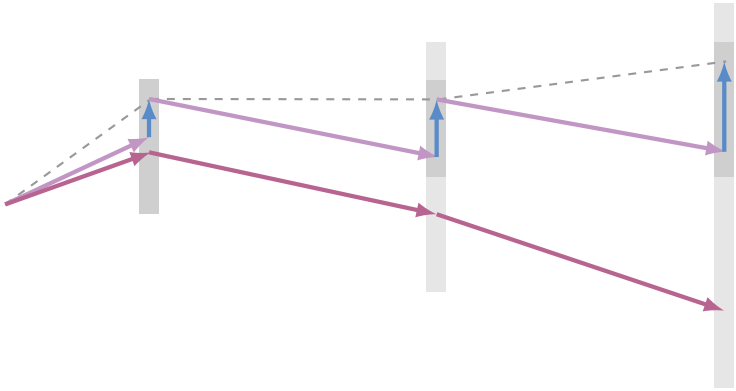
EPISTEMIC UNCERTAINTY FOR PRACTICAL DEEP MODEL-BASED REINFORCEMENT LEARNING

SEBASTIAN MARTIN CURI



DISS. ETH NO. 28649

DISS. ETH NO. 28649

EPISTEMIC UNCERTAINTY FOR PRACTICAL DEEP MODEL-BASED REINFORCEMENT LEARNING

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH
(Dr. Sc. ETH Zurich)

presented by

SEBASTIAN MARTIN CURI
M. Sc. in Robotics, Systems and Control, ETH Zurich

born on 2 July 1990
citizen of Argentina and Germany

accepted on the recommendation of

Prof. Dr. Andreas Krause, examiner
Prof. Dr. Melanie Zeilinger, co-examiner
Prof. Dr. Kfir Yehuda Levy, co-examiner

2022

To my family.

ABSTRACT

Reinforcement Learning (RL) has advanced the state-of-the-art in many applications in the last decade. The root of its success stems from having access to high-quality simulators, controlled environments, and massive compute power. Nonetheless, when the goal is to apply RL algorithms to real-world problems, many challenges remain unanswered. This dissertation focuses on three of them: data efficiency, robustness, and safety. On the one hand, practical algorithms that address these issues lack theoretical guarantees. On the other hand, theoretically-sound algorithms are impractical. This thesis aims to develop algorithms that achieve the best of both worlds. Namely, we propose *theoretically-sound* algorithms that can be *scaled* using state-of-the-art neural networks and are *easy to implement*. We take a model-based approach and learn models distinguishing between *aleatoric* and *epistemic* uncertainty. The former is uncertainty inherent to the system, such as sensor noise. In contrast, the latter stems from data scarcity, decreasing as we collect more data and expand our knowledge about the environment.

It is well-known that one needs to *plan using epistemic uncertainty* to achieve data-efficient exploration, robustness, and safety. Unfortunately, the algorithms that do so are impractical as they require optimizing over the set of plausible models. We *reparameterize* the set of plausible models to overcome this limitation. In particular, we add a *hallucinating control* policy that directly acts on the model's outputs and has as much authority as the epistemic uncertainty that the model affords. The reparameterization increases the action dimensions but reduces the intractable planning problem to one that standard RL algorithms can handle.

We first consider the problem of data-efficient exploration. In this setting, the objective is to find an *optimal policy* with *few interactions* with the environment. A theoretical approach to solve this problem is through optimism: an agent plans a policy using the most optimistic dynamics over the set of plausible models. Unfortunately, this requires *jointly optimizing policies and dynamics*, which is intractable. We propose the Hallucinated Upper Confidence RL (H-UCRL) algorithm. By augmenting the input space with the hallucinated inputs, we solve H-UCRL using *standard planners*. Hence, H-UCRL is practical while retaining its theoretical guarantees. In particular,

we show that H-UCRL attains near-optimal sample complexity guarantees, and we apply it to large-scale environments.

RL agents frequently encounter situations not present during training time in real-world tasks. The RL agents must exhibit *robustness* against worst-case situations to ensure *reliable* performance. The robust RL framework addresses this challenge via a worst-case optimization between an agent and an adversary. Previous robust RL algorithms are either sample inefficient, lack robustness guarantees, or do not scale to larger problems. We propose the Robust Hallucinated Upper-Confidence RL (RH-UCRL) algorithm to solve this problem *provably*. RH-UCRL combines *optimism with pessimism* when planning with the model to output a robust policy. Experimentally, we demonstrate that RH-UCRL outperforms other robust deep RL algorithms in various adversarial environments.

Finally, we address the problem of constraint satisfaction in RL. This challenge is crucial for the *safe deployment* of RL agents in real-world environments. We develop *confidence-based safety filters*, a control-theoretic approach for certifying state safety constraints for nominal policies learned via standard RL techniques. We *reformulate* state constraints in terms of *cost functions* to reduce safety verification to a standard RL task. The central idea of the safety filter is to filter the actions of the policy to ensure constraint satisfaction. The safety filter executes a *backup policy* when we cannot verify that the constraints are satisfied. This backup policy is assumed in most works, but we leverage the *hallucinating inputs* and *learn* the *backup policy* by solving a robust RL problem. We provide formal safety guarantees for the safety filter and empirically demonstrate the effectiveness of our approach.

ZUSAMMENFASSUNG

Das Verstärkungslernen (VL) hat in den letzten zehn Jahren in vielen Anwendungen den Stand der Technik vorangebracht. Die Grundlage seines Erfolgs liegt im Zugang zu sehr guten Simulatoren, kontrollierten Umgebungen und massiver Rechenleistung. Wenn es jedoch darum geht, VL-Algorithmen auf reale Probleme anzuwenden, bleiben viele Herausforderungen unbeantwortet. Diese Dissertation konzentriert sich auf drei davon: Dateneffizienz, Robustheit und Sicherheit. Einerseits fehlt es praktischen Algorithmen, die sich mit diesen Problemen befassen, an theoretischen Garantien. Andererseits sind theoretisch fundierte Algorithmen unpraktisch. Ziel dieser Arbeit ist es, Algorithmen zu entwickeln, die das Beste von beidem erreichen. Wir schlagen nämlich theoretisch fundierte Algorithmen vor, die mit modernsten neuronalen Netzen skaliert werden können und einfach zu implementieren sind. Wir gehen modellbasiert vor und lernen Modelle kennen, die zwischen aleatorischer und epistemischer Unsicherheit unterscheiden. Ersteres ist dem System innewohnende Unsicherheit, wie z. B. Sensorrauschen. Letzteres ist hingegen auf Datenknappheit zurückzuführen und nimmt ab, wenn wir mehr Daten sammeln und unser Wissen über die Umwelt erweitern.

Es ist bekannt, dass man epistemischer Unsicherheit modellieren muss, um eine dateneffiziente Exploration, Robustheit und Sicherheit zu erreichen. Leider sind die Algorithmen, die dies tun, unpraktisch, da sie eine Optimierung über alle plausibler Modelle erfordern. Wir schlagen daher eine Reparameterisierung aller plausiblen Modelle vor, um diese Einschränkung zu überwinden. Insbesondere nutzen wir eine “halluzinierende” Kontrollregel, die direkt auf die Ergebnisse des Modells wirkt und so viel Gewicht hat wie die epistemische Unsicherheit des Modells vorsieht. Insbesondere fügen wir eine halluzinierende Kontrollpolitik hinzu, die direkt auf die Ergebnisse des Modells wirkt und so viel Autorität hat wie die epistemische Unsicherheit, die das Modell bietet.

Wir betrachten zuerst das Problem der dateneffizienten Exploration. In diesem Umfeld ist das Ziel, eine optimale Regel mit wenigen Wechselwirkungen mit der Umwelt zu finden. Ein theoretischer Ansatz zur Lösung dieses Problems ist Optimismus: Ein Agent plant eine Richtlinie unter Verwendung der optimistischsten Dynamik gegenüber aller plausibler Modelle. Leider erfordert dies die gemeinsame Optimierung von Regeln und Dy-

namiken, was unlösbar ist. Deshalb schlagen wir den Hallucinated Upper Confidence RL (H-UCRL)-Algorithmus vor. Wir erweitern den Eingaberaum mit den halluzinierten Eingaben, und ermöglichen so, die Lösung von H-UCRL unter Verwendung von Standardplanern. Das macht ist H-UCRL praktisch, während es seine theoretischen Garantien behält. Insbesondere zeigen wir, dass H-UCRL nahezu optimale Stichprobenkomplexitätsgarantien erreicht, und wir wenden es auf groß angelegten Umgebungen an.

VL-Agenten begegnen häufig neuen Situationen, solchen die während der Trainingszeit in realen Aufgaben nicht vorhanden waren. Die VL-Agenten müssen Robustheit gegenüber ungünstigster Situationen aufweisen, um eine zuverlässige Leistung sicherzustellen. Das robuste VL-Framework begegnet dieser Herausforderung durch eine ungünstigster-Optimierung zwischen einem Agenten und einem Gegner. Frühere robuste VL-Algorithmen sind entweder probenineffizient, haben keine Robustheitsgarantien oder lassen sich nicht auf größere Probleme skalieren. Wir schlagen den Robust Hallucinated Upper-Confidence RL (RH-UCRL) Algorithmus vor, um dieses Problem nachweislich zu lösen. RH-UCRL kombiniert Optimismus mit Pessimismus bei der Planung mit dem Modell, um eine robuste Regel auszugeben. Experimentell zeigen wir, dass RH-UCRL andere robuste Deep-RL-Algorithmen in verschiedenen gegnerischen Umgebungen übertrifft.

Abschließend sprechen wir das Problem der Constraint-Erfüllung in VL an. Diese Herausforderung ist entscheidend für den sicheren Einsatz von VL-Agenten in realen Umgebungen. Wir entwickeln vertrauensbasierte Sicherheitsfilter, einen kontrolltheoretischen Ansatz zur Überprüfung von Sicherheitsbeschränkungen für nominale Regeln, die über Standard-VL-Techniken gelernt wurden. Wir formulieren Zustandsbeschränkungen in Bezug auf Kostenfunktionen um, um die Sicherheitsüberprüfung auf eine Standard-VL-Aufgabe zu reduzieren. Die zentrale Idee des Sicherheitsfilters besteht darin, die Aktionen der Regel zu filtern, um die Erfüllung von Einschränkungen sicherzustellen. Der Sicherheitsfilter führt eine Sicherungsregel aus, wenn wir nicht überprüfen können, ob die Einschränkungen erfüllt sind. Diese Sicherungsregel wird in den meisten Arbeiten angenommen, aber wir nutzen die halluzinierenden Eingaben und lernen die Sicherungsregel, indem wir ein robustes VL-Problem lösen. Wir stellen formelle Sicherheitsgarantien für den Sicherheitsfilter vor und weisen die Wirksamkeit unseres Ansatzes empirisch nach.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor Andreas Krause for his advice over the years. Thanks for feeding me with fantastic ideas, precise remarks, and critical thinking. But above all, I want to thank you for always making me feel valued inside the group and for building a cooperative and caring environment in which personal well-being is highly valued. I also want to thank Melanie Zeilinger and Kfir Levy for kindly serving as part of my Ph. D. committee.

I was extremely fortunate to have fantastic collaborators and friends throughout these years. I wanted to thank all members of the LAS group for their support and shared moments. I specially appreciate the discussions and work done together with Zalan Borsos, Mojmir Mutny, Armin Lederer, Felix Berkenkamp, Ilija Bogunovic, Johannes Kirschner, Ilnura Usmanova, Lenart Treven, Yarden As, Bhavya Sukhija, Scott Sussex, Max Paulus, and Matej Jusup. I also want to thank Joan Bas-Serrano and Gergely Neu for my time in Barcelona working with them, and Gabriel Dulac-Arnold and Marcin Andrychowi for my internship at Google Brain.

I am also grateful to have had the opportunity to work with many Bachelor's and Master's students. In particular, I wanted to thank Nuria Armengol Urpi, Marcello Fiducioso, Silvan Melchior, Yunshu Ouyang, Tristan Girard, Tobias Birchler for going the extra mile and writing publications beyond their respective projects. I also want to thank Jan Poland and Andrea Cortinovis for the collaborations with Hitachi and ABB that enabled several of these projects.

This endeavor would not have been possible without the support of institutions of funding agencies. I want to express my gratitude to the Department of Computer Science at ETH Zurich. I am happy to acknowledge the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation program grant "Reliable Data-Driven Decision Making in Cyber-Physical Systems (RADDICS)" agreement No. 815943 and to the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40180545.

Beyond academics, I wanted to thank the administrative staff at ETH, especially Rita, for all their help over the years. From a personal perspective, I first want to thank my family for their support and love. Without their focus on my academic education, I would most likely not be here today.

Next, I wanted to thank my group of friends in Zürich for being with me in good and bad times. I know I can count on you for the years to come. Por último, quiero agradecer a mis amigos de toda la vida dispersos por el mundo. Sé que a pesar de la distancia los llevo conmigo.

Martu, gracias por todo este tiempo juntos. Muchas gracias por mudarte a Zürich para que podamos hacer nuestra vida juntos cuando el COVID hizo imposible la distancia. Gracias por motivarme con el ejemplo y la palabra. Por acompañarme y apoyarme estos años, y por retarme cuando era necesario. Fuiste una guía este tiempo y estoy seguro de que seguirás siéndolo. Sos la mejor compañera de equipo que puedo soñar con tener.

CONTENTS

1

INTRODUCTION

Information is the resolution of uncertainty.
— Claude Shannon

Reinforcement Learning (RL) has become the state-of-the-art method for learning optimal control strategies by directly interacting with the environment. For example, RL algorithms surpassed human performance in Atari videogames (Mnih *et al.*, 2015), go and chess (Schrittwieser *et al.*, 2020), navigating stratospheric balloons (Bellemare *et al.*, 2020), and even controlling Fusion reactors (Degrave *et al.*, 2022). These success stories rely on a simulator to train the agents, as data collection is usually complex or expensive in real-world environments (Dulac-Arnold *et al.*, 2019).

Simulators are a way of doing Model-Based Reinforcement Learning (MBRL). MBRL can solve many challenging high-dimensional tasks with impressive sample efficiency (Chua *et al.*, 2018). These algorithms alternate between two phases: first, they collect data with a policy and fit a model to the data; then, they simulate transitions with the model and optimize the policy accordingly. A vital feature of the recent success of MBRL algorithms is the use of models that explicitly distinguish between *epistemic* and *aleatoric* uncertainty when learning a model (Gal, 2016). Aleatoric uncertainty is inherent to the system (noise), whereas epistemic uncertainty arises from data scarcity (Der Kiureghian & Ditlevsen, 2009). However, to optimize the policy, practical algorithms marginalize both the aleatoric and epistemic uncertainty to optimize the expected performance under the current model, as in PILCO (M. Deisenroth & Rasmussen, 2011).

This dissertation focuses on three central challenges outlined by Dulac-Arnold *et al.* (2019) when deploying Reinforcement Learning (RL) agents in real environments: *data efficiency, robustness, and safety*. Consider designing a braking system on an autonomous car as a motivating example. As the task of breaking is *highly complex*, we want to learn a policy that performs this maneuver. Testing an autonomous car is *expensive*; thereby, data efficiency is critical to reducing costs. Furthermore, the car is required to break in *many weather conditions* and with different laden weights; hence robustness to these changes is required to certify the proper behavior of the braking system. Finally, the car also needs to brake to *avoid collisions*, making safety

constraints essential to the deployment of the car in real-life. The central thesis of this dissertation is that an agent must plan using the models' epistemic uncertainty when tackling the three challenges mentioned earlier. Oppositely, marginalizing over both aleatoric and epistemic uncertainty fails in many scenarios.

Coming back to the braking system, learning a policy requires many exploratory trials, and one arrives at the well-known exploration-exploitation dilemma: the learning agent wants to execute a policy that has *high performance*, but to achieve this goal, it has to *learn about the dynamics*. Trading off these two objectives led to much research in recent years (Lattimore & Szepesvári, 2018). While optimistic exploration is a well-known solution for this dilemma, there is a lack of efficient, principled means of incorporating optimism in deep MBRL. On the other hand, there are many practical solvers for greedy exploitation of the current model. However, this algorithm can cause the optimization to get stuck in local minima, e.g., the agent might learn to brake aggressively and never explore other behaviors. Designing an algorithm that explores different behaviors efficiently and is compatible with deep RL is a central objective of this dissertation.

Even if various real-world conditions can be simulated during the training time, it is infeasible to consider *all* possible ones, such as road conditions, brightness, tire pressure, laden weight, or actuator wear, as these can all vary over time in potentially unpredictable ways. The main goal is to learn a policy that brakes to perform even if faced with new conditions *reliably*. While theoretical approaches for robust RL offer sample complexity guarantees, the existing algorithms are highly impractical. On the other hand, empirically motivated heuristic approaches lack provable robustness. The second objective is to develop an algorithm that outputs a provably robust policy, and it is not limited to linear dynamics.

Unlike in simulation, safety is a critical requirement in real environments. On the one hand, control theoretic methods consider safety through constraints on the system states but often *severely restrict* the allowed policy and dynamical system classes; thus, control methods might fail to model the complex behavior of the braking dynamics. On the other hand, RL methods enjoy better performance and may be applied to non-linear systems and policies. However, *constraint violation during training* cannot be excluded in general, which often prevents the usage in real-world applications, i.e., RL methods need to crash the car to learn about this behavior. The third objective is to develop an algorithm that achieves the best of both worlds by

using a safety filter. The idea is that we can learn a policy using RL methods and then filter the actions that the the policy suggests to ensure safety.

MODEL-BASED REINFORCEMENT LEARNING MBRL is a promising avenue toward applying RL methods to complex real-life decision problems due to its sample efficiency (M. P. Deisenroth *et al.*, 2013). For instance, Kaiser *et al.* (2019) use MBRL to solve the Atari suite, whereas Kamthe & M. Deisenroth (2018a) solve low-dimensional continuous-control problems using GP models and Chua *et al.* (2018) solve high-dimensional continuous-control problems using ensembles of probabilistic Neural Networks (NN). All these approaches perform *greedy exploitation* under the current model using a variant of PILCO (M. Deisenroth & Rasmussen, 2011). Unfortunately, greedy exploitation is *provably* optimal only in very limited cases such as linear quadratic regulators (LQR) (Mania *et al.*, 2019).

THOMPSON SAMPLING FOR EFFICIENT EXPLORATION *Thompson (posterior) sampling* and its variants are a common approach for *provable* exploration in reinforcement learning (Dearden *et al.*, 1999). In particular, Osband *et al.* (2013) propose Thompson sampling for *tabular* MDPs. Chowdhury & Gopalan (2019) prove a $\tilde{\mathcal{O}}(\sqrt{N})$ regret bound for continuous states and actions for this theoretical algorithm, where N is the number of episodes. However, Thompson sampling can be applied only when it is tractable to sample from the posterior distribution over dynamical models. For example, this is intractable for GP models with continuous domains. Moreover, Z. Wang *et al.* (2018) suggest that approximate inference methods may suffer from variance starvation and limited exploration.

OPTIMISM FOR EFFICIENT EXPLORATION The *Optimism-in-the-Face-of-Uncertainty* (OFU) principle is a classical approach towards *provable* exploration in the theory of RL. Notably, Brafman & Tennenholtz (2003) present the R-Max algorithm for *tabular* MDPs, where a learner is optimistic about the reward function and uses the *expected* dynamics to find a policy. R-Max has a sample complexity of $\mathcal{O}(1/\epsilon^3)$, which translates to a sub-optimal regret of $\tilde{\mathcal{O}}(N^{2/3})$. Jaksch *et al.* (2010) propose the UCRL algorithm that is optimistic on the transition dynamics and achieves an optimal $\tilde{\mathcal{O}}(\sqrt{N})$ regret rate for tabular MDPs. Recently, Zanette & Brunskill (2019), Efroni *et al.* (2019), and Domingues *et al.* (2020) provide refined UCRL algorithms for tabular MDPs. When the number of states and actions increase, these *tabular algorithms are inefficient* and practical algorithms must exploit the structure

of the problem. However, the use of optimism in continuous state/action MDPs is much less explored. Jin *et al.* (2019) present an optimistic algorithm for *linear* MDPs and Abbasi-Yadkori & Szepesvári (2011) for linear quadratic regulators (LQR), both achieving $\tilde{\mathcal{O}}(\sqrt{N})$ regret. Finally, Luo *et al.* (2018) propose a trust-region UCRL meta-algorithm that asymptotically finds an optimal policy, but it is intractable to implement.

Perhaps most closely related to the optimistic algorithm proposed in Chapter 3, Chowdhury & Gopalan (2019) present GP-UCRL for continuous state and action spaces. They use optimistic exploration for the policy optimization step with dynamical models that lie in a Reproducing Kernel Hilbert Space (RKHS). However, as mentioned by Chowdhury & Gopalan (2019), their algorithm is intractable to implement and cannot be used in practice. Instead, we build on an implementable but expensive strategy that was heuristically suggested by Moldovan *et al.* (2015) for planning on *deterministic* systems and develop a principled and highly efficient optimistic exploration approach for deep MBRL. Kakade *et al.* (2020) build tight confidence intervals based on information theoretical quantities. However, they assume an optimization oracle and do not provide a practical implementation (their experiments use Thompson sampling). Furthermore, in their experiments, Thompson Sampling fails unless strong priors are placed on the model classes. Ayoub *et al.* (2020) use value-targeted regression to learn a linear combination of tabular models and propose an efficient UCRL algorithm. Abeille & Lazaric (2020) propose an algorithm in the context of LQR and prove that the planning problem can be solved efficiently. In the same spirit, Neu & Pike-Burke (2020) reduce intractable optimistic exploration to greedy planning using well-selected reward bonuses under strong modeling assumptions. In particular, they prove an equivalence between optimistic reinforcement learning and exploration bonus (Azar *et al.*, 2017) for tabular and linear MDPs. How to generalize these exploration bonuses with neural network models is left for future work.

ROBUST REINFORCEMENT LEARNING Robust RL (Iyengar, 2005; Nilim & El Ghaoui, 2005; Wiesemann *et al.*, 2013) typically uses the Zero-Sum Markov Games formalism introduced by Littman (1994) and Littman & Szepesvári (1996). From a control engineering perspective, this is known as $\mathcal{H}_\infty$ control (Başar & Bernhard, 2008), and Bemporad *et al.* (2003) solve this problem for *known* linear systems where the optimal robust policy is an affine function of the state. For Markov Games with unknown systems, Lagoudakis & Parr (2002) introduce approximate value iteration whereas

Tamar *et al.* (2014) and Perolat *et al.* (2015) present an approximate policy iteration scheme. Although these works present error propagation schemes restricted to the linear setting, they do not address the problem of exploration explicitly and instead assume access to a sampling distribution that has sufficient state coverage. Zhang *et al.* (2020) propose a model-based algorithm for finding robust policies assuming access to a simulator that is able to sample at arbitrary state-action pairs. Finally, Bai & Jin (2020) recently introduce an algorithm that provably and efficiently outputs a policy, but it is limited to the tabular setting. In Chapter 4, we introduce RH-UCRL for this setting. Our approach does not require a generative model and considers the full exploration problem in a finite horizon scenario. Furthermore, our algorithm does not require tabular nor linear function approximation, and it is compatible with deep neural network dynamical models. In the bandit literature, Bogunovic *et al.* (2018) introduce STABLE-OPT, that generalizes the famous `GP-UCB` algorithm (Srinivas *et al.*, 2010) to the robust setting by combining pessimism with optimism. RH-UCRL uses similar ideas but in the RL setting.

MINIMAX OPTIMIZATION ALGORITHMS To solve robust RL problems, one needs to solve a minimax optimization algorithm. Pinto *et al.* (2017) propose to solve the minimax optimization via a stochastic gradient descent approach for both players for *adversarial-robust* RL algorithms. Tessler *et al.* (2019) introduce *action-robust* RL and policy and value iteration algorithms to solve these problems. Rajeswaran *et al.* (2017) introduce the `EPOpt` to solve *parameter-robust* problems. Finally, Kamalaruban *et al.* (2020) propose to use a Stochastic-Gradient Langevin Dynamics algorithm to solve such problems via sampling instead of optimization. These algorithms are generally sample-inefficient as they do not explicitly explore, but, given a model, they could be used to optimize a policy.

CONSTRAINED MARKOV DECISION PROCESSES Due to this high relevance of safety in reinforcement learning, it has been the focus of a variety of recent approaches (see (García & Fernández, 2015; Brunke *et al.*, 2021) for surveys). A common framework for safe reinforcement learning is constrained Markov decision processes (CMDP) (Altman, 1999). In the CMDP setting, constraints are posed on the expected cumulative cost along rollouts of a policy. This allows treating the cumulative cost analogously to rewards. Using this method, Achiam *et al.* (2017) adapt trust region policy optimization to maintain constraint satisfaction when initialized with a

safe policy. When no initially safe policy is known, Paternain *et al.* (2019) show that a Lagrangian relaxation asymptotically finds safe policies. Ding *et al.*, 2021 prove that this representation of the constrained optimization problem can also be combined with techniques such as UCRL to guarantee sublinear learning rates for linear CMDPs. However, constraint violation during training cannot be excluded in general, which often prevents its usage in real-world applications. Control theoretic methods consider safety through constraints on the system states but often severely restrict the allowed policy and system classes. For example, Dean *et al.* (2019) prove that linear quadratic regulators can be learned efficiently under polytopic constraints on the system states. The limitation to linear dynamics can be relaxed in the case of deterministic systems by employing model predictive control (MPC) techniques, such that the performance can be iteratively increased as shown by Rosolia & Borrelli (2018). Berkenkamp *et al.* (2017) learn Lyapunov stability regions using GP dynamics but assume access to a simulator. Koller *et al.* (2018) extends such work and combine MPC with reinforcement learning to allow active safe exploration.

SAFETY FILTERS To achieve the beneficial properties of both control theory and reinforcement learning approaches, Fisac *et al.* (2019) propose to employ reinforcement learning for finding the optimal policy, while in a second step a control method is used to certify the safety and, if necessary, adapt the applied action. We refer this two-step process as a safety filter. For example, Taylor *et al.* (2019) use control barrier functions to adapt the actions and (Bastani, 2021) use a "backup" policy, which is locally safe in some region of the state space While designing control barrier functions is challenging in general, determining locally safe policies often requires solving computationally expensive optimization problems on-line Bastani, 2021 or can only be applied to linearized systems Wabersich *et al.*, 2021. Therefore, the practical applicability of such safety filters in combination with highly flexible RL techniques is currently limited. In Chapter 5, we introduce H-UCSF, a safety filter algorithm that can be applied with non-linear dynamics and only requires a 1-step online optimization procedure.

1.1 CONTRIBUTIONS

In this dissertation, we focus on three central challenges when deploying Reinforcement Learning (RL) agents in real environments: data efficiency, robustness, and safety (Dulac-Arnold *et al.*, 2019). In particular, we show

that using the epistemic uncertainty that the agent has about the dynamics when optimizing a policy is *crucial* for the success of these agents.

Chapter 2 presents the main definitions of Model-Based Reinforcement Learning (MBRL). We present the model-learning aspects that we use in this dissertation and the policy optimization algorithms. The key contribution in this chapter is the concept of a *hallucinating policy*. The hallucinating policy is a method that reparameterizes the set of plausible models, similar to the famous reparameterization trick but over functions. Using this reparameterization trick, we reduce the problem of optimizing over the dynamics in a set of models to optimizing over a policy, where standard RL algorithms can be used. In particular, this trick enables the practical implementation of all the subsequent algorithms presented in this thesis.

In Chapter 3, we introduce *Hallucinated-UCRL* (H-UCRL), a novel optimistic MBRL algorithm, which can be applied together with state-of-the-art RL algorithms. In particular, we augment the control space of the agent with *hallucinated* control actions that directly control the agent's *epistemic* uncertainty about the 1-step ahead transition dynamics. We provide a general theoretical analysis for H-UCRL and prove sublinear regret bounds for the particular case of Gaussian Process (GP) dynamics models. Our key idea is to *reduce optimistic exploration to greedy exploitation* by reparameterizing the model space using the reparameterization trick introduced in the previous chapter. Finally, we evaluate H-UCRL in high-dimensional continuous control tasks that shed light on when optimistic exploration outperforms greedy exploitation and Thompson sampling. To the best of our knowledge, this is the first approach that successfully implements *optimistic* exploration with deep-MBRL.

In Chapter 4, we design *Robust Hallucinated-UCRL* (RH-UCRL), the first practical *provably* robust RL algorithm that is: (i) *sample-efficient*, (ii) compatible with *deep models*, and (iii) simulator-free as it addresses exploration on a real system. We establish rigorous general sample-complexity and regret guarantees for our algorithm, and we specialize them to Gaussian Process models, hence obtaining sublinear robust regret guarantees. A key algorithmic principle behind RH-UCRL is *hallucination*: In particular, the *agent* hallucinates an additional control input to *maximize* an *optimistic* estimate of the robust performance whereas the *adversary* hallucinates an additional control input to *minimize* a *pessimistic* estimate of the robust performance. The amount of "hallucination" is limited by the epistemic uncertainty of the model and it decreases as the learning algorithm collects more data. While previous robust RL works have focused on different individual settings,

we gather and summarize them all for the first time, and show particular instantiations of our algorithm in each of them: Adversarial-robust RL, Action-robust RL, and Parameter-robust RL. Finally, we provide experiments that include different environments and settings, and we empirically demonstrate that RH-UCRL outperforms or successfully competes with the state-of-the-art deep robust RL algorithms and other baselines.

In Chapter 5, we propose *Hallucinating Upper Confidence Safety Filters* (H-UCSF) for ensuring the safety of arbitrary policies applied to stochastic, nonlinear systems for which merely a model with high probability error bounds is known. To this end, we first establish a *relationship between state constraints and level sets of value functions*. Next, we show that these value functions can be efficiently estimated with standard reinforcement learning methods by optimizing the hallucinating policy. Our approach can be naturally extended to *finding safe policies*, by formulating it as a robust reinforcement learning problem. These safe policies can then be used for *computationally efficient online safety adaptation* of arbitrary reinforcement learning policies. We demonstrate the effectiveness of the proposed method on deep RL benchmark tasks.

1.1.1 *Publications and Collaborators*

This dissertation is, to large parts, based on the following publications and technical reports:

- **Curi, S.**, Berkenkamp, F., & Krause, A. *Efficient model-based reinforcement learning through optimistic policy search and planning* in *Advances in Neural Information Processing Systems (NeurIPS)* (2020).
- **Curi, S.**, Bogunovic, I., & Krause, A. *Combining Pessimism with Optimism for Robust and Efficient Model-Based Deep Reinforcement Learning* in *International Conference on Machine Learning (ICML)* (2021).
- **Curi, S.**, Lederer, A., Hirche, S., & Krause, A. *Safe Reinforcement Learning via Confidence-based Filters* in *IEEE 61th Annual Conference on Decision and Control (CDC)* (2022).

FURTHER PUBLICATIONS The following publications of the author and collaborators are more broadly relevant to the topic of this thesis but have not been directly included. The first set of papers investigates practical questions in model learning aspects that we present in Chapter 2. In particular, we develop in these papers algorithms for learning well-calibrated models

in partially observable systems with non-linear dynamics, and learning in unstable linear dynamical systems from a single trajectory, respectively:

- **Curi, S.**, Melchior, S., Berkenkamp, F, & Krause, A. *Structured Variational Inference in Partially Observable Unstable Gaussian Process State Space Models* in *Learning for Dynamics and Control (L4DC)* (2020).
- Treven, L., **Curi, S.**, Mutnỳ, M, & Krause, A. *Learning stabilizing controllers for unstable linear quadratic regulators from a single trajectory* in *Learning for Dynamics and Control (L4DC)* (2021).

In the second set of publications, we focused on practical aspects related to this thesis, including risk-averse decision making, learning-from-observations, learning from logged data, safe-exploration, provably efficient policy-optimization methods, and practical algorithms for constrained markov decission processes.

- **Curi, S.**, Levy, K. Y., & Krause, A. *Adaptive Input Estimation in Linear Dynamical Systems with Applications to Learning-from-Observations* in *IEEE 58th Conference on Decision and Control (CDC))* (2019).
- Fiducioso, M., **Curi, S.**, Schumacher, B., Gwerder, M, & Krause, A. *Safe contextual Bayesian optimization for sustainable room temperature PID control tuning* in *International Joint Conference on Artificial Intelligence (AAAI)* (2019).
- **Curi, S.**, Levy, K. Y., Jegelka, S., & Krause, A. *Adaptive sampling for stochastic risk-averse learning* in *Advances in Neural Information Processing Systems (NeurIPS)* (2020).
- Urpí, N., **Curi, S.**, & Krause, A. *Risk-Averse Offline Reinforcement Learning* in *International Conference on Learning Representations (ICLR)* (2021).
- Bas-Serrano, J., **Curi, S.** Krause, A., & Neu, G. *Logistic Q-learning* in *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2021).
- As, Y., Usmanova, I., **Curi, S.**, & Krause, A. *Constrained Policy Optimization via Bayesian World Models* in *International Conference on Learning Representations (ICLR)* (2022).

1.1.2 *Code Release*

Throughout this dissertation, we developed code associated with all publications. The main library developed is RL-Lib, hosted in `https://github.com/sebascuri/rllib`. RL-Lib is a pytorch-based (Paszke *et al.*, 2017) library with over 40 model-based and model-free algorithms implemented. Compared with other libraries, the main advantage is its ease of use and speed for prototyping new agents. Furthermore, it is one of the only open-source libraries with model-based implementations. For Chapter 3, we released `https://github.com/sebascuri/hucrl`, that is based on RL-Lib. Building upon these works, we released `https://github.com/sebascuri/rhucrl` with the algorithms and experiments in Chapter 4. Finally, the experiments and algorithms of Chapter 5 are hosted in `https://github.com/sebascuri/saferl`.

2

BACKGROUND

All models are wrong, but some are useful.
— George Box

In this chapter, we introduce the necessary background for this thesis. In Section 2.1, we present the basic notions and definitions used in the thesis, particularly the notion of Model-Based Reinforcement Learning (MBRL). In Section 2.2, we introduce the modelling techniques used in the thesis and prove basic properties of such models. Finally, in Section 2.3, we describe the model-based policy learning algorithms used in the thesis. All the papers in this dissertation use the notions developed in this chapter. In particular, many of the proofs in this chapter already appear in Berkenkamp (2019), Curi *et al.* (2020a), and Curi *et al.* (2021).

2.1 MODEL-BASED REINFORCEMENT LEARNING

DYNAMICS We consider a stochastic environment with states $\mathbf{s} \in \mathcal{S} \subseteq \mathbb{R}^{d_s}$, actions $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}^{d_a}$ within a compact set $\mathcal{A}$, and *i.i.d.*, additive transition noise $\boldsymbol{\omega}_h \in \mathbb{R}^{d_s}$. The resulting transition dynamics are

$$\mathbf{s}_{h+1} = f(\mathbf{s}_h, \mathbf{a}_h) + \boldsymbol{\omega}_h \tag{2.1}$$

with $f \colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$.

We make the following assumptions throughout the dissertation.

Assumption 1 (System Lipschitz Continuity)**.** The dynamics f in Equation (2.1) are L_f-Lipschitz continuous.

Assumption 2 (Noise is σ-sub-Gaussian.)**.** For all $h \in \{0, \dots, H-1\}$, the individual entries of the noise vector $\boldsymbol{\omega}_h$ are *i.i.d.* σ-sub-Gaussian and zero-mean, i.e., $\mathbb{E}_{\boldsymbol{\omega}}[\boldsymbol{\omega}] = 0$.

Assumption 3 (State Observation)**.** The state $\mathbf{s}$ is measured directly without observation noise.

For the sake of notational simplicity, we refer to $\mathbf{z} \coloneqq (\mathbf{s}, \mathbf{a})$ as the concatenated state-action vector and $\mathcal{Z} \coloneqq \mathcal{S} \times \mathcal{A}$ as the state-action space.

POLICY In this thesis, we use Markovian stationary policies, that are mappings from states to distributions over actions $\pi\colon \mathcal{S} \to \Delta(\mathcal{A})$. For deterministic policies, the distribution is simply the δ-dirac distirbution at the policy output.

Assumption 4 (Lipschitz continuity of policy)**.** Any policy $\pi \in \Pi$ is L_π-Lipschitz continuous.

This assumption is satisfied as we are able to choose the policy class Π that we want to optimize. In particular, we choose the class of L_π-Lipschitz continuous functions.

PERFORMANCE At every timestep, the agent receives a reward that depends on the current state and action. In the reinforcement learning literature there are many ways of formalizing the agent objective (Puterman, 2014). In this thesis, we consider the undiscounted finite-horizon objective and the discounted infinite-horizon discounted objective.

For any dynamical model $\tilde{f}\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ (e.g., the true dynamics f in eq. (2.1)), the **undiscounted finite-horizon performance** of a policy π is the total reward collected during an episode in expectation,

$$J(\tilde{f}, \pi) = \mathbb{E}_{\tau_{\tilde{f},\pi}}\left[\sum_{h=0}^{H} r(\tilde{\mathbf{s}}_h, \pi(\tilde{\mathbf{s}}_h))\right], \tag{2.2a}$$

$$\text{s.t. } \tilde{\mathbf{s}}_{h+1} = \tilde{f}(\tilde{\mathbf{s}}_h, \pi(\tilde{\mathbf{s}}_h)) + \tilde{\boldsymbol{\omega}}_h, \tag{2.2b}$$

$$\tilde{\mathbf{s}}_0 \sim \nu_0, \tag{2.2c}$$

where $\tau_{\tilde{f},\pi} = \{(\mathbf{s}_{h-1}, \mathbf{a}_{h-1}), \mathbf{s}_h\}_{h=0}^{H}$ is a random trajectory induced by the stochastic noise $\tilde{\boldsymbol{\omega}}$, the dynamics $\tilde{f}$, the policy π, and the initial state distribution ν_0.

