fusion (CF). It is composed of two parts – **(1)** a CNN takes o_t as an input and produce a vector representation, **(2)** an LSTM takes x as input and produces a vector representation. We then concatenate these two vectors, followed by an MLP to produce an action a_t.

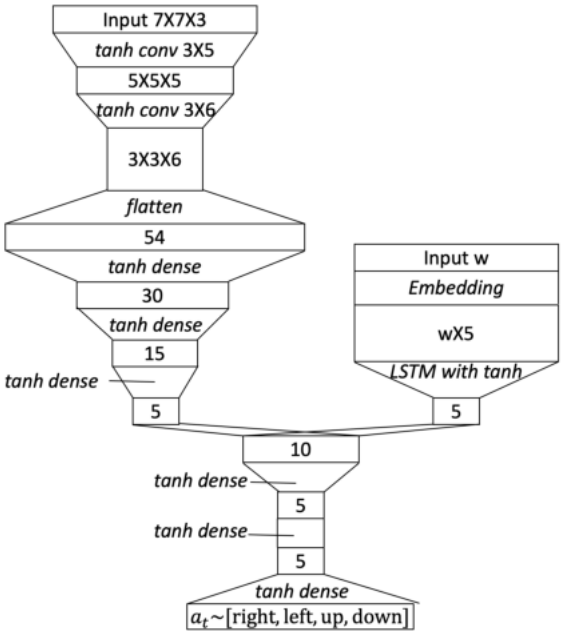


Figure 6.8: Description of our baseline model-Constraint Fusion (CF).

takes the environment embedding of o_t as input and produces a vector representation. These two vector representations are concatenated, followed by an MLP to produce an action a_t. We do not consider other baselines in [81] and [110]. This is because their models are designed to learn a multi-modal representation (*e.g.,* processing a 3D vision) and follow goal instructions. In contrast, our work focuses on learning a constraint-satisfying policy.

The parameters of the baseline is shown in Fig. 6.8. We use the same CNN parameters as in our policy network to process o_t. Then, we use the same LSTM parameters as in our constraint mask module to get a vector representation with size 5. Note that we use almost the same number of parameters to ensure that POLCO does not have an advantage over CF. Finally, we use dense layers to the categorical distribution with four classes. We then sample an action from this distribution.

6.6.3 Experiment Results

HazardWorld-grid. Fig. 6.9(1) shows results for all models in the first evaluation setting of transfer to new tasks. POLCO has lower constraint violations in excess of h_C while still achieving better reward performance in all cases. In comparison, the high cost values (Δ_C) obtained by RW and CF with TRPO indicate the challenges of the task. This supports our idea of using the learned constraint interpreter to learn a new task with similar textual constraints while ensuring constraint satisfaction. CF with PCPO has higher constraint violations, and in most cases, does not optimize the reward, which suggests that it cannot transfer the constraint understanding learned in $\mathcal{D}_{\text{train}}$ to $\mathcal{D}_{\text{eval}}$.

Fig. 6.9(2) shows our evaluation with multiple textual constraints. We see that POLCO achieves superior reward and cost performance compared to the baselines, while CF with PCPO has worse reward and cost performance. This shows that our approach is flexible enough to impose multiple constraints than that of existing

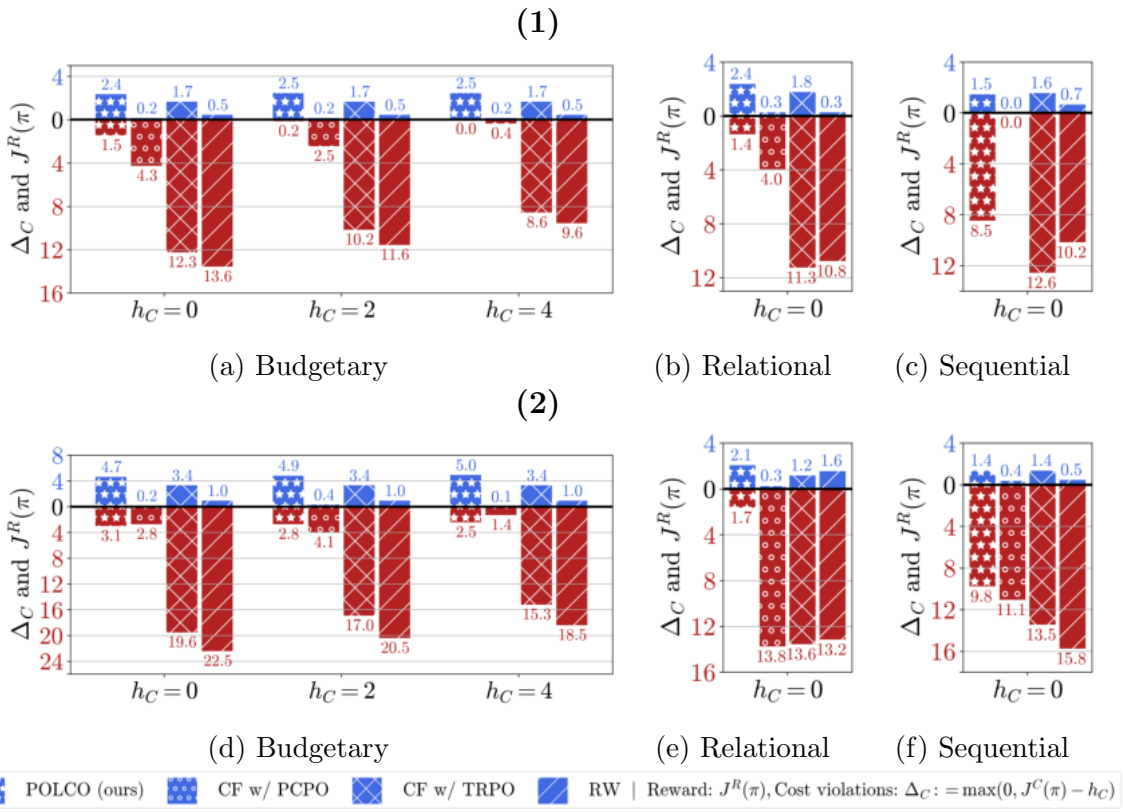


Figure 6.9: Results in HazardWorld-grid over different values of h_C. These graphs represent the results of budgetary, relational, and sequential constraints, respectively. The **blue bars** are the reward performance ($J^R(\pi)$) and the **red bars** are the constraint violations (Δ_C). For $J^R(\pi)$, higher values are better and for Δ_C, lower values are better. **(1)** Results for transfer to the new tasks. **(2)** Results for handling multiple textual constraints. POLCO generalizes to unseen reward structures and handles multiple constraints with minimal constraint violations in the new task.