Likewise, the **discounted infinite-horizon performance** is

$$J(\tilde{f}, \pi) = \mathbb{E}_{\tau_{\tilde{f},\pi}}\left[\sum_{h=0}^{\infty} \gamma^h r(\tilde{\mathbf{s}}_h, \pi(\tilde{\mathbf{s}}_h))\right], \tag{2.3a}$$

$$\text{s.t. } \tilde{\mathbf{s}}_{h+1} = \tilde{f}(\tilde{\mathbf{s}}_h, \pi(\tilde{\mathbf{s}}_h)) + \tilde{\boldsymbol{\omega}}_h, \tag{2.3b}$$

$$\tilde{\mathbf{s}}_0 \sim \nu_0. \tag{2.3c}$$

In this dissertation, we focus on finding a policy that maximizes the undiscounted finite-horizon performance measure (2.2) on the true *unknown* dynamics f from Equation (2.1). However standard RL algorithms are

designed to solve the discounted infinite-horizon performance measure (2.3). This is solved by selecting the discount factor as $\gamma = 1 - \frac{1}{H}$ (Kocsis & Szepesvári, 2006; Browne *et al.*, 2012; Jiang *et al.*, 2015).

Assumption 5 (Lipschitz continuity of rewards)**.** The reward function $r\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is L_r-Lipschitz continuous.

Although discontinuous reward functions are natural in some goal-reaching setting, optimizing policies for such functions is hard and a continuous relaxation is usually used (Tassa *et al.*, 2018).

When the reward function is unknown, the techniques developed in this dissertation still hold. Namely, one can also *learn* the reward function with the same model-learning techniques use for learning dynamical system. A simple way of doing so is increasing the dimension of the regression target by one, such that the model $\tilde{f}$ must output $\tilde{f}(\tilde{\mathbf{s}}_h, \tilde{\mathbf{a}}_h) = [\tilde{\mathbf{s}}_{h+1}, r_h]$.

2.1.1 *Approximate Dynamic Programming*

The main technique to estimate the performance (2.3) of a policy is through (approximate) dynamic programming (Bertsekas *et al.*, 1995; Sutton & Barto, 2018; Szepesvári, 2010; Puterman, 2014).

POLICY EVALUATION The key first quantity that we consider is the value function of a policy π, which is defined as the expected discounted sum of rewards starting from state $\mathbf{s}$ following the policy π, i.e.:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}_{\tau_{f,\pi}}\left[\sum_{h=0}^{\infty} \gamma^h r(\mathbf{s}_h, \mathbf{a}_h) \mid \mathbf{s}_0 = \mathbf{s}\right] \tag{2.4a}$$

$$= \mathbb{E}_{\pi}[r(\mathbf{s}_0, \mathbf{a}_0)] + \mathbb{E}_{\tau_{f,\pi}}\left[\sum_{h=1}^{\infty} \gamma^h r(\mathbf{s}_h, \mathbf{a}_h) \mid \mathbf{s}_1 = \mathbf{s}'\right], \tag{2.4b}$$

$$= \mathbb{E}_{\pi}[r(\mathbf{s}_0, \mathbf{a}_0)] + \gamma \mathbb{E}_{\tau_{f,\pi}}\left[\sum_{h=0}^{\infty} \gamma^h r(\mathbf{s}_h, \mathbf{a}_h) \mid \mathbf{s}_0 = \mathbf{s}'\right], \tag{2.4c}$$

$$= \mathbb{E}_{\pi}[r(\mathbf{s}, \pi(\mathbf{a}))] + \gamma \mathbb{E}_{\tau_{f,\pi}}\left[V^{\pi}(\mathbf{s}')\right]. \tag{2.4d}$$

Equation (2.4d) is the so called Bellman equation for policy evaluation.

The main idea in approximate dynamic programming is to learn the function V^{π}. When the dynamics f is known, this problem reduces to value

evaluation (Sutton & Barto, 2018; Bertsekas *et al.*, 1995). When the dynamics f is unknown, commonly a parametric approximation of $V^\pi(\cdot;\vartheta)$ is used.

To learn the parameters ϑ, the empirical loss function loss function

$$\mathcal{L}(\vartheta) = \mathbb{E}_{\mathbf{s}\sim\mathbb{P}[\mathbf{s}]}[d(V^\pi(\mathbf{s};\vartheta), \hat{y})], \tag{2.5}$$

is minimized, where d is a distance metric such as the $l2$ distance, $\mathbb{P}[\mathbf{s}]$ is a probability distribution over states, and $\hat{y}$ is a regression target that is computed from data. For example, $\hat{y}$ could be a a Monte-Carlo estimation of the value in Equation (2.4a). However, it is more common to use the TD-learning algorithm (Tesauro, 1994), which uses the Bellman equation as the regression target and bootstraps the value of the next state, i.e. $\hat{y} = r + V^\pi(\mathbf{s}';\vartheta')$, where ϑ' is a copy of the current parameters. Munos & Szepesvári (2008) analyze the sample complexity rates of this algorithm. The missing component is to specify the state distribution $\mathbb{P}[\mathbf{s}]$ in the learning loss (2.5). When the states are sampled from the dynamics using the policy π, then the expectation is replaced by a sum and the targets are called *on-policy*. When the states are sampled from another distribution these are called *off-policy* and many techniques exist to correct for biases in the target estimation (Munos *et al.*, 2016).

The second quantity of interest is the Q-function of a policy π, which is defined as the expected discounted sum of rewards starting from state $\mathbf{s}$ and action $\mathbf{a}$, and following the policy π thereafter, i.e.:

$$\begin{aligned}
Q^\pi(\mathbf{s},\mathbf{a}) &= \mathbb{E}_{\tau_{f,\pi}}\left[\sum_{h=0}^{\infty}\gamma^h r(\mathbf{s}_h,\mathbf{a}_h) \mid \mathbf{s}_0=\mathbf{s}, \mathbf{a}_0=\mathbf{a}\right] && \text{(2.6a)}\\
&= r(\mathbf{s}_0,\mathbf{a}_0) + \mathbb{E}_{\tau_{f,\pi}}\left[\sum_{h=1}^{\infty}\gamma^h r(\mathbf{s},\mathbf{a}) \mid \mathbf{s}_1=\mathbf{s}'\right], && \text{(2.6b)}\\
&= r(\mathbf{s}_0,\mathbf{a}_0) + \gamma\mathbb{E}_{\tau_{f,\pi}}\left[\sum_{h=0}^{\infty}\gamma^h r(\mathbf{s}_h,\mathbf{a}_h) \mid \mathbf{s}_0=\mathbf{s}'\right], && \text{(2.6c)}\\
&= r(\mathbf{s}_0,\mathbf{a}_0) + \gamma\mathbb{E}_{\tau_{f,\pi}}\left[V^\pi(\mathbf{s}')\right]. && \text{(2.6d)}
\end{aligned}$$

The Q-function has similar properties to the value function, in particular the Bellman property (2.6d). Thus, the techniques for for learning parametric Q-functions $Q^\pi(\cdot;\vartheta)$ are similar to value function learning.

POLICY IMPROVEMENT The optimal policy π^*, the optimal value function $V^* = V^{\pi^*}$, and the optimal Q-function $Q^* = Q^{\pi^*}$ satisfy the Bellman optimality principle (Sutton & Barto, 2018):

$$Q^*(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}') \tag{2.7a}$$

$$V^*(\mathbf{s}) = \max_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}). \tag{2.7b}$$

SARSA (Rummery & Niranjan, 1994) and Q-Learning (Watkins & Dayan, 1992) are algorithms that directly aim to learn $Q^*(\cdot; \vartheta)$ using on-policy or off-policy samples, respectively. The main idea is that they use the loss function (2.5) with target $\hat{y} = r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}'; \vartheta')$. However, these algorithms require to solve $\max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}')$ when computing the td-targets. When the action space is large, this optimization is intractable. Furthermore, as the Q-function is learned, the value of $Q(\mathbf{s}', \mathbf{a}')$ can be arbitrarily wrong for state-action pairs with no visitation, leading to maximization biases (Van Hasselt *et al.*, 2016)

Instead, one could hope to improve a parametric policy $\pi(\cdot; \theta)$ in an incremental fashion leveraging the policy evaluation step. In particular, Q-functions satisfy $\mathbb{E}_\pi[Q]^\pi(\mathbf{s}, \mathbf{a}) = V^\pi(\mathbf{s})$. Hence, if there are actions in which $Q^\pi(\mathbf{s}, \mathbf{a}) > V^\pi(\mathbf{s})$, then executing the action $\mathbf{a}$ at state $\mathbf{s}$ has higher expected returns than executing the action predicted by the policy π. The vanilla policy improvement equation is:

$$\max_\theta \mathbb{E}_{\mathbf{s} \sim \mathbb{P}[\mathbf{s}], \mathbf{a} \sim \pi}[Q^\pi(\mathbf{s}, \mathbf{a})]. \tag{2.8}$$

Where the goal is to optimize the policy parameters θ to maximize the estimate of the value function. REINFORCE (R. J. Williams, 1992) is an on-policy algorithm that estimates $Q^\pi(\mathbf{s}, \mathbf{a})$ with an on-policy Monte Carlo estimate and uses score-function gradient estimation (Mohamed *et al.*, 2019) to improve θ. The Policy Gradient algorithm by Sutton *et al.* (1999) also use the same score-function gradient estimation, but uses a Q^π function that is learned with td-learning. To reduce variance in the gradients, the advantage $A^\pi(\mathbf{s}, \mathbf{a}) = Q^\pi(\mathbf{s}, \mathbf{a}) - V^\pi(\mathbf{a})$ is used and the reader is referred to Schulman *et al.* (2015c) for advanced techniques on learning the advantage. CPI (Kakade & Langford, 2002) and TRPO (Schulman *et al.*, 2015a) use the same ideas but introduce a regularization term that limits the policy improvement step to increase stability in the optimization algorithm.

Another common technique is to use pathwise gradients instead of score-function gradients (Mohamed *et al.*, 2019) to maximize (2.8). DPG (Silver

Algorithm 1 Learning Protocol

Inputs: True dynamics f, horizon H, initial state distribution ν_0, number of episodes N.

1: Initialize dataset $\mathcal{D}_0 = \emptyset$.
2: Initialize statistical dynamical model.
3: **for** Episode $n = 1, 2, \ldots,$ N **do**
4: Select policy π_n.
5: Reset the system to $\mathbf{s}_{0,n} \sim \nu_0$.
6: Rollout dynamics:
7: **for** Timestep $h = 1, \ldots, H$ **do**
8: $\mathbf{a}_{h-1,n} = \pi_n(\mathbf{s}_{h-1,n})$
9: $\mathbf{s}_{h,n} = f(\mathbf{s}_{h-1,n}, \mathbf{a}_{h-1,n}) + \boldsymbol{\omega}_{h-1,n}$
10: Collect transition $\mathcal{D}_n = \mathcal{D}_n \cup \{(\mathbf{s}_{h-1,n}, \mathbf{a}_{h-1,n}), \mathbf{s}_{h,n}\}$
11: Update statistical dynamical model with the H transitions in $\mathcal{D}_n$.

et al., 2014), DDPG (Lillicrap *et al.*, 2015), TD3 (Fujimoto *et al.*, 2018) directly compute the gradient through of the policy parameters through the learned Q-function, omitting the second order term that arises from the change in the Q-function (Degris *et al.*, 2012). SAC (Haarnoja *et al.*, 2018) also use such gradient estimation and adds an entropy regularization term for increased stability.

2.1.2 *Episodic Learning Protocol*

In this thesis, we consider an episodic learning protocol following Algorithm 1. At the beginning of each episode n, the agent selects a policy π_n. Then, it executes such policy and collects transitions into a dataset $\mathcal{D}_n = \{(\mathbf{s}_{h-1,n}, \mathbf{a}_{h-1,n}), \mathbf{s}_{h,n}\}_{h=1}^{H}$. Before the beginning of the next episode, the agent updates the model using the data.

The main contribution of this dissertation is how to select the policy π_n in line 4 of Algorithm 1 to achieve the different objectives described in the introduction, namely data-efficiency, robustness, and safety.

2.2 MODEL LEARNING

The main idea of MBRL is to use a model to simulate state trajectories and use the simulated data to learn a policy via planning. Thus, a crucial part of any good MBRL algorithm is the model learning component. In all

chapters in this thesis, we use the model learning techniques described in this section. In particular, we present Gaussian Process (GP) and Neural Network (NN) models. GP models are better understood and we can prove properties such as calibration, concentration, and learning complexity rates. NN models are less understood but have better performance in large-scale settings.

The rest of the section reads as follows. In Section 2.2.1, we describe how we parameterize the models for the dynamics (2.1) using mean and variance functions. In Section 2.2.2, we introduce the notion of model complexity that we use throughout the dissertation. In Section 2.2.3, we instantiate these functions using GP models and prove various properties of these kinds of models. Next, we also instantiate these functions using NN models and describe the training and calibration procedures. Finally, in Section 2.2.5, we reparameterize the set of plausible models and introduce the key concept of hallucination.

2.2.1 *Model Parameterization*

Throughout this disseration, we use statistical estimation to probabilistically reason about dynamical models $\tilde{f}$ that are compatible with the observed data $\mathcal{D}_{1:n} = \left\{\tau_{f,\pi_{n'}}\right\}_{n'=1}^{n}$. This can be done, e.g., by frequentist estimation of mean $\boldsymbol{\mu}_n(\mathbf{z})$ and confidence $\boldsymbol{\Sigma}_n(\mathbf{z})$ estimators, or by taking a Bayesian perspective and considering the posterior distribution $\mathcal{F}$ over dynamical models that leads to $\boldsymbol{\mu}_n(\mathbf{z}) = \mathbb{E}_{\tilde{f}\sim\mathcal{F}}\left[\tilde{f}(\mathbf{z})\right]$ and $\boldsymbol{\Sigma}_n(\mathbf{z}) = \mathrm{Var}_{\tilde{f}\sim\mathcal{F}}\left[\tilde{f}(\mathbf{z})\right]$.

ALEATORIC VS. EPISTEMIC UNCERTAINTY Either using a frequentist or a Bayesian perspective, we learn a model parameterized with a mean prediction $\boldsymbol{\mu}$, epistemic uncertainty $\boldsymbol{\Sigma}$, and aleatoric uncertainty v. We show an illustration of our model in Figure 2.1. We distinguish aleatoric and epistemic uncertainty following Der Kiureghian & Ditlevsen (2009). On one hand, aleatoric uncertainty is inherent to the system (noise) and is used to model, for instance, sensor or actuator noise $\boldsymbol{\omega}$ in the transition dynamics (2.1). This is captured in the model by v. Following Assumption 2 we parameterize the aleatoric uncertainty using a normal distribution $\mathcal{N}(0, v^2)$, but we use v to refer to this distribution. This uncertainty is *irreducible*: if we were to repeat the experiment any number of times, we will get different results and we are not able to decrease the uncertainty about the next state. On the other hand, epistemic uncertainty arises from data scarcity and models the uncertainty that the agent has about the

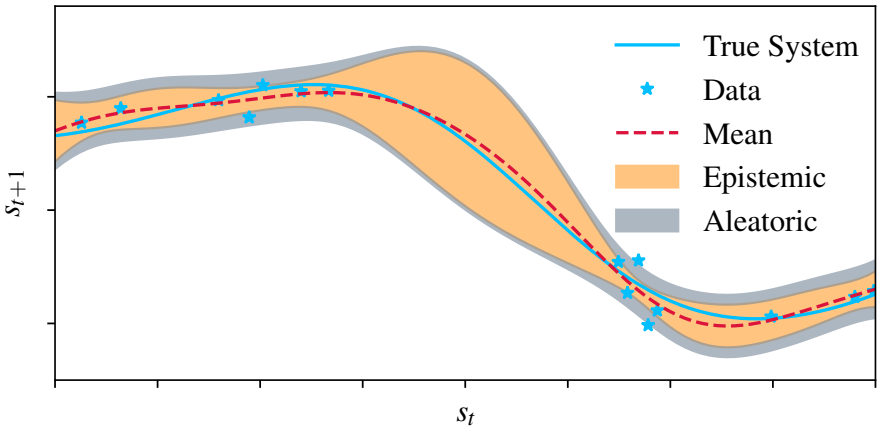


FIGURE 2.1: Illustration of the model used in this dissertation. We show the true dynamics that generates data in a solid light blue line and the noisy data points collected by the agent with light blue stars. In dashed red, we plot the mean predictions $\boldsymbol{\mu}$ of our model. We shade in orange the epistemic uncertainty σ and in gray the aleatoric uncertainty v. Where there is more data the epistemic uncertainty contracts. The aleatoric uncertainty is constant throughout the domain as we consider homoscedastic noise. Under Assumption 7, the true dynamics f is contained within the orange shaded region.

transition dynamics f. This is captured by $\boldsymbol{\sigma} = \text{diag}(\boldsymbol{\Sigma}^{1/2})$ and it represent the uncertainty of not knowing the dynamics f a priori. This uncertainty instead is *reducible*: if we were to repeat the experiment, the uncertainty about the dynamics f reduces as we explore and collect additional data.

PLAUSIBLE MODELS Given such a learned model, we define the **set of plausible models** as

$$\mathcal{M}_\beta := \left\{ \tilde{f} \quad \text{s.t.} \ \left| \tilde{f}(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z}) \right| \leq \beta \boldsymbol{\sigma}(\mathbf{z}) \right\}, \tag{2.9}$$

where $\boldsymbol{\sigma} = \text{diag}(\boldsymbol{\Sigma}^{1/2})$, and β is a calibration parameter. This set of models $\mathcal{M}_\beta$ describe the possible dynamics $\tilde{f}$ that is compatible with the data in a probabilistic sense. As $\beta \to 0$, only the mean function is a plausible model, whereas as $\beta \to \infty$ the set of plausible models increases. When the calibration parameter β, the model mean $\boldsymbol{\mu}$, and the model variance $\boldsymbol{\Sigma}$ depend on the episode n, then we use the subscript n instead of β to denote the set of plausible models $\mathcal{M}_{\beta_n} \equiv \mathcal{M}_n$.

We can use any $\tilde{f} \in \mathcal{M}_\beta$ to simulate a trajectory. In particular, given a starting state $\tilde{\mathbf{s}}_0$, a policy π, and a dynamical model $\tilde{f}$, the states are propagated according to:

$$\tilde{\mathbf{s}}_{h+1} = \tilde{f}(\tilde{\mathbf{z}}_h) + \tilde{\omega}_h, \tag{2.10}$$

where $\tilde{\mathbf{a}}_h \sim \pi(\tilde{\mathbf{s}}_h)$ and $\tilde{\omega}_h \sim \mathcal{N}(0, v^2)$.

As the dynamics is Lipschitz continuous due to Assumption 1, we only consider Lipschitz continuous models, which we formalize with the following assumption.

Assumption 6 (Continuity of model predictions)**.** The functions $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are $L_{\boldsymbol{\mu}}$ and $L_{\boldsymbol{\sigma}}$ Lipschitz continuous.

Common models such as GP models and NN models with Lipschitz continuous non-linearities are Lipschitz continuous.

Finally, we make the assumption that the model predictive uncertainty $\boldsymbol{\Sigma}$ is *large enough* in the sense that the confidence intervals that the model induces are *calibrated*.

Assumption 7 (Calibrated model)**.** The statistical model is *calibrated* w.r.t. f in Equation (2.1), so that with $\boldsymbol{\sigma}_n(\cdot) = \operatorname{diag}(\boldsymbol{\Sigma}_n(\cdot))$ and a non-decreasing sequence of parameters $\{\beta_n\}_{n\geq 1} \in \mathbb{R}_{>0}$, each depending on $\delta \in (0,1)$, it holds jointly for all $n \geq 1$ and $\mathbf{z} \in \mathcal{Z}$ that $|f(\mathbf{z}) - \boldsymbol{\mu}_{n-1}(\mathbf{z})| \leq \beta_n \boldsymbol{\sigma}_{n-1}(\mathbf{z})$ element-wise, with probability at least $1 - \delta$.

This assumption implies that the true dynamics $f \in \mathcal{M}_{\beta_n}$ for all n and it is crucial to reason about the uncertainty: if the model is not calibrated, then using the epistemic uncertainty of such a model will not *provably* guide exploration nor ensure safety nor robustness.

In upcoming sections, we introduce two sets of model families: GP and NN models. For dynamics with finite norm in a known RKHS space or that are sampled from a GP prior, we can use GP models, where Assumptions 6 and 7 are satisfied under technical conditions. For NN models, we do not have such guarantees for Assumption 7, but we can recalibrate one-step ahead predictions (Malik *et al.*, 2019). For NN models, Assumption 6 is satisfied by selecting Lipschitz continuous non-linearities.

2.2.2 *Model Complexity*

The sample complexity rates that we analyze in this dissertation depend on the difficulty of learning the underlying statistical model. Models that

are easy to learn typically require fewer samples and allow algorithms to make better decisions sooner. To express the difficulty of learning the imposed calibrated model class, we use the following model-based complexity measure:

$$\Gamma_N := \max_{\tilde{\mathcal{D}}_{1:N}} \sum_{n=1}^{N} \sum_{(\mathbf{z}) \in \tilde{\mathcal{D}}_n} \|\sigma_{n-1}(\mathbf{z})\|_2^2 \tag{2.11}$$

where each $\tilde{\mathcal{D}}_n \subset \{\mathcal{Z}\}^H$. This quantity has a worst-case flavor as it considers the data (collected during N episodes by any algorithm) that lead to maximal total predictive uncertainty of the model. For the special case of RKHS/GP dynamics models, we show below that this quantity can be effectively bounded, and the bound is sublinear (in the number of episodes N) for most commonly used kernel functions.

2.2.3 *Gaussian Process Models*

In this section, we formalize the setting in which the true dynamics f in Equation (2.1) has bounded norm in a Reproducing Kernel Hilbert Space (RKHS) induced by a continuous, symmetric positive definite kernel function $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. We denote by $\mathcal{K}$ the corresponding RKHS. Having a norm $\|f\|_{\mathcal{K}} \leq B_f$ for some finite $B_f > 0$ means that the RKHS is well-suited for capturing f (Durand *et al.*, 2018). The same results hold in a Bayesian setting where f is sampled from a GP $f \sim \mathcal{GP}(\cdot, k)$.

We first bound the model complexity. Next, we show that as the number of data points increases, the set of plausible models that the GP models induce contracts towards the true dynamics. Furthermore, we show that the set of plausible models satisfies Assumption 7, i.e., the model is well-calibrated. Finally, we show that the model posterior mean and covariance functions are Lipschitz continuous.

2.2.3.1 *Bounding the Model Complexity*

Most of the bounds on model complexity for sequential learning of GP models are based on regression to single dimensional targets (Srinivas *et al.*, 2010). However, due to the episodic nature of the problem, we follow the batch analysis from Desautels *et al.* (2014) and we need to generalize it to the RL setting, where multiple outputs are required. In particular, we observe H transitions per episode and at the beginning of each episode we

use the model to make decisions for other H steps. To extend to multiple outputs a possibility is to use multi-output GP kernels (Alvarez *et al.*, 2012). However, we simply build d_s copies of the dataset such that $\mathcal{D}_{1:h,i} = \left\{\left(\mathbf{z}_{n',h}, \mathbf{s}_{n',h+1,i}\right)\right\}_{h=0,n'=1}^{H-1,n}$, each with nH transitions. I.e., the i-th dataset has as covariates the state-action and as target the i-th coordinate of the next-state. We denote the covariates $z_{n,h} \equiv (\mathbf{s}_{n,h}, \mathbf{a}_{n,h})$ and the targets as $y_{n,h,i} \equiv \mathbf{s}_{n',h+1,i}$.

Finally, we build d_s models as

$$\mu_n(z, i) = k_n(z)^\top (K_n + \lambda I)^{-1} y_{1:Hn,i}, \tag{2.12a}$$

$$k_n(z, z', i) = k(z, z') - k_n(z)^\top (K_n + \lambda I)^{-1} k_n(z'), \tag{2.12b}$$

$$\Sigma_n(z, i) = k_n(z, z), \tag{2.12c}$$

where $\mathbf{s}'_{1:Hn,i}$ is the column vector of the i-th coordinate of all the next-states in the dataset, K_n is the kernel matrix, I is the identity matrix of appropriate dimension and we use $\lambda = d_s H$. This is stronger than the $\lambda = H$ from Desautels *et al.* (2014), and we need this as the same data is used in all the d_s models.

Stacking together the posterior mean and variance into column vectors we get:

$$\boldsymbol{\mu}_n(z) = \left[\mu_n(z, 1), \ldots, \mu_n(z, d_s)\right]^\top, \tag{2.12d}$$

$$\boldsymbol{\Sigma}_n(z) = \left[\Sigma_n(z, 1), \ldots, \Sigma_n(z, d_s)\right]^\top. \tag{2.12e}$$

Definition 1 (Information Gain (Cover & Thomas, 1991; Srinivas *et al.*, 2012; Durand *et al.*, 2018))**.** The information gain is the mutual information between the true function f and a set of observations at locations $Z = \{z_1, \ldots, z_n\}$. Hence, it is the difference between the entropy of such observations and the conditional entropy of the observations given function values i.e.,

$$I(f_Z; y_Z) = H(y_Z) - H(y_Z | f_Z), \tag{2.13a}$$

where f_Z is the noise-free evaluation of f at locations Z and y_Z is the noisy observation. In the case of GP models as in Equation (2.12), the information gain is:

$$I(f_Z; y_Z) = \frac{1}{2} \sum_{k=1}^{n} \ln(1 + \lambda^{-1} \sigma_{k-1}^2(z_k)). \tag{2.13b}$$

Next, we introduce the maximum information gain, which is a parameter that quantifies how hard the learning problem is and tightly upper bounds the *effective-dimensionality* of the problem (Valko *et al.*, 2013).

Definition 2 (Maximum Information Gain (Srinivas *et al.*, 2012))**.** The maximum information gain is the maximum of the information gain, taken over all datasets with a fixed size n, i.e.,

$$\gamma_n(k; Z) \coloneqq \max_{Z \subset \mathcal{Z}, |Z|=n} I(f_z; y_z). \tag{2.14a}$$

In the particular case of GP models, this reduces to:

$$\gamma_n(f; Z) = \max_{\{z_1, \ldots, z_n\} \subset \mathcal{Z}} \frac{1}{2} \sum_{k=1}^{n} \ln(1 + \lambda^{-1} \sigma_{k-1}^2(z_k)). \tag{2.14b}$$

Srinivas *et al.* (2012) show that the Maximum Information Gain (MIG) is sub-linear in the number of observations for commonly used kernels.

The main idea now is to bound the complexity measure Γ_N defined in Equation (2.11) in terms of the MIG and, for commonly used kernels, we achieve no-regret algorithms. Towards this end, we recall two results related to GP-models.

Lemma 1. *Posterior variance bound (Chowdhury & Gopalan, 2019) Let $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be a symmetric positive semi-definite kernel with bounded variance, i.e., $k(z, z) \leq 1, \forall z \in \mathcal{Z}$ and $f \sim GP_{\mathcal{Z}}(0, k)$ be a sample from the associated GP, then for all $n \geq 1$ and $z \in \mathcal{Z}$:*

$$\sigma_{n-1}^2(z) \leq (1 + \lambda^{-1}) \sigma_n^2(z), \tag{2.15a}$$

$$\begin{aligned} \sum_{k=1}^{n} \sigma_{k-1}^2(z_k) &\leq (1 + 2\lambda) \sum_{k=1}^{n} \frac{1}{2} \ln \left[1 + \lambda^{-1} \sigma_{k-1}(z_k)\right] \\ &= (1 + 2\lambda) I(f_Z; y_Z). \end{aligned} \tag{2.15b}$$

Proof. See Chowdhury & Gopalan (2019, Lemma 2) □

Although the left-hand-side in eq. (2.15b) has the flavor of the complexity measure Γ_N defined in Equation (2.11) it is not exactly the same as we only update the posterior once every Hd_s observations. This is related to the batch setting analyzed in Desautels *et al.* (2014). The next lemma bounds the sum of posterior variances in terms of the information gain.

Lemma 2 (Complexity measure Γ_N is upper bounded by MIG)**.** *Let $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be a symmetric positive semi-definite kernel with bounded variance, i.e., $k(\mathbf{z}, \mathbf{z}) \leq 1, \forall \mathbf{z} \in \mathcal{Z}$ and $f \sim \mathrm{GP}_{\mathcal{Z}}(\mathbf{0}, k)$ be a sample from the associated GP, then for all $n \geq 1$ and $\mathbf{z} \in \mathcal{Z}$ for the GP model given in Equation (2.12) with $\lambda = Hd_s$ we have that:*

$$\Gamma_n \leq 2ed_sH\gamma_{d_sHn}(k, \mathcal{Z}) \tag{2.16}$$

Proof. The proof is based on Chowdhury & Gopalan (2019, Lemma 11) and adapted to our setting.

$$\sum_{n=1}^{N} \sum_{(\mathbf{s},\mathbf{a}) \in \tilde{\mathcal{D}}_n} \|\boldsymbol{\Sigma}_{n-1}(\mathbf{s}, \mathbf{a})\|_2^2 \tag{2.17a}$$

$$= \sum_{n'=1}^{n} \sum_{h=0}^{H-1} \sum_{i=1}^{d_s} \sigma^2_{(n'-1)Hd_s}(z_{n',h,i}) \tag{2.17b}$$

$$\leq \sum_{n'=1}^{n} \sum_{h=0}^{H-1} \sum_{i=1}^{d_s} (1 + \lambda^{-1})^{d_s h + i - 1} \sigma^2_{(n'-1)Hd_s + hd_s + i}(z_{n',h,i}) \tag{2.17c}$$

$$\leq (1 + \lambda^{-1})^{d_s(H-1)+d_s-1} \sum_{n'=1}^{n} \sum_{h=0}^{H-1} \sum_{i=1}^{d_s} \sigma^2_{(n'-1)Hd_s + hd_s + i}(z_{n',h,i}) \tag{2.17d}$$

$$\leq (1 + \lambda^{-1})^{d_sH-1}(2\lambda + 1) I(f_Z; y_Z) \tag{2.17e}$$

$$\leq 2ed_sH I(f_Z; y_Z) \tag{2.17f}$$

Here, equality (2.17b) is the definition of the 2-norm; inequality (2.17c) is due to eq. (2.15a) in Lemma 1; inequality (2.17d) is due to $1 + \lambda^{-1} \geq 1$; inequality (2.17e) is due to eq. (2.15b) in Lemma 1; finally the last inequality (2.17f) is due to $(1 + \lambda^{-1})^{\lambda} \leq e$ and $(1 + \lambda^{-1})^{-1}(2\lambda + 1) \leq 2\lambda$. The statement follows by taking the maximum over data sets. □

Next, we will show that, under some technical conditions, GP models are calibrated and satisfy Assumption 7.

2.2.3.2 *Concentration of GP Models*

Lemma 3. *Concentration of an RKHS member (Durand* et al., *2018, Theorem 1) Given Assumption 1, $\|f\|_{\mathcal{K}} \leq B_f$, and $k(\cdot,\cdot) \leq 1$, then for all $\delta \in [0,1]$, with probability at least $1-\delta$, it holds simultaneously over all $z \in Z$ and $n \geq 0$,*

$$|f(\mathbf{z}) - \boldsymbol{\mu}_n(\mathbf{z})| \leq \left(B_f + \frac{\sigma}{\lambda}\sqrt{2\ln(1/\delta) + 2\gamma_n}\right)\sigma_n(\mathbf{z}), \tag{2.18}$$

where $\boldsymbol{\mu}_n(\mathbf{z})$ and $\boldsymbol{\sigma}_n(\mathbf{z})$ are given by Equation (2.12d) *and Equation* (2.12e).

Thus we know that, using $\beta_n = \left(B_f + \frac{\sigma}{\lambda}\sqrt{2\ln(1/\delta) + 2\gamma_n}\right)$, Assumption 7 holds for a single dimension. The extension to multiple dimensions is straightforward and has been done by (Chowdhury & Gopalan, 2019, Lemma 10) and (Curi *et al.*, 2020a, Lemma 11), using $\lambda \leftarrow Hd_s$ and $t \leftarrow nHd_s$.

Putting together results of the previous sections, we know by Lemma 3 that, under Assumption 1 and bounded norm $\|f\|_{\mathcal{K}}$, GP models satisfy Assumption 7.

UNBOUNDED DOMAINS Most bounds on Γ_H assume that the domain $\mathcal{Z}$ is compact (Chowdhury & Gopalan, 2019). However, it is incompatible with Assumption 2, which allows for potentially unbounded noise $\boldsymbol{\omega}$. While this is a technical detail, Berkenkamp (2019, Appendix D.1) prove that, with high-probability, the domain can be bounded within a norm-ball of radius $b_n = \mathcal{O}(L_f^H Hd_s \log(Hn^2))$. Thus, the model complexity Γ_N only increases by a polylog factor on N for common kernels (e.g., the squared-exponential kernel).

2.2.3.3 *Lipschitz Continuity of GP Predictions*

Since the mean function is a linear combination of kernel evaluations (features), it is easy to show that it is Lipschitz continuous if the kernel function is Lipschitz continuous (Lederer *et al.*, 2019). However, existing bounds for the Lipschitz constant for the posterior standard deviation $\sigma(\cdot)$ depend on the number of data points. As our sample complexity bounds depend on L_σ^N, this would render our regret bound superlinear and thus meaningless.

In the following, we show that the GP standard deviation is Lipschitz-continuous with respect to the kernel metric.

Definition 3 (Kernel metric). $d_k(\mathbf{z}, \mathbf{z}') = \sqrt{k(\mathbf{z}, \mathbf{z}) + k(\mathbf{z}', \mathbf{z}') - 2k(\mathbf{z}, \mathbf{z}')}$.

We start with the standard deviation.

Lemma 4. *For all* $\mathbf{z}$ *and* $\mathbf{z}'$ *in* $\mathcal{Z}$*, we have*

$$\left|\sigma(\mathbf{z}) - \sigma(\mathbf{z}')\right| \leq d_k(\mathbf{z}, \mathbf{z}') \tag{2.19}$$

Proof. This is an adaptation from proof in Berkenkamp (2019, Lemma 50). From Mercer's theorem we know that each kernel can be equivalently written in terms of an infinite-dimensional inner product, so that $k(\mathbf{z}, \mathbf{z}') = \langle k(\mathbf{z}, \cdot), k(\mathbf{z}', \cdot) \rangle_k$, where $< \cdot, \cdot >_k$ is the inner product in the RKHS corresponding to the kernel k. We can think of GP regression as linear regression based on these infinite-dimensional feature vectors. In particular, it follows from (Kirschner & Krause, 2018, Appendix D) that we can write the GP posterior standard deviation $\sigma(\mathbf{z})$ as the weighted norm of the infinite-dimensional feature vectors $k(\mathbf{z}, \cdot)$,

$$\sigma(\mathbf{z}) = \|k(\mathbf{z}, \cdot)\|_{\mathbf{V}^{-1}}, \tag{2.20}$$

where $\mathbf{V} = \sigma^2 \mathbf{M}^* \mathbf{M} + \mathbf{I}$ and $\mathbf{M}$ is a linear operator that corresponds to the infinite-dimensional feature vectors $k(\mathbf{z}_i, \cdot)$ of the data points $\mathbf{z}_i$ in $\mathcal{Z}$ so that $[\mathbf{M}\mathbf{M}^*]_{(i,j)} = k(\mathbf{z}_i, \mathbf{z}_j)$, where $\mathbf{z}_i$ and $\mathbf{z}_j$ are the ith and jth data point in $\mathcal{Z}$. Now we have that the minimum eigenvalue of $\mathbf{V}$ is larger or equal to one, which implies that the maximum eigenvalue of $\mathbf{V}^{-1}$ is less or equal to one. Thus,

$$\begin{aligned}
|\sigma(\mathbf{z}) - \sigma(\mathbf{z}')| &= \left| \|k(\mathbf{z}, \cdot)\|_{\mathbf{V}^{-1}} - \|k(\mathbf{z}', \cdot)\|_{\mathbf{V}^{-1}} \right| && \text{(2.21a)} \\
&\leq \|k(\mathbf{z}, \cdot) - k(\mathbf{z}', \cdot)\|_{\mathbf{V}^{-1}}, && \text{(2.21b)} \\
&\leq \|k(\mathbf{z}, \cdot) - k(\mathbf{z}', \cdot)\|_k, && \text{(2.21c)} \\
&= \sqrt{\langle k(\mathbf{z}, \cdot) - k(\mathbf{z}', \cdot), k(\mathbf{z}, \cdot) - k(\mathbf{z}', \cdot) \rangle_k}, && \text{(2.21d)} \\
&= \sqrt{k(\mathbf{z}, \mathbf{z}) - k(\mathbf{z}, \mathbf{z}') - k(\mathbf{z}', \mathbf{z}) + k(\mathbf{z}', \mathbf{z}')}, && \text{(2.21e)} \\
&= \sqrt{k(\mathbf{z}, \mathbf{z}) + k(\mathbf{z}', \mathbf{z}') - 2k(\mathbf{z}, \mathbf{z}')}, && \text{(2.21f)} \\
&= d_k(\mathbf{z}, \mathbf{z}'), && \text{(2.21g)}
\end{aligned}$$

where *Equation* (2.21a) $\rightarrow$ *Equation* (2.21b) follows from the reverse triangle inequality. □

To show that Lemma 4 implies Lipschitz continuity of the variance, the key observation is that standard deviation $\sigma(\mathbf{z})$ is bounded. In particular,

$$\sigma(\mathbf{z}) \leq \sigma_0(\mathbf{z}) = \sqrt{k(\mathbf{z},\mathbf{z})} \leq \max_{\mathbf{z},\mathbf{z}' \in \mathbb{R}^d} \sqrt{k(\mathbf{z},\mathbf{z}')} := \sqrt{|k|_\infty} \tag{2.22}$$

Based on this, we have the following result.

Lemma 5. *For all* $\mathbf{z}$ *and* $\mathbf{z}'$ *in* $\mathcal{Z}$*, we have*

$$|\sigma^2(\mathbf{z}) - \sigma^2(\mathbf{z}')| \leq 2\sqrt{|k|_\infty}\, d_k(\mathbf{z},\mathbf{z}') \tag{2.23}$$

Proof. Since $0 \leq \sigma(\mathbf{z}) \leq \sqrt{|k|_\infty}$, we have

$$|\sigma^2(\mathbf{z}) - \sigma^2(\mathbf{z}')| = |\left(\sigma(\mathbf{z}) + \sigma(\mathbf{z}')\right)\left(\sigma(\mathbf{z}) - \sigma(\mathbf{z}')\right)| \tag{2.24a}$$

$$\leq 2\sqrt{|k|_\infty}\,\left|\sigma(\mathbf{z}) - \sigma(\mathbf{z}')\right| \tag{2.24b}$$

$$\leq 2\sqrt{|k|_\infty}\, d_k(\mathbf{z},\mathbf{z}') \tag{2.24c}$$

□

2.2.4 *Neural Network Models*

Neural network models have the advantage that they scale to higher dimensions and larger datasets better than GP models. To represent the set of plausible models $\mathcal{M}$ in (2.9) we use ensembles of I independent neural networks (Lakshminarayanan *et al.*, 2017a; Chua *et al.*, 2018). As the dataset increases with each episode, bootstrapping is impractical and instead we follow Osband *et al.* (2016) and simulate bootstrapping by sampling a weight from a Poison distribution $w_{i,j} \sim \text{Poi}(1)$, for each ensemble member i and each datapoint j. This weight $w_{i,j}$ modules the losses as seen below. We either use deterministic ensembles (DE) or probabilistic ensembles (PE), which we develop next. The main difference between these two model families is the underlying dynamics they represent. As DE models ignore aleatoric uncertainty, they represent deterministic environments better. Instead, PE models are better suited for stochastic environments. We train each ensemble member using type-II maximum likelihood estimation, and after training we recalibrate them using temperature scaling. We describe training and recalibration below.

DETERMINISTIC ENSEMBLES Each member i of the ensemble outputs only a mean prediction $\boldsymbol{\mu}_i(\cdot;\theta_i)$. The training loss in the setting is:

$$\mathcal{L}_i(\theta) = \frac{1}{N}\sum_{j=1}^{N} w_{i,j}\left(\boldsymbol{\mu}_i(z_j;\theta_i) - y_j\right)^2. \tag{2.25}$$

Given I trained ensemble members that output $\boldsymbol{\mu}_i(\cdot;\theta)$, we combine the predictions as a mixture of dirac distributions.

$$\boldsymbol{\mu}(\cdot;\theta) = \frac{1}{I}\sum_{i=1}^{I}\boldsymbol{\mu}_i(\cdot;\theta), \tag{2.26a}$$

$$\boldsymbol{\Sigma}(\cdot;\theta) = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{\mu}_i(\cdot;\theta) - \boldsymbol{\mu}(\cdot;\theta))(\boldsymbol{\mu}_i(\cdot;\theta) - \boldsymbol{\mu}(\cdot;\theta))^\top, \tag{2.26b}$$

$$v = \mathbf{0}. \tag{2.26c}$$

PROBABILISTIC ENSEMBLES Each member i of the ensemble outputs a normal distribution parameterized as $\mathcal{N}(\boldsymbol{\mu}_i(\cdot;\theta_i), v_i^2(\cdot;\theta_i))$. Hence, the log-likelihood loss in the setting is:

$$\mathcal{L}_i(\theta) = \frac{1}{2Nv_i^2(z_j;\theta)}\sum_{j=1}^{N} w_{i,j}\left(\boldsymbol{\mu}_i(z_j;\theta) - y_j\right)^2 + \log\left(2\pi v_i^2(z_j;\theta)\right). \tag{2.27}$$

Given I trained ensemble members that output $\mathcal{N}(\boldsymbol{\mu}_i(\cdot;\theta_i), v_i^2(\cdot;\theta_i))$, we combine the predictions as a mixture of Gaussian distributions.

$$\boldsymbol{\mu}(\cdot;\theta) = \frac{1}{I}\sum_{i=1}^{I}\boldsymbol{\mu}_i(\cdot;\theta), \tag{2.28a}$$

$$\boldsymbol{\Sigma}(\cdot;\theta) = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{\mu}_i(\cdot;\theta) - \boldsymbol{\mu}(\cdot;\theta))(\boldsymbol{\mu}_i(\cdot;\theta) - \boldsymbol{\mu}(\cdot;\theta))^\top, \tag{2.28b}$$

$$v = \frac{1}{I}\sum_{i=1}^{I} v_i(\cdot;\theta) \tag{2.28c}$$

2.2.4.1 *Neural Network Calibration*

After training, we recalibrate on the validation set using temperature scaling with parameter a α per output coordinate. This is equivalent to selecting β in the set of plausible models (2.9). We follow Malik *et al.* (2019) and choose

m thresholds, such that $0 \leq p_1 \cdots \leq p_m = 1$ and compute the empirical frequency of the prediction continuous distribution function, i.e.

$$\hat{p}_j = \frac{1}{|\mathcal{D}|} \left| \frac{\mathbf{s}_{i,h+1} - \boldsymbol{\mu}_h(\mathbf{z}_{i,h})}{\alpha \boldsymbol{\sigma}_{i,h}} \quad \forall i \in \mathcal{D} \leq p_j \right|, \tag{2.29}$$

elementwise. Then, we chose a temperature parameter that minimizes the expected calibration error:

$$\mathcal{L}(\alpha) = \sum_{j=1}^{m} \left(\hat{p}_j - p_j \right)^2, \tag{2.30}$$

and find α that minimize such loss. Finally, the epistemic variance of the recalibrated model is $\alpha^2 \boldsymbol{\Sigma}(\cdot)$.

2.2.5 *Model Reparameterization and Hallucination*

In the subsequent chapters, we will use the set of plausible models $\mathcal{M}$ to reason about the uncertainty. To achieve the goals of data-efficiency, robustness, and safety, we will need to optimize over $\tilde{f} \in \mathcal{M}$. Unfortunately, this optimization is usually intractable as the $\mathcal{M}$ is not convex, even in bandit settings (Dani *et al.*, 2008). Instead, we leverage the structure in $\mathcal{M}$ and reparameterize it introducing functions $\eta \in \mathcal{U} \colon \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to [-1, 1]^{d_s}$, which we call hallucination policies. We use this name because η exerts a hallucinated control authority on the *outputs* of the model to select the dynamics $\tilde{f} \in \mathcal{M}$. The **key idea** is that optimizing over $\eta \in \mathcal{U}$ is simpler than optimizing over $\tilde{f} \in \mathcal{M}$. We reparameterize the dynamics as:

$$\tilde{f}(\cdot) = \boldsymbol{\mu}(\cdot) + \beta \sigma(\cdot) \eta(\cdot). \tag{2.31}$$

Given a $\eta \in \mathcal{U}$ function, an initial starting state $\tilde{\mathbf{s}}_0$, a policy π, the states are propagated according to:

$$\tilde{\mathbf{s}}_{h+1} = \boldsymbol{\mu}(\tilde{\mathbf{z}}_h) + \beta \sigma(\tilde{\mathbf{z}}_h) \eta(\tilde{\mathbf{z}}_h) + \tilde{\boldsymbol{\omega}}_h, \tag{2.32}$$

$$\tag{2.33}$$

where $\tilde{\mathbf{a}}_h \sim \pi(\tilde{\mathbf{s}}_h)$ and $\tilde{\boldsymbol{\omega}}_h \sim \mathcal{N}(0, v^2)$. In Figure 2.2, we show an illustration of the true and hallucinated state trajectories, which we denote $\mathbf{s}_{0:3}$ and $\tilde{\mathbf{s}}_{0:3}$, respectively. In red, we show how the policy acts on the true state

trajectory, and the state realization is contained within the state distribution, which grows with time. In blue, we show how the true and hallucinated policies act on the hallucinated trajectory. First, the true policy controls the inputs to the dynamics and selects the next-state distribution. Then, the hallucination policy acts on the *output* of the 1-step ahead distribution and exherts a *hallucinated control authority* to select any outcome from within the 1-step confidence intervals. By changing the hallucination policy η, the set of plausible models $\mathcal{M}$ is covered.

An important observation is that both $\mathbf{s}_{0:3}$ and $\tilde{\mathbf{s}}_{0:3}$ are contained within the state distribution, but computing this distribution is usually hard for non-linear models and thus we do not have access to it. Furthermore, although the hallucinated state $\tilde{\mathbf{s}}_h$ is contained within the confidence interval centered at $\boldsymbol{\mu}(\tilde{\mathbf{s}}_{h-1})$ with width $\beta\boldsymbol{\sigma}(\tilde{\mathbf{s}}_{h-1})$, the true state $\mathbf{s}_h$ is not inside such set. On one hand, this makes the analysis more complicated. On the other hand, this makes the algorithms practical because we know how to compute the 1-step ahead predictive distribution of the model.

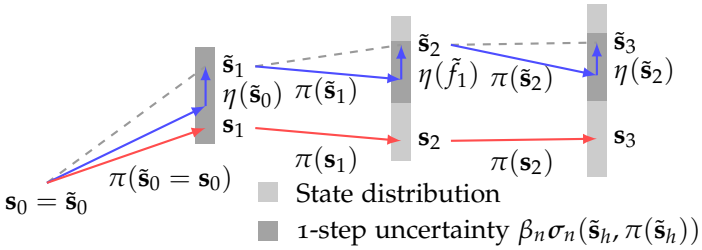


FIGURE 2.2: Illustration of the true state trajectory $\mathbf{s}_{0:3}$ generated by the true dynamics f and the policy π, and the hallucinated state trajectory $\tilde{\mathbf{s}}_{0:3}$ generated by the true dynamics $\tilde{f}$ and the same policy π. In blue, we show the true and hallucinated policies along the hallucinated trajectory. In red, we show the true policies along the true trajectory. For the first time step, both $\mathbf{s}_1$ and $\tilde{\mathbf{s}}_1$ are contained within the models 1-step ahead confidence interval. Afterwards, both $\mathbf{s}_h$ and $\tilde{\mathbf{s}}_h$ are contained within the state predictive distribution, but $\mathbf{s}_h$ is *outside* the 1-step ahead confidence interval evaluated at $\mathbf{s}_{h-1}$.

A NOTE ON UNKNOWN REWARDS When the reward function is unknown, the agent must learn the it together with the dynamics. This implies that the agent has epistemic uncertainty about it. The reparameterization trick can also be used in this setting, however the dimensionality increases

by one, i.e., the hallucination polices are $\eta \in \mathcal{U} \colon \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to [-1,1]^{d_s+\mathbf{1}}$. The reward and state dynamics are propagated simulated as:

$$\tilde{\mathbf{s}}_{h+1}, \tilde{r}_h = \boldsymbol{\mu}(\tilde{\mathbf{z}}_h) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_h)\eta(\tilde{\mathbf{z}}_h) + \tilde{\boldsymbol{\omega}}_h, \tag{2.34}$$

$$. \tag{2.35}$$

We now prove that the hallucination policies η exist.

Lemma 6 (Existence of η policies.)**.** *Under Assumption 7, for any sequence $\mathbf{s}_{h,n}$ generated by the true system (2.1), there exists a function $\eta \in \mathcal{U} \colon \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to [-1,1]^{d_s}$ such that $\mathbf{s}_{h,n} = \tilde{\mathbf{s}}_{h,n}$ if $\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}}$.*

Proof. By Assumption 7 we have $|f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z})| \leq \beta\boldsymbol{\sigma}(\mathbf{z})$, elementwise. Thus for each $\mathbf{z}$ there exists a vector $\mathbf{u}$ with values in $[-1,1]^{d_s}$ such that $f(\mathbf{z}) = \boldsymbol{\mu}(\mathbf{z}) + \boldsymbol{\sigma}(\mathbf{z})\mathbf{u}$. Let the function $\eta(\cdot)$ return this vector for each state and action, then the result follows. □

So far, we have considered general functions $\eta \in \mathcal{U} \colon \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to [-1,1]^{d_s}$, which can potentially be discontinuous. However, as long as Lemma 6 holds, we can use a more restrictive function class.

It is clear that it is sufficient to consider functions η such that $\boldsymbol{\sigma}(\cdot)\eta(\cdot)$ is Lipschitz continuous, since it aims to approximate a Lipschitz continuous function f:

Lemma 7. *With Assumptions 1, 6 and 7 let $\eta(\cdot)$ be a function such that $f(\cdot) - \boldsymbol{\mu}(\cdot) = \beta\boldsymbol{\sigma}(\cdot)\eta(\cdot)$ as in Lemma 6. Then $\boldsymbol{\sigma}(\cdot)\eta(\cdot)$ is Lipschitz continuous.*

Proof.

$$\|\boldsymbol{\sigma}(\mathbf{z})\eta(\mathbf{z}) - \boldsymbol{\sigma}(\mathbf{z}')\eta(\mathbf{z}')\| = \frac{1}{\beta}\|f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z}) - (f(\mathbf{z}') - \boldsymbol{\mu}(\mathbf{z}'))\| \tag{2.36}$$

$$\leq \frac{L_f + L_{\boldsymbol{\mu}}}{\beta}\|\mathbf{z} - \mathbf{z}'\| \tag{2.37}$$

□

Unfortunately, the same is not true for η on its own in general. However, if the predictive standard deviation $\boldsymbol{\sigma}$ does not decay to zero, this holds.

Lemma 8. *Under the assumptions of Lemma 7 let $0 < \boldsymbol{\sigma}_{\min} \leq \boldsymbol{\sigma}(\mathbf{z}) \leq \boldsymbol{\sigma}_{\max}$ elementwise for all $\mathbf{z} \in \mathcal{Z}$. Then, with probability at least $(1-\delta)$, there exists a Lipschitz-continuous function $\eta(\cdot)$ with $\|\eta(\cdot)\|_\infty = 1$ such that $f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z}) = \beta\boldsymbol{\sigma}(\mathbf{z})\eta(\mathbf{z})$ for all $\mathbf{z} \in \mathbb{R}^{d_s}$.*

Proof. By contradiction. By Assumption 7, we know that, with probability $(1-\delta)$, $f \in \mathcal{M}_\beta$. Let $\eta(\cdot)$ be a function that is not Lipschitz continuous such that $f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z}) = \beta\boldsymbol{\sigma}(\mathbf{z})\eta(\mathbf{z})$. By assumption we know that $\boldsymbol{\sigma}(\mathbf{z})$ is strictly larger than zero and bounded element-wise from above by some constant. As a consequence, $\boldsymbol{\sigma}^{-1}(\mathbf{z})$ exists and is $L_\sigma/\sigma_{\min}^2$-Lipschitz continuous w.r.t. the Frobenius norm. Thus, we have

$$
\begin{aligned}
&\|\eta(\mathbf{z}) - \eta(\mathbf{z}')\|_2 \\
&= \|\frac{1}{\beta}\boldsymbol{\sigma}^{-1}(\mathbf{z})(f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z})) - \frac{1}{\beta}\boldsymbol{\sigma}^{-1}(\mathbf{z}')(f(\mathbf{z}') - \boldsymbol{\mu}(\mathbf{z}'))\|_2 \\
&\leq |\frac{1}{\beta}|\|\boldsymbol{\sigma}^{-1}(\mathbf{z})((f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z})) - (f(\mathbf{z}') - \boldsymbol{\mu}(\mathbf{z}')))\|_2 \\
&\quad + |\frac{1}{\beta}|\|\left(\boldsymbol{\sigma}^{-1}(\mathbf{z}) - \boldsymbol{\sigma}^{-1}(\mathbf{z}')\right)(f(\mathbf{z}') - \boldsymbol{\mu}(\mathbf{z}'))\|_2 \\
&\leq |\frac{1}{\beta}|\|\boldsymbol{\sigma}^{-1}(\mathbf{z})\|_{\mathrm{F}}\|(f(\mathbf{z}) - \boldsymbol{\mu}(\mathbf{z})) - (f(\mathbf{z}') - \boldsymbol{\mu}(\mathbf{z}'))\|_2 \\
&\quad + |\frac{1}{\beta}|\|f(\mathbf{z}') - \boldsymbol{\mu}(\mathbf{z}')\|_2\|\boldsymbol{\sigma}^{-1}(\mathbf{z}) - \boldsymbol{\sigma}^{-1}(\mathbf{z}')\|_{\mathrm{F}} \\
&\leq |\frac{1}{\beta}|\|\boldsymbol{\sigma}^{-1}(\mathbf{z})\|_{\mathrm{F}}(L_f + L_\mu)\sqrt{1+L_\pi}\|\mathbf{z} - \mathbf{z}'\|_2 \\
&\quad + |\frac{1}{\beta}|\|\beta\boldsymbol{\sigma}(\mathbf{z}')\|_2\|\boldsymbol{\sigma}^{-1}(\mathbf{z}) - \boldsymbol{\sigma}^{-1}(\mathbf{z}')\|_{\mathrm{F}} \\
&\leq \frac{\sqrt{d_s}}{\beta\sigma_{\min}}(L_f + L_\mu)\sqrt{1+L_\pi}\|\mathbf{z} - \mathbf{z}'\|_2 + \frac{\sqrt{d_s}\sigma_{\max}}{\sigma_{\min}^2}L_\sigma\sqrt{1+L_\pi}\|\mathbf{z} - \mathbf{z}'\|_2
\end{aligned}
$$

Since $\beta > 0$ we have that $\eta(\mathbf{z})$ is Lipschitz continuous, which is a contradiction. □

Thus, it is generally sufficient to optimize over Lipschitz continuous functions $\eta \in \mathcal{U}$ instead of over dynamics $\tilde{f} \in \mathcal{M}$. However, it is important to note that the complexity of the function (i.e., its Lipschitz constant) will generally increase as the predictive variance decreases. It is easy to construct cases where $\boldsymbol{\sigma}(\cdot) = 0$ implies that η has to be discontinuous. However, at least in theory $\boldsymbol{\sigma}(\cdot) = 0$ is impossible with finite data when the system is noisy ($\omega > 0$). Also note that as $\boldsymbol{\sigma}$ decreases, the effect of η on the dynamics also decreases.

2.3 MODEL-BASED POLICY OPTIMIZATION TECHNIQUES

After learning a model, the next task is to select a policy in Algorithm 1. This is usually done either by using the model to simulate transitions and learn a parametric policy in a pure offline fashion or by using the model to plan online. In this section, we augment common MBRL algorithms using the Hallucinated models in Section 2.2.5. In Section 2.3.1, we see offline policy search algorithms by considering parametric hallucinated policies. Instead, in Section 2.3.2, we consider online planning where η is defined implicitly as the solution of an online optimization problem. Finally, in Section 2.3.3, we combine offline policy search with online planning and introduce Dyna-MPC.

2.3.1 *Offline Policy Search*

Off-line policy search usually parameterize the control policy $\pi(\cdot;\theta_\pi)$ and the hallucination policy $\eta(\cdot;\theta_\eta)$ using a function approximation method (e.g., neural networks). The main idea is to use the control policy only $\pi(\cdot;\theta_\pi)$ to interact with the environment, and the both the control policy $\pi(\cdot;\theta_\pi)$ and the hallucination policy $\eta(\cdot;\theta_\eta)]$ to interact with the model in an offline fashion. Next, we describe how to *hallucinate* common policy-search algorithms. We leave unspecified how to update the hallucination policies as this depends on the particular problem that we are tackling.

Hallucinated Data Augmentation consists of using the model to simulate data and then use these data to learn a policy using a model-free RL method. For example, the celebrated Dyna algorithm from Sutton (1990), DAD from Venkatraman *et al.* (2016), IB from Kalweit & Boedecker (2017), and I2A Racanière *et al.* (2017) generate data by sampling from expected models. In Algorithm 2, we show Hallucinated Data Augmentation (HDA). In HDA, we generate data using the dynamics in (2.2) and then call any model-free RL algorithm such as SAC (Haarnoja *et al.*, 2018), MPO (Abdolmaleki *et al.*, 2018), TD3 (Fujimoto *et al.*, 2018), TRPO (Schulman *et al.*, 2015b), or PPO (Schulman *et al.*, 2017). As there is no need to propagate gradients through the data-generation process, there is no need to re-parameterize the sampling procedure in Line 10 of Algorithm 2. Furthermore, the initial state distribution can be made arbitrary to make the planning problem easier to solve, i.e., trajectories might start from any exploratory distribution. This greatly simplifies the task of the `ModelFree` algorithm. Usually these strategies combine simulated with real data buffers. This strategy usually

suffers from model-bias as model errors compound throughout a trajectory, yielding highly biased estimates that hinder the policy optimization (van Hasselt *et al.*, 2019).

Algorithm 2 Hallucinated Data Augmentation

Inputs: Calibrated dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, reward function $r(\cdot)$, initial state distribution $d(\mathbf{s}_0)$, number of iterations N_{iter}, simulation horizon $\tilde{H}$ initial parameters $\theta_{0,\pi}, theta_{0,\eta}, \vartheta_0$, model-free algorithm `ModelFree`, Real data set $\mathcal{D}$

1: **for** $i = 1, \ldots, N_{\text{iter}}$ **do**
2: /* Simulate Data */
3: Initialize data buffer $\tilde{\mathcal{D}} = \{\emptyset\}$
4: **for** $i' = 1, \ldots, N_{\text{data}}$ **do**
5: Start from initial state distribution $\tilde{\mathbf{s}}_0 \sim d(\mathbf{s}_0)$
6: **for** $h = 0, \ldots, \tilde{H} - 1$ **do**
7: Sample action $\tilde{\mathbf{a}}_h \sim \pi(\tilde{\mathbf{s}}_h; \theta_{i,\pi})$
8: Sample hallucinated action $\tilde{\mathbf{u}}_h \sim \eta(\tilde{\mathbf{s}}_h; \theta_{i,\eta})$
9: Concatenate $\tilde{\mathbf{z}}_h = [\tilde{\mathbf{s}}_h, \tilde{\mathbf{a}}_h]$
10: Sample next state $\tilde{\mathbf{s}}_{h+1} \sim \mathcal{N}(\boldsymbol{\mu}(\tilde{\mathbf{z}}_h) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_h)\tilde{\mathbf{u}}_h; \boldsymbol{\omega}_h^2)$
11: Query Reward $r_h = r(\tilde{\mathbf{z}}_h)$
12: Append transition to buffer $\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \cup \{(\tilde{\mathbf{s}}_h, \tilde{\mathbf{s}}_{h+1}, \tilde{\mathbf{a}}_h, r_h)\}$.
13: /* Optimize Policy */
14: $\theta_{i+1,\pi}, \theta_{i+1,\eta} \vartheta_{i+1} \leftarrow$ `ModelFree`$(\tilde{\mathcal{D}} \cup \mathcal{D}, \theta_{i,\pi}, \theta_{i,\eta}, \vartheta_i)$

Outputs: Final policy $\theta_\pi = \theta_{N_{\text{iter}},\pi}$.

Hallucinated Back-Propagation Through Time is an algorithm that updates the policy parameters by computing the derivatives of the performance w.r.t. the parameters directly. In Algorithm 3, we show Hallucinated Back-Propagation Through Time (H-BPTT). For instance, PILCO from M. Deisenroth & Rasmussen (2011) and MBAC from Clavera *et al.* (2020) are different examples of practical algorithms that BPTT using GPs and ensembles of neural networks, respectively. Like in BPTT, it samples the trajectories in a differentiable way, i.e., using the reparameterization trick (Kingma & Welling, 2013). Under some assumptions (such as moment matching), the sampling step in Line 10 of Algorithm 3 can be replaced by exact integration as in PILCO (M. Deisenroth & Rasmussen, 2011). While performing the rollout, it computes the performance and at the end it bootstrapped with a critic. This critic is learned using a policy evaluation `PolEval` algorithm such as Fitted Value Iteration (Antos *et al.*, 2008). This strategy usually

suffers from high variance due to the stochasticity of the sampled trajectories and the compounding of gradients (McHutchon, 2014). Interestingly, Parmas *et al.* (2018) propose a method to combine the model-free gradients given by any HDA strategy together with the model-based gradients given by HBPTT, but we leave this for future work. We found that limiting the KL-divergence between the policies in different episodes, as suggested by Schulman *et al.* (2015a), helps to regularize the optimization problem.

Algorithm 3 Hallucinated Back-Propagation Through Time

Inputs: Calibrated dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, reward function $r(\cdot)$, initial state distribution $d(\mathbf{s}_0)$, number of iterations N_{iter}, simulation horizon $\tilde{H}$ initial parameters $\theta_{0,\pi}, \theta_{0,\eta}, \vartheta_0$, learning rate η_{lr}, policy evaluation algorithm `PolEval`, regularization λ, Real data set $\mathcal{D}$.

for $i = 1, \ldots, N_{\text{iter}}$ **do**
  Initialize data buffer $\tilde{\mathcal{D}} = \{\emptyset\}$
  /* Simulate Data */
  Start from initial state distribution $\tilde{\mathbf{s}}_0 \sim d(\mathbf{s}_0)$.
  Restart $J \leftarrow 0$
  for $h = 0, \ldots, \tilde{H} - 1$ **do**
    Sample action $\tilde{\mathbf{a}}_h \sim \pi(\tilde{\mathbf{s}}_h; \theta_{i,\pi})$
    Sample hallucinated action $\tilde{\mathbf{u}}_h \sim \eta(\tilde{\mathbf{s}}_h; \theta_{i,\eta})$
    Concatenate $\tilde{\mathbf{z}}_h = [\tilde{\mathbf{s}}_h, \tilde{\mathbf{a}}_h]$
    Sample next state $\tilde{\mathbf{s}}_{h+1} \sim \mathcal{N}(\boldsymbol{\mu}(\tilde{\mathbf{z}}_h) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_h)\tilde{\mathbf{u}}_h; \boldsymbol{\omega}_h^2)$
    Query Reward $r_h = r(\tilde{\mathbf{z}})$
    Accumulate $J \leftarrow J + \gamma^h r_h - \lambda \operatorname{KL}\left(\pi(\tilde{\mathbf{s}}_h; \theta_{i,\pi}) \,||\, \pi(\tilde{\mathbf{s}}_h; \theta_{0,\pi})\right)$
    Append transition to buffer $\tilde{\mathcal{D}} \leftarrow \tilde{\mathcal{D}} \cup \left\{(\tilde{\mathbf{s}}_h, \tilde{\mathbf{s}}_{h+1}, \tilde{\mathbf{a}}_h, r_h\right\}$
  Bootstrap $J \leftarrow J + \gamma^{\tilde{H}} Q(\tilde{\mathbf{s}}_{\tilde{H}}, \pi(\tilde{\mathbf{s}}_{\tilde{H}}; \theta_{i,\pi}), \eta(\tilde{\mathbf{s}}_{\tilde{H}}; \theta_{i,\eta}); \vartheta_i)$
  /* Optimize Policy */
  Compute gradient $\partial J / \partial \theta_\pi$ with BPTT
  Compute gradient $\partial J / \partial \theta_\eta$ with BPTT
  Do gradient step $\theta_{i+1,\pi} \leftarrow \theta_{i,\pi} + \eta_{\text{lr}} \partial J / \partial \theta_{i,\pi}$
  Do gradient step $\theta_{i+1,\eta} \leftarrow \theta_{i,\eta} \pm \eta_{\text{lr}} \partial J / \partial \theta_{i,\eta}$
  Update Critic $\vartheta_{i+1} \leftarrow \texttt{PolEval}(\tilde{\mathcal{D}}, \vartheta_i)$

Outputs: Final policy $\theta_\pi = \theta_{N_{\text{iter}},\pi}$.

Hallucinated Model-Based Value Expansion is an Actor-Critic approach that uses the model to compute the next-states for the Bellman target when learning the action-value function. It then uses pathwise derivatives (Mohamed *et al.*, 2019) through the learned action-value function. For example MVE from (Feinberg *et al.*, 2018) and STEVE from Buckman *et al.*

(2018) use such strategy. In Algorithm 4, we show H-MVE (Hallucinated-Model Based Value Expansion). Here we use hallucinated trajectories only to learn the Bellman target. This strategy is usually less data efficient than BPTT or IDA as it uses the model only to compute targets, but suffers less from model bias. To address data efficiency, one can combine HMVE and HDA to compute optimistic value functions as well as simulating optimistic data.

Algorithm 4 Hallucinated-Model Value Expansion

Inputs: Calibrated dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, reward function $r(\cdot)$, number of iterations N_{iter}, simulation horizon $\tilde{H}$ initial parameters $\theta_{0,\pi}, \theta_{0,\eta}, \vartheta_0$, learning rate η_{lr}, Real data set $\mathcal{D}$.

Initialize target parameters $\bar{\vartheta}_0 \leftarrow \vartheta_0$.
for $i = 1, \ldots, N_{\text{iter}}$ **do**
  /* Simulate Data */
  Start from buffer $\tilde{\mathbf{s}}_0 \sim \mathcal{D}$
  Initialize target $Q_{\text{target}} \leftarrow 0$
  Compute prediction $Q_{\text{pred}} = Q(\tilde{\mathbf{s}}_0; \vartheta_i)$
  for $h = 0, \ldots, \tilde{H} - 1$ **do**
    Sample action $\tilde{\mathbf{a}}_h \sim \pi(\tilde{\mathbf{s}}_h; \theta_{i,\pi})$
    Sample hallucinated action $\tilde{\mathbf{u}}_h \sim \eta(\tilde{\mathbf{s}}_h; \theta_{i,\eta})$
    Concatenate $\tilde{\mathbf{z}}_h = [\tilde{\mathbf{s}}_h, \tilde{\mathbf{a}}_h]$
    Sample next state $\tilde{\mathbf{s}}_{h+1} \sim \mathcal{N}(\boldsymbol{\mu}(\tilde{\mathbf{z}}_h) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_h)\tilde{\mathbf{u}}_h; \boldsymbol{\omega}_h^2)$
    Query Reward $r_h = r(\tilde{\mathbf{z}})$
    Accumulate target $Q_{\text{target}} \leftarrow \gamma^h r_h$.
  Bootstrap $Q_{\text{target}} \leftarrow Q_{\text{target}} + \gamma^{\tilde{H}} Q(\tilde{\mathbf{s}}_{\tilde{H}}, \pi(\tilde{\mathbf{s}}_{\tilde{H}}; \theta_{i,\pi}), \eta(\tilde{\mathbf{s}}_{\tilde{H}}; \theta_{i,\eta}); \vartheta_i)$
  /* Optimize Critic */
  $\vartheta_{i+1} \leftarrow \vartheta_i - \eta_{\text{lr}} \nabla_\vartheta (Q_{\text{pred}} - Q_{\text{target}})^2$
  Update target parameters $\bar{\vartheta}_{i+1} \leftarrow \tau\bar{\vartheta}_i + (1-\tau)\vartheta_{i+1}$
  /* Optimize Policy */
  $\theta_{i+1,\pi} \leftarrow \theta_{i,\pi} + \eta_{\text{lr}} \nabla_{\theta_{i,\pi}} Q(\tilde{\mathbf{s}}_0; \vartheta_i)$
  $\theta_{i+1,\eta} \leftarrow \theta_{i,\eta} \pm \eta_{\text{lr}} \nabla_{\theta_{i,\eta}} Q(\tilde{\mathbf{s}}_0; \vartheta_i)$

Outputs: Final policy $\theta_\pi = \theta_{N_{\text{iter}},\pi}$.

2.3.2 *Online Planning*

An alternative approach is to consider non-parametric policies and directly optimize the true actions as $\mathbf{a}_{h,n} \in [-1, 1]^{d_a}$, and the hallucinated action

$\mathbf{u}_{h,n} \in [-1,1]^{d_s}$. This is usually called Model-Predictive Control (MPC) and it is implemented in a receding horizon fashion (Morari & H. Lee, 1999; Zeilinger *et al.*, 2011). That means that for each new state encounter online, the planning problem (2.2) is solved using the actions as decission variables. Due to the receding horizon application, the effect of model errors compounding is lower as the starting states are evaluated through the real trajectories. However, the receding horizon comes at high online computational costs, which limits the applicability of such algorithms to simulations.

GP-MPC Kamthe & M. Deisenroth (2018a) and PETS Chua *et al.* (2018) are MPC-based methods that use GP and neural networks ensembles, respectively. Other MPC solvers such as POPLIN T. Wang & Ba (2019) or POLO (Lowrey *et al.*, 2019) are also compatible with such dynamical models. In H-MPC (Hallucinated-MPC), we directly optimize both the control and hallucinated inputs jointly and any of the previous methods can be used as the MPC solver. Moldovan *et al.* (2015) also use MPC to solve an optimistic exploration scheme but only on linear models and, like other on-line planning methods, are extremely slow for real-time deployment.

To solve the optimization problem, approximate local solvers are usually used that rely either on sampling or on linearization.

HALLUCINATED SAMPLING METHOD An approximate way of solving MPC problems is to exhaustively sample the decision variables. Shooting methods sample the actions and then propagate the trajectory through the model whereas collocation methods sample both the states and the actions (Hargraves & Paris, 1987). For simplicity, we only consider shooting methods. This method initializes particles at the current state. For each particle, it samples a sequence of actions from a proposal distribution and rollouts each particle independently, computing the returns of such sequence. This process is repeated updating the proposal distribution. Random Shooting (Richards & How, 2006), the Cross-Entropy Method (Botev *et al.*, 2013a), and Model-Predictive Path Integral Control (G. Williams *et al.*, 2016) differ in the ways to update the action sampling distributions. POPLIN from T. Wang & Ba (2019) instead maintains a distribution over the weights of a policy network and samples different policies. The main advantage of POPLIN method is that it correlates the random samples along a trajectory (as they come from the same policy), possibly scalling to higher dimensions. We show in Algorithm 5 the pseudo-code for Hallucinated Shooting (HS) shooting.

Algorithm 5 Hallucinated Shooting

Inputs: Calibrated dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, reward function $r(\cdot)$, terminal reward V, current state $\mathbf{s}_h$, simulation horizon $\tilde{H}$ number of particles $N_{\text{particles}}$, number of iterations N_{iter}, number of elite particles N_{elite}, initial sampling distribution $d(\cdot)$, algorithm to evaluate actions `EliteActions`, algorithm to update distribution `UpdateDistribution`.

for $i = 1, \ldots, N_{\text{iter}}$ **do**
  /* Simulate Data */
  Initialize $N_{\text{particles}}$ at the current state $\tilde{\mathbf{s}}_{p,0} = \mathbf{s}_h$
  Initialize $J_p \leftarrow 0$
  for $h = 0, \ldots, \tilde{H} - 1$ **do**
    Sample action $\tilde{\mathbf{a}}_{p,h}, \tilde{\mathbf{u}}_{p,h} \sim d(\cdot)$
    Concatenate $\tilde{\mathbf{z}}_{p,h} = [\tilde{\mathbf{s}}_{p,h}, \tilde{\mathbf{a}}_{p,h}]$
    Sample next state $\tilde{\mathbf{s}}_{p,h+1} \sim \mathcal{N}(\boldsymbol{\mu}(\tilde{\mathbf{z}}_{p,h}) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_{p,h})\tilde{\mathbf{u}}_{p,h}; \boldsymbol{\omega}^2_{p,h})$
    Query Reward $r_{p,h} = r(\tilde{\mathbf{z}}_{p,h})$
    Accumulate $J_p \leftarrow J_p + \gamma^h r_{p,h}$
  Bootstrap $J_p \leftarrow J_p + \gamma^{\tilde{H}} V(\tilde{\mathbf{s}}_{p,\tilde{H}})$.
  $\hat{\mathbf{a}}_i \leftarrow$ `EliteActions`$(J_p, \tilde{\mathbf{a}}_{p,0:\tilde{H}-1}, N_{\text{elite}})$
  $\hat{\mathbf{u}}_i \leftarrow$ `EliteActions`$(\pm J_p, \tilde{\mathbf{u}}_{p,0:\tilde{H}-1}, N_{\text{elite}})$
  /* Optimize Policy */
  Update distribution $d(\cdot) \leftarrow$ `UpdateDistribution`$(d(\cdot), \hat{\mathbf{a}}_i, \hat{\mathbf{u}}_i)$

Outputs: Return best action $\mathbf{a} \leftarrow$ `EliteActions`$(J_p, \tilde{\mathbf{a}}_{p,0:\tilde{H}-1})$.

HALLUCINATED DIFFERENTIAL DYNAMIC PROGRAMMING (DDP) DDP can be interpreted as a second-order shooting method Jacobson (1968) for dynamical systems. For linear dynamical models with quadratic costs, problems Equations (2.2) and (2.3) is a quadratic program (QP) that enjoys a closed form solution (Morari & H. Lee, 1999). To address non-linear systems and other cost functions, a common strategy is to use a variant of iLQR W. Li & Todorov (2004), Todorov & W. Li (2005), and Tassa *et al.* (2012) which linearizes the system and uses a second order approximation to the cost function to solve sequential QPs (SQP) that approximate the original problem. When the rewards and dynamical model are differentiable, DDP method is faster to sampling methods as it uses the problem structure to update the sampling distribution. We show an example of Hallucinated Gradient-Based MPC in Algorithm 6.

Algorithm 6 Hallucinated Gradient Based MPC

Inputs: Calibrated dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, reward function $r(\cdot)$, terminal reward V, current state $\mathbf{s}_h$, simulation horizon $\tilde{H}$ number of iterations N_{iter}, initial action sequence $\tilde{\mathbf{a}}_{0:\tilde{H}-1}$, initial action sequence $\tilde{\mathbf{u}}_{0:\tilde{H}-1}$,

Initialize $\tilde{\mathbf{a}}_{0,0:\tilde{H}-1} \leftarrow \tilde{\mathbf{a}}_{0:\tilde{H}-1}$
Initialize $\tilde{\mathbf{u}}_{0,0:\tilde{H}-1} \leftarrow \tilde{\mathbf{u}}_{0:\tilde{H}-1}$
for $i = 1, \ldots, N_{\text{iter}}$ **do**
    /* Simulate Data */
    Initialize $J \leftarrow 0$
    for $h = 0, \ldots, \tilde{H} - 1$ **do**
        Concatenate $\tilde{\mathbf{z}}_{i,h} = [\tilde{\mathbf{s}}_h, \tilde{\mathbf{a}}_{i,h}]$
        Sample next state $\tilde{\mathbf{s}}_{h+1} \sim \mathcal{N}(\boldsymbol{\mu}(\tilde{\mathbf{z}}_{i,h}) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_{i,h})\tilde{\mathbf{u}}_{i,h}; \boldsymbol{\omega}_h^2$
        Query Reward $r_h = r(\tilde{\mathbf{z}}_{i,h})$
        Accumulate $J \leftarrow J + \gamma^h r_h$
    Bootstrap $J \leftarrow J + \gamma^{\tilde{H}} V(\tilde{\mathbf{s}}_{\tilde{H}})$.
    /* Optimize actions */
    Compute gradient $\partial J / \partial \tilde{\mathbf{a}}_{0:\tilde{H}-1}$ with BPTT.
    Do gradient step $\tilde{\mathbf{a}}_{i+1,0:\tilde{H}-1} \leftarrow \tilde{\mathbf{a}}_{i,0:\tilde{H}-1} + \eta_{\text{lr}} \partial J / \partial \tilde{\mathbf{a}}_{0:\tilde{H}-1}$
    Compute gradient $\partial J / \partial \tilde{\mathbf{u}}_{0:\tilde{H}-1}$ with BPTT.
    Do gradient step $\tilde{\mathbf{u}}_{i+1,0:\tilde{H}-1} \leftarrow \tilde{\mathbf{u}}_{i,0:\tilde{H}-1} \pm \eta_{\text{lr}} \partial J / \partial \tilde{\mathbf{u}}_{0:\tilde{H}-1}$

Outputs: Return best action $\mathbf{a} \leftarrow \tilde{\mathbf{a}}_{N_{\text{iter}},0:\tilde{H}-1}$.