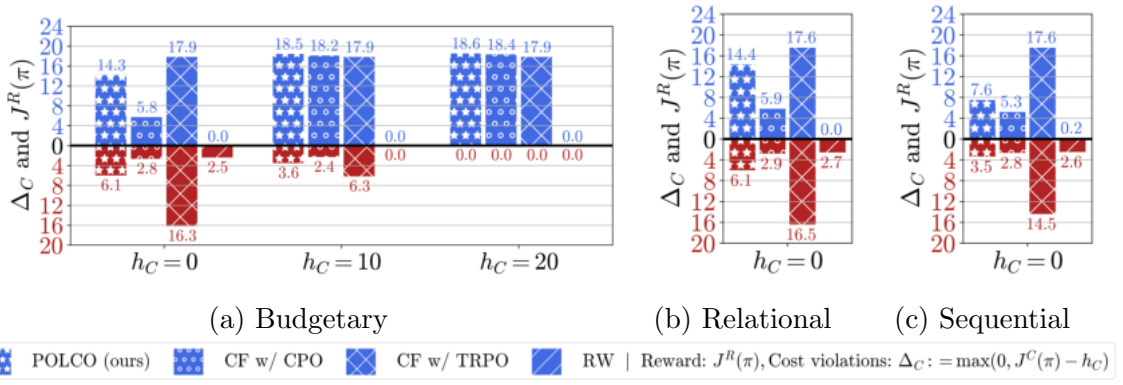


Figure 6.10: Results in HazardWorld-robot over different values of h_C for transfer to the new tasks. POLCO achieves competitive results with higher rewards and lower cost violations.

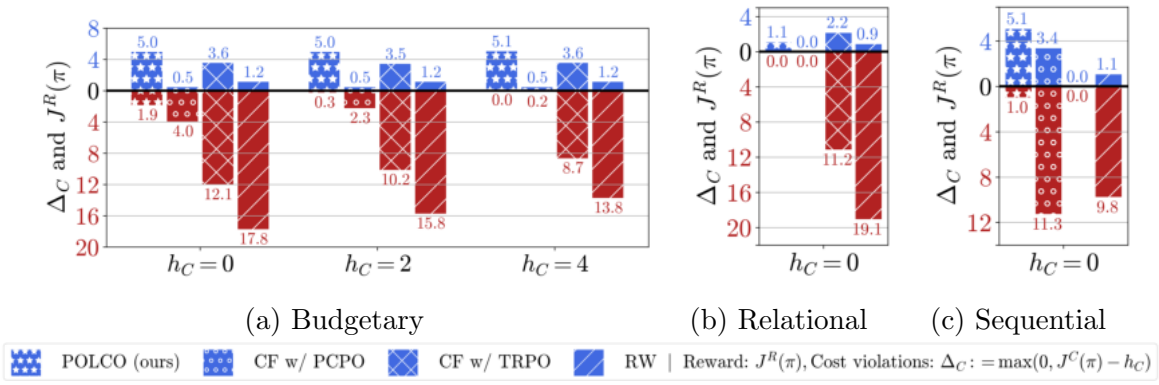


Figure 6.11: Results in HazardWorld-grid for the setting of evaluation with the same reward function as seen in training. POLCO achieves higher rewards and lower constraint violations over the baselines.

safe RL methods which requires retraining the policy for each unique combination of constraints.

HazardWorld-robot. Fig. 6.10 shows transfer to new tasks in HazardWorld-robot. The $J^R(\pi)$ and Δ_C of RW are relatively small since the agent does not move much because of random force applied to each actuator. For the budgetary constraints, although CF with TRPO achieves the best reward when $h_C = 0$, it has very large constraint violations. POLCO performs better than the baselines–it induces policies with higher rewards under fewer constraint violations in most cases. In contrast, CF with CPO has lower reward performance.

Having demonstrated the overall effectiveness of POLCO, our remaining experiments analyze **(1)** the learned models' performance evaluated on the same reward function as in $\mathcal{D}_{\text{train}}$, and **(2)** the importance of each component–M_B, M_C and h_C embedding in POLCO. For compactness, we restrict our consideration in HazardWorld-grid.

Evaluation with Reward Function from $\mathcal{D}_{\text{train}}$. To provide another point of comparison in addition to our main results, we evaluate all models using the same reward function as in $\mathcal{D}_{\text{train}}$, but with unseen textual constraints from $\mathcal{D}_{\text{eval}}$. (Fig. 6.11)

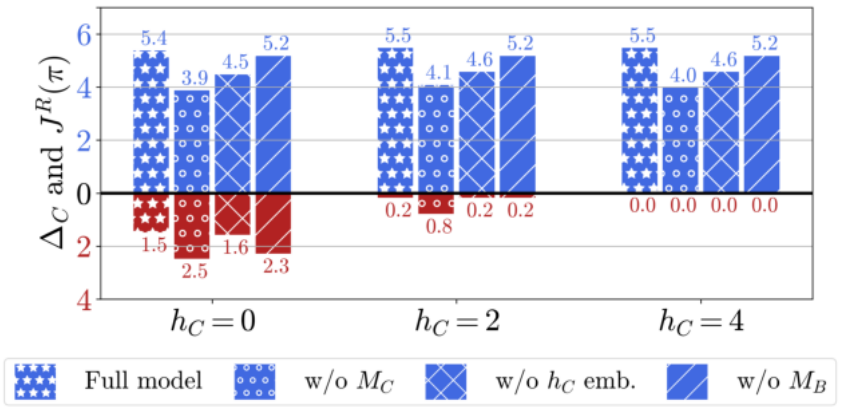
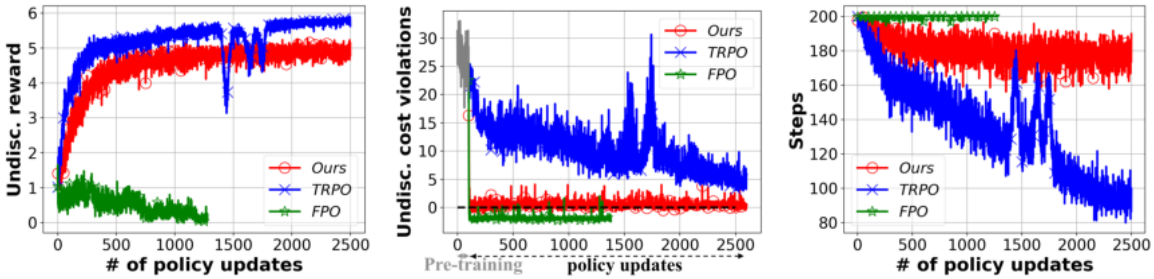


Figure 6.12: Ablations showing the effect of each component in POLCO for the budgetary constraint.

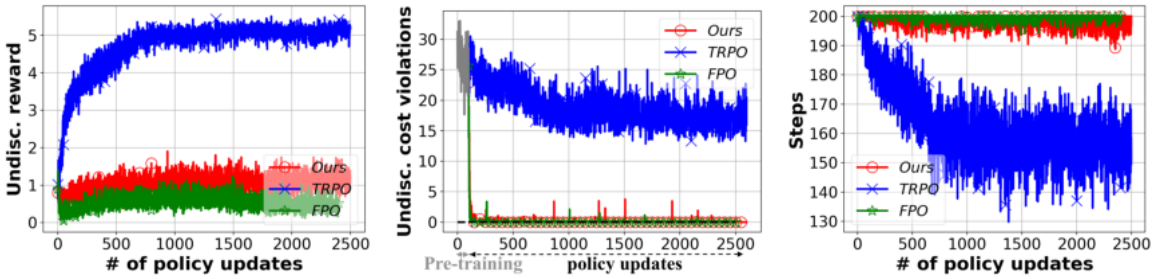
We observe POLCO achieves the lowest violations across different choices of h_C compared to the baselines. This implies that merely combining the observations and the text is not sufficient to learn an effective representation for parsing the constraints. In addition, POLCO achieves the best reward performance under cost satisfaction for the more complex relational and sequential constraints. For the relational case, although the CF agent trained with PCPO satisfies the constraints, it has a relatively low reward.

Ablation Studies. We also examine the importance of each part in POLCO (Fig. 6.12). To eliminate prediction errors from the constraint interpreter, we use the *true* M_C and h_C here. Our full model achieves the best performance in all cases, averaging 5.12% more reward and 2.22% fewer constraint violations. Without M_C, the agent cannot recognize cost entities effectively, which causes the agent to incur 66.67% higher Δ_C compared with the full model (which has a Δ_C close to zero). This shows that h_C embedding and the M_B mask are useful in enabling constraint satisfaction given textual constraints.

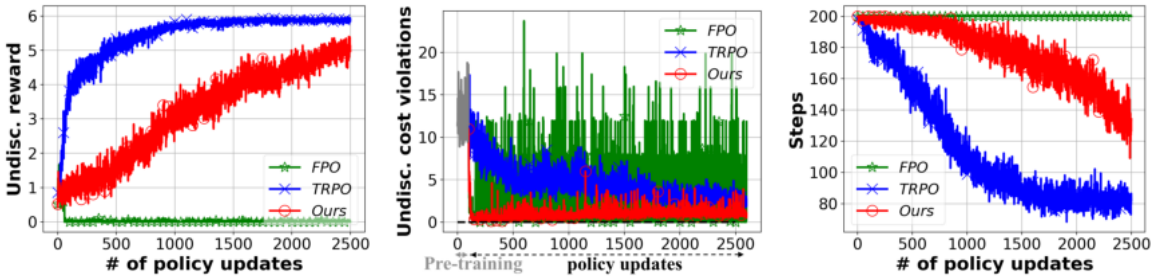
Learning Curves of Training the Policy Network. The learning curves of the undiscounted constraint cost, the discounted reward, and the number of steps over policy updates are shown for all tested algorithms and the constraints in Fig.



(a) Budgetary constraints



(b) Relational constraints



(c) Sequential constraints

Figure 6.13: **Learning Curves of Training the Policy Network.** The undiscounted reward, the undiscounted cost violations (*i.e.*, $\Delta_C = J_C(\pi) - h_C$), and the number of steps over policy updates for the tested algorithms and the constraints. In the undiscounted cost violations plots, we further include the numbers for the interpreter pre-training stage in the first 100 points. This is equal to 5000 trajectories. The maximum allowable step for each trajectory is 200. We observe that POLCO satisfies the cost constraints throughout training while improving the reward. In contrast, the policy network trained with TRPO suffers from violating the constraints and the one trained with FPO cannot effectively improve the reward. (Best viewed in color.)

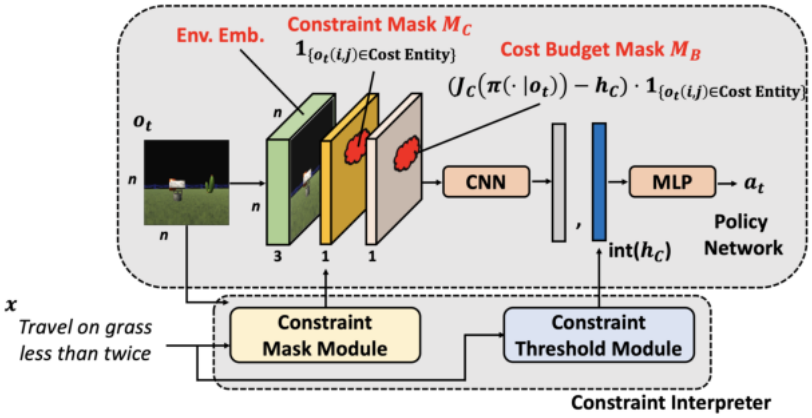


Figure 6.14: POLCO for pixel observations and 3D ego-centric observations. The red cloud area represents the bounding box of each object in o_t.

6.13. Overall, we observe that

(1) POLCO improves the reward performance while satisfying the cost constraints during training in all cases,

(2) the policy network trained with TRPO has substantial cost constraint violations during training,

(3) the policy network trained with FPO is overly restricted, hindering the reward improvement.