2.3.3 *Combining Offline Policy Search with Online Planning*

MPC methods suffer less from model bias, but typically require substantial computation. Furthermore, they are limited to the planning horizon unless a *learned* terminal reward is used to approximate the reward-to-go (Lowrey *et al.*, 2019). On the other hand, off-policy search approaches yield policies and value function estimates (critics) that are fast to evaluate, but suffer from bias (van Hasselt *et al.*, 2019). We propose to combine these methods to get the best of both worlds: First, we learn a parametric control policy π and hallucination policy η using an offline policy search algorithm. Then, we use such policies as a warm-start for the sampling distributions of the planning algorithm. We name this planning algorithm Dyna-MPC, as it resembles the Dyna architecture proposed by Sutton (1990) and we show the pseudo-code in Algorithm 7.

Closely related to Dyna-MPC is POPLIN (T. Wang & Ba, 2019). We also use a policy to initialize actions and and then refine them with a shooting

method. Nevertheless, we use a policy search algorithm to optimize the policy parameters instead of the cross-entropy method. Hong *et al.* (2019) also uses MPC to refine an off-line learned policy. However, they use a model-free algorithm directly from real data instead of model-based policy search.

Algorithm 7 Dyna-MPC

Inputs: Calibrated dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, reward function $r(\cdot)$, learned control policy $\pi(\cdot; \theta_\pi)$, learned hallucination policy $\eta(\cdot; \theta_\eta)$, learned critic $Q(\cdot; \vartheta)$, current state $\mathbf{s}_h$, simulation horizon $\tilde{H}$ number of iterations N_{iter}, number of particles $N_{\text{particles}}$, number of elite particles N_{elite}, algorithm to evaluate actions `EliteActions`, algorithm to update distribution `UpdateDistribution`.

for $i = 1, \ldots, N_{\text{iter}}$ **do**
  /* Simulate Data */
  Initialize $N_{\text{particles}}$ at the current state $\tilde{\mathbf{s}}_{p,0} = \mathbf{s}_h$
  Initialize $J_p \leftarrow 0$
  for $h = 0, \ldots, \tilde{H} - 1$ **do**
    Compute action $\tilde{\mathbf{a}}_{p,h} \sim \pi(\tilde{\mathbf{s}}_{p,h}; \theta_i)$
    Compute hallucinated action $\tilde{\mathbf{u}}_{p,h} \sim \pi(\tilde{\mathbf{s}}_{p,h}; \theta_i)$
    Concatenate $\tilde{\mathbf{z}}_{p,h} = [\tilde{\mathbf{s}}_{p,h}, \tilde{\mathbf{a}}_{p,h}]$
    Sample next state $\tilde{\mathbf{s}}_{p,h+1} \sim \mathcal{N}(\boldsymbol{\mu}(\tilde{\mathbf{z}}_{p,h}) + \beta\boldsymbol{\sigma}(\tilde{\mathbf{z}}_{p,h})\tilde{\mathbf{u}}_{p,h}; \boldsymbol{\omega}^2_{p,h}$
    Query Reward $r_{p,h} = r(\tilde{\mathbf{z}}_{p,h})$
    Accumulate $J_p \leftarrow J_p + \gamma^h r_{p,h}$
  Bootstrap $J_p \leftarrow J_p + \gamma^{\tilde{H}} Q(\tilde{\mathbf{s}}_{p,\tilde{H}}, \pi(\tilde{\mathbf{s}}_{p,\tilde{H}}; \theta_i); \vartheta_i)$
  $\hat{\mathbf{a}}_i \leftarrow$ `EliteActions`$(J_p, \tilde{\mathbf{a}}_{p,0:\tilde{H}-1}, N_{\text{elite}})$
  /* Optimize Policy */

Outputs: Return best action $\mathbf{a} \leftarrow$ `EliteActions`$(J_p, \tilde{\mathbf{a}}_{p,0:\tilde{H}-1})$.

3

EPISTEMIC UNCERTAINTY FOR PROVABLE DATA EFFICIENCY

A hallucination is a fact, not an error – what is erroneous is a judgment based upon it.
— Bertrand Russell

In this chapter, we use the epistemic uncertainty to *efficiently explore* in a dynamical system with continuous states and actions. Data-efficient exploration aims to reduce the number of interactions with the environment needed to learn an optimal policy for a given task. In Section 3.1, we formalize the problem setup and the objective we are trying to solve. In Section 3.2, we describe commonly used exploration strategies, and their shortcomings in practical large-scale settings. In Section 3.3, we present the main contribution of the chapter: the H-UCRL algorithm, and, in Section 3.4, we provide the theoretical analysis of H-UCRL. Finally, in Section 3.5, we demonstrate the practical aspects of the algorithm in experiments. We defer the technical proofs to Appendix A.
The results in this chapter have been previously published in:

- Curi, S., Berkenkamp, F., & Krause, A. *Efficient model-based reinforcement learning through optimistic policy search and planning* in *Advances in Neural Information Processing Systems (NeurIPS)* (2020).

3.1 PROBLEM SETUP

In this chapter, we use the epistemic uncertainty in our model to promote exploration so that the agent trades-off good performance with learning about the dynamics. Consider executing policy π_n at episode n. On the one hand, we hope for π_n to achieve good performance. On the other hand, we hope to learn about the dynamics f to contract its *epistemic* uncertainty. We

formalize this by aiming to find the optimal policy for the true dynamics f in a finite horizon setting,

$$\begin{aligned} \pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} J(f, \pi) = \mathbb{E}_{\tau_{f,\pi}} \left[\sum_{h=0}^{H} r(\mathbf{s}_h, \pi(\mathbf{s}_h)) \right], \\ \text{s.t. } \mathbf{s}_{h+1} = f(\mathbf{s}_h, \pi(\mathbf{s}_h)) + \boldsymbol{\omega}_h \\ \mathbf{s}_0 \sim \nu_0, \end{aligned} \tag{3.1}$$

where $\tau_{f,\pi} = \{(\mathbf{s}_{h-1}, \mathbf{a}_{h-1}), \mathbf{s}_h\}_{h=0}^{H}$ is a random trajectory induced by the stochastic noise $\boldsymbol{\omega}$, the dynamics f, the policy π, and the initial state distribution ν_0.

We cannot solve Equation (3.1) as the system dynamics f is unknown,. However, as the agent collects more data *and* the set of plausible models $\mathcal{M}$ contracts towards the true dynamics, the estimate of the performance (2.2) contracts towards the true performance (3.1). As we do not have enough knowledge to solve this optimization problem, we focus on regret, defined as:

$$\mathfrak{R}_N = \sum_{n=0}^{N} J(f, \pi^\star) - J(f, \pi_n). \tag{3.2}$$

Regret quantifies the performance difference between the unknown optimal policy $\pi^\star$ and the policy applied in each episode π_n on the true system f. An algorithm that chooses the policy π_n has no-regret if $\mathfrak{R}_N = o(N)$. If such algorithm has no-regret, then the sequence of policies $\{\pi_n\}_{n=0}^{N}$ performs asymptotically as the optimal policy $\pi^\star$. The goal is to design a *practical* algorithm with no-regret.

Looking at the regret (3.2), it is clear that the policy π_n should try to maximize the performance on the true dynamics $J(f, \pi_n)$, and at the same time it should try to maximize the knowledge about f, or contract the set $\mathcal{M}$, to bound the performance difference between such policy π_n and the optimal policy $\pi^\star$.

3.2 EXPLORATION STRATEGIES IN REINFORCEMENT LEARNING

Ultimately the performance of any algorithm depends on the choice of π_n. We now provide an overview of existing exploration schemes and summarize the MBRL procedure in Algorithm 1.

GREEDY EXPLOITATION One of the most commonly used algorithms is to select the policy π_n that greedily maximizes the expected performance over the aleatoric uncertainty *and* epistemic uncertainty induced by the dynamical model. Other exploration strategies, such as dithering (e.g., epsilon-greedy, Boltzmann exploration) (Sutton & Barto, 1998) or certainty equivalent control (Bertsekas *et al.*, 1995, Chapter 6.1), can be grouped into this class. The greedy policy is

$$\pi_n^{\text{Greedy}} = \operatorname*{argmax}_{\pi \in \Pi} \mathbb{E}_{\tilde{f} \sim \mathcal{F}_n} \left[J(\tilde{f}, \pi) \right]. \tag{3.3}$$

For example, PILCO (M. Deisenroth & Rasmussen, 2011) and GP-MPC (Kamthe & M. Deisenroth, 2018a) use moment matching to approximate $\mathcal{F}$ and use *greedy* exploitation to optimize the policy. Likewise, PETS-1 and PETS-∞ from Chua *et al.* (2018) also lie in this category, in which $\mathcal{F}$ is represented via ensembles. The main difference between PETS-∞ and other algorithms is that PETS-∞ ensures that the same function is used throughout the rollout, whereas PETS-1, PILCO, and GP-MPC sample a new function at each time step for computational reasons. In the tabular RL setting, dithering takes an exponential number of episodes to find an optimal policy (Osband *et al.*, 2014). As such, it is *not* an efficient exploration scheme for reinforcement learning. Nevertheless, for some specific reward and dynamics structure, such as linear-quadratic control, greedy exploitation indeed achieves no-regret (Mania *et al.*, 2019). In addition, it is the most common exploration strategy and there exist many practical algorithms to efficiently solve optimization problem eq. (3.3) (cf. Section 3.3.1).

THOMPSON SAMPLING A theoretically grounded exploration strategy is Thompson sampling, which optimizes the policy w.r.t. a single model that is sampled from $\mathcal{F}_n$ at every episode n. Formally,

$$\tilde{f}_n \sim \mathcal{F}_n, \quad \pi_n^{\text{TS}} = \operatorname*{argmax}_{\pi \in \Pi} J(\tilde{f}_n, \pi). \tag{3.4}$$

This is different to PETS-∞, as the former algorithm optimizes w.r.t. the average of the (consistent) model trajectories instead of a single model. In general, it is intractable to sample from $\mathcal{F}$. Nevertheless, after the sampling step, the optimization problem is equivalent to greedy exploitation of the sampled model. Thus, the same optimization algorithms can be used to solve Equation (3.3) and Equation (3.4).

Name	Objective Function	Efficient Exploration	Practical Optimization	NN Implementation
Greedy	(3.3)	✗	✓	✓
TS	(3.4)	✓	✓	✗
UCRL	(3.5)	✓	✗	✗
H-UCRL	(3.6)	✓	✓	✓

TABLE 3.1: Comparison of different exploration strategies in reinforcement learning problems. H-UCRL is the only algorithm that is simultaneously provable efficient, has a practical optimization problem, and can be used with large scale models such as Neural Networks.

UPPER-CONFIDENCE REINFORCEMENT LEARNING (UCRL) The final exploration strategy we address is UCRL exploration (Jaksch *et al.*, 2010), which optimizes jointly over policies and models inside the set $\mathcal{M}_n = \{\tilde{f} \text{s.t. } |\tilde{f}(\mathbf{s},\mathbf{a}) - \boldsymbol{\mu}_n(\mathbf{s},\mathbf{a})| \leq \beta_n \boldsymbol{\sigma}_n(\mathbf{s},\mathbf{a}) \, \forall \mathbf{s},\mathbf{a} \in \mathcal{S} \times \mathcal{A}\}$ that contains all statistically-plausible models compatible with Assumption 7. The UCRL algorithm is

$$\pi_n^{\text{UCRL}} = \underset{\pi \in \Pi}{\text{argmax}} \max_{\tilde{f} \in \mathcal{M}_n} J(\tilde{f}, \pi). \tag{3.5}$$

Instead of greedy exploitation, these algorithms optimize an optimistic policy that maximizes performance over all plausible models. Unfortunately, this joint optimization is in general *intractable* and algorithms designed for greedy exploitation (3.3) do *not* generally solve the UCRL objective (3.5).

3.3 HALLUCINATED UPPER CONFIDENCE REINFORMCENT LEARNING (H-UCRL)

We propose a practical variant of the UCRL exploration (3.5) algorithm. Namely, we reparameterize the functions $\tilde{f} \in \mathcal{M}_n$ as $\tilde{f} = \boldsymbol{\mu}_{n-1}(\cdot) + \beta_{n-1}\boldsymbol{\sigma}_{n-1}(\cdot)\eta(\cdot)$, for some function $\eta \in \mathcal{U} \colon \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to [-1,1]^{d_s}$. This transformation is similar in spirit to the re-parameterization trick from Kingma & Welling (2013), except that $\eta(\cdot)$ is a function. The key insight is that instead of optimizing over dynamics in $\tilde{f} \in \mathcal{M}_n$ as in UCRL, it

suffices to optimize over the functions $\eta(\cdot)$. We call this algorithm H-UCRL, formally:

$$\begin{aligned} \pi_n^{\text{H-UCRL}} = \underset{\pi \in \Pi}{\operatorname{argmax}} \max_{\eta(\cdot) \in \mathcal{U}} J(\tilde{f}, \pi), \\ \text{s.t. } \tilde{f}(\cdot) = \boldsymbol{\mu}_{n-1}(\cdot) + \beta_{n-1} \boldsymbol{\sigma}_{n-1}(\cdot) \eta(\cdot). \end{aligned} \tag{3.6}$$

At a high level, the policy π acts on the *inputs* (actions) of the dynamics and chooses the next-state distribution. In turn, the optimization variables η act in the *outputs* of the dynamics to select the most-optimistic outcome from within the confidence intervals. We call the optimization variables the *hallucinated* controls as the agent hallucinates control authority to find the most-optimistic model.

The H-UCRL algorithm *does not explicitly propagate uncertainty* over the horizon. Instead, it does so *implicitly* by using the pointwise uncertainty estimates from the model to recursively plan an optimistic trajectory, as illustrated in fig. 3.1. This has the practical advantage that the model only has to be well-calibrated for 1-step predictions and not H-step predictions. In practice, the parameter β_n trades off between exploration and exploitation.

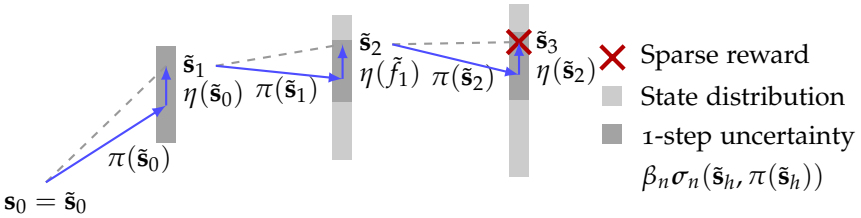


FIGURE 3.1: Illustration of the optimistic trajectory $\tilde{f}_h$ from H-UCRL. The policy π is used to choose the next-state distribution, and the hallucination inputs η to choose the next state optimistically inside the one-step confidence interval (dark grey bars). Greedy exploitation strategies fail because they marginalize the reward over the epistemic uncertainty (red cross compared to light grey bar). Instead, H-UCRL efficiently finds the high-reward (red cross) region by optimizing w.r.t η.

3.3.1 *Solving the Optimization Problem*

Problem (3.6) is still intractable as it requires to optimize over general functions. The *crucial* insight is that we can make the H-UCRL (3.6) practical by

optimizing over a smaller class of functions η. In Section 2.2.5, we prove that it suffices to optimize over Lipschitz-continuous bounded functions instead of general bounded functions. Therefore, we can optimize jointly over policies and Lipschitz-continuous, bounded *functions* $\eta(\cdot)$. As we consider deterministic policies, we re-write $\eta(\mathbf{s}_h, \mathbf{a}_h) = \eta(\mathbf{s}_h, \pi(\mathbf{s}_h)) = \eta(\mathbf{s}_h)$. This allows to reduce the intractable optimistic problem (3.6) to *greedy exploitation* (3.3): We simply treat $\eta(\cdot) \in \mathcal{U}$ as an additional *hallucinated* control input that has no associated control penalties. Furthermore, the hallucination policy η can exert as much control as the current *epistemic* uncertainty that the model affords. With this observation in mind, H-UCRL greedily exploits a *hallucinated* system with the extended dynamics $\tilde{f}$ in Equation (3.6) and a corresponding augmented control policy (π, η). This means that we can now use the *same* efficient MBRL approaches for optimistic exploration that were previously restricted to greedy exploitation and Thompson sampling (albeit on a larger action space, since the dimension of the action space increases from d_a to $d_a + d_s$).

In practice, if we have access to a greedy oracle $\pi = \texttt{GreedyOracle}(\tilde{f})$, we simply access it using $\pi, \eta = \texttt{GreedyOracle}(\boldsymbol{\mu}_{n-1} + \beta_{n-1}\boldsymbol{\sigma}_{n-1}\eta)$. Broadly speaking, greedy oracles are implemented using offline-policy search or online planning algorithms. Next, we discuss how to use these strategies independently to solve the H-UCRL planning problem (3.6).

Offline Policy Search is any algorithm that optimizes a parametric policy to maximize performance of the current dynamical model. As inputs, it takes the dynamical model and a parametric family for the policy and the critic (the value function). It outputs the optimized policy and the corresponding critic of the optimized policy. These algorithms have fast inference time and scale to large dimensions but can suffer from model bias and inductive bias from the parametric policies and critics (van Hasselt *et al.*, 2019).

Online Planning or Model Predictive Control (Morari & H. Lee, 1999) is a local planning algorithm that outputs the best action for the current state. This method solves the H-UCRL planning problem (3.6) in a receding-horizon fashion. The planning horizon is usually shorter than H and the reward-to-go is bootstrapped using a terminal reward. In most cases, however, this terminal reward is unknown and must be learned (Lowrey *et al.*, 2019). As the planner observes the *true* transitions during deployment, it suffers less from model errors. However, its running time is too slow for real-time implementation.

Algorithm 8 H-UCRL combining Optimistic Policy Search and Planning

Inputs: Mean $\boldsymbol{\mu}(\cdot,\cdot)$ and variance $\boldsymbol{\sigma}^2(\cdot,\cdot)$, policies $\pi_\theta(\cdot)$ and $\eta_\theta(\cdot)$, critic $Q_\vartheta(\cdot)$, horizon H, policy search algorithm `PolicySearch`, planning algorithm `Plan` .

1: **for** $n = 1, 2, \ldots$ **do**
2: $(\pi_{\theta,n}, \eta_{\theta,n}), Q_{\vartheta,n} \leftarrow$ `PolicySearch`$(\boldsymbol{\mu}_{n-1}; \boldsymbol{\sigma}^2_{n-1}; (\pi_{\theta,n-1}, \eta_{\theta,n-1}))$
3: **for** $h = 1, \ldots, H$ **do**
4: $(\mathbf{a}_{h-1,n}, \mathbf{u}_{h-1,n}) =$ `Plan`$(\mathbf{s}_{h-1,n}; \boldsymbol{\mu}_{n-1}; \boldsymbol{\sigma}^2_{n-1}; (\pi_{\theta,n}, \eta_{\theta,n}), Q_\vartheta)$
5: $\mathbf{s}_{h,n} = f(\mathbf{s}_{h-1,n}, \mathbf{a}_{h-1,n}) + \boldsymbol{\omega}_{h-1,n}$
6: Update statistical model with the H observed transitions in $\mathcal{D}_n$.

COMBINING OFFLINE POLICY SEARCH WITH ONLINE PLANNING In Algorithm 8, we propose to combine the best of both worlds to solve the H-UCRL planning problem (3.6). In particular, Algorithm 8 takes as inputs a policy search algorithm and a planning algorithm. After each episode, it optimizes parametric (e.g. neural networks) control and hallucination policies $(\pi_\theta, \eta_\theta)$ using the policy search algorithm. As a by-product of the policy search algorithm we have the *learned* critic Q_ϑ. At deployment, the planning algorithm returns the true and hallucinated actions $(\mathbf{a}, \mathbf{a}')$, and we only execute the true action $\mathbf{a}$ to the true system. We initialize the planning algorithm using the learned policies $(\pi_\theta, \eta_\theta)$ and use the *learned* critic to bootstrap at the end of the prediction horizon as the reward-to-go. In this way, we achieve the best of both worlds. The policy search algorithm accelerates the planning algorithm by shortening the planning horizon with the learned critic and by using the learned policies to warm-start the optimization. The planning algorithm reduces the model-bias that a pure policy search algorithm has.

3.4 THEORETICAL ANALYSIS

In this section, we analyze the H-UCRL algorithm (3.6). A natural quality criterion to evaluate exploration schemes is the *cumulative regret* (3.2). If we can show that $\mathfrak{R}_N$ is sublinear in N, then we know that the performance $J(f, \pi_n)$ of our chosen policies π_n converges to the performance of the optimal policy π^*.

MODEL COMPLEXITY In general, we expect that the regret $\mathfrak{R}_N$ depends on the complexity of the statistical model in Assumption 7. If we can quickly

estimate the true model using a few data-points, then the regret would be lower than if the model is slower to learn. To account for these differences, we use the following complexity measure introduced in Equation (2.11),

$$\Gamma_N = \max_{\tilde{\mathcal{D}}_{1:N}} \sum_{n=1}^{N} \sum_{\mathbf{s},\mathbf{a} \in \tilde{\mathcal{D}}_n} \|\sigma_{n-1}(\mathbf{s}, \mathbf{a})\|_2^2. \tag{3.7}$$

While in general impossible to compute, this complexity measure considers the "worst-case" datasets $\mathcal{D}_1$ to $\mathcal{D}_N$, with $|\mathcal{D}_n| = H$ elements each, that we could collect at each iteration of Algorithm 1 in order to maximize the predictive uncertainty of our statistical model. Intuitively, if $\sigma(\mathbf{s}, \mathbf{a})$ shrinks sufficiently quickly after observing a transition and if the model generalizes well over $\mathcal{S} \times \mathcal{A}$, then the model complexity Γ in Equation (2.11) will be small. In contrast, if our model does not learn or generalize at all, then Γ_N will be $\mathcal{O}(NHd_s)$ and we cannot hope to succeed in finding the optimal policy. For the special case of Gaussian process (GP) models, we use the results in Section 2.2.2 to show that Γ_N is indeed sublinear in N.

GENERAL REGRET BOUND The true sequence of states $\mathbf{s}_{h,n}$ at which we obtain data during our rollout in Algorithm 1 lies somewhere withing the light-gray shaded state distribution with epistemic uncertainty in Figure 3.1.

While this is generally difficult to compute, we can bound it in terms of the predictive variance $\sigma_{n-1}(\mathbf{s}_{h,n}, \pi_n(\mathbf{s}_{h,n}))$, which is directly related to Γ_H. However, the optimistically planned trajectory instead depends on $\sigma_{n-1}(\tilde{\mathbf{s}}_{h,n}, \pi(\tilde{\mathbf{s}}_{h,n}))$ in Equation (3.6), which enables policy optimization without explicitly constructing the state distribution. How the predictive uncertainties of these two trajectories relate depends on the generalization properties of our statistical model; specifically on L_σ in Assumption 6. We can use this observation to obtain the following bound on $\mathfrak{R}_N$:

Theorem 1. *Under Assumptions 1 to 7, let* $C = (1 + L_f + 2L_\sigma)(1 + L_\pi^2 + L_{\hat{\pi}}^2)^{1/2}$ *and* $\mathbf{s}_{h,n} \in \mathcal{S}$ *and* $\mathbf{a}_{h,n} \in \mathcal{A}$ *for all* $h, n > 0$. *Then, for all* $N \geq 1$, *with probability at least* $(1 - \delta)$, *the regret of* H-UCRL *in eq.* (3.6) *is at most*

$$\mathfrak{R}_N \leq \mathcal{O}\Big(L_r C^H \beta_N^H H^{3/2} \sqrt{N \Gamma_N}\Big). \tag{3.8}$$

We provide a proof of theorem 1 in appendix A. The theorem ensures that, if we evaluate optimistic policies according to Equation (3.6), we eventually achieve performance $J(f, \pi_n)$ arbitrarily close to the optimal performance of $J(f, \pi^*)$ if $\Gamma_N(\mathcal{S}, \mathcal{A})$ grows at a rate smaller than N. As

one would expect, the regret bound in Theorem 1 depends on constant factors like the prediction horizon H, the relevant Lipschitz constants of the dynamics, policy, reward, and the predictive uncertainty. The dependence on the dimensionality of the state space d_s is hidden inside Γ_N, while β_N is a function of δ.

GAUSSIAN PROCESS MODELS For the bound in Theorem 1 to be useful, we must show that Γ_N is sublinear in N. Proving this is impossible for general models, but can be proven for GP models. In particular, we show in Section 2.2.2 that Γ_N is bounded by the worst-case mutual information (information capacity) of the GP model. Srinivas *et al.* (2012) and Krause & Ong (2011) derive upper-bounds for the information capacity for commonly-used kernels. For example, when we use their results for independent GP models with squared exponential kernels for each component $[f(\mathbf{s}, \mathbf{a})]_i$, we obtain a regret of $\mathcal{O}\Big((1 + B_f)^N L_\sigma^N H^{3/2} (d_s^2 (d_s + d_a) \log(d_s N H))^{(H+1)/2}\Big)$, where B_f is a bound on the functional complexity of the function f. Specifically, B_f is the norm of f in the RKHS that corresponds to the kernel.

A similar optimistic exploration scheme was analyzed by Chowdhury & Gopalan (2019), but for an algorithm that is not implementable as we discussed at the beginning of section 3.3. Their exploration scheme *depends* on the (generally unknown) Lipschitz constant of the value function, which corresponds to knowing L_f *a priori* in our setting. While this is a restrictive and impractical requirement, we show in Curi *et al.* (2020a, Appendix H.3) that under this assumption we can improve the dependence on $L_\sigma^H \beta_N^H$ in the regret bound in theorem 1 to $(L_f \beta_N)^{1/2}$. This matches the bounds derived by Chowdhury & Gopalan (2019) up to constant factors. Thus we can consider the regret term $L_\sigma^H \beta_N^H$ to be the additional cost that we have to pay for a practical algorithm.

3.5 EXPERIMENTS

We now evaluate H-UCRL in a set of different environments. Throughout the experiments, we consider reward functions of the form $r(\mathbf{s}, \mathbf{a}) = r_{\text{state}}(\mathbf{s}) - \rho c_{\text{action}}(\mathbf{a})$, where $r_{\text{state}}(\mathbf{s})$ is the reward for being in a "good" state, and $\rho \in [0, \infty)$ is a parameter that scales the action costs $c_{\text{action}}(\mathbf{a})$. We specify $r_{\text{state}}(\mathbf{s})$ and $c_{\text{action}}(\mathbf{a})$ for each environment later. We evaluate how H-UCRL, greedy exploitation, and Thompson sampling perform for different values of ρ in different environments. We expect greedy exploitation to struggle for larger ρ and find the local optima with $\mathbf{a} = 0$,

whereas H-UCRL and Thompson sampling should perform well. We provide an open-source implementation of our method, which is available at `http://github.com/sebascuri/hucrl`. In Section 3.5.1, we evaluate H-UCRL in a small scale setting to provide insight about the algorithm and, in Section 3.5.2, we evaluate RH-UCRL in a large scale setting to demonstrate its scalability. Finally, in Section 3.5.3, we provide experimental ablations to understand why Thompson Sampling underperforms compared to H-UCRL.

In both small and large-scale experiments, we use 5-head probabilistic ensembles (PE) of NN as in Chua *et al.* (2018). For greedy exploitation, we sample the next-state from the ensemble mean and covariance (PE-DS algorithm in Chua *et al.* (2018)). We use ensemble sampling (Lu & Van Roy, 2017) to approximate Thompson sampling. For H-UCRL, we follow Lakshminarayanan *et al.* (2017b) and use the ensemble mean and covariance as the next-state predictive distribution. In small scale experiments, we also evaluate deterministic ensembles (DE) and GP models with RBF kernels.

3.5.1 *Small-Scale: Sparse Inverted Pendulum*

We first investigate a swing-up pendulum with sparse rewards. In this task, the policy must perform a complex maneuver to swing the pendulum to the upwards position. A policy that does not act obtains zero state rewards but suffers zero action costs. Slightly moving the pendulum still has zero state reward but the actions are penalized. Hence, a zero-action policy is locally optimal, but it fails to complete the task.

The pendulum has $d_s = 2$ and $d_a = 1$, with actions bounded in $[-1, 1]$ and each episode lasts 400 time steps. We transform the angles to a quaternion representation via $[\sin(\theta), \cos(\theta)]$. The pendulum starts at $\theta_0 = \pi$, $\omega_0 = 0$ and the objective is to swing it up to $\theta_0 = 0$, $\omega_0 = 0$. The reward function is $r(\theta, \omega, \mathbf{a}) = r_\theta \cdot r_\omega - \rho c_{\mathbf{a}}$, where $r_\theta = \textsc{tolerance}(\cos(\theta), \mathrm{b} = (0.95, 1.), \mathrm{m} = 0.1)$, $r_\omega = \textsc{tolerance}(\omega, \mathrm{b} = (-0.5, 0.5), \mathrm{m} = 0.5)$, and $c_{\mathbf{a}} = -\textsc{tolerance}(\mathbf{a}, \mathrm{b} = (-0.1, 0.1), \mathrm{m} = 0.1) - 1$. The TOLERANCE is defined in Tassa *et al.* (2018).

We show the final results in Figure 3.2 for different models and the learning curves for PE models in Figure 3.3. With no action penalty, all exploration methods perform equally well – the randomness is enough to explore and find a quasi-optimal sequence. For $\rho = 0.1$, greedy exploitation struggles: sometimes it finds the swing-up sequence, which explains the

large error bars. Finally, for $\rho = 0.2$ only H-UCRL is able to successfully swing up the pendulum.

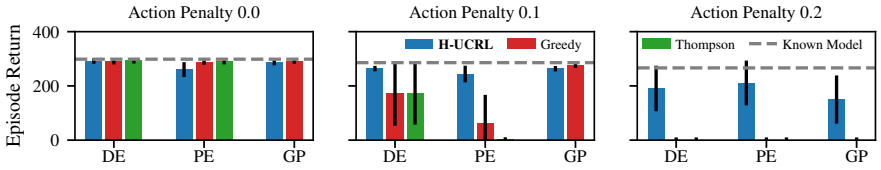


FIGURE 3.2: Final returns in an inverted pendulum swing-up task with sparse rewards. As the action penalty increases, exploration through noise is penalized and algorithms get stuck in a local minimum, where the pendulum is kept at the bottom position. Instead, H-UCRL is able to solve the swing-up task reliably. This holds for for all considered dynamical models: Deterministic- (DE) and Probabilistic Ensembles (PE) of neural networks as well as Gaussian Processes (GP) models.

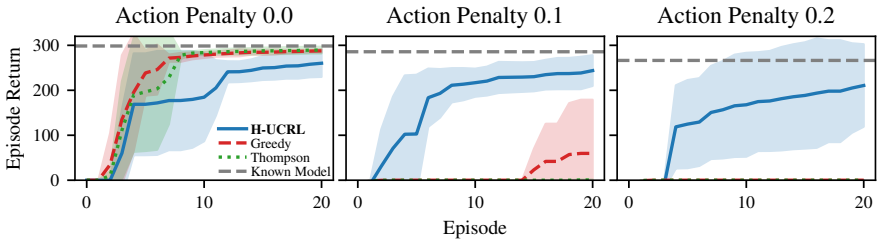


FIGURE 3.3: Learning curves of the inverted pendulum with Probabilistic Ensemble dynamical models. H-UCRL outperforms other algorithms during learning.

In Figure 3.4, we plot the real and simulated trajectories of the H-UCRL algorithm. In the first episode, data is collected only around the bottom position and the agent learns a model, as shown in the first subplot, left figure. Next, it plans an optimistic trajectory that reaches the top-up position, as shown in the first subplot, right figure. In the second episode, the agent intends to execute the planned trajectory but it fails – the planned trajectory was *too* optimistic and it did not manage to swing up the pendulum. In the fourth episode, the agent succesfully swings-up the pendulum, but it is not able to break as no data was collected in the top-up position. Finally, after six episodes, the planned and the simulated trajectories almost match perfectly.

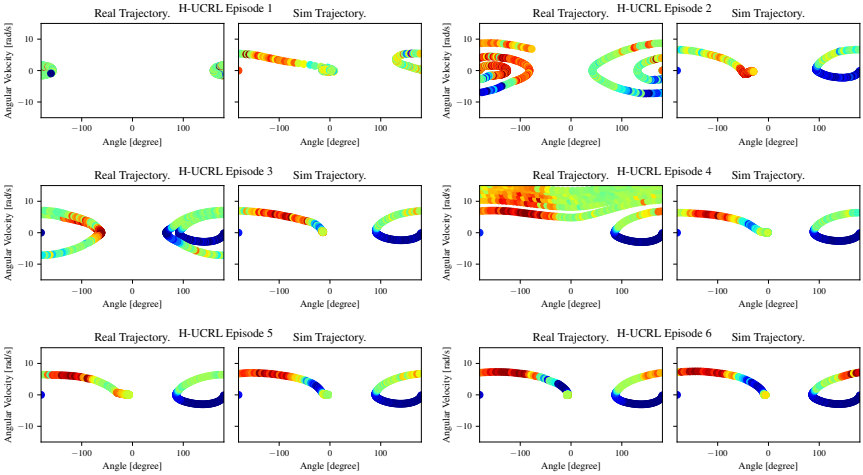


FIGURE 3.4: Real and simulated trajectories for the first 6 episodes with H-UCRL (0.2 action penalty). We plot the trajectory in phase space, and use color coding to denote the action magnitude.

3.5.2 *Large-Scale: Mujoco Tasks*

Next, we evaluate how H-UCRL performs in higher-dimensional problems, namely, the Reacher and Pusher environments proposed by Chua *et al.* (2018), and the standard Mujoco Half-Cheetah (Todorov *et al.*, 2012). We plot the results in Figure 3.5.

7-DOF PR2 ROBOT The PR2 robot is a 7DOF robot with $d_s = 14$ and $d_a = 7$, with actions bounded in $[-20, 20]^{d_a}$ and each episode lasts 150 time steps for 50 episodes. We consider three tasks: pusher, reacher and sparse reacher. To learn the model, we transform the angles to a quaternion representation via $[\sin(\theta), \cos(\theta)]$. We plot the final results in Figure 3.5 and the learning curves in Figure 3.6.

In the reacher task, the goal is to move the robot's end-effector to a goal sampled at the beggining of each episode at location $(x, y, z) = (0, 0.25, 0) + \omega$, where ω is a zero-mean normal noise with 0.1 standard deviation. The reward signal is $r = -\sum_{i=x,y,z} (\text{ee} - \text{goal})_i^2 - \rho \sum_{i=1}^{7} \mathbf{a}_i^2$, where $\text{ee} - \text{goal}$ is the vector that measures the distance between the end-effector and the goal. For the sparse reacher, we use a reward signal given by $r = e^{-\sum_{i=x,y,z} (\text{ee}-\text{goal})_i^2 / 0.45^2} + \rho(e^{-\sum_{i=1}^{7} \mathbf{a}_i^2} - 1)$.

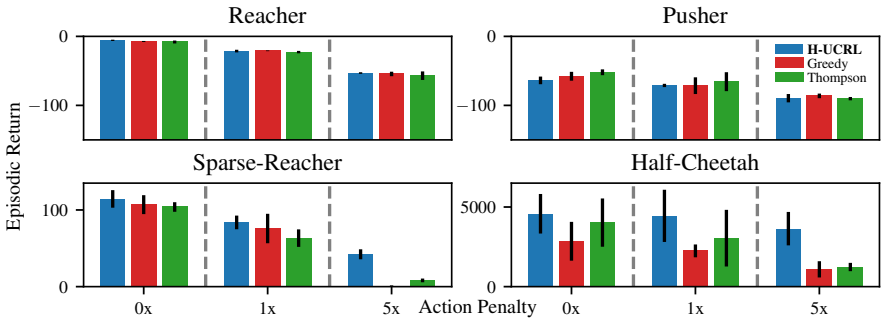


FIGURE 3.5: Mean final episodic returns on Mujoco tasks averaged over five different random seeds. For Reacher and Pusher (50 episodes), all exploration strategies perform equally. For Sparse-Reacher (50 episodes) and Half-Cheetah (250 episodes), H-UCRL outperforms other exploration algorithms.

In the pusher task, the goal is to move a puckle to the $(0, 0)$ location. As the object is free to move, this introduces 3 more states to the environment. The robot starts with zero angles, an angular velocity sampled uniformly at random from $[-0.005, 0.005]$, and the puckle is sampled from $(x, y) = (-0.25, 0.15) + \omega$, where ω is a zero-mean normal noise with 0.025 standard deviation. The reward signal is given by $r = -0.5 \sum_{i=x,y,z} (\text{ee} - \text{obj})_i^2 - 1.25 \sum_{i=x,y,z} (\text{obj} - \text{goal})_i^2 - \rho \sum_{i=1}^{7} \mathbf{a}_i^2$, where $\text{ee} - \text{obj}$ is the distance between the end-effector and the object and $\text{obj} - \text{goal}$ is the distance between the object and the goal.

All exploration strategies achieve state-of-the-art performance in the pusher and reacher environment, which seems to indicate that greedy exploitation is indeed sufficient for these tasks. Presumably, this is due to the over-actuated dynamics and the reward structure. This is in line with the theoretical results for linear-quadratic control by Mania *et al.* (2019). However, on the sparse reacher task, H-UCRL outperforms alternative methods, particularly for larger action penalties, showing that exploration is beneficial in the sparse-reward setting.