6.7 Discussion and Conclusion

Our work provides a view of machines that can interoperate with humans. As autonomous agents proliferate into our world, they should be able to understand safety constraints set by human agents around them. Accordingly, we proposed the problem of safe RL with natural language constraints, created a new benchmark called HazardWorld to test agents, and developed a new algorithm for the task (POLCO) that learns to interpret constraints. Our paper defines and trains machine agents that

understand what *not to do* in natural language, much like instruction following tasks enable agents in understanding what *to do.*

The thesis of our POLCO approach is that modularity enables reuse. By bootstrapping a modular constraint interpreter through exploration, our model scales easily to multiple constraints and to shifts in the environment's reward structure, all while exploring new environments safely. We applied POLCO within HazardWorld to train an agent that navigates safely by obeying natural language constraints. This agent is a step towards creating applications like cleaning robots that can obey free form constraints, such as "*don't get too close to the TV*" – a relational constraint in our formulation.

No model is without limitations. The absolute scores of POLCO on HazardWorld still leave a lot of room for improvement using better models or training techniques. The current version of HazardWorld is also not all-encompassing – we envision it as a benchmark that evolves over time, with the addition of new types of constraints and new environments. Future work can investigate training without explicit labels for the constraint interpreter, potentially using techniques like Gumbel softmax [79], or extending POLCO to tasks with more realistic visuals. In addition, for POLCO for in robotics tasks, to deal with pixel observations o_t, we can still use the proposed architecture to process o_t as shown in Fig. 6.14. To predict the cost constraint mask $\hat{M}_C$, we use the object segmentation method to get the bounding box of each object in the scene. As a result, the area of that bounding box will be one if there is a cost entity (*i.e.,* the forbidden states mentioned in the text). Otherwise, the bounding box contains a zero. For $\hat{M}_B$, we can use a similar approach to compute the cumulative cost violations at each step. In addition, to deal with navigation environments with 3D ego-centric observations, we propose shifting the o_t, $\hat{M}_C$ and $\hat{M}_B$ matrices to be the first-person view. The bounding box approach for the image case can still be applied here. We leave this proposal to future work.

Acknowledgements. We thank Michael Hu for collecting the data and writing the paper, and Dr. Yinlam Chow, Prof. Peter J. Ramadge, and Prof. Karthik Narasimhan for the helpful discussion.

Chapter 7

Conclusion

In this thesis, we study the problem of safe reinforcement learning (RL). We provide theoretical analysis and test the proposed algorithms in a wide range of simulated and real-world tasks. In Chapter 2, we use the idea of projection onto the safety constraint set to ensure constraint satisfaction during the policy update. We provide worst-case constraint violations for each policy update. We test the algorithm in the Mujoco robot tasks with safety constraints and the traffic management tasks with fairness constraints. The results suggest that the model is able to improve the reward while substantially reducing the constraint violations. In Chapter 3, we extend the problem in Chapter 2 when provided with baseline policies. The baseline policy provides a useful cue for learning, but it is not guaranteed to produce higher rewards or satisfy the safety constraint at hand. We extend PCPO in Chapter 2 with a projection step that constrains the distance to the baseline policy. We provide a theoretical analysis to quantify the distance between the learner policy and the baseline policy to ensure safe and efficient learning. The theory ensures reward improvement and constraint satisfaction for each update. We test the algorithm in the experiment where the learning agent aims to learn human driving log data. The results show that SPACE achieves better learning efficiency and fewer safety constraint violations.

In Chapter 4, we deploy a safe RL algorithm in quadrupedal locomotion to show its real-world applicability. We start by proposing a safe RL algorithm that switches between the learner policy and the safe recovery policy. The learner policy is used to learn a task while the safe recovery policy is used to take over the control when the agent is going to violate the safety constraint. We further design switch criteria between the two policies by rolling out the state trajectory based on the provided system dynamics. If the state enters an unsafe set, we use the safe recovery policy; otherwise, we use the learner policy. We verify the proposed algorithm in three locomotion tasks (efficient gaits, two-leg balance, and catwalk) in real hardware. The results show that we are able to learn agile locomotion skills without falling or near-zero falls during the entire learning process. In Chapter 5, we relax the assumption of knowing the system dynamics in the previous chapters and propose an approach that learns to estimate the state and the system parameters when provided with the equation of the system dynamics. Knowing the state and the system parameters aids the learning a control policy. To address this problem, we propose a model that estimates the states and physical parameters of the system using two main components. First, an autoencoder compresses a sequence of observations (*e.g.,* sensor measurements, pixel images) into a sequence for the state representation that is consistent with physics by including a simulation of the dynamic equation. Second, an estimator is coupled with the autoencoder to predict the values of the physical parameters. We also theoretically and empirically show that using Fourier feature mappings improves the generalization of the estimator in predicting physical parameters compared to raw state sequences. In our experiments on the demonstrated example and three visual and one sensor measurement tasks, our model imposes interpretability on latent states and achieves improved generalization performance for long-term prediction of system dynamics over state-of-the-art baselines.

Finally, in Chapter 6, while safe RL holds great promise for many practical appli-

cations like robotics or autonomous cars, current approaches require specifying constraints in mathematical form. Such specifications demand domain expertise, limiting the adoption of safe RL. In addition, current autonomous systems do not have the ability to correct their unsafe behavior after receiving feedback from humans. We thus propose learning to interpret natural language constraints for safe RL. To this end, we first introduce HazardWorld, a new multi-task benchmark that requires an agent to optimize reward while not violating constraints specified in free-form text. We then develop an agent with a modular architecture that can interpret and adhere to such textual constraints while learning new tasks. Across different domains in HazardWorld, we show that our method achieves higher rewards (up to 11x) and fewer constraint violations (by 1.8x) compared to existing approaches. We hope this thesis has not only shed light on how to use RL algorithms in real-world applications safely but will also lead to further understanding and progress in using other techniques to ensure safety when learning control policies.

We discuss several future research directions. On the algorithm side, combing Hamilton-Jacobi reachability methods [51] with safety analysis with the consideration of model error and noise from the environment for safe RL algorithms can provide stronger safety guarantees. In addition, the current safety certificate methods of the control policy such as the Hamilton-Jacobi reachability analysis require a large amount of computation. Such computation prevents deployment in a real-time embedded system such as drones and self-driving cars. An approximate safety analysis but a fast computation method is needed to scale up the safe RL algorithm in more complicated run-time systems. Furthermore, extending our safe RL algorithms to multiple constraints, diverse textual constraints would develop more general-purposed and easy deployment of safe RL algorithms. Moreover, while it is possible to come up with safety guarantees during training and designing of the control policy, the real-world deployment of the control policy makes pre-defined or

pre-programmed safety guarantees weaker or even diminished due to uncertainty and noise from the real world. Incorporating the uncertainty measurement into safe RL is a promising direction. On the application side, safe RL algorithms can be potentially deployed in several applications such as recommendation systems with user preference constraints and robot control tasks with safety constraints. In addition, we show that it is possible to learn a control policy autonomously with minimal human effort in Chapter 4. Scaling this approach to a wide range of applications would enable easy and wide deployment of autonomous systems.

As human moves toward an automated society with a plethora of deployment of self-driving cars, personalized robot assistants, and unmanned aerial vehicles, the safety of these autonomous systems is increasingly important. According to the White House Office of Science and Technology Policy's National Science and Technology Council in 2019, there are several key areas of priority focus for the Federal agencies that invest in AI. One of them is to ensure the safety and security of AI. This shows that ensuring AI safety is a key to the prosperity and health of society. We hope that everyone on earth can safely benefit from an automated society in the future.

Appendix A

Prior Presentations and Publications

A.1 Prior Presentations During Ph.D.

1. Tsung-Yen Yang, Christopher Brinton, Prateek Mittal, Mung Chiang, and Andrew S. Lan. Learning informative and private representations via generative adversarial networks. In IEEE International Conference on Big Data (Big Data), 2018

2. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Projection-Based Constrained Policy Optimization. In International Conference on Learning Representations (ICLR), 2020. Also presented in AAAI 2019 Fall Symposium Series–Human-centered AI: Trustworthness of AI Models and Data

3. Tsung-Yen Yang, Andrew S. Lan, Karthik Narasimhan. Robust and Interpretable Grounding of Spatial References with Relation Networks. In Findings of Conference on Empirical Methods in Natural Language Processing (Findings of EMNLP), 2020. Also presented in Third International Workshop on Spatial Language Understanding In conjunction with The Conference on Empirical

Methods in Natural Language Processing 2020

4. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Accelerating Safe Reinforcement Learning with Constraint-mismatched Policies. In International Conference on Machine Learning (ICML), 2021. Also presented in NeurIPS 2020 Workshop Challenges of Real World Reinforcement Learning

5. Tsung-Yen Yang*, Michael Hu*, Yinlam Chow, Peter J. Ramadge, Karthik Narasimhan. Safe Reinforcement Learning with Natural Language Constraints. In Neural Information Processing System (NeurIPS), 2021 (Spotlight Talk) (*Equal Contribution). Also presented in NeurIPS 2020 Workshop Deep Reinforcement Learning Workshop

A.2 Prior Publications During Ph.D.

1. Tsung-Yen Yang, Christopher G. Brinton, Carlee Joe-Wong, and Mung Chiang. Behavior-based grade prediction for MOOCs via time series neural networks. In IEEE Journal of Selected Topics in Signal Processing (JSTSP), 2017

2. Andrew S. Lan, Christopher G. Brinton, Tsung-Yen Yang, and Mung Chiang. Behavior-based latent variable model for learner engagement. In International Conference on Educational Data Mining (EDM), 2018

3. Madhumitha Shridharan, Ashley Willingham, Jonathan Spencer, Tsung-Yen Yang, and Christopher Brinton. Predictive learning analytics for video-watching behavior in MOOCs. In Conference on Information Sciences and Systems (CISS), 2018

4. Tsung-Yen Yang, Christopher G. Brinton, and Carlee Joe-Wong. Predicting learner interactions in social learning networks. In IEEE Conference on Computer Communications (INFOCOM), 2018

5. Tsung-Yen Yang, Christopher Brinton, Prateek Mittal, Mung Chiang, and Andrew S. Lan. Learning informative and private representations via generative adversarial networks. In IEEE International Conference on Big Data (Big Data), 2018

6. Tsung-Yen Yang, Ryan S. Baker, Christoph Studer, Neil Heffernan, and Andrew S. Lan. Active learning for student affect detection. In International Conference on Educational Data Mining (EDM), 2019

7. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Projection-Based Constrained Policy Optimization. In International Conference on Learning Representations (ICLR), 2020

8. Patrick Hansen, Richard Junior Bustamante, Tsung-Yen Yang, Elizabeth Tenorio, Christopher G. Brinton, Mung Chiang, and Andrew S. Lan. Predicting the timing and quality of responsesin online discussion forums. In IEEE International Conference on Distributed Computing Systems (ICDCS), 2019

9. Tsung-Yen Yang, Andrew S. Lan, Karthik Narasimhan. Robust and Interpretable Grounding of Spatial References with Relation Networks. In Findings of Conference on Empirical Methods in Natural Language Processing (Findings of EMNLP), 2020

10. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Accelerating Safe Reinforcement Learning with Constraint-mismatched Policies. In International Conference on Machine Learning (ICML), 2021 (also in NeurIPS 2020 Workshop Challenges of Real World Reinforcement Learning, Spotlight Talk)

11. Tsung-Yen Yang*, Michael Hu*, Yinlam Chow, Peter J. Ramadge, Karthik Narasimhan. Safe Reinforcement Learning with Natural Language Constraints. In Neural Information Processing System (NeurIPS), 2021 (Spotlight Talk) (*Equal Contribution)

12. Tsung-Yen Yang, Tingnan Zhang, Linda Luu, Sehoon Ha, Jie Tan, Wenhao Yu. Safe Reinforcement Learning for Legged Locomotion. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022

A.3 Prior Presentations Included in the Dissertation

1. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Projection-Based Constrained Policy Optimization. In International Conference on Learning Representations (ICLR), 2020. Also presented in AAAI 2019 Fall Symposium Series–Human-centered AI: Trustworthness of AI Models and Data

2. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Accelerating Safe Reinforcement Learning with Constraint-mismatched Policies. In International Conference on Machine Learning (ICML), 2021. Also presented in NeurIPS 2020 Workshop Challenges of Real World Reinforcement Learning

3. Tsung-Yen Yang*, Michael Hu*, Yinlam Chow, Peter J. Ramadge, Karthik Narasimhan. Safe Reinforcement Learning with Natural Language Constraints. In Neural Information Processing System (NeurIPS), 2021 (Spotlight Talk) (*Equal Contribution). Also presented in NeurIPS 2020 Workshop Deep Reinforcement Learning Workshop

4. Tsung-Yen Yang, Tingnan Zhang, Linda Luu, Sehoon Ha, Jie Tan, Wenhao Yu. Safe Reinforcement Learning for Legged Locomotion. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022

A.4 Prior Publications Included in the Dissertation

1. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Projection-Based Constrained Policy Optimization. In International Conference on Learning Representations (ICLR), 2020

2. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Accelerating Safe Reinforcement Learning with Constraint-mismatched Policies. In International Conference on Machine Learning (ICML), 2021 (also in NeurIPS 2020 Workshop Challenges of Real World Reinforcement Learning, Spotlight Talk)

3. Tsung-Yen Yang*, Michael Hu*, Yinlam Chow, Peter J. Ramadge, Karthik Narasimhan. Safe Reinforcement Learning with Natural Language Constraints. In Neural Information Processing System (NeurIPS), 2021 (Spotlight Talk) (*Equal Contribution)

4. Tsung-Yen Yang, Tingnan Zhang, Linda Luu, Sehoon Ha, Jie Tan, Wenhao Yu. Safe Reinforcement Learning for Legged Locomotion. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022

5. Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, Peter J. Ramadge. Learning Physics Constrained Dynamics Using Autoencoders. Under review in Neural Information Processing System (NeurIPS), 2022

Bibliography

[1] Unitree a1. https://www.unitree.com/products/a1/. Accessed: 2021.

[2] Unitree laikago. https://www.unitree.com/. Accessed: 2021.

[3] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[4] D. Abel, J. Salvatier, A. Stuhlmüller, and O. Evans. Agent-agnostic human-in-the-loop reinforcement learning. *arXiv preprint arXiv:1701.04079*, 2017.

[5] J. Achiam and D. Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019.

[6] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 2017.

[7] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2):4606–4613, 2022.

[8] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[9] AlphaStar. Alphastar: Mastering the real-time strategy game starcraft ii, 2019.

[10] E. Altman. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

[11] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3674–3683. IEEE Computer Society, 2018.

[12] J. Andreas and D. Klein. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1165–1174, Lisbon, Portugal, 2015. Association for Computational Linguistics.

[13] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

[14] P. Bacher and H. Madsen. Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings*, 43(7):1511–1522, 2011.

[15] A. Balakrishna, B. Thananjeyan, J. Lee, A. Zahed, F. Li, J. E. Gonzalez, and K. Goldberg. On-policy robot imitation learning from a converging supervisor. In *Proceedings of the Conference on Robot Learning*, 2019.

[16] O. Bastani, S. Li, and A. Xu. Safe reinforcement learning via statistical model predictive shielding. In *Robotics: Science and Systems*, 2021.

[17] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4502–4510, 2016.

[18] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 908–918, 2017.

[19] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[20] A. Bietti and J. Mairal. On the inductive bias of neural tangent kernels. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12873–12884, 2019.

[21] Y. Bisk, K. J. Shih, Y. Choi, and D. Marcu. Learning interpretable spatial operations in a rich 3d blocks world. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intel-*

ligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5028–5036. AAAI Press, 2018.

[22] V. Blukis, D. Misra, R. A. Knepper, and Y. Artzi. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Proceedings of the Conference on Robot Learning*, pages 505–518, 2018.

[23] S. Branavan, D. Silver, and R. Barzilay. Learning to win by reading manuals in a Monte-Carlo framework. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 268–277, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

[24] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[25] J. Brynjarsdottir and A. OHagan. Learning about physical parameters: The importance of model discrepancy. *Inverse problems*, 30(11):114007, 2014.

[26] M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality data? 2016.

[27] G. Camps-Valls, M. Reichstein, X. Zhu, and D. Tuia. Advancing deep learning for earth sciences: From hybrid modeling to interpretability. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 3979–3982. IEEE, 2020.

[28] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In W. Burgard and D. Roth, editors, *Proceedings*

of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011. AAAI Press, 2011.

[29] G. Chen and M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using bregman functions. *SIAM Journal on Optimization*, 1993.

[30] H. Chen, Z. Hong, S. Yang, P. M. Wensing, and W. Zhang. Quadruped capturability and push recovery via a switched-systems characterization of dynamic balance. *arXiv preprint arXiv:2201.11928*, 2022.

[31] H. Chen, A. Suhr, D. Misra, N. Snavely, and Y. Artzi. TOUCHDOWN: natural language navigation and spatial reasoning in visual street environments. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12538–12547. Computer Vision Foundation / IEEE, 2019.

[32] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman. Using predictions in online optimization: Looking forward with an eye on the past. *ACM SIGMETRICS Performance Evaluation Review*, 44(1):193–206, 2016.

[33] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583, 2018.

[34] S. Chernova and A. L. Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[35] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency

of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[36] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

[37] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

[38] Y. Chow, O. Nachum, E. A. Duéñez-Guzmán, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8103–8112, 2018.

[39] Y. Chow, O. Nachum, A. Faust, M. Ghavamzadeh, and E. Duenez-Guzman. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[40] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning, 2017.

[41] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[42] H. de Vries, K. Shuster, D. Batra, D. Parikh, J. Weston, and D. Kiela. Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*, 2018.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[44] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018.

[45] K. Driessens and S. Džeroski. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57(3):271–304, 2004.

[46] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1329–1338. JMLR.org, 2016.

[47] G. Dulac-Arnold, D. Mankowitz, and T. Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[48] M. El Chamie, Y. Yu, and B. Açıkmeşe. Convex synthesis of randomized policies for controlled markov chains with density safety upper bound constraints. In *2016 American Control Conference (ACC)*, pages 6290–6295, 2016.

[49] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. 2005.

[50] M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[51] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, (7), 2018.

[52] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3318–3329, 2018.

[53] J. Fu, A. Korattikara, S. Levine, and S. Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[54] D. Gaddy and D. Klein. Pre-learning environment representations for data-efficient neural instruction following. In *Proceedings of the 57th Annual Meeting*

of the Association for Computational Linguistics, pages 1946–1956, Florence, Italy, 2019. Association for Computational Linguistics.

[55] Y. Gao, J. Lin, F. Yu, S. Levine, and T. Darrell. Reinforcement learning from imperfect demonstrations. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[56] J. Garcia and F. Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

[57] J. Garcia and F. Fernandez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[58] M. Grewal and K. Glover. Identifiability of linear and nonlinear dynamical systems. *IEEE Transactions on automatic control*, 21(6):833–837, 1976.

[59] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15353–15363, 2019.

[60] V. L. Guen and N. Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11474–11484, 2020.

[61] G. Gui, W. Peng, and F. Adachi. Adaptive system identification using robust lms/f algorithm. *International Journal of Communication Systems*, 27(11):2956–2963, 2014.