HALF-CHEETAH Our final experiment demonstrates H-UCRL on a common deep-RL benchmark, the Half-Cheetah. The Half-Cheetah is a mobile robot with $d_s = 17$ and $d_a = 6$, with actions bounded in $[-2, 2]^{d_a}$ and each episode lasts 1000 time steps for 250 episodes. The objective is to make the cheetah run as fast as possible forwards up to a maximum of 10m/s.

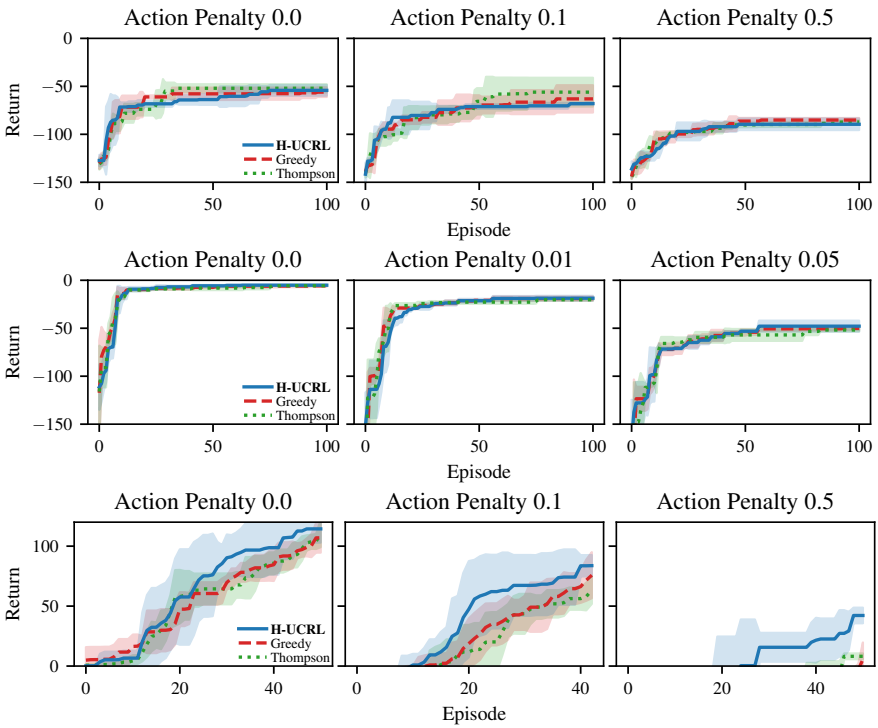


FIGURE 3.6: Learning curves in 7-DOF PR2 Robot using PE models for pusher, reacher, and sparse-reacher tasks. In the pusher and reacher tasks Greedy, Thompson sampling, and H-UCRL perform equally well. H-UCRL outperforms greedy and Thompson sampling, particularly when the action penalty increases in the sparse-reacher task.

The reward function is given by $r = \max(v, 10)$ and the actuation costs are $c_{\mathbf{a}} = -\mathbf{a}^2$. The actuators have to interact in a complex manner to achieve running. In Figure 3.7, we can see a clear advantage of using H-UCRL at different action penalties, even at zero. This indicates that H-UCRL not only addresses action penalties, but also explores through complex dynamics.

3.5.3 *Further Experiments on Thompson Sampling*

Surprisingly, Thompson Sampling under-performs compared to optimistic exploration. To understand better why this happens, we perform different experiments in this section.

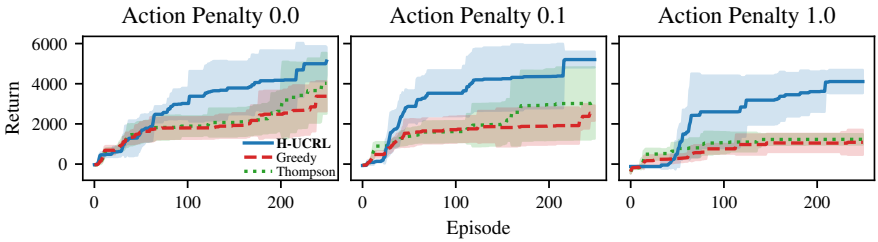


FIGURE 3.7: Learning curves in Half-Cheetah environment. For all action penalties, H-UCRL learns faster than greedy and Thompson sampling strategies. For larger action penalties, greedy and Thompson lead to insufficient exploration and get stuck in local optima with poor performance.

3.5.3.1 *Can the sampled models solve the task?*

One possibility is that, when doing posterior sampling, the agent learns a policy for the sampled model which might be biased. If this was the case, we would expect to see large *simulated* returns, i.e., the returns of the optimal policy in the sampled system $\tilde{f}_i$.

In Figure 3.8, we show the returns of the last simulated trajectory starting from the bottom position of each episode. This figure indicates that there is no model bias, i.e., the simulated returns for Thompson sampling are also low. We conclude that it is not over-fitting to the sampled model, but rather, the algorithm cannot solve the task with the sampled model.

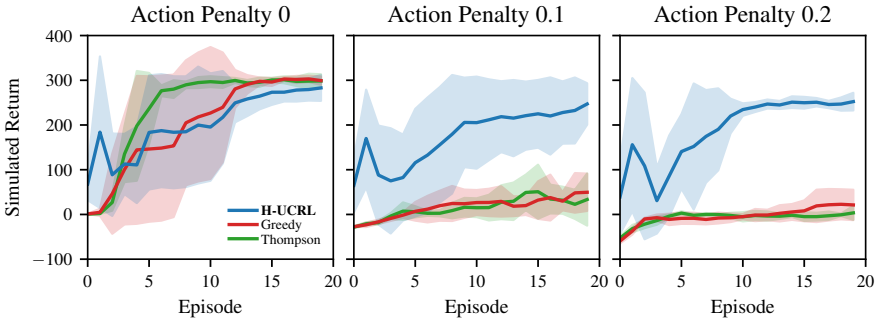


FIGURE 3.8: Total return from last simulated trajectory with the same initial state as the environment initial state. H-UCRL has higher simulated returns than Greedy and Thompson as the action penalty increases.

3.5.3.2 *Is it variance starvation?*

Another possibility is Thompson Sampling suffers variance starvation, i.e., all ensemble members' predictions are identical. Variance starvation means that the approximate posterior variance is smaller than the true posterior variance. When this happens, (approximate) Thompson Sampling fails because of a lack of exploration (Z. Wang *et al.*, 2018). In contrast to UCRL-stye algorithms where the optimism is implemented *deterministically*, Thompson sampling implements optimism *stochastically*. Thus, it is crucial that the variance is not underestimated.

If there was variance starvation, we would expect to see the epistemic variance along simulated trajectories shrink. In Figure 3.9 we show the average simulated uncertainty during training, considered as the predictive variance of the ensemble. To summarize the predictive uncertainty into a scalar, we consider the trace of the Cholesky factorization of the covariance matrix. From the figure, we see that H-UCRL starts with the same predictive uncertainty as greedy and Thompson sampling. Furthermore, the variance of Thompson sampling does not shrink. We conclude that there is no variance starvation in the one-step ahead predictions.

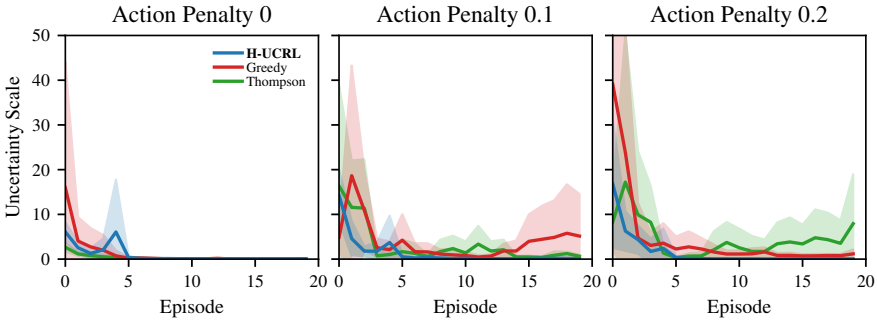


FIGURE 3.9: Epistemic model uncertainty along simulated trajectories. Thompson and Greedy have the same or more uncertainty than H-UCRL.

3.5.3.3 *Is the number of ensemble members enough?*

In order to verify this hypothesis, we ran the same experiments with 5, 10, 20, 50, and 100 ensemble members. All models swing-up the pendulum with 0 action penalty. With 0.1 action penalty, the 20, 50, and 100 ensembles find a swing up in only one run out of five. With 0.2 action penalty, no model finds

a swing-up strategy. This suggests that having larger ensembles could help, but it is not convincing. Furthermore, the model training computational complexity increases linearly with the number of ensemble members, which limits the practicality of larger ensembles.

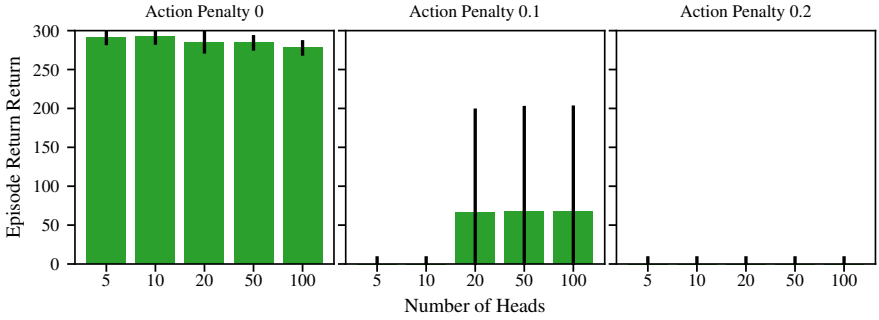


FIGURE 3.10: Episodic returns using Thompson Sampling for different number of ensemble members

3.5.3.4 *Is it the bootstrapping procedure during Training?*

Yet another possibility is that the bootstrap procedure yields inconsistent models for Thompson sampling. To simulate bootstrapping, for each transition and ensemble member, we sample a mask from a Poisson distribution (Osband *et al.*, 2016). Then, we train using the loss of each transition multiplied by this mask. This yields correct one-step ahead confidence intervals. However, the model is used for multi-step ahead predictions. To test if this is the reason of the failure we repeat the experiment *without* simulated bootstrapping the transitions. The only source of discrepancy between the models comes from the initialization of the model. This is how Chua *et al.* (2018) train their probabilistic models and the models learn from *consistent* trajectories.

In Figure 3.11 we show the results when training without simulated bootstrapping. The learning curves closely follow those with bootstrapping in Figure 3.3. We conclude that the bootstrapping procedure is likely not the cause of the failure of Thompson Sampling.

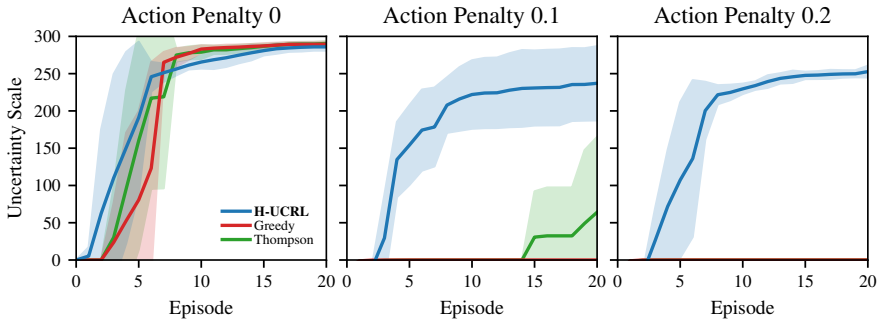


FIGURE 3.11: Episodic Returns in inverted pendulum *without* bootstrapping data while learning the model.

3.5.3.5 *Are probabilistic ensembles not a good approximation to the posterior in Thompson sampling?*

We next investigate the possibility that Probabilistic Ensembles are not a good approximation for $\mathcal{F}$. To this end, we consider the Random Fourier Features (RFF) proposed by Rahimi & Recht (2008) for GP Models. To sample a posterior, we sample a set of random features and use the same features throughout the episodes as required by theoretical results for Thompson sampling and suggested by Hewing *et al.* (2019) to simulate trajectories. RFFs, however, are known to suffer from variance starvation. We also consider Quadrature Fourier Features (QFF) proposed by Mutny & Krause (2018). QFFs have provable no-regret guarantees in the Bandit setting as well as a uniform approximation bound.

In Figure 3.12, we show the results for both RFF (1296 features), and QFFs (625 features). Neither QFFs nor RFFs find a swing-up maneuver for action penalties larger than zero, whereas optimistic exploration with both QFFs and RFFs do. For no action penalty, optimistic exploration with RFFs underperforms compared to greedy exploitation and Thompson sampling. This might be due to variance starvation of RFFs because we do not see the same effect on QFFs. We conclude that PEs are as good as other approximate posterior methods such as random feature models.

3.5.3.6 *Is it the optimization procedure?*

We run optimistic exploration with five ensemble heads and save snapshots of the models after the first, fifth and tenth episode. Then, we optimize a

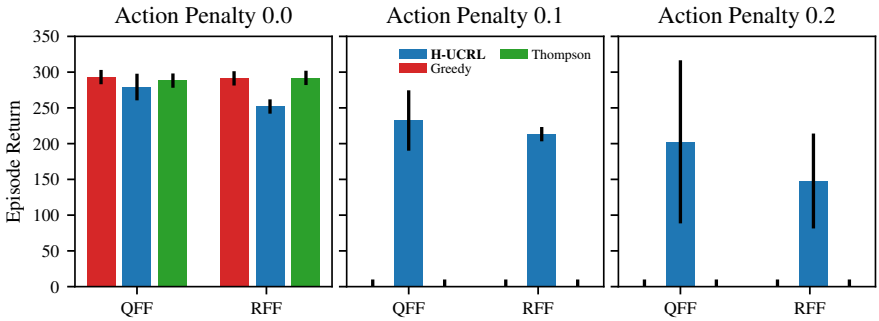


FIGURE 3.12: Episodic Returns in inverted pendulum using Random Fourier Features (RFF) and Quadrature Fourier Features (QFF).

different policy for each of the models separately. In Figure 3.13 we compare the simulated returns using optimistic exploration on the ensemble at each episode against the *maximum* return obtained by the best head.

After the first episode, the simulated returns using optimistic exploration always find an optimistic swing-up trajectory, whereas the optimizing all the heads and getting the best-performing one always returns zero. This indicates that, when the uncertainty is large, optimistic exploration finds a better policy than approximate Thompson sampling. Without the action penalty, the best head return quickly catches up to the simulated ones with optimistic exploration. For an action penalty of 0.1, after five episodes the best head is not able to find a swing-up trajectory. However, after ten episodes it does. This shows that the H-UCRL optimization algorithm (3.6) is able to find the policy that swings-up a single model. However, when Thompson sampling is used to collect data, the optimization does not find such a policy. This indicates that the models learned using H-UCRL better reduce the uncertainty around the high-reward region and each member of the ensemble has *sharper* predictions. For 0.2 action penalty, the best head never finds a swing-up policy in ten episodes.

3.5.3.7 *Conclusions*

We believe that the poor performance of Thompson sampling relative to H-UCRL suggests that a probabilistic ensemble with five members is sufficient to construct reasonable confidence intervals (hence H-UCRL finds good policies), but does not comprise a rich enough posterior distribution for Thompson Sampling. Phan *et al.* (2019) in the Bandit Setting and Kakade *et*

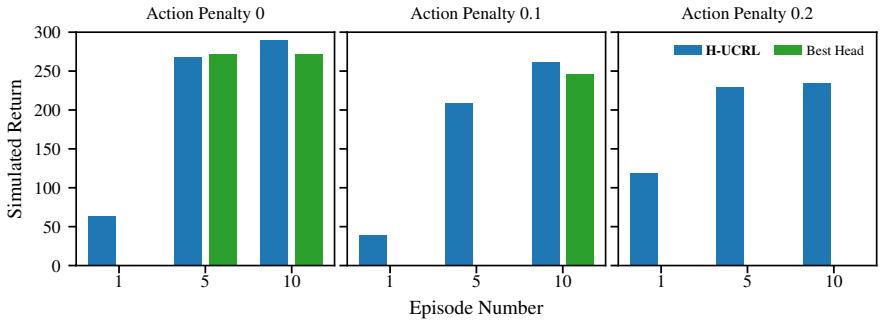


FIGURE 3.13: Simulated Returns using H-UCRL vs. Maximum simulated return over all ensemble members using the same model as H-UCRL.

al. (2020) in the RL setting also report that approximate Thompson sampling fails unless strong modelling priors are used.

We suspect that this effect is inherent to the multi-step RL setting. It seems to be the case that an approximate posterior model whose variance is rich enough for one-step predictions does not sufficiently cover the diversity of plausible trajectories in the multi-step setting. As an example, in H-UCRL we use the five members of the ensemble to construct the 1-step ahead confidence interval at every time-step. On the other hand, in Thompson sampling we sample a *single* model from the *approximate* posterior for the full horizon. It is possible that in some regions of the state-space one member is more optimistic than others, and in a different region the situation reverses. This is not only a property of ensembles, but also other approximate models such as random-feature GP models (c.f. Section 3.5.3.5) that exhibit the same behaviour. Thompson sampling implements optimism *stochastically*: for it to work, we must be able to sample a model that solves the task using multi-step predictions. Designing *tractable* approximate posteriors with *sufficient* variance for multi-step prediction is still a challenging problem. For instance, an ensemble model with I members that has sufficient variance for 1-step predictions, requires I^N members for N-step predictions, this quickly becomes intractable.

Compared to Thompson sampling, UCRL algorithms in general, and H-UCRL in particular, only require one-step ahead calibrated predictive uncertainties in order to successfully implement optimism. This is because the optimism is implemented *deterministically* and it can be used recursively in a computationally efficient way. Furthermore, we know how to train and calibrate models (c.f. Malik *et al.* (2019)) to capture the uncertainty. Due

to the multi-step nature of the problem, constructing *scalable* approximate posteriors that have enough variance to sufficiently explore is still an open problem. This hints that optimism might be better suited than approximate Thompson sampling in MBRL.

4

EPISTEMIC UNCERTAINTY FOR PROVABLE ROBUSTNESS

All the adversity I've had in my life, all my troubles and obstacles, have strengthened me. You may not realize it when it happens, but a kick in the teeth may be the best thing in the world for you.

— Walt Disney

In this chapter, we use the epistemic uncertainty to output a *provably robust* policy in a dynamical system with continuous states and actions. To model robustness, we consider an adversary that acts on the environment together with the agent, albeit with possibly different action spaces. The high-level objective is to output a policy $\hat{\pi}$ that performs well even in the presence of such adversary. In Section 4.1, we formalize the problem setup and the objective that we are trying to solve. In Section 4.2, we present the main contribution of the section: the RH-UCRL algorithm and, in Section 4.3, we provide the theoretical analysis of RH-UCRL. Finally, in Section 4.4, we show the practical aspects of the algorithm in experiments. We defer the technical proofs to Appendix B.
The results in this chapter have been previously published in:

- Curi, S., Bogunovic, I., & Krause, A. *Combining Pessimism with Optimism for Robust and Efficient Model-Based Deep Reinforcement Learning* in *International Conference on Machine Learning (ICML)* (2021).

4.1 PROBLEM SETUP

We consider a stochastic environment with states $\mathbf{s} \in \mathcal{S} \subseteq \mathbb{R}^{d_s}$, agent actions $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}^{d_a}$, adversary actions $\hat{\mathbf{a}} \in \hat{\mathcal{A}} \subset \mathbb{R}^{d_{\hat{a}}}$, and *i.i.d.* additive transition noise vector $\boldsymbol{\omega}_h \in \mathbb{R}^{d_s}$. Both action sets are assumed to be compact, and the dynamics are given by:

$$\mathbf{s}_{h+1} = f(\mathbf{s}_h, \mathbf{a}_h, \hat{\mathbf{a}}_h) + \boldsymbol{\omega}_h \tag{4.1}$$

with $f \colon \mathcal{S} \times \mathcal{A} \times \hat{\mathcal{A}} \to \mathcal{S}$. We assume the true dynamics f are *unknown* and consider the episodic setting over a finite time horizon H. After every

episode (i.e., every H time steps), the system is reset to a known state $\mathbf{s}_0$. In this work, we assume that the dynamics is Lipschitz continuous using Assumption 1. Furthermore, we assume that the adversarial policy is Lipschitz continuous with constant $L_{\hat{\pi}}$.

Before stating our main theoretical results, we add an additional assumption on the adversarial policy:

Assumption 8 (Lipschitz continuity of Adversary policy)**.** The adversary policy $\hat{\pi}_n \in \hat{\Pi}$ is Lipschitz continuous with constant $L_{\hat{\pi}}$.

The previous assumption is mild and complements Assumption 4. The policy class $\hat{\Pi}$ is typically known and designed in a way that is compatible with the previous assumption.

At every time-step, the system returns a deterministic reward $r(\mathbf{s}_h, \mathbf{a}_h, \hat{\mathbf{a}}_h)$, where $r : \mathcal{S} \times \mathcal{A} \times \hat{\mathcal{A}} \to \mathbb{R}$ is assumed to be known to the agent. As discussed in Chapter 2, when the rewards are not known, then these could also be learned. The algorithms in this chapter still work using the simulation dynamics in Equation (2.35) instead of the one in Equation (2.33).

We consider time-homogeneous agent policies $\pi \in \Pi$, $\pi \colon \mathbf{s} \to \mathbf{a}$, that select actions according to $\mathbf{a}_h = \pi(\mathbf{s}_h)$. Similarly, we consider adversary policies $\hat{\pi} \in \hat{\Pi}$ on the common state space, i.e., $\hat{\pi} \colon \mathcal{S} \to \hat{\mathcal{A}}$, that select actions as $\hat{\mathbf{a}}_h = \hat{\pi}(\mathbf{s}_h)$. Our method also supports time-indexed policies by considering H different agent polices $\pi_h \in \Pi$ and adversarial policies $\hat{\pi}_h \in \hat{\Pi}$. We omit this for the sake of notational simplicity. For now, we leave both Π and $\hat{\Pi}$ unspecified, but in Section 4.4, we parameterize them via neural networks.

The performance of a pair of policies $(\pi, \hat{\pi})$ on a given dynamical system $\tilde{f}$ is the episodic expected sum of returns:

$$\begin{aligned} J(\tilde{f}, \pi, \hat{\pi}) &\coloneqq \mathbb{E}_{\tau_{\tilde{f},\pi,\hat{\pi}}}\left[\sum_{h=0}^{H} r(\mathbf{s}_h, \mathbf{a}_h, \hat{\mathbf{a}}_h)\right], \\ \text{s.t. } \mathbf{s}_{h+1} &= \tilde{f}(\mathbf{s}_h, \mathbf{a}_h, \hat{\mathbf{a}}_h) + \tilde{\boldsymbol{\omega}}_h, \\ \tilde{\mathbf{s}}_0 &\sim \nu_0, \end{aligned} \tag{4.2}$$

where $\tau_{\tilde{f},\pi,\hat{\pi}} = \{(\mathbf{s}_{h-1}, \mathbf{a}_{h-1}, \hat{\mathbf{a}}_{h-1}), \mathbf{s}_h\}_{h=0}^{H}$ is a random trajectory induced by the stochastic noise $\tilde{\boldsymbol{\omega}}$, the dynamics $\tilde{f}$, the policies π and $\hat{\pi}$, and the initial state distribution ν_0.

We use $\pi^\star$ to denote the optimal deterministic *robust* policy from set Π in case of true dynamics f, i.e.,

$$\pi^\star \in \arg\max_{\pi \in \Pi} \min_{\hat{\pi} \in \hat{\Pi}} J(f, \pi, \hat{\pi}). \tag{4.3}$$

Even when the true system dynamics are known, finding a robust policy is generally a challenging task for arbitrary policy sets, reward and transition functions. In the rest, we make an assumption that Equation (4.3) can be solved for a given dynamics, and in Section 4.2.3, we propose a concrete problem instantiation and algorithmic solution.

LEARNING PROTOCOL We consider the episodic setting in which, at every episode n, the learning algorithm selects both the agent's π_n and a fictitious adversary's $\hat{\pi}_n$ policies. The pair of policies $(\pi_n, \hat{\pi}_n)$ is then deployed on the true system f, and a single realization of the trajectory $\tau_{f,\pi_n,\hat{\pi}_n}$ is observed and used to update the underlying statistical model. We summarize the general learning protocol in Algorithm 9. In the braking system example, this learning protocol implies that during training we are allowed to execute braking maneuvers as well as possible adversarial policies, e.g., changing the braking surface. The execution of both policies during training is crucial to guarantee robust performance: The learner can actively look for the *worst-case* adversarial policies that it might encounter during deployment and learn what to do when faced upon them. The **main contribution of this chapter** is an algorithm that selects agent and adversary policies $(\pi_n, \hat{\pi}_n)$, in such a way that the output policy is provably robust.

Algorithm 9 Robust Model-based Reinforcement Learning

Inputs: True dynamics f, horizon H, initial state $\mathbf{s}_0$, number of episodes N.

Initialize dataset $\mathcal{D}_0 = \{\emptyset\}$.
Initialize statistical dynamical model $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
for Episode $n = 1, 2, \ldots,$ N **do**
  Select agent and adversary policies $(\pi_n, \hat{\pi}_n)$.
  Reset the system to $\mathbf{s}_{0,n} \sim \nu_0$.
  for Timestep $h = 1, \ldots, H$ **do**
    $\mathbf{a}_{h-1,n}, \hat{\mathbf{a}}_{h-1,n} = \pi_n(\mathbf{s}_{h-1,n}), \hat{\pi}_n(\mathbf{s}_{h-1,n})$
    $\mathbf{s}_{h,n} = f(\mathbf{s}_{h-1,n}, \mathbf{a}_{h-1,n}, \hat{\mathbf{a}}_{h-1,n}) + \boldsymbol{\omega}_{h-1,n}$
    Collect transition $\mathcal{D}_n = \mathcal{D}_n \cup \{(\mathbf{s}_{h-1,n}, \mathbf{a}_{h-1,n}, \hat{\mathbf{a}}_{h-1,n}), \mathbf{s}_{h,n}\}$
  Update statistical dynamical model with the H transitions in $\mathcal{D}_n$.

Outputs: Final policy $\hat{\pi}$

MODEL LEARNING The model learning techniques are identical to the ones in Section 2.2, but the covariates are $\mathbf{z} \equiv (\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}})$. All the results and

training protocol from the aformentioned section hold with this slight modification.

PERFORMANCE METRIC For a small fixed $\epsilon > 0$, the goal is to output a robust policy $\hat{\pi}$ after N episodes such that:

$$\min_{\hat{\pi} \in \hat{\Pi}} J(f, \hat{\pi}, \hat{\pi}) \geq \min_{\hat{\pi} \in \hat{\Pi}} J(f, \pi^\star, \hat{\pi}) - \epsilon, \tag{4.4}$$

where $\pi^\star$ is defined as in Equation (4.3). Hence, we consider the task of near-optimal robust policy identification, but we note that one can also measure the performance in terms of the robust cumulative regret as discussed in section 4.3. Thus, the goal is to output the agent's policy with near-optimal robust performance when facing its own *worst-case* adversary, and the adversary selects $\hat{\pi}$ *after* the agent selects $\hat{\pi}$. Note that this is a stronger robustness notion than just considering the worst-case adversary of the optimal policy, since, by letting $\hat{\pi}^* \in \operatorname{argmin}_{\hat{\pi} \in \hat{\Pi}} J(f, \pi^\star, \hat{\pi})$, we have $J(f, \hat{\pi}, \hat{\pi}^*) \geq \min_{\hat{\pi} \in \hat{\Pi}} J(f, \hat{\pi}, \hat{\pi})$.

4.2 ROBUST HALLUCINATED UPPER CONFIDENCE REINFORMCENT LEARNING (RH-UCRL)

We now develop our RH-UCRL algorithm that can be used in Algorithm 9, for selecting policies π_n and $\hat{\pi}_n$. RH-UCRL takes the sequence of confidence parameters $\{\beta_n\}_{n \geq 1}$ from Assumption 7 as input. The main idea is to use our probabilistic model of f to *optimistically* select π_n and *pessimistically* select $\hat{\pi}_n$ w.r.t. all plausible dynamics.

4.2.1 *Optimistic and Pessimistic Policy Evaluation*

For any two policies π and $\hat{\pi}$, we provide the **(o)**ptimistic and **(p)**essimistic estimate of $J(f, \pi, \hat{\pi})$ at episode n, and we denote them with $J_n^{(o)}(\pi, \hat{\pi})$ and $J_n^{(p)}(\pi, \hat{\pi})$, respectively. Such estimates are constructed considering the epistemic uncertainty in the dynamical model. For instance, the optimistic estimate is the maximum performance of a given policy, where the maximum is taken over the dynamical models in $\mathcal{M}_n$. In general, such optimization problem is intractable. However, we introduce an auxiliary function $\eta : \mathcal{S} \times \mathcal{A} \times \hat{\mathcal{A}} \to [-1, 1]^{d_s}$ and reparameterize the set of plausible

models as $\tilde{f} = \boldsymbol{\mu}_{n-1}(\cdot) + \beta_{n-1}\boldsymbol{\sigma}_{n-1}(\cdot)\eta(\cdot)$. Using this reparameterization, the optimistic estimate is given by:

$$J_n^{(o)}(\pi, \hat{\pi}) \coloneqq \max_{\eta^{(o)}} J(f^{(o)}, \pi, \hat{\pi}), \tag{4.5a}$$

$$\text{s.t. } f^{(o)}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}) = \boldsymbol{\mu}_{n-1}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}) + \beta_{n-1}\eta^{(o)}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}})\boldsymbol{\sigma}_{n-1}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}). \tag{4.5b}$$

Similarly, the pessimistic estimate is given by:

$$J_n^{(p)}(\pi, \hat{\pi}) \coloneqq \min_{\eta^{(p)}} J(f^{(p)}, \pi, \hat{\pi}) \tag{4.6a}$$

$$\text{s.t. } f^{(p)}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}) = \boldsymbol{\mu}_{n-1}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}) + \beta_{n-1}\eta^{(p)}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}})\boldsymbol{\sigma}_{n-1}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}). \tag{4.6b}$$

We note that $J_n^{(o)}$ and $J_n^{(p)}$ represent upper and lower bounds on the performance of the policies π, $\hat{\pi}$ in case of the true dynamics f. These estimates are computed by finding the most optimistic (pessimistic) dynamics compatible with the data. Note that the optimistic/pessimistic outcome is selected via decision variables $\eta^{(o)}/\eta^{(p)} : \mathcal{S} \times \mathcal{A} \times \hat{\mathcal{A}} \to [-1, 1]^{d_s}$, which are functions of the state as well as actions of both players. These select among all plausible outcomes of the dynamics bounded within the epistemic uncertainty over f. When the policies are fixed and clear from context, with slight abuse of notation we write $\eta(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}}) = \eta(\mathbf{s}, \pi(\mathbf{s}), \hat{\pi}(\mathbf{s})) = \eta(\mathbf{s})$. A crucial observation is that both eqs. (4.5) and (4.6) can be viewed as two optimal control problems, where the decision variables $\eta^{(o)}/\eta^{(p)}$ are hallucinated control policies, whose effect is bounded by the model epistemic uncertainty. We can use optimal control algorithms (Camacho & Alba, 2013) to maximize/minimize the sum of rewards following the reparameterized dynamics $f^{(o)}/f^{(p)}$.

4.2.2 *The* RH-UCRL *Algorithm*

Given both the pessimistic and optimistic performance estimates from the previous section, we are now ready to state our algorithm. At each episode n, RH-UCRL selects the agent and adversary policies as follows:

$$\pi_n \in \operatorname*{argmax}_{\pi \in \Pi} \min_{\hat{\pi} \in \hat{\Pi}} J_n^{(o)}(\pi, \hat{\pi}), \tag{4.7a}$$

$$\hat{\pi}_n \in \operatorname*{argmin}_{\hat{\pi} \in \hat{\Pi}} J_n^{(p)}(\pi_n, \hat{\pi}). \tag{4.7b}$$

Thus, RH-UCRL selects the most optimistic robust policy for the agent player in eq. (4.7a). The adversary player picks the most pessimistic policy given the selected agent policy in eq. (4.7b). When the adversarial policy space $\hat{\Pi}$ is a singleton, RH-UCRL reduces to the H-UCRL algorithm.

Finally, after a total of N episodes, the algorithm outputs an agent policy $\hat{\pi}$ given by:

$$\hat{\pi} = \pi_{n^\star} \;\text{ s.t. }\; n^\star \in \operatorname*{argmax}_{n \in \{1,\dots,N\}} J_n^{(p)}(\pi_n, \hat{\pi}_n). \tag{4.8}$$

There is no extra computational cost in identifying the output policy as $J_n^{(p)}(\pi_n, \hat{\pi}_n)$ is already computed by the learner in eq. (4.7b) in every episode n. Thus, the algorithm simply returns the encountered agent policy with maximum pessimistic robust performance.

4.2.3 *Practical Implementation*

Policy Learning. To implement RH-UCRL, we parameterize π, $\hat{\pi}$, and η using neural network policies. We remark that $\hat{\pi}$ in the agent optimization (4.7a) and $\hat{\pi}$ in the adversary optimization (4.7b) are different so, for the sake of clarity, we call $\hat{\pi}'$ the policy in the agent optimization (4.7a). We approximate the finite-horizon RL problem with a discounted infinite-horizon problem using $\gamma = 1/(1-H)$ as discount factor. We use an actor-critic approach where we learn two separate critics, one for the optimistic performance in (4.7a) and the other one for the pessimistic performance in (4.7b) via fitted Q-iteration (Perolat *et al.*, 2015; Antos *et al.*, 2008). Finally, we do stochastic gradient ascent/descent using pathwise gradients through such learned critics (Mohamed *et al.*, 2019; Silver *et al.*, 2014). Namely, we compute the gradients of π, $\eta^{(o)}$, and $\hat{\pi}'$ through the learned optimistic critic, then we update π and $\eta^{(o)}$ via gradient ascent, whereas $\hat{\pi}'$ via

gradient descent. Likewise, we compute the gradients of $\hat{\pi}$ and $\eta^{(p)}$ through the learned pessimistic critic for the fixed π and update both $\hat{\pi}$ and $\eta^{(p)}$ via gradient descent.

4.3 THEORETICAL ANALYSIS

In this section, we theoretically analyze the performance of the RH-UCRL algorithm. First, we use the notion of *robust cumulative regret*[1]

$$\bar{\mathfrak{R}}_N = \sum_{n=1}^{N} \min_{\hat{\pi} \in \hat{\Pi}} J(f\pi^\star, \hat{\pi}) - \min_{\hat{\pi} \in \hat{\Pi}} J(f\pi_n, \hat{\pi}), \tag{4.9}$$

which measures the difference in performance between the optimal robust policy and the sequence of agent's policies $\{\pi_1, \dots, \pi_N\}$ selected at every episode in eq. (4.7a). Below (see Theorem 2) we establish that RH-UCRL achieves sublinear regret, i.e., $\bar{\mathfrak{R}}_N / N \to 0$ for $N \to \infty$. In addition to the robust regret notion, we also analyze the recommendation rule of RH-UCRL via eq. (4.8), and the number of episodes N required to output a near-optimal robust policy (see Corollary 1). We start by analyzing a general robust model-based RL framework, and later on, we demonstrate the utility of the obtained results by specializing them to the important special case of Gaussian Process dynamics models. We defer all the proofs from this section to Appendix B.

Both the robust regret and sample complexity rates that we analyze depend on the difficulty of learning the underlying statistical model. Models that are easy to learn typically require fewer samples and allow algorithms to make better decisions sooner. To express the difficulty of learning the imposed calibrated model class, we use the following model-based complexity measure introduced in Equation (2.11):

$$\Gamma_N \coloneqq \max_{\tilde{\mathcal{D}}_{1:N}} \sum_{n=1}^{N} \sum_{(\mathbf{s},\mathbf{a},\hat{\mathbf{a}}) \in \tilde{\mathcal{D}}_n} \|\boldsymbol{\sigma}_{n-1}(\mathbf{s}, \mathbf{a}, \hat{\mathbf{a}})\|_2^2 \tag{4.10}$$

where each $\tilde{\mathcal{D}}_n \subset \{\mathcal{S} \times \mathcal{A} \times \hat{\mathcal{A}}\}^H$. This quantity has a worst-case flavor as it considers the data (collected during N episodes by any algorithm) that lead to maximal total predictive uncertainty of the model. For the special case of RKHS/GP dynamics models, we show below that this quantity can be

1 Similar notions of robust cumulative regret have been analyzed before in bandit optimization (see, e.g., Kirschner *et al.*, 2020).

effectively bounded, and the bound is sublinear (in the number of episodes N) for most commonly used kernel functions.

GENERAL RESULTS Now, we can state the main result of this section. In the following theorem, we bound the robust cumulative regret incurred by the policies from eq. (4.7a).

Theorem 2. *Under Assumptions 1 to 8, let* $C = (1 + L_f + 2L_\sigma)(1 + L_\pi^2 + L_{\hat{\pi}}^2)^{1/2}$ *and let* $\mathbf{s}_{n,h} \in \mathcal{S}$, $\mathbf{a}_{n,h} \in \mathcal{A}$, $\hat{\mathbf{a}}_{n,h} \in \hat{\mathcal{A}}$ *for all* $n, h > 0$. *Then, for any fixed* $H \geq 1$, *with probability at least* $1 - \delta$, *the robust cumulative regret of* RH-UCRL *is upper bounded by:*

$$\bar{\mathfrak{R}}_N = \mathcal{O}\Big(L_r C^H \beta_N^H H^{3/2} \sqrt{N \Gamma_N}\Big).$$

This regret bound shows that RH-UCRL achieves sublinear robust regret when $\beta_N^H \sqrt{\Gamma_N} = o(\sqrt{N})$. Below, we show a concrete example of GP models where this is indeed the case. The obtained bound also depends on the Lipschitz constants from Assumption 8, as well as the episode length H that we assume is constant. The dependency of the regret bound on the problem dimension is hidden in Γ_N, while β_N depends also on δ (see Assumption 7).

Next, we characterize the number of episodes (samples) required by RH-UCRL to output ϵ-optimal robust policy. Our analysis upper bounds the optimal robust performance according to the confidence bounds from Assumption 7, but also addresses the challenge of characterizing the impact of exploring different adversary policies in eq. (4.7b).