[62] J. K. Gupta, K. Menda, Z. Manchester, and M. J. Kochenderfer. A general framework for structured learning of mechanical systems. *arXiv preprint arXiv:1902.08705*, 2019.

[63] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

[64] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[65] D. Hafner, T. P. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019.

[66] W. Hao, C. Li, X. Li, L. Carin, and J. Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13134–13143. IEEE, 2020.

[67] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630, Berlin, Germany, 2016. Association for Computational Linguistics.

[68] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. *arXiv preprint arXiv:2011.04217*, 2020.

[69] E. Heiden, D. Millard, H. Zhang, and G. S. Sukhatme. Interactive differentiable simulation. *arXiv preprint arXiv:1905.10706*, 2019.

[70] K. M. Hermann, M. Malinowski, P. Mirowski, A. Banki-Horvath, K. Anderson, and R. Hadsell. Learning to follow directions in street view. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11773–11781. AAAI Press, 2020.

[71] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. P. Agapiou, J. Z. Leibo, and A. Gruslys. Deep q-learning from demonstrations. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3223–3230. AAAI Press, 2018.

[72] I. Higgins, P. Wirnsberger, A. Jaegle, and A. Botev. Symetric: Measuring the quality of learnt hamiltonian dynamics inferred from vision. *Advances in Neural Information Processing Systems*, 34, 2021.

[73] J. Ho and S. Ermon. Generative adversarial imitation learning. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4565–4573, 2016.

[74] T. M. Howard, S. Tellex, and N. Roy. A natural language planner interface for mobile manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659. IEEE, 2014.

[75] Y. Hristov, D. Angelov, M. Burke, A. Lascarides, and S. Ramamoorthy. Disentangled relational representations for explaining and learning from demonstration. In *Proceedings of the Conference on Robot Learning*, 2019.

[76] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

[77] A. Jacot, C. Hongler, and F. Gabriel. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8580–8589, 2018.

[78] V. Jain, G. Magalhaes, A. Ku, A. Vaswani, E. Ie, and J. Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1862–1872, Florence, Italy, 2019. Association for Computational Linguistics.

[79] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[80] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu. Reasoning about physical interactions with object-oriented prediction and planning. *arXiv preprint arXiv:1812.10972*, 2018.

[81] M. Janner, K. Narasimhan, and R. Barzilay. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61, 2018.

[82] M. Jaques, M. Burke, and T. M. Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[83] N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[84] K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Levesque, K. Xie, K. Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*, 2021.

[85] W. Jin, S. Mou, and G. J. Pappas. Safe pontryagin differentiable programming. *Advances in Neural Information Processing Systems*, 34:16034–16050, 2021.

[86] S. M. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In C. Sammut and A. G. Hoffmann, editors, *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 267–274. Morgan Kaufmann, 2002.

[87] K. Kashinath, M. Mustafa, A. Albert, J. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

[88] J. Kim and R. Mooney. Adapting discriminative reranking to grounded language learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 218–227, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

[89] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[90] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[91] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings*, 2004.

[92] R. Koppejan and S. Whiteson. Neuroevolutionary reinforcement learning for generalized control of simulated helicopters. *Evolutionary intelligence*, 4(4):219–241, 2011.

[93] N. R. Kristensen, H. Madsen, and S. B. Jørgensen. Parameter estimation in stochastic grey-box models. *Automatica*, 40(2):225–237, 2004.

[94] M. Kwon, E. Biyik, A. Talati, K. Bhasin, D. P. Losey, and D. Sadigh. When humans aren't optimal: Robots that collaborate with risk-aware humans. In *Proceedings of ACM/IEEE International Conference on Human-Robot Interaction*, 2020.

[95] H. M. Le, C. Voloshin, and Y. Yue. Batch policy learning under constraints. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3703–3712. PMLR, 2019.

[96] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier. Differentiable simulation for physical system identification. *IEEE Robotics and Automation Letters*, 6(2):3413–3420, 2021.

[97] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 2020.

[98] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5757–5766. PMLR, 2020.

[99] K. Lee, N. Trask, and P. Stinis. Machine learning structure preserving brackets for forecasting irreversible processes. *Advances in Neural Information Processing Systems*, 34, 2021.

[100] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021.

[101] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[102] J. Liang, M. Lin, and V. Koltun. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems*, 32, 2019.

[103] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[104] O. Linial, N. Ravid, D. Eytan, and U. Shalit. Generative ode modeling with known unknowns. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 79–94, 2021.

[105] N. Liu, M. Du, and X. Hu. Representation interpretation with spatial encoding and multimodal analytics. In J. S. Culpepper, A. Moffat, P. N. Bennett, and K. Lerman, editors, *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 60–68. ACM, 2019.

[106] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6309–6317. ijcai.org, 2019.

[107] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *7th International Conference on Learning*

Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

[108] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Conference on Artificial Intelligence*, pages 1475–1482, 2006.

[109] H. Miao, X. Xia, A. S. Perelson, and H. Wu. On identifiability of nonlinear ode models and applications in viral dynamics. *SIAM review*, 53(1):3–39, 2011.

[110] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678, Brussels, Belgium, 2018. Association for Computational Linguistics.

[111] D. Misra, J. Langford, and Y. Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1004–1015, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

[112] 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.

[113] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[114] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie,

A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[115] A. Mokhtari, A. E. Ozdaglar, and A. Jadbabaie. Escaping saddle points in constrained optimization. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3633–3643, 2018.

[116] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 2013.

[117] T. Ogunfunmi. *Adaptive nonlinear system identification: The Volterra and Wiener model approaches*. Springer Science & Business Media, 2007.

[118] H. Oliff, Y. Liu, M. Kumar, M. Williams, and M. Ryan. Reinforcement learning for facilitating human-robot-interaction in manufacturing. *Journal of Manufacturing Systems*, 56:326–340, 2020.

[119] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[120] H. Pohjanpalo. System identifiability based on the power series expansion of the solution. *Mathematical biosciences*, 41(1-2):21–33, 1978.

[121] B. Prakash, N. Waytowich, A. Ganesan, T. Oates, and T. Mohsenin. Guiding safe reinforcement learning policies using structured language constraints. *UMBC Student Collection*, 2020.

[122] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168*, 2020.

[123] A. Queiruga, N. B. Erichson, L. Hodgkinson, and M. W. Mahoney. Stateful ode-nets using basis function expansions. *Advances in Neural Information Processing Systems*, 34, 2021.