Corollary 1. *Consider the assumptions and setup of Theorem 2, and suppose that*

$$\frac{N}{\beta_N^{2H} \Gamma_N} \geq \frac{16 L_r^2 H^3 C^{2H}}{\epsilon^2}, \tag{4.11}$$

for some fixed $\epsilon > 0$ *and* $H \geq 1$. *Then, with probability at least* $1 - \delta$ *after* N *episodes,* RH-UCRL *achieves:*

$$\min_{\hat{\pi} \in \hat{\Pi}} J(f, \hat{\pi}_N, \hat{\pi}) \geq \min_{\hat{\pi} \in \hat{\Pi}} J(f, \pi^\star, \hat{\pi}) - \epsilon, \tag{4.12}$$

where $\hat{\pi}$ *is the output of* RH-UCRL, *reported according to eq.* (4.8), *and* $\pi^\star$ *is the optimal robust policy given in eq.* (4.3).

GAUSSIAN PROCESS MODELS We specialize the regret bound obtained in theorem 2 to the case of Gaussian Process (GP) models. GPs are popular

statistical models that are frequently used to model unknown dynamics (M. Deisenroth & Rasmussen, 2011; Kamthe & M. Deisenroth, 2018b; Curi *et al.*, 2020b). These models are very expressive due to a versatility of possible kernel functions, and can naturally differentiate between aleatoric noise and epistemic uncertainty. Moreover, GPs are known to be provably well-calibrated when the unknown dynamics f are B_f-smooth as measured by the GP kernel.

In Section 2.2.3, we recall the GP maximum information gain (MIG) which is a kernel-dependent quantity (first introduced by Srinivas *et al.* (2010)), that is frequently used in various GP optimization works to characterize complexity of learning a GP model. Sublinear upper bounds for MIG are known (c.f. Srinivas *et al.* (2010)) for most popularly used kernels (e.g., linear, squared-exponential, etc.), as well as for their compositions, e.g., additive kernels Krause & Ong, 2011. We recall the known results and use MIG to express β_N and upper bound Γ_N in theorem 2. For example, when we use independent GP models with either (i) linear or (ii) squared-exponential kernels, for every component, we obtain the following *sublinear* (in N) regret bounds $O(H^{3/2} d_s \left[(d_s + d_a + d_{\hat{a}}) \ln(d_s N H)\right]^{(H+1)/2} \sqrt{N})$ and $O(H^{3/2} d_s \left[\ln(d_s N H)\right]^{(d_s+d_a+d_{\hat{a}})(H+1)/2} \sqrt{N})$, respectively.

Finally, we note that the previously used MIG bounds require $\mathcal{S}$ to be compact, which does not hold under the considered noise model in Assumption 2. By bounding the domain w.h.p., Curi *et al.* (2020a) show that this only increases the MIG bounds (e.g., in case of the squared-exponential kernel) by at most a polylog(N) factor.

4.4 EXPERIMENTS

We now discuss concrete instantiations of RH-UCRL for three important robust RL scenarios: (i) *adversarial-robustness*, (ii) *action-robustness*, and (iii) *parameter-robustness*. In all of the above scenarios, we experimentally demonstrate that RH-UCRL outperforms or successfully competes with the state-of-the-art variants designed specifically for these settings. In Section 4.4.1, we evaluate RH-UCRL in a small scale setting to provide insight about the algorithm and, in Section 4.4.2, we evaluate RH-UCRL in a large scale setting to demonstrate its scalability. We provide an open-source implementation of our method, which is available at `http://github.com/sebascuri/rhucrl`.

4.4.1 *Small-Scale: Inverted Pendulum Swing-Up Task*

The pendulum swing up task has a reward function given by $r(\mathbf{s}, \mathbf{a}) = -(\theta^2 + 0.1 * \dot{\theta}^2)$, where θ is the angle and $\dot{\theta}$ is the angular velocity. The Pendulum always starts from $\theta = \pi$ in the bottom down position and the goal is to swing the pendulum to the top-up position at $\theta = 0$. Crucially, the initial distribution is a dirac-distribution located at $\theta = \pi$, i.e., it does not have enough coverage for algorithms to explore with it.

ADVERSARIAL-ROBUST. In this setting, the adversary can change the relative gravity and the relative mass of the environment at every timestep of each episode between $[1 - \alpha, 1 + \alpha]$, for varying α. We train RH-UCRL for 200 episodes, H-UCRL with the nominal gravity and mass for 200 episodes, and the baseline in this setting is RARL (Pinto *et al.*, 2017), which we train for 1000 episodes. To evaluate the robust performance, we train SAC for 200 episodes, fixing the agent policy of the algorithms.

ACTION-ROBUST. In this setting, the action is a mixture sampled with probability α of the learner and the adversary, i.e., the adversary only affects the input torque to the pendulum. The training and evaluation procedure is the same as in the adversarial robust setting. The baseline in this setting is AR-DDPG, which we train for 200 episodes.

PARAMETER-ROBUST. In this setting, we consider robustness to mass change. Compared to the adversarial-robust setting, here the adversary is only allowed to change the mass once per episode. In this setting, H-UCRL is trained for 200 episodes with the nominal mass, and then it is evaluated for varying masses. RH-UCRL and the baseline, EP-OPT are allowed to change the mass also during training. In this setting, there is no worst-case adversary during evaluation.

We show the experimental results in Figure 4.1. The baselines never learns a successful swing-up strategy as dithering is not enough to explore in this environment. On the other hand, RH-UCRL and H-UCRL both learn a swing-up strategy. However, RH-UCRL outperforms H-UCRL as the level of adversarial perturbation increases.

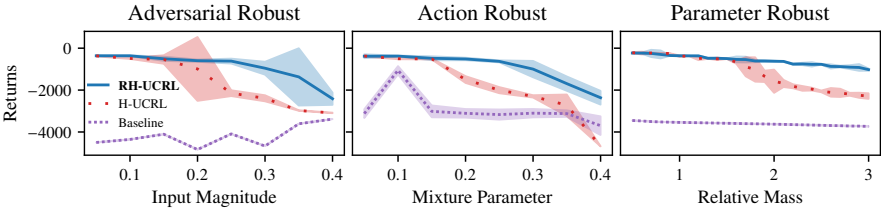


FIGURE 4.1: Performance of RH-UCRL (this work), H-UCRL, and a baseline with no exploration in adversarial-, action-, and parameter-robust settings on a pendulum swing-up task. The baseline never learns a successful swing-up strategy due to insufficient exploration. RH-UCRL and H-UCRL learn a swing-up strategy but RH-UCRL outperforms H-UCRL as the perturbation increases.

4.4.2 *Large-Scale: Mujoco Environments*

EXPERIMENTAL ENVIRONMENTS We use the Mujoco suite (Todorov *et al.*, 2012) to demonstrate the effectiveness of our algorithms in all the considered robust-RL settings. In particular, we use the Half Cheetah, Hopper, Inverted Pendulum, Reacher, Swimmer, and Walker robots.

TRAINING AND EVALUATION Unless stated otherwise, we train all algorithms with an adversarial environment for 200 episodes. To evaluate the *robust performance* (4.4) of each algorithm, we freeze the output policy of the training step and train an adversary using SAC for 200 episodes. We perform five independent runs and report the mean and standard deviation over the runs.

ALGORITHM HYPER-PARAMETERS. For RH-UCRL and its variants, we fix $\beta = 1.0$, we train every time step and do two gradient steps with Adam (Kingma & Ba, 2014) with learning rate $= 3 \times 10^{-4}$. To compute a policy gradient, we take pathwise derivatives of a learned critic using the learned model for 3 time steps and weight each estimates using td-λ, with $\lambda = 0.1$ (Sutton & Barto, 2018). We also add entropy regularization with parameter 0.2. We did not do a hyper parameter search, but rather use the software default values.

BASELINES Besides the specific algorithms designed for each setting, we use H-UCRL (Curi *et al.*, 2020a) as a non-robust baseline and three

ablations derived from RH-UCRL, namely MINIMAX-MB, MINIMAX-MF and BESTRESPONSE. The MINIMAX-MB algorithm is:

$$(\pi_n, \hat{\pi}_n) \in \operatorname*{argmax}_{\pi \in \Pi} \min_{\hat{\pi} \in \hat{\Pi}} J_n^{(e)}(\pi, \hat{\pi}), \tag{4.13}$$

where $J_n^{(e)}(\pi, \hat{\pi})$ corresponds to the *expected* performance, where the expectation is taken with respect to the aleatoric and epistemic uncertainty, i.e., none of the players actively explore. Next, the MINIMAX-MF algorithm is a model-free implementation of Equation (4.13) that uses SAC (Haarnoja *et al.*, 2018) as the optimizer for each player. The BESTRESPONSE algorithm is:

$$\pi_n \in \operatorname*{argmax}_{\pi \in \Pi} \min_{\hat{\pi} \in \hat{\Pi}} J_n^{(o)}(\pi, \hat{\pi}), \tag{4.14a}$$

$$\hat{\pi}_n \in \operatorname*{argmin}_{\hat{\pi} \in \hat{\Pi}} J_n^{(e)}(\pi_n, \hat{\pi}). \tag{4.14b}$$

Thus, the agent is the same as in RH-UCRL, whereas the adversary simply plays the best-response to the agent's policy and does not perform exploration with pessimism. The goal of BESTRESPONSE is to analyze if exploration of the adversary through pessimism is empirically important, of MINIMAX-MB is to analyze if any exploration is empirically important, and of MINIMAX-MF is to analyze if using a model of the dynamics is beneficial.

4.4.2.1 *Adversarial-Robust Reinforcement Learning*

This setting is the most general one that we also consider in Section 4.2. The agent and the adversary can have distinct action spaces, which can also be seen as a particular instance of multi-agent RL with two competing agents. In the braking system motivating example, this can be used to model an adversarial state-dependent friction coefficient, e.g., icy roads. Having good robust performance in this setting implies braking robustly even with changing conditions.

The deep robust RL algorithms that we compare with are RARL (Pinto *et al.*, 2017) and RAP (Vinitsky *et al.*, 2020), and we use the adversarial action space proposed by Pinto *et al.* (2017). We train all algorithms for 200 episodes except for RARL and RAP that we train for 1000 episodes since they are on-policy algorithms and thus less sample-efficient. For RARL and RAP, we use the PPO algorithm from Schulman *et al.* (2017) as this

performed better than TRPO from Schulman *et al.* (2015b). We train PPO after collecting a batch of 4 episodes, for 80 gradient steps, using early stopping once the KL divergence between the initial and the current policy is more than 0.0075. To evaluate *robust performance* (recall Equation (4.4)), we freeze the output policy and train only its adversary by using SAC for 200 episodes.

ENVIRONMENTS For the Half-Cheetah environment, the adversary acts on the torso, the front foot and the back foot. For the Hopper environment, the adversary acts on the torso. For the Inverted Pendulum, the adversary acts on the pole. The Inverted Pendulum task is different here as it starts from a perturbation of the top-up position and the task is to stabilize the pendulum. For the Reacher2d environment, the adversary acts on the body0 link. For the Swimmer, the adversary acts on the torso. For the Walker, the adversary acts on the torso. For all environments, we use the adversarial input magnitude $\hat{\mathcal{A}} = [-10, 10]^{d_{\hat{a}}}$, where $d_{\hat{a}}$ is environment dependent.

In Figure 4.2, we show the *worst-case* and *average* returns on the different environments. In terms of *average* performance, there is no algorithm that performs better than others in all of the environments. On the other hand, comparing *worst-case* performance, RH-UCRL clearly outperforms the robust ablations, deep robust RL and non-robust baselines. For example, in the Inverted Pendulum stabilization task, RH-UCRL is the *only* algorithm that discovers a robust policy while all other algorithms severely fail. BESTRESPONSE and RAP manage to learn a policy that stabilizes the pendulum even when they learn with an adversary. However, when facing a *worst-case* adversary, they fail to complete the task.

Comparing RH-UCRL with non-robust H-UCRL, we see that in most environments it has comparable or better *worst-case* and *average* performance. This indicates that RH-UCRL is not only robust, but using an adversary during training practically helps with exploration. Pinto *et al.* (2017) also report similar findings regarding robust training. Comparing RH-UCRL with the ablations, we see that RH-UCRL achieves higher robust performance. From here, we conclude that exploring with both the agent and the adversary during training is crucial to achieve high robust performance in this setting. Finally, we see that both RARL and RAP have poor robust performance when trained for 1000 episodes, which demonstrates their sample inefficiency.

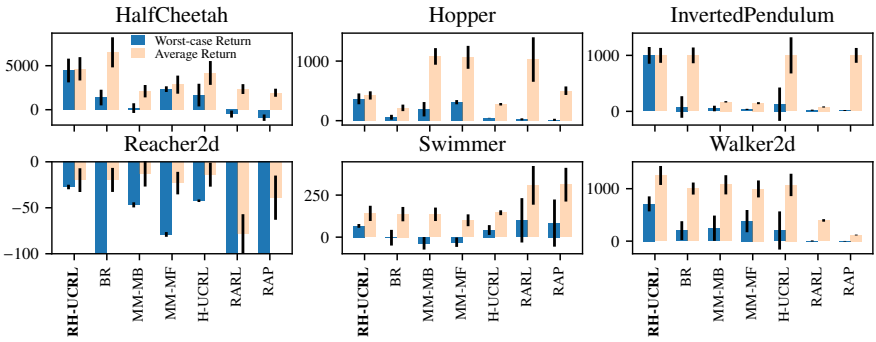


FIGURE 4.2: Worst-case and average return of different algorithms in the Adversarial-Robust Setting in Mujoco tasks. RH-UCRL outperforms the other algorithms in terms of worst-case return. The non-robust baseline, H-UCRL, has good average performance but poor worst-case performance (e.g., Inverted Pendulum). The deep robust RL baselines have worse sample complexity and often underperform. Our ablations are also non-robust, since exploration of both *agent* and *adversary* is crucial here to achieve robust performance.

4.4.2.2 *Action-Robust Reinforcement Learning*

Tessler *et al.* (2019) introduce the action-robust setting, where both the agent and the adversary share the action space $\mathcal{A}$ and jointly execute a single action in the environment. This is useful, e.g., to model robustness to changes in the actuator dynamics, e.g., due to tire wear or incorrect pressure in a braking system. The action is sampled from a mixture policy $\mathbf{a}^{\text{mix}} \sim \pi^{\text{mix}} = \Upsilon_\alpha(\pi, \hat{\pi})$, where $\alpha \in [0,1]$ is a known parameter that controls the mixture proportion. One example of the mixture policy is the noisy-robust setting, in which $\Upsilon_\alpha(\pi, \hat{\pi}) = (1-\alpha)\pi + \alpha\hat{\pi}$. Another example is the noisy-robust setting, in which $\Upsilon_\alpha = \pi$ with probability $(1-\alpha)$ and $\Upsilon_\alpha = \hat{\pi}$ with probability α. The system evolves according to $\mathbf{s}_{h+1} = f'(\mathbf{s}_h, \mathbf{a}_h^{\text{mix}}) + \boldsymbol{\omega}_h$.

Besides the previous baselines, we compare to AR-DDPG Tessler *et al.*, 2019, and show the results of the experiment in Figure 4.3. Here, RH-UCRL is also comparable or better than the baselines in terms of *average* and *worst-case* returns. However, the ablations perform better than in the adversarial-robust setting. This is possibly due to the agent and adversary sharing the action space: The agent injects "enough" exploration to successfully learn both policies.

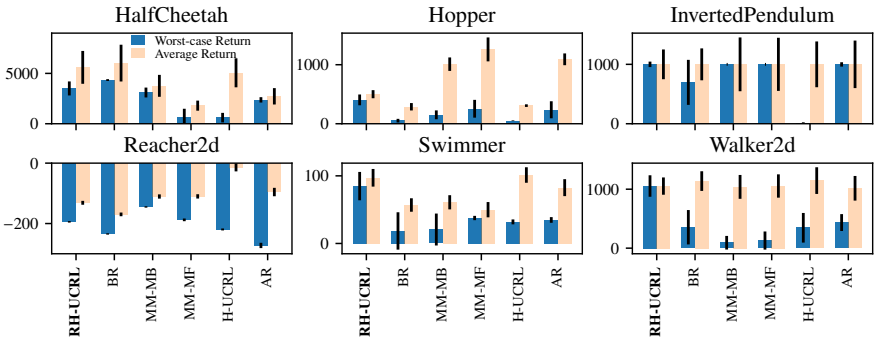


FIGURE 4.3: Average and worst-case return of different algorithms in the Noisy Action-Robust Setting in Mujoco tasks. RH-UCRL mostly outperforms other algorithms in terms of worst-case return. The non-robust baseline, H-UCRL, has good average performance but has an extreme drop in worst-case performance. Overall, the ablations perform better here than in the Adversarial-Robust setting.

4.4.2.3 *Parameter-Robust Reinforcement Learning*

The goal in this setting is to be robust to changes in parameters, such as mass or friction, that can occur between training and test time. Being robust to a fixed parameter is equivalent to considering a stateless adversary policy in the RH-UCRL algorithm (4.7), i.e., $\hat{\Pi} : \varnothing \to \mathcal{A}$. Common benchmarks in this setting are DOMAINRANDOMIZATION (Peng *et al.*, 2018; Tobin *et al.*, 2017) and EP-OPT Rajeswaran *et al.*, 2017. The former randomizes the parameters in the simulation and uses the *average* over these parameters as a surrogate of the maximum. The latter also randomizes the parameters but considers the CVaR as a surrogate of the maximum. As they are on-policy procedures, we train them using data for 1000 episodes. Finally, we evaluate the policies in different environments by varying the corresponding mass parameters.

We show the results of this setting in Figure 4.4. Although RH-UCRL optimizes for the worst-case parameter, it performs well over different mass parameter values, and, except in the Walker environment, its performance remains robust and nearly constant for different values of the mass parameter. H-UCRL is trained with nominal mass only (relative mass = 1), and it suffers in performance when varying the mass. This is most notable in the Half Cheetah environment (see Figure 4.4). The robust variants, instead, can alter the mass during training and often perform better than H-UCRL. A particular case happens with the BESTRESPONSE algorithm in the Inverted Pen-

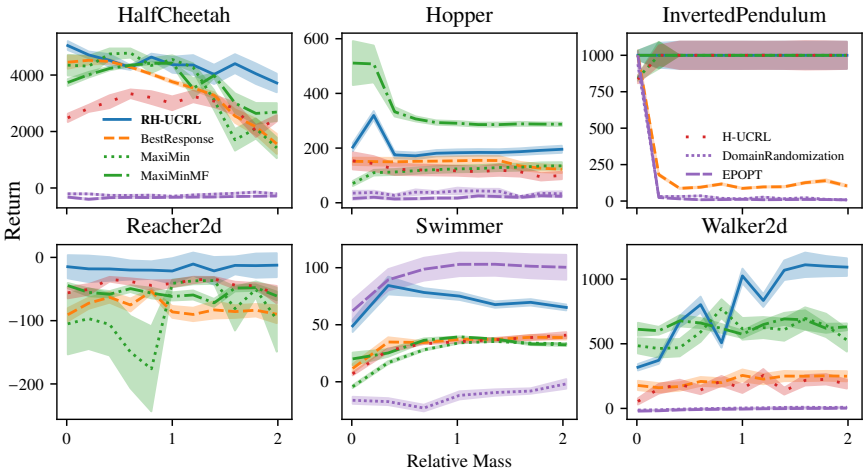


FIGURE 4.4: Returns of different algorithms in the Parameter-Robust Setting in Mujoco tasks for different masses during evaluation. Although RH-UCRL optimizes for the worst-case relative mass in this setting, it also performs well over different value of mass parameters.

dulum, where the adversary is greedy and so it swiftly chooses a small mass and never changes it during training. The agent learns only for this small mass and, when evaluated with different ones, it performs poorly. We also observe that in the Hopper, the MINIMAX-MF outperforms the MINIMAX-MB. The reason for this might be due to early stopping of the environment, as it is possible that the transitions collected in 200 episodes are not sufficient for learning the model, but allow for learning a policy in a model-free way.

5

EPISTEMIC UNCERTAINTY FOR PROVABLE SAFETY

An Ounce of Prevention Is Worth A Pound of Cure.
— Benjamin Franklin

In this chapter, we use the models' epistemic uncertainty to provide *provable safety guarantees* in a dynamical system with continuous states and actions. We define safety as constraints that must be met at every time step of the execution. In Section 5.1, we formalize the problem setup and the objective we are trying to solve. In Section 5.2, we prove that we can express the safety constraints as a sum of discounted costs and transform the problem into a constrained MDP by inner approximating the safe set of states. In Section 5.3, we show how to incorporate epistemic uncertainty through confidence-based safety filters and introduce the H-UCSF algorithm, together with the theoretical analysis. Finally, in Section 5.4, we show the practical aspects of the algorithm in experiments. We defer the technical proofs to Appendix C.
The results in this chapter have been previously published in:

- Curi, S., Lederer, A., Hirche, S., & Krause, A. *Safe Reinforcement Learning via Confidence-based Filters* in *IEEE 61th Annual Conference on Decision and Control (CDC)* (2022).

5.1 PROBLEM SETUP

We consider a discrete-time dynamical system as in Equation (5.1),

$$\mathbf{s}_{h+1} = f(\mathbf{s}_h, \mathbf{a}_h) + \boldsymbol{\omega}_h, \tag{5.1}$$

where $\mathbf{s} \in \mathcal{S} \subset \mathbb{R}^{d_s}$ are states, $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}^{d_a}$ control actions, $\boldsymbol{\omega}$ is process noise sampled from a zero-mean probability distribution, and $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the unknown deterministic transition function. The

control actions $\mathbf{a}_h$ are determined using a policy $\pi\colon \mathcal{S} \to \mathcal{A}$, with the goal to maximize an expected cumulative return

$$J(f, \pi; \mathbf{s}) = \mathbb{E}_\tau \left[\sum_{h=0}^{\infty} \gamma^h r(\mathbf{s}_h, \pi(\mathbf{s}_h)) \right], \tag{5.2a}$$

$$\text{s.t. } \mathbf{s}_{h+1} = f(\mathbf{s}_h, \pi(\mathbf{s}_h)) + \boldsymbol{\omega}_h \tag{5.2b}$$

$$\mathbf{s}_0 = \mathbf{s}, \tag{5.2c}$$

where $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a known immediate reward function, $\gamma \in (0, 1)$ is a discount factor, and $\tau_{\tilde{f},\pi} = \{(\mathbf{s}_{h-1}, \mathbf{a}_{h-1}), \mathbf{s}_h\}_{h=0}^{H}$ is a random trajectory induced by the stochastic noise $\tilde{\boldsymbol{\omega}}$, the dynamics $\tilde{f}$, and the policy π. Notice that in this chapter we consider fixed starting states instead of distributions, thus we index the returns with $J(f, \pi; \mathbf{s})$.

In practice, the policy π must additionally ensure safety of the closed-loop dynamical system, e.g., because damage to the system described by f must be avoided. In the RL literature, this is typically addressed through Constrained Markov Decision Processes (CMDPs), which additionally consider a constraint on a cumulative cost function

$$C(f, \pi; \mathbf{s}) = \mathbb{E}_\tau \left[\sum_{h=0}^{\infty} \gamma^h c(\mathbf{s}_h) \right] < \xi, \tag{5.3}$$

where $c : \mathcal{S} \to \mathbb{R}$ is an immediate cost, $\xi \in \mathbb{R}$ is a constant specifying the constraint, and $\mathbf{s}_h$ is defined iteratively through the dynamics (5.1) with actions $\mathbf{a}_h = \pi(\mathbf{s}_h)$ and initial state $\mathbf{s}_0 = \mathbf{s}$. Therefore, an optimization problem of the form

$$\pi^* = \operatorname*{argmax}_{\pi} \mathbb{E}_{\mathbf{s}\sim\nu_0}[J(f, \pi; \mathbf{s})] \tag{5.4a}$$

$$\text{s.t. } \mathbb{E}_{\mathbf{s}\sim\nu_0}[C(f, \pi; \mathbf{s})] < \xi \tag{5.4b}$$

is usually solved to determine safe policies.

While this problem can be directly solved by adapting standard RL algorithms with techniques akin to Lagrangian relaxation Paternain *et al.*, 2019, this approach generally cannot ensure safety *during* training. Moreover, it does not reflect the fact that the safety of many systems is defined in terms of safe and unsafe states classified into a set of safe states $\mathcal{S}_{\text{safe}} \subset \mathcal{S}$ and its complement $\mathcal{S}_{\text{unsafe}} = \mathcal{S} \setminus \mathcal{S}_{\text{safe}}$. For example, an autonomously driving car should not leave the road, which directly defines the road as

$\mathcal{S}_{\text{safe}}$. When using the natural indicator $\mathbf{1}_{\boldsymbol{x}\in\mathcal{S}_{\text{unsafe}}}$ as cost function, satisfying Equation (5.4b) bounds the discounted probability of violating the constraints by ξ. Nonetheless, this does not guarantee that constraints will not be violated when deploying π^*.

Therefore, we consider safety in terms of state constraints $\mathbf{s}_h \in \mathcal{S}_{\text{safe}}$, which we require to hold with high probability, since the process noise $\boldsymbol{\omega}$ generally prevents deterministic guarantees. This leads to the following definition of safety.

Definition 4 (K-step δ-safety)**.** A policy π is K-step δ-safe for a state $\mathbf{s} \in \mathcal{S}$ if it holds that $\mathbb{P}[\mathbf{s}_h \in \mathcal{S}_{\text{safe}}\ \forall k = 0, \dots, K \mid \mathbf{s}_0 = \mathbf{s}] \geq 1 - \delta$, where states $\mathbf{s}_h$ are defined in the dynamics (5.1).

The concept of K-step δ-safety is commonly found in stochastic model predictive control, where it is typically referred to as joint chance constraint (Mesbah, 2016). We consider finite values of K because ensuring δ-safety over an infinite horizon, i.e., $K = \infty$, is not possible for unbounded process noise $\boldsymbol{\omega}$ in general. This can be easily seen for a system with $f = \mathbf{0}$ and i.i.d. zero mean Gaussian noise $\boldsymbol{\omega}$, which almost surely leaves any compact safe set $\mathcal{S}_{\text{safe}}$ eventually.

In order to obtain the optimal policy π^*_{safe} ensuring δ-safety, we generally need to consider the optimization problem

$$\pi^*_{\text{safe}} = \underset{\pi}{\operatorname{argmax}}\, J(f, \pi) \tag{5.5a}$$

$$\text{s.t. } \mathbb{P}[\mathbf{s}_h \in \mathcal{S}_{\text{safe}} \forall h = 0, \dots, K | \mathbf{s}_0 = \mathbf{s}] \geq 1 - \delta. \tag{5.5b}$$

Solving this optimization problem is challenging since there usually exists no closed-form expression for the probability constraint (5.5b), such that computationally expensive uncertainty propagation methods have to be employed, e.g., generalized polynomial chaos expansions (Kim & Braatz, 2013).

To find approximate solutions for the safety problem (5.5), we follow the idea of Wabersich *et al.* (2021) and split it into two phases: an initial phase in which arbitrary methods can be used to determine a nominal policy π^*, followed by an on-line phase, in which a safety filter is employed to adapt the policy π^* to ensure K-step δ-safety. Since we cannot ensure safety on-line without any knowledge about f, we assume to have access to a set of *plausible* models $\mathcal{M}_\beta$ from Equation (2.9), and assume that such set is *well-calibrated* using Assumption 7.

We assume that this set of models is *well-calibrated*, i.e., $f \in \mathcal{M}$ with high probability, as formalized in the following. Since the statistical model is often obtained by applying supervised machine learning, e.g., deep ensembles, to the data obtained from policy roll-outs Curi *et al.*, 2020a, the uncertainty usually decreases with the number of roll-outs. Thereby, this assumption typically enables less conservative and higher performant policies over time.

Using calibrated models from Assumption 7, we investigate the following sub-problems for the derivation of the safety filter.

STATE CONSTRAINTS AS CUMULATIVE COST In order to enable the application of reinforcement learning methods, we consider the problem of converting the K-step δ-safety constraint (5.5b) for known dynamics f into a constraint on an expected cumulative cost function. We show that this can be achieved by deriving a condition of the form

$$\mathbb{E}_{\omega}[C(f, \pi; f(\mathbf{s}, \mathbf{a}) + \omega)] < \xi. \tag{5.6}$$

for suitably chosen immediate costs c, cf. Section 5.2. This conditions means that after we compute C, we should evaluate C at the next state to verify the safety of the policy π.

SAFETY FILTER Using this condition, we derive a novel approach for computing safe policies π_{safe} for systems with unknown dynamics f. This allows us to address the problem of ensuring the safety of a possibly unsafe nominal policy π^* on-line using a confidence-based filter

$$\hat{\pi}(\mathbf{s}) = \operatorname*{argmin}_{\mathbf{a} \in \mathcal{A}} \|\pi^*(\mathbf{s}) - \mathbf{a}\|, \tag{5.7a}$$

$$\text{s.t.} \max_{\tilde{f} \in \mathcal{M}} \mathbb{E}_{\omega}\left[C(\tilde{f}, \pi_{\text{safe}}; \mathbf{s}')\right] < \xi, \tag{5.7b}$$

$$\mathbf{s}' = \tilde{f}(\mathbf{s}, \mathbf{a}) + \omega, \tag{5.7c}$$

which outputs the closest action $\hat{\pi}(\mathbf{s})$ to π^*. We derive tractable formulations for these optimization problems in Section 5.3.

5.2 EXPRESSING STATE CONSTRAINTS THROUGH COST FUNCTIONS

To reformulate the δ-safety constraint into a constraint on cumulative costs, we first show in Section 5.2.1 that sub-level sets of C contained in $\mathcal{S}_{\text{safe}}$

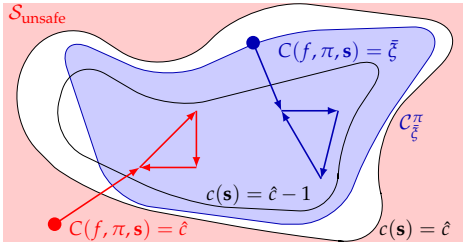


FIGURE 5.1: The expected cumulative cost can be 0 even if the immediate cost c at the first state is greater than 0, as this positive cost can be compensated by negative costs afterwards (red trajectory). Therefore, $\mathcal{C}^{\pi}_{\hat{c}} \not\subset \mathcal{S}_{\text{safe}}$, such that we have to consider the tightened threshold $\bar{\xi}$, which ensures that states $\mathbf{s} \in \mathcal{C}^{\pi}_{\bar{\xi}}$ start with immediate cost $c(\mathbf{s}) \leq \hat{c}$ (blue trajectory).

can be easily defined. Based on this result, we derive sufficient conditions on the cost function, which allow to conclude safety from cumulative cost constraints in Section 5.2.2, providing useful design freedom.

5.2.1 *Safe Sub-Level Sets of the Cumulative Cost*

For deriving the sub-level set $\mathcal{C}^{\pi}_{\bar{\xi}} = \{\mathbf{s} \in \mathcal{S} : C(f,\pi;\mathbf{s}) < \bar{\xi}\}$, $\bar{\xi} \in \mathbb{R}$, which is contained in the set of safe states $\mathcal{S}_{\text{safe}}$, we consider an immediate cost function $c : \mathcal{S} \to \mathbb{R}$ satisfying

$$\underline{c} \leq c(\mathbf{s}) \leq \bar{c} \quad \forall \mathbf{s} \in \mathcal{S}_{\text{safe}}, \quad c(\mathbf{s}) \geq \hat{c} \quad \text{if } \mathbf{s} \in \mathcal{S}_{\text{unsafe}} \tag{5.8}$$

for constants $\underline{c}, \bar{c}, \hat{c} \in \mathbb{R}$. For example, using the indicator function $\mathbf{1}_{\mathbf{s} \in \mathcal{S}_{\text{unsafe}}}$ as cost, which equals 1 for $\mathbf{s} \in \mathcal{S}_{\text{unsafe}}$ and 0 otherwise, implies $\underline{c} = 0$ and $\bar{c} = \hat{c} = 1$. Using this definition, we can define an inner-approximation of the safe set of states $\mathcal{S}_{\text{safe}}$ through the $\hat{c}$ sub-level set of the immediate cost c, which becomes exact if $c(\mathbf{s}) < \hat{c}$ for all $\mathbf{s} \in \mathcal{S}_{\text{safe}}$. Moreover, we can define the expected cumulative cost using Equation (5.3).

While one might think that the definition of the immediate cost c in Equation (5.8) ensures that the $\hat{c}$ sub-level set $\mathcal{C}^{\pi}_{\hat{c}}$ of C is also contained in the safe set of states $\mathcal{S}_{\text{safe}}$, this is not true in general. As illustrated by the red trajectory in Figure 5.1, the cumulative cost C can equal $\hat{c}$ even if the immediate cost c in the initial state is greater than $\hat{c}$, since negative costs of following states along the trajectory can compensate it. Therefore, the sub-level set $\mathcal{C}^{\pi}_{\hat{c}}$ is generally not completely contained in the set of safe states $\mathcal{S}_{\text{safe}}$, such that we must consider a tightened threshold $\bar{\xi}$. Due to the

lower bound $\underline{c}$ of the cost c, this constant $\bar{\xi}$ can be determined using the following lemma.

Lemma 9. *Consider an immediate cost function $c : \mathcal{S} \to \mathbb{R}$ satisfying the conditions* (5.8)*. Then, it holds that $\mathcal{C}^{\pi}_{\bar{\xi}} \subset \mathcal{S}_{\text{safe}}$, where $\bar{\xi} = \hat{c} + \gamma \min_{\mathbf{s} \in \mathcal{S}} C(f, \pi; \mathbf{s})$.*

This lemma relies on the idea that the cumulative cost can be lower bounded by $\min_{\mathbf{s} \in \mathcal{S}} C(f, \pi; \mathbf{s})$, such that any state with immediate cost c greater than $\hat{c}$ also must have an expected cumulative cost greater than $\bar{\xi}$. For the example of the indicator cost, $\bar{\xi}$ can be straightforwardly computed as $\bar{\xi} = 1$ since C is trivially lower bounded by 0. It is straightforward to see that this choice of cost function generally allows to accurately approximate $\mathcal{S}_{\text{safe}}$ using $\mathcal{C}^{\pi}_{\bar{\xi}}$, and indeed $\mathcal{S}_{\text{safe}} = \mathcal{C}^{\pi}_{\bar{\xi}}$ is possible for deterministic dynamics with $\boldsymbol{\omega} = \mathbf{0}$. However, Lemma 9 is not limited to indicator type cost functions, but applies to arbitrary costs c satisfying the conditions (5.8). For example, when a safety set is defined as a polytope e.g., $\mathbf{s} > 0$, either the indicator $c(\mathbf{s}) = \mathbf{s} >$ or even the argument $c(\mathbf{s}) = \mathbf{s}$ are valid cost functions. This is particularly beneficial for computing optimal policies using C, where informative gradients may aid the convergence of common RL techniques. Thus, Lemma 9 allows a flexible approximation of the safe set $\mathcal{S}_{\text{safe}}$ suitable for the optimization-based approaches employed in the following sections.

5.2.2 *Cumulative Cost Safety Conditions*

To express K-step δ-safety through expected cumulative costs C, it remains to derive conditions which ensure that the system state $\mathbf{s}_h$ stays inside the sub-level set $\mathcal{C}^{\pi}_{\bar{\xi}}$ for all $h = 1, \ldots, K$ with probability δ. For this purpose, we employ techniques from stochastic stability analysis (Y. Li *et al.*, 2013), which are exploited in the following result.

Proposition 1. *Consider an immediate cost function $c : \mathcal{S} \to \mathbb{R}$, which satisfies the conditions* (5.8)*. Define $C_\pi(\mathbf{s}) \equiv C(f, \pi; \mathbf{s})$. Assume there exists a class $\mathcal{K}$ function [1] $\alpha : \mathbb{R} \to \mathbb{R}_{0,+}$, such that*

$$\mathbb{E}_{\boldsymbol{\omega}}[C_\pi(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] \leq C_\pi(\mathbf{s}) - \alpha(C_\pi(\mathbf{s}) - C_{\min})$$

holds for all $\mathbf{s} \in \mathcal{S}_{\text{safe}}$. Then,

$$\mathbb{E}_{\boldsymbol{\omega}}[C_\pi(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] \leq \xi < \bar{\xi} \tag{5.9}$$

1 A function $\alpha : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$ is a class $\mathcal{K}$ function, if it is monotonically increasing and $\alpha(0) = 0$.

guarantees that the policy π is K-step $\delta(\xi)$-safe.

Condition (5.9) effectively resembles a Foster-Lyapunov drift condition, which is satisfied if stochastic stability can be shown with C_π as a Lyapunov function (Meyn & Tweedie, 1993). Since stability is a well-studied problem, it has been shown that this condition can be satisfied for many dynamics f, e.g., systems which are asymptotically controllable with respect to the immediate cost c (Gaitsgory *et al.*, 2018). In contrast to stability theory, Proposition 1 does require C_π to be positive definite or the existence of a class $\mathcal{K}$ function lower bounding C_π. Therefore, the conditions of Proposition 1 are slightly weaker than for stability.

Due to the close relationship to stability, it is straightforward to see that the increase rate of α determines the convergence rate of the system. If α is only slowly growing, a relatively small noise realization can cause an increase in the expected cumulative cost, and thereby, increases the probability δ of leaving the safe set. This can be compensated by choosing a smaller value of ξ, such that there essentially is a larger margin between the safe initial states $\mathbf{s}_0$ and the unsafe set $\mathcal{S}_{\text{unsafe}}$. Note that the noise distribution also affects the probability δ through Equation (5.9), since flat distributions with heavy tails generally cause higher values of $\mathbb{E}_\omega[C(f,\pi)]$ leading to smaller increase rates of α.