[124] P. Quintía Vidal, R. Iglesias Rodríguez, M. Á. Rodríguez González, and C. Vázquez Regueiro. Learning on real robots from experience and simple user feedback. 2013.

[125] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[126] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, 2017.

[127] A. Ray, J. Achiam, and D. Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. *arXiv preprint arXiv:1910.01708*, 2019.

[128] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. Meshsdf: Differentiable iso-surface extraction. *Advances in Neural Information Processing Systems*, 33:22468–22478, 2020.

[129] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[130] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8459–8468. PMLR, 2020.

[131] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. A. Riedmiller, R. Hadsell, and P. W. Battaglia. Graph networks as learnable physics engines for inference and control. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4467–4476. PMLR, 2018.

[132] C. Schenck and D. Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR, 2018.

[133] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org, 2015.

[134] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[135] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[136] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[137] D. Silver, R. S. Sutton, and M. Müller. Reinforcement learning of local shape in the game of go. In *Proceedings of International Joint Conferences on Artificial Intelligence*, volume 7, pages 1053–1058, 2007.

[138] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 903–910. Morgan Kaufmann, 2000.

[139] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

[140] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. In S. P. Singh and S. Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 2576–2582. AAAI Press, 2017.

[141] Y. Sui, A. Gotovos, J. W. Burdick, and A. Krause. Safe exploration for optimization with gaussian processes. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 997–1005. JMLR.org, 2015.

[142] W. Sun, G. J. Gordon, B. Boots, and J. A. Bagnell. Dual policy iteration. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31:*

Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 7059–7069, 2018.

[143] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

[144] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[145] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[146] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55, 2020.

[147] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In W. Burgard and D. Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

[148] C. Tessler, D. J. Mankowitz, and S. Mannor. Reward constrained policy optimization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[149] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg. Recovery rl: Safe reinforce-

ment learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021.

[150] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J. E. Gonzalez, S. Levine, F. Borrelli, and K. Goldberg. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *IEEE Robotics and Automation Letters*, 2020.

[151] S. Thapa, N. Li, and J. Ye. Dynamic fluid surface reconstruction using deep neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21–30, 2020.

[152] P. S. Thomas, B. C. da Silva, A. G. Barto, S. Giguere, Y. Brun, and E. Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.

[153] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. Vision-and-dialog navigation. In *Proceedings of Conference on Robot Learning*, pages 394–406, 2020.

[154] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins. Hamiltonian generative networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[155] M. Turchetta, F. Berkenkamp, and A. Krause. Safe exploration in finite markov decision processes with gaussian processes. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4305–4313, 2016.

[156] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal. Safe reinforcement learning via curriculum induction. *arXiv preprint arXiv:2006.12136*, 2020.

[157] K. Um, R. Brand, P. Holl, N. Thuerey, et al. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *arXiv preprint arXiv:2007.00016*, 2020.

[158] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[159] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[160] S. G. Venkatesh, A. Biswas, R. Upadrashta, V. Srinivasan, P. Talukdar, and B. Amrutur. Spatial reasoning from natural language instructions for robot manipulation. *arXiv preprint arXiv:2012.13693*, 2020.

[161] E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, C. Wu, F. Wu, R. Liaw, E. Liang, and A. M. Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Proceedings of Conference on Robot Learning*, pages 399–409, 2018.

[162] A. Vogel and D. Jurafsky. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Lin-*

guistics, pages 806–814, Uppsala, Sweden, 2010. Association for Computational Linguistics.

[163] A. Wachi and Y. Sui. Safe reinforcement learning in constrained markov decision processes. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9797–9806. PMLR, 2020.

[164] A. Walsman, Y. Bisk, S. Gabriel, D. Misra, Y. Artzi, Y. Choi, and D. Fox. Early fusion for goal directed robotic vision. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1025–1031. IEEE, 2019.

[165] C. Wang, C. Ross, Y.-L. Kuo, B. Katz, and A. Barbu. Learning a natural-language to ltl executable semantic parser for grounded robotics. *arXiv preprint arXiv:2008.03277*, 2020.

[166] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards physics-informed deep learning for turbulent flow prediction. In R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1457–1466. ACM, 2020.

[167] R. Wang, D. Maddix, C. Faloutsos, Y. Wang, and R. Yu. Bridging physics-based and data-driven modeling for learning dynamical systems. *arXiv preprint arXiv:2011.10616*, 2020.

[168] R. Wang, R. Walters, and R. Yu. Meta-learning dynamics forecasting using task inference. *arXiv preprint arXiv:2102.10271*, 2021.

[169] N. Watters, D. Zoran, T. Weber, P. W. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video.

In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4539–4547, 2017.

[170] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via visual de-animation. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 153–164, 2017.

[171] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

[172] T. Xue, A. Beatson, S. Adriaenssens, and R. P. Adams. Amortized finite element analysis for fast pde-constrained optimization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10638–10647. PMLR, 2020.

[173] T. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. Projection-based constrained policy optimization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[174] T.-Y. Yang, M. Hu, Y. Chow, P. J. Ramadge, and K. Narasimhan. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems*, 34, 2021.

[175] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. Accelerating safe reinforcement learning with constraint-mismatched policies. *International Conference on Machine Learning*, 2020.

[176] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan, and W. Yu. Safe reinforcement learning for legged locomotion. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[177] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[178] W. Yu, C. K. Liu, and G. Turk. Protective policy transfer. *arXiv preprint arXiv:2012.06662*, 2020.

[179] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, (2), 2020.

[180] J. Zhang, B. Cheung, C. Finn, S. Levine, and D. Jayaraman. Cautious adaptation for reinforcement learning in safety-critical settings. In *International Conference on Machine Learning*. PMLR, 2020.

[181] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone. Leveraging human guidance for deep reinforcement learning tasks. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6339–6346. ijcai.org, 2019.

[182] T. Zhao and M. Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*, 2016.

[183] Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[184] Y. D. Zhong, B. Dey, and A. Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. In *Learning for Dynamics and Control*, pages 1218–1229. PMLR, 2021.

[185] Y. D. Zhong, B. Dey, and A. Chakraborty. Extending lagrangian and hamiltonian neural networks with differentiable contact models. *Advances in Neural Information Processing Systems*, 34, 2021.

[186] Y. D. Zhong and N. Leonard. Unsupervised learning of lagrangian dynamics from images for prediction and control. *Advances in Neural Information Processing Systems*, 33, 2020.

[187] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.