5.3 HALLUCINATING UPPER CONFIDENCE SAFETY FILTERS (H-UCSF)

In the previous section, we derived conditions on the cumulative cost function C that ensured K-step δ-safety with known dynamics f. In this section, we extend the problem to unknown dynamics, but where we know a well-calibrated set of plausible models $\mathcal{M}$ that satisfy Assumption 7. We first tackle the question of how to verify that a policy is safe with unknown dynamics, i.e., how to lift Proposition 1 to the unknown dynamics setting. When we cannot verify that a policy is safe, we would like to have a safe backup policy that might not be available. The second question that we address is how to find a safe backup policy given $\mathcal{M}$. While the backup policy is safe, the performance could be arbitrarily poor. In the last subsection, we develop H-UCSF, a confidence based safety filter that ensures constraint satisfaction of any arbitrary policy π. The hope is that when the policy π has high performance, the safety filter will minimally adjust the actions of π to ensure safety yet achieve high performance.

5.3.1 *Safety Certification with Unknown Dynamics*

Since we assume only the availability of a set of plausible models $\mathcal{M}$, but not the true dynamics f, we cannot determine C and consequently cannot directly exploit Proposition 1 for determining a safe policy.

To overcome this issue, it is straightforward to see that

$$\max_{\tilde{f}\in\mathcal{M}} C(\tilde{f},\pi;\mathbf{s}) \leq \theta \quad \Rightarrow \quad C(f,\pi;\mathbf{s}) \leq \theta \tag{5.10}$$

for every $\theta \in \mathbb{R}$ due to Assumption 7. To solve the optimization problem on the left hand side of the safety condition (5.10), we use the hallucination reparameterization (2.31) and reformulate the left side of (5.10) as a policy optimization problem:

$$\max_{\tilde{f}\in\mathcal{M}} C(\tilde{f},\pi;\mathbf{s}) = \max_{\eta\in\mathcal{U}} C(\tilde{f},\pi;\mathbf{s}), \quad \text{s.t.} \quad \tilde{f}(\cdot) = \boldsymbol{\mu}(\cdot) + \beta\boldsymbol{\sigma}(\cdot)\eta(\cdot) \tag{5.11}$$

Using this formulation as an optimization of the hallucinating policy η, it is straightforward to extend Proposition 1 to unknown dynamics f. Namely, let's define the pessimistic cumulative cost as

$$C_\pi^{(p)}(\mathbf{s}) \equiv \max_{\eta\in\mathcal{U}} C(\tilde{f},\pi;\mathbf{s}), \quad \text{s.t.} \quad \tilde{f}(\cdot) = \boldsymbol{\mu}(\cdot) + \beta\boldsymbol{\sigma}(\cdot)\eta(\cdot). \tag{5.12}$$

Then the following proposition can be used to certify that a policy is K-step δ-safe.

Proposition 2. *Consider a set of plausible models $\mathcal{M}$ satisfying Assumption 7 and an immediate cost c, which satisfies* (5.8)*. If*

$$\max_{\eta\in\mathcal{U}} \mathbb{E}_{\boldsymbol{\omega}}\left[C_\pi^{(p)}(\mathbf{s}')\right] \leq \xi, \tag{5.13}$$

with $\mathbf{s}'$ is the next-state defined through the reparameterized dynamics (2.33) *and $\xi \leq \bar{\xi}$, and $C_\pi^{(p)}(\mathbf{s})$ satisfies* (5.9)*, then, the π is K-step δ-safe.*

Proposition 2 is a generalization of Proposition 1 for unknown dynamics using the pessimistic estimate $C_\pi^{(p)}(\cdot)$ of the cumulative costs instead of the true $C_\pi(\cdot)$. When the model is known accurately, i.e., $\boldsymbol{\sigma}(\mathbf{s},\mathbf{a}) = \mathbf{0}$ for all $\mathbf{s},\mathbf{a} \in \mathcal{S}\times\mathcal{A}$, the conditions of Proposition 2 intuitively reduce to the conditions of Proposition 1.

Solving the verification condition (5.13) for a fixed policy π reduces to solving a standard RL problem for the hallucination η. To see this, we define

$$\hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi, \eta) = \mathbb{E}_{\boldsymbol{\omega}} \left[\sum_{h=0}^{\infty} \gamma^h c(\mathbf{s}_h, \pi(\mathbf{s}_h)) \right], \tag{5.14}$$

$$\text{s.t. } \mathbf{s}_{h+1} = \tilde{f}(\mathbf{s}_h, \pi(\mathbf{s}_h)) + \boldsymbol{\omega}_h \tag{5.15}$$

$$\tilde{f}(\cdot) = \boldsymbol{\mu}(\cdot) + \beta \boldsymbol{\sigma}(\cdot) \eta(\cdot) \tag{5.16}$$

$$\mathbf{s}_0 = \mathbf{s}. \tag{5.17}$$

Thus, for a fixed policy π, $\hat{C}$ is a standard cumulative reward function, such that actor-critic methods as those described in Section 2.3 can be used to learn $\hat{C}$.

5.3.2 *Learning Safe Policies with Robust Reinforcement Learning*

Based on the formulation of the K-step δ-safety as an optimization problem in Proposition 2, it is natural to augment the optimization problem to directly find δ-safe policies.

$$\pi_{\text{safe}} = \operatorname*{argmin}_{\pi \in \Pi} \max_{\eta \in \mathcal{U}} \hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi, \eta) \tag{5.18}$$

$$\eta_{\text{adv}} = \operatorname*{argmax}_{\eta \in \mathcal{U}} \hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi_{\text{safe}}, \eta), \tag{5.19}$$

where the solution for the hallucinating policy η_{adv} acts adversarially on the system. Therefore, we refer to η_{adv} as the hallucinating adversarial policy in the sequel. In contrast to the safety certification using Proposition 2, determining safe policies π_{safe} using Equation (5.18) cannot be formulated as a standard reinforcement learning problem due to the minimization over policies π and the maximization over hallucination policies η. However, it can be straightforwardly addressed using robust reinforcement learning techniques (Pinto *et al.*, 2017; Curi *et al.*, 2021), which perform gradient descent for π and gradient ascent for η. Therefore, we refer to π_{safe} as the *learned safe policy*.

Moreover, if the cost c and the discount γ allow to establish the safety of this system for some policy π, it is straightforward to show that the maximally safe problem (5.18) yields a δ-safe policy.

Proposition 3. *If there exists a policy π and a class $\mathcal{K}$ function α such that $C_{\pi}^{(p)}$ satisfies condition (5.9), then, the learned safe policy π_{safe} is K-step δ-safe for all $\mathbf{s} \in \mathcal{S}_{\text{safe}}$ if $C_{\pi_{\text{safe}}}^{(p)} \leq \xi < \bar{\xi}$.*

Since there exist combinations of dynamics f and safe sets $\mathcal{S}_{\text{safe}}$ for which safety cannot be ensured, Proposition 3 obviously cannot guarantee the existence of a learned safe policy π_{safe} to be always K-step δ-safe. However, as discussed in Section 5.2.2, there exist system classes for which condition (5.9) is satisfied.

5.3.3 *Ensuring Constraint Satisfaction with Safety Filters*

While the learned safe policy π_{safe} is safe, the performance can be arbitrarily poor since it does not consider the reward function r when optimizing π_{safe}. Given any policy π with high performance, the goal of this section is to design a filter for the policy to ensure constraint satisfaction while preserving as much as possible its performance. The core idea for achieving this relies on a continuous monitoring of every nominal action $\pi(\mathbf{s})$, such that they can be adapted to ensure a safe roll-out of π_{safe} afterwards. Using the the hallucination reparameterization (2.31) of the set of plausible models $\tilde{f}$, it yields our *confidence-based safety filter*

$$\hat{\pi}(\mathbf{s}) = \underset{\mathbf{a} \in \mathcal{A}}{\operatorname{argmin}} \|\pi(\mathbf{s}) - \mathbf{a}\|, \tag{5.20a}$$

$$\text{s.t.} \max_{\mathbf{u} \in [-1,1]^{d_x}} \mathbb{E}_{\boldsymbol{\omega}} \left[C_{\pi_{\text{safe}}}^{(p)}(\mathbf{s}') \right] \leq \xi, \tag{5.20b}$$

$$\mathbf{s}' = \boldsymbol{\mu}(\mathbf{s}, \mathbf{a}) + \beta \boldsymbol{\sigma}(\mathbf{s}, \mathbf{a}) \mathbf{u} + \boldsymbol{\omega}. \tag{5.20c}$$

The safety filter executes an action $\mathbf{a}$ that is as closed as possible to the policy action $\pi(\mathbf{s})$, such that the next state is safe if the policy π_{safe} is executed thereafter. This is then executed in a receding-horizon fashion. Finally, we can see that when $\mathbf{s} \in \mathcal{C}_{\xi}^{\pi}$ and π_{safe} satisfies Proposition 3, the constraint (5.20b) is satisfied by setting $\mathbf{a} = \pi_{\text{safe}}(\mathbf{s})$. The expected cumulative cost function $C_{\pi_{\text{safe}}}^{(p)}$ is computed when solving for $(\pi_{\text{safe}}, \eta_{\text{adv}})$ in Section 5.3.2 and it can be thought of as a terminal constraint that must be satisfied. Furthermore, one could extend the safety filter to k actions by including the constraints (5.8) in the safety filter (5.20) for all the intermediate hallucinated states and evaluating (5.20b) only at the final state. We do not include this as it increases the real-time computational cost of solving the safety filter (5.20).

We still need to address the cases in which $\mathbf{s} \notin \mathcal{C}^{\pi}_{\xi}$ but $\mathbf{s} \in \mathcal{S}_{\text{safe}}$. In such cases, the safety filter just deploys the safe policy which is safe if it satisfies the conditions of Proposition 3. The final H-UCSF policy is.

$$\pi_{\text{H-UCSF}}(\mathbf{s}) = \begin{cases} \hat{\pi}(\mathbf{s}) & \text{if } C^{(p)}_{\pi_{\text{safe}}}(\mathbf{s}) \leq \xi, \\ \pi_{\text{safe}}(\mathbf{s}) & \text{if } C^{(p)}_{\pi_{\text{safe}}}(\mathbf{s}) \geq \xi. \end{cases} \tag{5.21}$$

Due to its strong foundation on the learned safe policy π_{safe}, the roll-out policy $\pi_{\text{H-UCSF}}$ inherits its theoretical safety guarantees as shown in the following theorem.

Theorem 3. *Consider a set of plausible models $\mathcal{M}$ satisfying Assumption 7 and assume that the learned safe policy π_{safe} satisfies the conditions of Proposition 3. Then, the confidence-based safety filtered policy* (5.21) *is K-step δ-safe for all states $\mathbf{s} \in \mathcal{S}_{\text{safe}}$.*

While the safety filter problem (5.20) is not compatible with standard reinforcement learning methods, it can easily be solved online using model predictive control. Hence, (5.20) requires optimization merely for one time step and consequently only for a single actual and hallucinating adversarial action in contrast to similar predictive safety filter approaches Bastani, 2021; Wabersich *et al.*, 2021, which require optimization over a sequence of actions. Therefore, the safety filter (5.20) can be solved with comparatively low computational complexity using numerical optimization schemes, which allows a straightforward online application as safety filter.

Practically, ξ in the safety filter (5.20) can be considered a tuning parameter. The smaller its value, the higher the probability of safety. However, a small ξ will lead to more conservatism of the safety filter, such that it must be carefully chosen to trade-off safety and performance.

5.4 EXPERIMENTS

In this section, we evaluate the safety filter and compare it with three competing algorithms: the constraint-free model-free algorithm SAC Haarnoja *et al.*, 2018, a Lagrangian primal-dual approach with SAC as the base algorithm, which we call CMDP Paternain *et al.*, 2019, and the model-based alternative Safe-CEM Liu *et al.*, 2020. We consider two widely used environments to test our approach. First, we test it on an airplane pitch control Hafner & Riedmiller, 2011, where the pitch angle θ starts at -0.2 radians and the constraint function is simply $c_t = \theta_t$ such that the angle

should never exceed 0. The reward is given by $r_t = -2\theta^2 + 0.02u^2$, where u is the control input. Second, we use the Mujoco Half-Cheetah environment with the default reward function Todorov *et al.*, 2012. The constraint is that the forward speed is less than 2. Due to the Cheetah's trot, the penalty is on the average forward speed, calculated as $\bar{v}_t = 0.1v_t + 0.9\bar{v}_{t-1}$, $\bar{v}_0 = 0$, where v_t is the instantaneous speed and $\bar{v}_t$ is the average speed. Thus we use $c_t = \bar{v}_t - 2$. We run each environment for 100 episodes, each episode for 1000 time steps, using $\gamma = 0.99$ as a discount factor.

To learn the model, we use deterministic ensembles of five members following Curi *et al.*, 2020a. Each member is a neural network with 3 fully connected layers of width 200 and Swish non-linearities. For the first ten episodes, data is collected using a random policy. Such random policy was safe in these environments but only at the given initial conditions, i.e., it is not the learned safe policy used by the safety filter. After the initial exploration phase, the model is pre-trained for 100 iterations using Adam with learning rate 0.0005 and weight decay 0.0001. Then, after each subsequent episode, the model is updated using the additional data collected during the episode. We store the data using an experience replay buffer of at most 100000 transitions. Finally, to solve the safety filter problem (5.20) we use the cross-entropy method Botev *et al.*, 2013b with 1000 particles and 5 iterations per time-step.

In Figure 5.2, we show the results in the pitch control environment. In this setting, only the Safety Filter algorithm avoids any constraint violation while achieving comparable performance in terms of returns and costs. In Figure 5.3, we show the results for the Half-Cheetah. Here, both Safe-CEM and the safety filter avoid any constraint violations. However, the safety filter achieves higher returns than Safe-CEM. The main difference between these two environments relies on the backup policy. While in the Cheetah it is enough to *do nothing* in order to stop it, in the Pitch Control environment this is not the case and the *learned* safe backup policy is crucial to ensure safety. Thus, with these two environments we demonstrate the scalability of our method in the Half Cheetah environment as well as the ability to satisfy constraints in the Pitch Control environment.

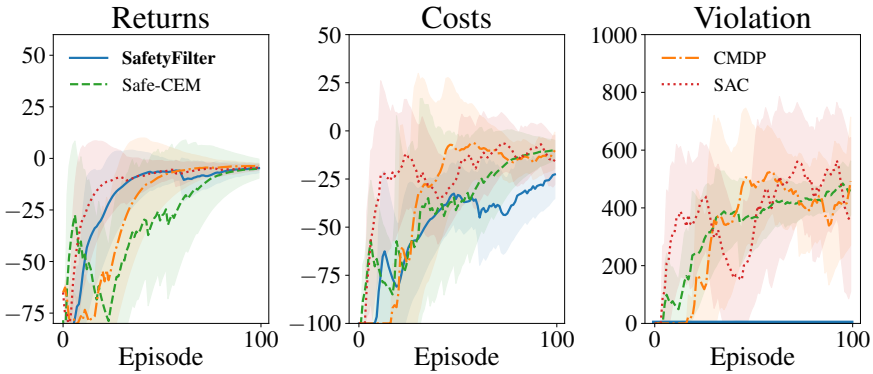


FIGURE 5.2: Total returns, costs, and constraint violation in the Pitch Control environment. Only the safety filter attains **no** constraint violations and achieves comparable performance to the benchmarks.

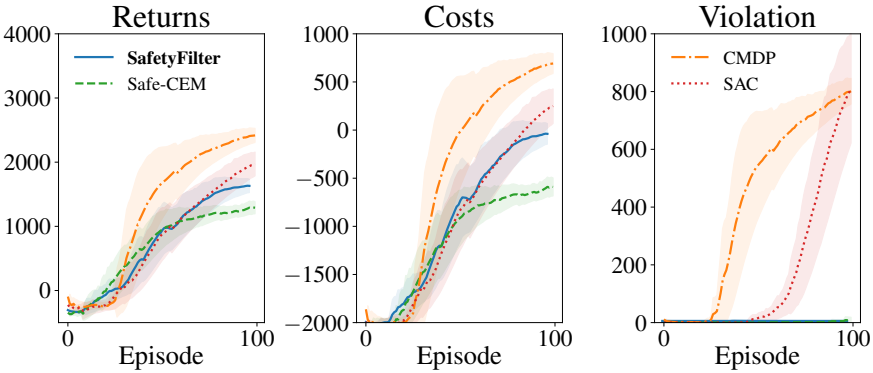


FIGURE 5.3: Total returns, costs, and constraint violation in the Half-Cheetah environment. The safety filter and Safe-CEM achieve **no** constraint violation. These two algorithms perform slightly worse than the benchmarks in terms of returns, but the safety filter performs better than Safe-CEM.

6

CONCLUSION

El éxito no es la victoria, sino todo lo que has peleado por ganar.

— Rafa Nadal

In this dissertation, we used MBRL to derive algorithms for efficient exploration, robustness, and safety. These goals were achieved while reasoning about the models' epistemic uncertainty.

The main technical contribution that enables a practical implementation of all the algorithms is the concept of hallucinating policies, which we introduce in Chapter 2. Instead of optimizing over the set of plausible models which is intractable, we reparameterize such set introducing an additional *hallucinated* control input that has no associated control penalties and can exert as much control as the current *epistemic* uncertainty that the model affords.

Using the hallucinated controls, we derive in Chapter 3 the H-UCRL algorithm for data-efficient exploration. H-UCRL is provably efficient for GP models and lends itself to a gradient-based implementation with NN models, scaling to larger problems. The critical insight is that with the reparameterization policies, we reduce the intractable theoretical optimistic problem to a standard planning problem where many practical algorithms exist.

Next, we tackle the problem of outputting a robust policy in Chapter 4 and introduce the RH-UCRL algorithm. By carefully training with a fictitious adversary, we can certify the performance of the policy even RH-UCRL uses optimistic and pessimistic estimates of the robust performance to efficiently explore both the agent and fictitious adversary decision spaces during policy learning. To compute the optimistic and pessimistic estimates of the performance, RH-UCRL relies on the *hallucinating* inputs reducing the optimistic/pessimistic problem to traditional robust planning problems. We show that RH-UCRL is provably robust, and we established sample complexity and regret guarantees. We instantiated our algorithm in important robust-RL settings such as adversarial-robust RL, parameter-robust RL, and action-robust RL.

Finally, in Chapter 5 we address the problem of filtering a policy to ensure safety with the H-UCSF algorithm. Here, the models' epistemic uncertainty plays two roles. First, we use it to find a *maximally safe* learned policy by solving a robust RL problem: the control inputs minimize the constraint violations, whereas the hallucinating inputs maximize the constraint violations. Under some conditions, we can guarantee that the learned backup policy is safe. Second, we use the model to filter the actions of any policy, solving a 1-step horizon MPC algorithm online with the safe learned backup policy.

6.1 FUTURE WORK

We finish the main body of this dissertation with open questions and exciting directions for future work.

In Chapter 3, we focus on using epistemic uncertainty to explore in a data-efficient fashion. However, in many settings, we already have logged data collected with an offline policy. This is called the offline RL setting. Instead of being *optimistic*, in this setting, one hopes to use *pessimism* to learn a policy to certify the behavior w.r.t. the worst case model compatible with data. This opens the possibility of using data from different sources, meta-learn model priors, and transfer learning between different tasks.

When we consider robustness in Chapter 4, we focus on a notion of robustness that compares against a worst-case adversary. Although this is a strong notion, it could also be an unlikely adversary. To tackle this, one could frame the problem using distributional robustness, where the goal is to be robust with respect to a worst-case distribution. The distributional robust setting suits sim-to-real applications, in which the worst-case simulation might be too unreal to be true, whereas a worst-case distribution of simulated environments could better capture reality.

In the safety filter setting in Chapter 5, we consider that we were given a model set, and we use such set to guarantee safety. However, it is unclear how we got that set in the first place. An interesting direction is to use transfer learning or offline RL to learn these sets. Another important aspect is that we were *pessimistic* w.r.t. the epistemic uncertainty to find the safe backup policy. However, when the model set is too large this could be *overly-pessimistic*, and we will not be able to guarantee safety. An important open question is how to find tight model sets.

Finally, our results in this thesis are statistical and rely on the performance of model learning and policy learning algorithms. Perhaps the most exciting question in model learning is designing neural network representations

that correctly capture epistemic uncertainty while allowing for efficient sampling. Such models would enable Thompson sampling implementations of our algorithms that would yield straightforward optimization problems. There are still theoretical and practical questions in the policy learning scope to be addressed. From a theoretical point of view, it is unclear if any algorithm can find the best policies from within a policy class. From a practical perspective, successful RL algorithms require large amounts of hyper-parameter tuning, restricting their applicability. Designing out-of-the-box RL algorithms with theoretical guarantees is a task of utmost importance.

A

PROOFS OF CHAPTER 3

We start by bounding the simple regret $\mathfrak{r}_n$ at episode n as the difference between the value of the policy on the optimistic dynamics and the value of policy on the true dynamics

Lemma 10 (Simple regret bound)**.** *Under Assumptions 1 to 7, with probability at least* $(1-\delta)$ *we have for all* $n \geq 0$ *that the simple regret* $\mathfrak{r}_n$ *is bounded by*

$$\mathfrak{r}_n = J(f, \pi^*) - J(f, \pi_n) \leq J(\tilde{f}_n, \pi_n) - J(f, \pi_n) \tag{A.1}$$

Proof. By Assumption 7, we know that the true dynamics are contained within set of plausible models, i.e., $f \in \mathcal{M}_n$. Furthermore, from Lemma 6, we know that there exists an $\hat{\eta} \in \mathcal{U} \colon \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to [-1, 1]^{d_s}$ such that with $\hat{f}_n(\cdot) = \boldsymbol{\mu}_{n-1}(\cdot) + \beta_{n-1}\boldsymbol{\sigma}_{n-1}(\cdot)\hat{\eta}(\cdot)$ we have $J(f, \pi^*) = J(\hat{f}_n, \pi^*)$. As a consequence, we have from the joint maximization in the H-UCRL algorithm (3.6) that $J(f, \pi^*) \leq J(\tilde{f}_n, \pi_n)$ and the result follows. □

Thus, to bound the instantaneous regret $\mathfrak{r}_n$, we must bound the difference between the optimistic value estimate $J(\tilde{f}_n, \pi_n)$ and the true value of the policy $J(f_n, \pi_n)$. The following step is to bound this difference in term of the models' predictive variance. The main idea is to bound the difference in performance by the difference in state trajectories, and the difference in state trajectories by the predictive variance. For many models, the predictive variance decreases with the number of data points and this technique yields sublinear regret bounds.

We can use the Lipschitz continuity properties to obtain the following bound

Lemma 11 (Lipschitz continuity of closed loop dynamics)**.** *Let the open-loop dynamics* f *in Equation* (2.1) *be* L_f*-Lipschitz continuous by Assumption 1 and the policy* $\pi \in \Pi$ *be* L_π*-Lipschitz continuous w.r.t. to the 2-norm. Then the closed-loop system is* $L_{f,\pi}$*-Lipschitz continuous with* $L_{f,\pi} = L_f\sqrt{1 + L_\pi}$.

Proof.

$$\|f(\mathbf{s}, \pi(\mathbf{s})) - f(\mathbf{s}', \pi(\mathbf{s}'))\|_2 \leq L_f \|(\mathbf{s} - \mathbf{s}', \pi(\mathbf{s}) - \pi(\mathbf{s}'))\|_2 \tag{A.2}$$

$$= L_f \sqrt{\|(\mathbf{s} - \mathbf{s}'\|_2^2 + \|\pi(\mathbf{s}) - \pi(\mathbf{s}'))\|_2^2} \tag{A.3}$$

$$\leq L_f \sqrt{\|(\mathbf{s} - \mathbf{s}'\|_2^2 + L_\pi \|\mathbf{s} - \mathbf{s}'))\|_2^2} \tag{A.4}$$

$$= \underbrace{L_f \sqrt{1 + L_\pi}}_{:= L_{f,\pi}} \|\mathbf{s} - \mathbf{s}'\|_2 \tag{A.5}$$

□

Lemma 12 (Performance difference as difference in state trajectory)**.** *Based on Assumptions 1 to 7 we have*

$$\left|J(\tilde{f}_n, \pi_n) - J(f, \pi_n)\right| \leq L_r \sqrt{1 + L_\pi} \sum_{h=0}^{H} \mathbb{E}_{\omega = \tilde{\omega}}[\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2] \tag{A.6}$$

Proof.

$$\left|J(\tilde{f}_n, \pi_n) - J(f, \pi_n)\right| = \left| \mathbb{E}_{\tilde{\omega}} \left[\sum_{h=0}^{H} r(\tilde{\mathbf{s}}_h, \pi_n(\tilde{\mathbf{s}}_h)) \right] - \mathbb{E}_{\omega} \left[\sum_{h=0}^{H} r(\mathbf{s}_h, \pi_n(\mathbf{s}_h)) \right] \right| \tag{A.7a}$$

$$= \left| \mathbb{E}_{\omega = \tilde{\omega}} \left[\sum_{h=0}^{H} r(\tilde{\mathbf{s}}_{h,n}, \pi_n(\tilde{\mathbf{s}}_{h,n})) - r(\mathbf{s}_{h,n}, \pi_n(\mathbf{s}_{h,n})) \right] \right| \tag{A.7b}$$

$$\leq L_r \sqrt{1 + L_\pi} \sum_{h=0}^{H} \mathbb{E}_{\omega = \tilde{\omega}}[\|\tilde{\mathbf{s}}_{h,n} - \mathbf{s}_{h,n}\|_2], \tag{A.7c}$$

where $\mathbb{E}_{\omega = \tilde{\omega}}[\cdot]$ means in expectation over ω and with $\tilde{\omega} = \omega$; that is, $\tilde{\omega}$ and ω are the same random variable. □

What remains is to bound the deviation of the optimistic and the true trajectory. We exploit the Lipschitz continuity of σ from Assumption 6 in order to bound the deviation in terms of σ_{n-1} at states of the true state trajectory $\mathbf{s}_{h,n}$.

Lemma 13 (Difference in state trajectory as sum of predictive variance along true trajectory)**.** *Under Assumptions 1 to 7, for all episodes $n \geq 1$, $h \in \{1, \dots, H\}$, and $\pi \in \Pi$ it holds:*

$$\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \leq 2\beta_{n-1}\kappa_{n-1}^{h-1} \sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h,n})\|_2, \tag{A.8}$$

where $\kappa_{n-1} \coloneqq \left(1 + (L_f + 2\beta_{n-1}L_\sigma)\sqrt{1 + L_\pi^2}\right)$, the closed-loop epistemic uncertainty as $\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h,n}) \coloneqq \boldsymbol{\sigma}_{n-1}(\mathbf{s}_{h,n}, \pi_n(\mathbf{s}_{h,n}))$, the simulated state $\tilde{\mathbf{s}}_{h,n}$ is generated by any system $\tilde{f} \in \mathcal{M}_n$ (cf., Section 2.2.1), and the true state $\mathbf{s}_{h,n}$ is generated by the true dynamics f, with $\boldsymbol{\omega}_{h,n} = \tilde{\boldsymbol{\omega}}_{h,n}$.

Proof. To avoid notational clutter, we denote the closed-loop dynamics as $f^\pi(\mathbf{s}) = f(\mathbf{s}, \pi(\mathbf{s}))$. Likewise, we use the following Lipschitz constants shorthands $L_{f,\pi} \equiv L_f\sqrt{1 + L_\pi^2}$ and $L_{\sigma,\pi} \equiv L_\sigma\sqrt{1 + L_\pi^2}$.

We first prove by induction that

$$\begin{aligned}
&\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \\
&\qquad \leq 2\beta_{n-1} \sum_{h'=0}^{h-1} \left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1-h'} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|
\end{aligned} \tag{A.9a}$$

For $h = 0$, clearly $\mathbf{s}_{0,n} = \tilde{\mathbf{s}}_{0,n}$, while the right-hand-side of inequality (A.9a) is always non-negative. We assume that for h the inductive hypothesis (A.9a) holds. For $h+1$ we have:

$$
\begin{aligned}
&\|\mathbf{s}_{h+1,n} - \tilde{\mathbf{s}}_{h+1,n}\|_2 \\
&\quad = \|f^{\pi_n}(\mathbf{s}_{h,n}) - \tilde{f}^{\pi_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(A.9b)}\\
&\quad = \|f^{\pi_n}(\mathbf{s}_{h,n}) - \tilde{f}^{\pi_n}(\tilde{\mathbf{s}}_{h,n}) + f^{\pi_n}(\tilde{\mathbf{s}}_{h,n}) - f^{\pi_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(A.9c)}\\
&\quad \le \|f^{\pi_n}(\mathbf{s}_{h,n}) - f^{\pi_n}(\tilde{\mathbf{s}}_{h,n})\|_2 + \|f^{\pi_n}(\tilde{\mathbf{s}}_{h,n}) - \tilde{f}^{\pi_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(A.9d)}\\
&\quad \le L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 + \|f^{\pi_n}(\tilde{\mathbf{s}}_{h,n}) - \tilde{f}^{\pi_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(A.9e)}\\
&\quad \le L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 + 2\beta_{n-1}\|\sigma^{\pi_n}_{n-1}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(A.9f)}\\
&\quad = L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \\
&\qquad + 2\beta_{n-1}\|\sigma^{\pi_n}_{n-1}(\tilde{\mathbf{s}}_{h,n}) + \sigma^{\pi_n}_{n-1}(\mathbf{s}_{h,n}) - \sigma^{\pi_n}_{n-1}(\mathbf{s}_{h,n})\|_2 && \text{(A.9g)}\\
&\quad \le L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \\
&\qquad + 2\beta_{n-1}\left(\|\sigma^{\pi_n}_{n-1}(\tilde{\mathbf{s}}_{h,n}) - \sigma^{\pi_n}_{n-1}(\mathbf{s}_{h,n})\|_2 + \|\sigma^{\pi_n}_{n-1}(\mathbf{s}_{h,n})\|_2\right) && \text{(A.9h)}\\
&\quad \le \left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 + 2\beta_{n-1}\|\sigma^{\pi_n}_{n-1}(\mathbf{s}_{h,n})\|_2 && \text{(A.9i)}\\
&\quad \le 2\beta_{n-1}\sum_{h'=0}^{(h+1)-1}\left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{(h+1)-1-h'}\|\sigma^{\pi_n}_{n-1}(\mathbf{s}_{h',n})\|_2 && \text{(A.9j)}
\end{aligned}
$$

Here, Equation (A.9b) holds by applying the transition dynamics f^{π_n} and $\tilde{f}^{\pi_n}$ with the same noise realization $\boldsymbol{\omega}_h = \tilde{\boldsymbol{\omega}}_h$; Equation (A.9c) holds by adding and subtracting $f^{\pi_n}(\tilde{\mathbf{s}}_{h,n})$; inequality (A.9d) follows from the triangular inequality; inequality (A.9e) comes from Lemma 11; inequality (A.9f) holds due to both f and $\tilde{f}$ belonging to the set of plausible models $\mathcal{M}_n$; Equation (A.9g) holds by adding and substracting $\sigma^{\pi_n}(\mathbf{s}_{h,n})$; inequality eq. (A.9h) holds by applying the triangular inequality once more; inequality eq. (A.9i) is due to the Lipschitz continuity of σ as per Assumption 6; and inequality (A.9j) holds by replacing the inductive hypothesis (A.9a).

Finally, with simple algebraic manipulation we can see that,

$$
\begin{aligned}
\left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1-h'} &< \left(1 + L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1-h'} \\
&\le \left(1 + L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1},
\end{aligned}
$$

and the main result follows by combining this with Equation (A.9j). □

As a direct consequence of these lemmas, we can bound the simple regret in terms of the predictive uncertainty of our statistical model in expectation over the states visited under the true dynamics.

Lemma 14 (Difference between optimistic and pessimistic performance as sum of predictive variance along true trajectory)**.** *Under Assumptions 1 to 7, let π_n be the policies selected by* H-UCRL *at episode n. Then, the following holds for the difference between its optimistic and pessimistic performance:*

$$J(\tilde{f}_n, \pi_n) - J(f, \pi_n) \leq 2L_r \beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2\right], \tag{A.10}$$

where $\boldsymbol{\sigma}^{\pi}(\mathbf{s}) = \boldsymbol{\sigma}(\mathbf{s}, \pi(\mathbf{s}))$ and $C \coloneqq (1 + L_f + L_\sigma)(1 + L_\pi^2)^{1/2}$.

Proof. From Lemma 12 we know that:

$$J(\tilde{f}_n, \pi_n) - J(f, \pi_n) \leq L_r \sqrt{1 + L_\pi^2} \sum_{h=0}^{H} \mathbb{E}[\|\tilde{\mathbf{s}}_{h,n} - \mathbf{s}_{h,n}\|_2]. \tag{A.11a}$$

We recall the upper bound on $\|\tilde{\mathbf{s}}_{h,n} - \mathbf{s}_{h,n}\|_2$ from Lemma 13:

$$\|\mathbf{s}_{h,n} - \mathbf{s}_{h,n}^{(o)}\|_2 \leq 2\beta_{n-1}\kappa_{n-1}^{h-1} \sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2.$$

As $\tilde{f} \in \mathcal{M}_n$, and by denoting $C \coloneqq (1 + L_f + 2L_\sigma)(1 + L_\pi^2)^{1/2}$, we arrive at:

$$J(\tilde{f}_n, \pi_n) - J(f, \pi_n) \leq 2L_r \beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2\right],$$

where we used $n \leq N$ and $1 \leq \beta_n$ is non-decreasing in n. □

Now we are finally ready to prove our main result.

Theorem 1. *Under Assumptions 1 to 7, let $C = (1 + L_f + 2L_\sigma)(1 + L_\pi^2 + L_{\hat{\pi}}^2)^{1/2}$ and $\mathbf{s}_{h,n} \in \mathcal{S}$ and $\mathbf{a}_{h,n} \in \mathcal{A}$ for all $h, n > 0$. Then, for all $N \geq 1$, with probability at least $(1 - \delta)$, the regret of* H-UCRL *in eq. (3.6) is at most*

$$\mathfrak{R}_N \leq \mathcal{O}\Big(L_r C^H \beta_N^H H^{3/2} \sqrt{N \Gamma_{N}}\Big). \tag{3.8}$$

Proof of Theorem 1. We bound the cumulative regret as follows:

$$\mathfrak{R}_N = \sum_{n=1}^{N} \underbrace{J(f,\pi^\star) - J(f,\pi_n)}_{:=\mathfrak{r}_n} \tag{A.12a}$$

$$\leq \sqrt{N \sum_{n=1}^{N} r_n^2} \tag{A.12b}$$

$$\leq \sqrt{N \sum_{n=1}^{N} (2L_r \beta_N^H C^H)^2 \left(\sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2 \right] \right)^2} \tag{A.12c}$$

$$= 2L_r \beta_N^H C^H \sqrt{N} \sqrt{\sum_{n=1}^{N} \left(\sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2 \right] \right)^2} \tag{A.12d}$$

$$\leq 2L_r \beta_N^H C^H H \sqrt{N} \sqrt{\sum_{n=1}^{N} \left(\mathbb{E}\left[\sum_{h'=0}^{H} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2 \right] \right)^2} \tag{A.12e}$$

$$\leq 2L_r \beta_N^H C^H H \sqrt{N} \sqrt{\sum_{n=1}^{N} \mathbb{E}\left[\left(\sum_{h'=0}^{H} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2 \right)^2 \right]} \tag{A.12f}$$

$$\leq 2L_r \beta_N^H C^H H^{3/2} \sqrt{N} \sqrt{\sum_{n=1}^{N} \mathbb{E}\left[\sum_{h'=0}^{H} \|\boldsymbol{\sigma}_{n-1}^{\pi_n}(\mathbf{s}_{h',n})\|_2^2 \right]} \tag{A.12g}$$

$$\leq 2L_r \beta_N^H C^H H^{3/2} \sqrt{N\Gamma_N}, \tag{A.12h}$$

where eq. (A.12b) is due to the Cauchy-Schwarz's inequality; eq. (A.12c) is due to Lemma 14. Finally, eq. (A.12f) follows from Jensen's inequality, eq. (A.12g) follows from Cauchy-Schwarz's inequality, and eq. (A.12h) follows from the definition of Γ_N in Equation (2.11). □

B

PROOFS OF CHAPTER 4

To prove Theorem 2 and Corollary 1, we follow a similar structure to the proof of Theorem 1. We start by bounding its simple robust-regret by the difference between optimistic and pessimistic performance estimates.

Lemma 15 (Simple robust-regret bound)**.** *Let $\pi^\star$ be the benchmark policy from eq.* (4.3)*, and let π_n and $\hat{\pi}_n$ be the policies selected by* RH-UCRL *at episode n. Under the callibrated model Assumption 7, the following holds with probability at least $1-\delta$:*

$$\bar{\mathfrak{r}}_n \coloneqq \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi^\star,\hat{\pi}) - \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi_n,\hat{\pi}) \leq J_n^{(o)}(\pi_n,\hat{\pi}_n) - J^{(p)}(\pi_n,\hat{\pi}_n). \quad \text{(B.1)}$$

Proof. We refer to the considered quantity $\bar{\mathfrak{r}}_n$ as the simple robust-regret of the selected policy π_n, and we proceed by providing its upper bound:

$$\begin{aligned}
\bar{\mathfrak{r}}_n &\coloneqq \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi^\star,\hat{\pi}) - \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi_n,\hat{\pi}) && \text{(B.2a)}\\
&\leq \min_{\hat{\pi}\in\hat{\Pi}} J_n^{(o)}(\pi^\star,\hat{\pi}) - \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi_n,\hat{\pi}) && \text{(B.2b)}\\
&\leq \min_{\hat{\pi}\in\hat{\Pi}} J_n^{(o)}(\pi_n,\hat{\pi}) - \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi_n,\hat{\pi}) && \text{(B.2c)}\\
&\leq J_n^{(o)}(\pi_n,\hat{\pi}_n) - \min_{\hat{\pi}\in\hat{\Pi}} J(f,\pi_n,\hat{\pi}) && \text{(B.2d)}\\
&\leq J_n^{(o)}(\pi_n,\hat{\pi}_n) - \min_{\hat{\pi}\in\hat{\Pi}} J^{(p)}(\pi_n,\hat{\pi}) && \text{(B.2e)}\\
&= J_n^{(o)}(\pi_n,\hat{\pi}_n) - J^{(p)}(\pi_n,\hat{\pi}_n). && \text{(B.2f)}
\end{aligned}$$

Here, inequality (B.2b) holds by definition of the optimistic estimate in eq. (4.5a); inequality (B.2c) holds by definition of protagonist policy in the RH-UCRL algorithm (4.7a); and inequality (B.2e) holds by definition of the pessimistic estimate in eq. (4.6a); finally, equality (B.2f) holds by definition of the antagonist policy in the RH-UCRL algorithm (4.7b). □

The following step is to bound this difference in term of the models' predictive variance. The main idea is to bound the difference in performance by the difference in state trajectories, and the difference in state trajectories

by the predictive variance. For this we use only Lipschitz continuity of the dynamics, rewards, and policies.

We will first show that the performance difference of a pair of policies $(\pi, \hat{\pi})$ on the true dynamics f and any dynamics $\tilde{f}$ is bounded by a constant times the sum of the difference between the true and the simulated state trajectories.

Lemma 16 (Lipschitz continuity of closed loop dynamics)**.** *Under Assumptions 1, 4 and 8, for every $\mathbf{s}, \mathbf{s}' \in \mathcal{S}$, it holds:*

$$\|f(\mathbf{s}, \pi(\mathbf{s}), \hat{\pi}(\mathbf{s})) - f(\mathbf{s}', \pi(\mathbf{s}'), \hat{\pi}(\mathbf{s}'))\|_2 \leq L_f \sqrt{1 + L_\pi^2 + L_{\hat{\pi}}^2} \|\mathbf{s} - \mathbf{s}'\|_2. \quad \text{(B.3)}$$

Proof.

$$\begin{aligned}
&\|f(\mathbf{s}, \pi(\mathbf{s}), \hat{\pi}(\mathbf{s})) - f(\mathbf{s}', \pi(\mathbf{s}'), \hat{\pi}(\mathbf{s}'))\|_2 \\
&\quad \leq L_f \sqrt{\|\mathbf{s} - \mathbf{s}'\|_2^2 + \|\pi(\mathbf{s}) - \pi(\mathbf{s}')\|_2^2 + \|\hat{\pi}(\mathbf{s}') - \hat{\pi}(\mathbf{s})\|_2^2} && \text{(B.4a)} \\
&\quad \leq \sqrt{\|\mathbf{s} - \mathbf{s}'\|_2^2 + L_\pi^2 \|\mathbf{s} - \mathbf{s}'\|_2^2 + L_{\hat{\pi}}^2 \|\mathbf{s} - \mathbf{s}'\|_2^2} && \text{(B.4b)} \\
&\quad = L_f \sqrt{1 + L_\pi^2 + L_{\hat{\pi}}^2} \|\mathbf{s} - \mathbf{s}'\|_2. && \text{(B.4c)}
\end{aligned}$$

Equation (B.4a) holds due to Lipschitz continuity of f and Equation (B.4b) is due to Lipschitz continuity of π and $\hat{\pi}$, which we assume in Assumptions 1, 4 and 8. □

Lemma 17 (Performance difference as difference in state trajectory)**.** *Under Assumptions 1, 4, 5 and 8, it holds:*

$$\left|J(f, \pi, \hat{\pi}) - J(\tilde{f}, \pi, \hat{\pi})\right| \leq L_r \sqrt{1 + L_\pi^2 + L_{\hat{\pi}}^2} \sum_{h=0}^{H} \mathbb{E}[\|\mathbf{s}_h - \tilde{\mathbf{s}}_h\|_2], \quad \text{(B.5)}$$

where $\tilde{\mathbf{s}}_h$ for $h = 0, \ldots, H$ is the trajectory generated by the dynamics $\tilde{f}$, starting from $\tilde{\mathbf{s}}_0 = \mathbf{s}_0$ with $\boldsymbol{\omega}_h = \tilde{\boldsymbol{\omega}}_h$.

Proof.

$$\left|J(f,\pi,\hat{\pi}) - J(\tilde{f},\pi,\hat{\pi})\right| = \left|\mathbb{E}\left[\sum_{h=0}^{H} r(\mathbf{s},\mathbf{a},\hat{\mathbf{a}}) - \sum_{h=0}^{H} r(\tilde{\mathbf{s}},\tilde{\mathbf{a}},\hat{\tilde{\mathbf{a}}})\right]\right| \tag{B.6a}$$

$$= \left|\sum_{h=0}^{H} \mathbb{E}\left[r(\mathbf{s},\mathbf{a},\hat{\mathbf{a}}) - r(\tilde{\mathbf{s}},\tilde{\mathbf{a}},\hat{\tilde{\mathbf{a}}})\right]\right| \tag{B.6b}$$

$$\leq L_r\sqrt{1+L_\pi^2+L_{\hat{\pi}}^2}\sum_{h=0}^{H}\mathbb{E}[\|\mathbf{s}_h - \tilde{\mathbf{s}}_h\|_2]. \tag{B.6c}$$

Equation (B.6a) follows by definition of J, Equation (B.6b) from linearity of expectation, and eq. (B.6c) from Lipschitzness of the closed loop dynamics which we prove in Lemma 16 and Lipschitzness of the reward function, which we assume in Assumption 5. □

The following lemma bounds the deviation between any trajectory generated by $\tilde{f} \in \mathcal{M}_n$ and the true trajectory in terms of the predictive variance.

Lemma 18 (Difference in state trajectory as sum of predictive variance along true trajectory)**.** *Under Assumptions 1 to 8, for all episodes $n \geq 1$, $h \in \{1,\dots,H\}$, $\pi \in \Pi$ and $\hat{\pi} \in \hat{\Pi}$ it holds:*

$$\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \leq 2\beta_{n-1}\kappa_{n-1}^{h-1}\sum_{h'=0}^{h-1}\|\sigma_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n})\|_2, \tag{B.7}$$

where $\kappa_{n-1} := \left(1 + (L_f + 2\beta_{n-1}L_\sigma)\sqrt{1+L_\pi^2+L_{\hat{\pi}}^2}\right)$, the closed-loop epistemic uncertainty as $\sigma_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n}) := \sigma_{n-1}(\mathbf{s}_{h,n}, \pi_n(\mathbf{s}_{h,n}), \hat{\pi}_n(\mathbf{s}_{h,n}))$, the simulated state $\tilde{\mathbf{s}}_{h,n}$ is generated by any system $\tilde{f} \in \mathcal{M}_n$ (cf., Section 2.2.1), and the true state $\mathbf{s}_{h,n}$ is generated by the true dynamics f, with $\boldsymbol{\omega}_{h,n} = \tilde{\boldsymbol{\omega}}_{h,n}$.

Proof. To avoid notational clutter, we denote the closed-loop dynamics as $f^{\pi,\hat{\pi}}(\mathbf{s}) = f(\mathbf{s}, \pi(\mathbf{s}), \hat{\pi}(\mathbf{s}))$. Likewise, we use the following Lipschitz constants shorthands $L_{f,\pi} \equiv L_f\sqrt{1+L_\pi^2+L_{\hat{\pi}}^2}$ and $L_{\sigma,\pi} \equiv L_\sigma\sqrt{1+L_\pi^2+L_{\hat{\pi}}^2}$.

We first prove by induction that

$$\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \leq 2\beta_{n-1} \sum_{h'=0}^{h-1} \left(L_{f,\pi} + 2\beta_{n-1} L_{\sigma,\pi}\right)^{h-1-h'} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\| \tag{B.8a}$$

For $h = 0$, clearly $\mathbf{s}_{0,n} = \tilde{\mathbf{s}}_{0,n}$, while the right-hand-side of inequality (B.8a) is always non-negative. We assume that for h the inductive hypothesis (B.8a) holds. For $h+1$ we have:

$$\begin{aligned}
&\|\mathbf{s}_{h+1,n} - \tilde{\mathbf{s}}_{h+1,n}\|_2 \\
&= \|f^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n}) - \tilde{f}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(B.8b)}\\
&= \|f^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n}) - \tilde{f}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n}) + f^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n}) - f^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(B.8c)}\\
&\leq \|f^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n}) - f^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})\|_2 + \|f^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n}) - \tilde{f}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(B.8d)}\\
&\leq L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 + \|f^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n}) - \tilde{f}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(B.8e)}\\
&\leq L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 + 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})\|_2 && \text{(B.8f)}\\
&= L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \\
&\quad + 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n}) + \boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n}) - \boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n})\|_2 && \text{(B.8g)}\\
&\leq L_{f,\pi}\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 \\
&\quad + 2\beta_{n-1}\left(\|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n}) - \boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n})\|_2 + \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n})\|_2\right) && \text{(B.8h)}\\
&\leq \left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)\|\mathbf{s}_{h,n} - \tilde{\mathbf{s}}_{h,n}\|_2 + 2\beta_{n-1}\|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n})\|_2 && \text{(B.8i)}\\
&\leq 2\beta_{n-1}\sum_{h'=0}^{(h+1)-1}\left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{(h+1)-1-h'}\|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2 && \text{(B.8j)}
\end{aligned}$$

Here, Equation (B.8b) holds by applying the transition dynamics $f^{\pi_n,\hat{\pi}_n}$ and $\tilde{f}^{\pi_n,\hat{\pi}_n}$ with the same noise realization $\boldsymbol{\omega}_h = \tilde{\boldsymbol{\omega}}_h$; Equation (B.8c) holds by adding and subtracting $f^{\pi_n,\hat{\pi}_n}(\tilde{\mathbf{s}}_{h,n})$; inequality (B.8d) follows from the triangular inequality; inequality (B.8e) comes from Lemma 16; inequality (B.8f) holds due to both f and $\tilde{f}$ belonging to the set of plausible models $\mathcal{M}_n$; Equation (B.8g) holds by adding and substracting $\boldsymbol{\sigma}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h,n})$; inequality eq. (B.8h) holds by applying the triangular inequality once more; inequality eq. (B.8i) is due to the Lipschitz continuity of $\boldsymbol{\sigma}$ as per Assumption 6; and inequality (B.8j) holds by replacing the inductive hypothesis (B.8a).

Finally, with simple algebraic manipulation we can see that,

$$\left(L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1-h'} < \left(1 + L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1-h'} \leq \left(1 + L_{f,\pi} + 2\beta_{n-1}L_{\sigma,\pi}\right)^{h-1},$$

and the main result follows by combining this with Equation (B.8j). □

Lemma 19 (Difference between optimistic and pessimistic performance as sum of predictive variance along true trajectory)**.** *Under Assumptions 1 to 8, let π_n and $\hat{\pi}_n$ be the policies selected by* RH-UCRL *at episode n. Then, the following holds for the difference between its optimistic and pessimistic performance:*

$$J_n^{(o)}(\pi_n, \hat{\pi}_n) - J^{(p)}(\pi_n, \hat{\pi}_n) \leq 4L_r\beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\sigma_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right], \quad \text{(B.9)}$$

where $\sigma^{\pi,\hat{\pi}}(\mathbf{s}) = \sigma(\mathbf{s}, \pi(\mathbf{s}), \hat{\pi}(\mathbf{s}))$ and $C := (1 + L_f + L_\sigma)(1 + L_\pi^2 + L_{\hat{\pi}}^2)^{1/2}$.

Proof.

$$\begin{aligned} &J_n^{(o)}(\pi_n, \hat{\pi}_n) - J^{(p)}(\pi_n, \hat{\pi}_n) \\ &\leq \left|J_n^{(o)}(\pi_n, \hat{\pi}_n) - J(f, \pi_n, \hat{\pi}_n)\right| + \left|J_n^{(p)}(\pi_n, \hat{\pi}_n) - J(f, \pi_n, \hat{\pi}_n)\right| && \text{(B.10a)} \\ &\leq L_r\sqrt{1 + L_\pi^2 + L_{\hat{\pi}}^2} \sum_{h=0}^{H} \left(\mathbb{E}\left[\|\mathbf{s}_{h,n} - \mathbf{s}_{h,n}^{(o)}\|_2\right] + \mathbb{E}\left[\|\mathbf{s}_{h,n} - \mathbf{s}_{h,n}^{(p)}\|_2\right]\right) && \text{(B.10b)} \end{aligned}$$

Here, inequality (B.10a) holds by the triangle inequality and inequality (B.10b) follows from Lemma 17.

We proceed to upper bound terms $\|\mathbf{s}_{h,n} - \mathbf{s}_{h,n}^{(o)}\|_2$ and $\|\mathbf{s}_{h,n} - \mathbf{s}_{h,n}^{(p)}\|_2$. From Lemma 18, it follows that both terms can be bounded in the same way as follows:

$$\|\mathbf{s}_{h,n} - \mathbf{s}_{h,n}^{(o)}\|_2 \leq 2\beta_{n-1}\kappa_{n-1}^{h-1} \sum_{h'=0}^{h-1} \|\sigma_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2$$

as $f^{(o)}$ and $f^{(p)}$ belong to the set of plausible models $\mathcal{M}_n$. By applying the previous bound twice in Equation (B.10b), and by denoting $C := (1 + L_f + 2L_\sigma)(1 + L_\pi^2 + L_{\hat{\pi}}^2)^{1/2}$, we arrive at:

$$J_n^{(o)}(\pi_n, \hat{\pi}_n) - J^{(p)}(\pi_n, \hat{\pi}_n) \leq 4L_r\beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\sigma_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right],$$

where we used $n \leq N$ and $1 \leq \beta_n$ is non-decreasing in n. □

Theorem 2. *Under Assumptions 1 to 8, let* $C = (1 + L_f + 2L_\sigma)(1 + L_\pi^2 + L_{\hat{\pi}}^2)^{1/2}$ *and let* $\mathbf{s}_{n,h} \in \mathcal{S}$, $\mathbf{a}_{n,h} \in \mathcal{A}$, $\hat{\mathbf{a}}_{n,h} \in \hat{\mathcal{A}}$ *for all* $n, h > 0$. *Then, for any fixed* $H \geq 1$, *with probability at least* $1 - \delta$, *the robust cumulative regret of* RH-UCRL *is upper bounded by:*

$$\bar{\mathfrak{R}}_N = \mathcal{O}\Big(L_r C^H \beta_N^H H^{3/2} \sqrt{N \Gamma_N}\Big).$$

Proof of Theorem 2. We bound the cumulative robust-regret as follows:

$$\bar{\mathfrak{R}}_N = \sum_{n=1}^{N} \underbrace{\min_{\hat{\pi} \in \hat{\Pi}} J(f, \pi^\star, \hat{\pi}) - \min_{\hat{\pi} \in \hat{\Pi}} J(f, \pi_n, \hat{\pi}_n)}_{:= \bar{\mathfrak{r}}_n} \tag{B.11a}$$

$$\leq \sqrt{N \sum_{n=1}^{N} r_n^2} \tag{B.11b}$$

$$\leq \sqrt{N \sum_{n=1}^{N} (4 L_r \beta_N^H C^H)^2 \left(\sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \| \boldsymbol{\sigma}_{n-1}^{\pi_n, \hat{\pi}_n}(\mathbf{s}_{h',n}) \|_2 \right] \right)^2} \tag{B.11c}$$

$$= 4 L_r \beta_N^H C^H \sqrt{N} \sqrt{\sum_{n=1}^{N} \left(\sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \| \boldsymbol{\sigma}_{n-1}^{\pi_n, \hat{\pi}_n}(\mathbf{s}_{h',n}) \|_2 \right] \right)^2} \tag{B.11d}$$

$$\leq 4 L_r \beta_N^H C^H H \sqrt{N} \sqrt{\sum_{n=1}^{N} \left(\mathbb{E}\left[\sum_{h'=0}^{H} \| \boldsymbol{\sigma}_{n-1}^{\pi_n, \hat{\pi}_n}(\mathbf{s}_{h',n}) \|_2 \right] \right)^2} \tag{B.11e}$$

$$\leq 4 L_r \beta_N^H C^H H \sqrt{N} \sqrt{\sum_{n=1}^{N} \mathbb{E}\left[\left(\sum_{h'=0}^{H} \| \boldsymbol{\sigma}_{n-1}^{\pi_n, \hat{\pi}_n}(\mathbf{s}_{h',n}) \|_2 \right)^2 \right]} \tag{B.11f}$$

$$\leq 4 L_r \beta_N^H C^H H^{3/2} \sqrt{N} \sqrt{\sum_{n=1}^{N} \mathbb{E}\left[\sum_{h'=0}^{H} \| \boldsymbol{\sigma}_{n-1}^{\pi_n, \hat{\pi}_n}(\mathbf{s}_{h',n}) \|_2^2 \right]} \tag{B.11g}$$

$$\leq 4 L_r \beta_N^H C^H H^{3/2} \sqrt{N \Gamma_N}, \tag{B.11h}$$

where eq. (B.11b) is due to the Cauchy-Schwarz's inequality; eq. (B.11c) is due to Lemmas 15 and 19. Finally, eq. (B.11f) follows from Jensen's inequality, eq. (B.11g) follows from Cauchy-Schwarz's inequality, and eq. (B.11h) follows from the definition of Γ_N in Equation (2.11). □

Corollary 1. *Consider the assumptions and setup of Theorem 2, and suppose that*

$$\frac{N}{\beta_N^{2H}\Gamma_N} \geq \frac{16L_r^2H^3C^{2H}}{\epsilon^2}, \tag{4.11}$$

for some fixed $\epsilon > 0$ and $H \geq 1$. Then, with probability at least $1-\delta$ after N episodes, RH-UCRL *achieves:*

$$\min_{\hat{\pi}\in\hat{\Pi}} J(f, \hat{\pi}_N, \hat{\pi}) \geq \min_{\hat{\pi}\in\hat{\Pi}} J(f, \pi^\star, \hat{\pi}) - \epsilon, \tag{4.12}$$

where $\hat{\pi}$ is the output of RH-UCRL*, reported according to eq.* (4.8)*, and $\pi^\star$ is the optimal robust policy given in eq.* (4.3)*.*

Proof of Corollary 1. We start the proof by recalling some of the previously obtained results. The simple robust-regret $\bar{\mathfrak{r}}_n(\pi_n)$ of a policy π_n selected at episode n by the RH-UCRL learner algorithm (4.7a) is given by:

$$\bar{\mathfrak{r}}(\pi_n) = \min_{\hat{\pi}\in\hat{\Pi}} J(f, \pi^\star, \hat{\pi}) - \min_{\hat{\pi}\in\hat{\Pi}} J(f, \pi_n, \hat{\pi}). \tag{B.12}$$

From Lemmas 15 and 19, it follows that

$$\bar{\mathfrak{r}}(\pi_n) \leq 4L_r\beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right]. \tag{B.13}$$

We also define the pessimistic regret of a policy π_n selected at episode n

$$\bar{\mathfrak{r}}^{(p)}(\pi_n) \coloneqq \min_{\hat{\pi}\in\hat{\Pi}} J(f, \pi^\star, \hat{\pi}) - \min_{\hat{\pi}\in\hat{\Pi}} J^{(p)}(\pi_n, \hat{\pi}), \tag{B.14}$$

and note that $\bar{\mathfrak{r}}(\pi_n) \leq \bar{\mathfrak{r}}^{(p)}(\pi_n)$ for every π_n, since $J^{(p)}(\pi_n, \hat{\pi}) \leq J(f, \pi_n, \hat{\pi})$ for any $\pi \in \Pi$ and $\hat{\pi} \in \hat{\Pi}$. Another useful observation is that the same bound obtained in Equation (B.13) also holds in case of $\bar{\mathfrak{r}}_n^{(p)}$, i.e.,

$$\bar{\mathfrak{r}}(\pi_n) \leq \bar{\mathfrak{r}}^{(p)}(\pi_n) \leq 4L_r\beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right]. \tag{B.15}$$

Recall that the reported policy $\hat{\pi}_N$ from eq. (4.8) is chosen among the previously selected episodic policies $\{\pi_1, \dots, \pi_N\}$, such that

$$\hat{\pi}_N = \operatorname*{argmin}_{n\in\{1,\dots,N\}} \bar{\mathfrak{r}}^{(p)}(\pi_n). \tag{B.16}$$

It follows that:

$$\bar{\mathfrak{r}}(\hat{\pi}_N) \leq \bar{\mathfrak{r}}^{(p)}(\hat{\pi}_N) \tag{B.17a}$$

$$\leq \frac{1}{N}\sum_{n=1}^{N} \bar{\mathfrak{r}}(\pi_n) \tag{B.17b}$$

$$\leq \frac{1}{N}\sum_{n=1}^{N} 4L_r\beta_N^H C^H \sum_{h=0}^{H} \mathbb{E}\left[\sum_{h'=0}^{h-1} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right] \tag{B.17c}$$

$$\leq \frac{1}{N}4L_r\beta_N^H C^H H \sum_{n=1}^{N} \mathbb{E}\left[\sum_{h'=0}^{H} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right] \tag{B.17d}$$

$$\leq \frac{1}{N}4L_r\beta_N^H C^H H\sqrt{N}\sqrt{\sum_{n=1}^{N} \mathbb{E}\left[\left(\sum_{h'=0}^{H} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2\right)^2\right]} \tag{B.17e}$$

$$\leq \frac{1}{N}4L_r\beta_N^H C^H H\sqrt{N}\sqrt{\sum_{n=1}^{N} \mathbb{E}\left[\sum_{h'=0}^{H} \|\boldsymbol{\sigma}_{n-1}^{\pi_n,\hat{\pi}_n}(\mathbf{s}_{h',n})\|_2^2\right]} \tag{B.17f}$$

$$\leq \frac{4L_r\beta_N^H C^H H^{3/2}\sqrt{N\Gamma_N}}{N} \tag{B.17g}$$

where inequality (B.17a) follows from inequality eq. (B.15); inequality (B.17b) follows from the policy reporting rule in Equation (B.16) and by upper bounding minimum with average. Finally, inequality (B.17c) is due to inequality (B.15), and Equations (B.17d) to (B.17g) follow the same argument as in the proof of theorem 2.

To achieve $r(\hat{\pi}_N) \leq \epsilon$ for some given $\epsilon > 0$, we require that

$$\frac{4L_r\beta_N^H C^H H^{3/2}\sqrt{N\Gamma_N}}{N} \leq \epsilon.$$

By simple inversion it follows that we require the following number of episodes N:

$$\frac{N}{\beta_N^{2H}\Gamma_N} \geq \frac{16L_r^2 H^3 C^{2H}}{\epsilon^2}$$

to achieve $\bar{\mathfrak{r}}(\hat{\pi}_N) \leq \epsilon$. □

C

PROOFS OF CHAPTER 5

We start by proving that the sub-level sets at value $\bar{\xi}$ of the cumulative cost function C are completely contained within the safe set $\mathcal{S}_{\text{safe}}$.

Lemma 9. *Consider an immediate cost function $c : \mathcal{S} \to \mathbb{R}$ satisfying the conditions* (5.8)*. Then, it holds that $\mathcal{C}^{\pi}_{\bar{\xi}} \subset \mathcal{S}_{\text{safe}}$, where $\bar{\xi} = \hat{c} + \gamma \min_{\mathbf{s}\in\mathcal{S}} C(f, \pi; \mathbf{s})$.*

Proof. Due to the lower bound for c, C is lower bounded by $C_{\min} = \min_{\mathbf{s}\in\mathcal{S}} C(f, \pi; \mathbf{s}) \geq \frac{c}{1-\gamma}$. Moreover, due to condition (5.8) we have $c(\mathbf{s}) > \hat{c}$ for $\mathcal{S}_{\text{unsafe}}$, which yields $C(f, \pi, \mathbf{s}) > \hat{c} + \gamma C_{\min}$ for all $\mathbf{s} \in \mathcal{S}_{\text{unsafe}}$. Using Bellman equation, we know that $C(f, \pi; \mathbf{s}) = c(\mathbf{s}) + \gamma C(f, \pi; \mathbf{s}')$, where $\mathbf{s}'$ is the next state following policy π. Thus, for any unsafe state $\mathbf{s}$ we know that

$$\begin{aligned} C(f, \pi; \mathbf{s}) &= c(\mathbf{s}) + \gamma C(f, \pi; \mathbf{s}') \\ &\geq \hat{c} + \gamma C_{\min} \end{aligned}$$

Therefore, the level set $\mathcal{C}^{\pi}_{\bar{\xi}}$, with $\bar{\xi} := \hat{c} + \gamma C_{\min}$ is completely contained in $\mathcal{S}_{\text{safe}}$, i.e., $\mathcal{C}^{\pi}_{\bar{\xi}} \subset \mathcal{S}_{\text{safe}}$, which concludes the proof. □

While we now that the sub-level set $\bar{\xi}$ of the cumulative cost function C is safe, we still need to prove that the stochastic dynamical system (5.1) will remain in such set with some probability. For this we need the two following technical lemmas.

Lemma 20. *If there exists a function $V : \mathcal{S} \to \mathbb{R}$ such that*

$$\mathbb{E}_{\boldsymbol{\omega}}[V(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] \leq \theta_1 \tag{C.1}$$

for $\theta_1 \in \mathbb{R}$, then, it holds that

$$\frac{\theta_2 - \theta_1}{\theta_2 - \underline{V}} \leq \mathbb{P}\left[\mathbf{s}_{h+1} \in \mathcal{V}_{\theta_2} | \mathbf{s}_h = \mathbf{s}\right] \leq \frac{\theta_2 - \theta_1}{\bar{V} - \theta_2} \tag{C.2}$$

for every $\underline{V} \leq \theta_1 < \theta_2 < \bar{V}$, where $\underline{V} = \min_{\mathbf{s}\in\mathcal{S}} V(\mathbf{s})$ and $\bar{V} = \max_{\mathbf{s}\in\mathcal{S}} V(\mathbf{s})$.

Proof. In order to prove this lemma, we follow the ideas of Y. Li *et al.*, 2013. It is straightforward to see that

$$(\theta_2 - \underline{V})\mathbb{P}\left[\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2} | \ \mathbf{s}_h = \mathbf{s}\right] \leq \mathbb{E}_{\boldsymbol{\omega}_h}\left[\mathbb{I}_{\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2}}(V(\mathbf{s}_{h+1}) - \underline{V}) | \ \mathbf{s}_h = \mathbf{s}\right], \tag{C.3}$$

where $\mathcal{V}_{\theta_2} = \{\mathbf{s} \in \mathcal{S} : V(\mathbf{s}) < \theta_2\}$, and $\mathbb{I}_{\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2}} = 1$ if $\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2}$ and 0 otherwise, since $\mathbb{I}_{\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2}} V(\mathbf{s}_{h+1}) \geq \mathbb{I}_{\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2}} \theta_2$. Moreover, we trivially have

$$\mathbb{E}_{\boldsymbol{\omega}_k}\left[\mathbb{I}_{\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2}}(V(\mathbf{s}_{h+1}) - \underline{V}) | \ \mathbf{s}_h = \mathbf{s}\right] \leq \mathbb{E}_{\boldsymbol{\omega}_k}[(V(\mathbf{s}_{h+1}) - \underline{V}) | \ \mathbf{s}_h = \mathbf{s}]. \tag{C.4}$$

By combining (C.1), (C.3) and (C.4), we therefore obtain

$$(\theta_2 - \underline{V})\mathbb{P}\left[\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2} | \ \mathbf{s}_h = \mathbf{s}\right] \leq -\underline{V} + \theta_1,$$

which results in

$$\mathbb{P}\left[\mathbf{s}_{h+1} \notin \mathcal{V}_{\theta_2} | \ \mathbf{s}_h = \mathbf{s}\right] \leq \frac{\theta_1 - \underline{V}}{\theta_2 - \underline{V}}. \tag{C.5}$$

The proof for the upper bound is analogous. □

Lemma 21. *Assume there exists a function $V : \mathcal{S} \to \mathbb{R}$ and a class $\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}_{0,+}$, such that*

$$\mathbb{E}_{\boldsymbol{\omega}}[V(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] - V(\mathbf{s}) \leq -\alpha(V(\mathbf{s})) \tag{C.6}$$

holds for all $\mathbf{s} \in \mathcal{V}_{\bar{\xi}}$ for $\bar{\xi} \in \mathbb{R}$. Then, $\mathbb{E}_{\boldsymbol{\omega}}[V(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] \leq \xi$ with $\xi < \bar{\xi}$ ensures

$$\mathbb{P}\left[V(\mathbf{s}_h) \leq \bar{\xi} \ \forall k = 1, \ldots, K | \mathbf{s}_0 = \mathbf{s}\right] \geq 1 - \delta_{\mathrm{FL}}(\xi) \tag{C.7}$$

with

$$\delta_{\mathrm{FL}}(\xi) = \begin{bmatrix} 1 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{1}^T(\boldsymbol{I} - [\boldsymbol{P}]_+) \\ \mathbf{0} & [\boldsymbol{P}]_+ \end{bmatrix}^K \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}, \tag{C.8}$$

where the elements of $\boldsymbol{P}$ *are defined as*

$$p_{i,j} = \begin{cases} \frac{\theta^i - \theta^j + \alpha(\theta^{j+1} + \underline{V})}{\theta^i - \underline{V}} - \frac{\theta^{i+1} - \theta^j + \alpha(\theta^{j+1} + \underline{V})}{\bar{V} - \theta^j + \alpha(\theta^{j+1} + \underline{V})} & \text{if } i \leq j \\ \frac{(1-\vartheta)\alpha(\theta^{j+1} + \underline{V})}{\theta^j - \alpha(\theta^{j+1} + \underline{V}) - \underline{V}} & \text{if } i = j+1. \\ 0 & \text{if } i > j+1 \end{cases} \tag{C.9}$$

and M *is the largest integer such that* $\theta^1 \leq \bar{\xi}$ *for* θ^i *recursively defined by* $\theta^{i-1} = \theta^i + \vartheta\alpha(\theta^i + \underline{V})$ *with* θ^{M+1} *implicitly defined via* $\theta^{M+1} + (\vartheta - 1)\alpha(\theta^{M+1} + \underline{V}) = \xi$ *and sufficiently small* $\vartheta \in (0, 1)$.

Proof. For proving this proposition, we construct a sequence of sub-level sets $\mathcal{V}_{\theta^j}$ as illustrated in Figure C.1 and bound the transition probabilities between them using Lemma 20. Given a sub-level set $\mathcal{V}_{\theta^j}$, the probability of transitioning into sub-level set $\mathcal{V}_{\theta^{j+1}}$ can be lower bounded using $\theta_2 = \theta^j - \vartheta\alpha(\theta^{j+1} - \underline{V})$, $\theta_1 = \theta^j - \alpha(\theta^{j+1} - \underline{V})$, which yields

$$p_{j+1,j} = \frac{(1-\vartheta)\alpha(\theta^{j+1} + \underline{V})}{\theta^j - \alpha(\theta^{j+1} + \underline{V}) - C_{\min}}.$$

For transitioning from the sub-level set $\mathcal{V}_{\theta^j}$ to a sub-level set $\mathcal{V}_{\theta^i}$, $i \leq j$, we have

$$\begin{aligned} &\mathbb{P}\left[\mathbf{s}_{h+1} \in \mathcal{V}_{\theta^i} \setminus \mathcal{V}_{\theta^{i+1}} | \mathbf{s}_h \in \mathcal{V}_{\theta^j}\right] = \\ &\quad \mathbb{P}\left[\mathbf{s}_{h+1} \in \mathcal{V}_{\theta^i} | \mathbf{s}_h \in \mathcal{V}_{\theta^j}\right] - \mathbb{P}\left[\mathbf{s}_{h+1} \in \mathcal{V}_{\theta^{i+1}} | \mathbf{s}_h \in \mathcal{V}_{\theta^j}\right], \end{aligned}$$

such that applying Lemma 20 to both summands with $\theta_2 = \theta^i$, $\theta_1 = \theta^j - \alpha(\theta^{j+1} - \underline{V})$ and $\theta_2 = \theta^{i+1}$, $\theta_1 = \theta^j - \alpha(\theta^{j+1} - \underline{V})$, respectively, yields

$$p_{i,j} = \frac{\theta^i - \theta^j + \alpha(\theta^{j+1} + \underline{V})}{\theta^i - \underline{V}} - \frac{\theta^{i+1} - \theta^j + \alpha(\theta^{j+1} + \underline{V})}{\bar{V} - \theta^j + \alpha(\theta^{j+1} + \underline{V})}.$$

Note that for $i = M$ we have $\theta_1 = \xi$. Since we cannot guarantee to directly transition from sub-level sets $\mathcal{V}_{\theta^j}$ to sub-level sets $\mathcal{V}_{\theta^i}$ with $i \geq j + 2$, we obtain the trivial bound $p_{i,j} = 0$ in this case, which results in (C.9). Based on the bounds $p_{i,j}$, we can construct a left stochastic matrix similar to the transition matrix of a Markov chain, whose first row corresponds to an absorbing state as shown in (C.8). Since the first state is absorbing and the transition probabilities to all other states are lower bounds, multiplying this matrix K times with itself and multiplying the initial probability distribu-

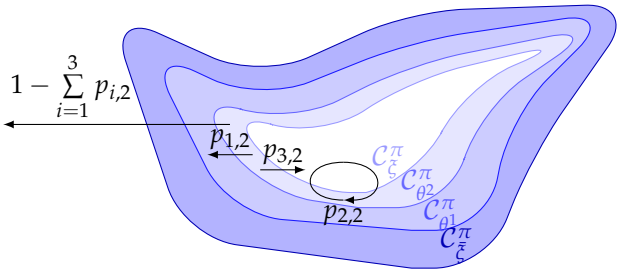


FIGURE C.1: In order to certify the K-step δ-safety of a policy π, we define a sequence of sub-level sets $\mathcal{C}^{\pi}_{\theta^i}$ with decreasing thresholds θ^i. We can bound the probabilities for transitioning to other sub-level sets in each time step using Lemma 20 as example illustrated for $\mathcal{C}^{\pi}_{\theta^2}$, such that the probability of leaving the $\mathcal{C}^{\pi}_{\bar{\xi}}$ can be bounded using methods for Markov chains.

tion from the right yields the upper bound δ for leaving the sub-level set $\mathcal{V}_{\bar{\xi}}$ within K time steps. □

We use the previous lemma to prove Proposition 1

Proposition 1. *Consider an immediate cost function $c : \mathcal{S} \to \mathbb{R}$, which satisfies the conditions* (5.8)*. Define $C_\pi(\mathbf{s}) \equiv C(f, \pi; \mathbf{s})$. Assume there exists a class $\mathcal{K}$ function* [1] *$\alpha : \mathbb{R} \to \mathbb{R}_{0,+}$, such that*

$$\mathbb{E}_{\boldsymbol{\omega}}[C_\pi(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] \leq C_\pi(\mathbf{s}) - \alpha(C_\pi(\mathbf{s}) - C_{\min})$$

holds for all $\mathbf{s} \in \mathcal{S}_{\text{safe}}$. Then,

$$\mathbb{E}_{\boldsymbol{\omega}}[C_\pi(f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega})] \leq \xi < \bar{\xi} \tag{5.9}$$

guarantees that the policy π is K-step $\delta(\xi)$-safe.

Proof of Proposition 1. The result directly follows from Lemma 9 and Lemma 21, which ensure δ-safety with $\delta = \delta_{\text{FL}}(\xi)$. □

Proposition 2. *Consider a set of plausible models $\mathcal{M}$ satisfying Assumption 7 and an immediate cost c, which satisfies* (5.8)*. If*

$$\max_{\eta \in \mathcal{U}} \mathbb{E}_{\boldsymbol{\omega}}\left[C^{(p)}_\pi(\mathbf{s}')\right] \leq \xi, \tag{5.13}$$

1 A function $\alpha : \mathbb{R}_{0,+} \to \mathbb{R}_{0,+}$ is a class $\mathcal{K}$ function, if it is monotonically increasing and $\alpha(0) = 0$.

with $\mathbf{s}'$ is the next-state defined through the reparameterized dynamics (2.33) *and $\xi \leq \bar{\xi}$, and $C_\pi^{(p)}(\mathbf{s})$ satisfies* (5.9)*, then, the π is K-step δ-safe.*

Proof of Proposition 2. Due to Assumption 7, $\max_{\eta \in \mathcal{U}} C(\tilde{f}, \pi; \mathbf{s}) \leq \bar{\xi}$ implies $C(f, \pi; \mathbf{s}_0) \leq \bar{\xi}$ with probability at least $1 - \delta_f$. Thus, $\max_{\eta \in \mathcal{U}} C(\tilde{f}, \pi, ; \mathbf{s}) \leq \bar{\xi}$ implies that $\mathbf{s} \in \mathcal{S}_{\text{safe}}$ due to Lemma 9. Moreover,

$$\max_{\eta \in \mathcal{U}} \mathbb{E}_{\boldsymbol{\omega}} \left[C(\tilde{f}, \pi; f(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega}) \right] \leq \max_{\eta \in \mathcal{U}} \mathbb{E}_{\boldsymbol{\omega}} \left[C(\tilde{f}, \pi; \tilde{f}(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega}) \right]$$

with probability at least $(1 - \delta_f)$ due to Assumption 7.

Since $\max_{\eta \in \mathcal{U}} C(\tilde{f}, \pi; \mathbf{s})$ satisfying (5.9) ensures

$$\max_{\eta \in \mathcal{U}} \mathbb{E}_{\boldsymbol{\omega}} \left[C(\tilde{f}, \pi; \tilde{f}, \pi; \tilde{f}(\mathbf{s}, \pi(\mathbf{s})) + \boldsymbol{\omega}) \right] \leq C(\tilde{f}, \pi; \mathbf{s}) - \alpha(C(\tilde{f}, \pi; \mathbf{s}) - C_{\min}),$$

we can apply Lemma 21, such that safety follows with $\delta = \delta_{\text{FL}}(\xi) + \delta_f - \delta_{\text{FL}}(\xi)\delta_f$. □

Proposition 3. *If there exists a policy π and a class $\mathcal{K}$ function α such that $C_\pi^{(p)}$ satisfies condition* (5.9)*, then, the learned safe policy π_{safe} is K-step δ-safe for all $\mathbf{s} \in \mathcal{S}_{\text{safe}}$ if $C_{\pi_{\text{safe}}}^{(p)} \leq \xi < \bar{\xi}$.*

Proof of Proposition 3. Due to the definition of the learned safe policy π_{safe} in the maximally safe problem (5.18), it directly follows that

$$\hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi_{\text{safe}}, \eta) \leq \max_{\eta \in \mathcal{U}} \hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi_{\text{safe}}, \eta). \tag{C.10}$$

Therefore, $\hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi_{\text{safe}}, \eta_{\text{adv}})$ satisfies the requirements of Proposition 2 □

Theorem 3. *Consider a set of plausible models $\mathcal{M}$ satisfying Assumption 7 and assume that the learned safe policy π_{safe} satisfies the conditions of Proposition 3. Then, the confidence-based safety filtered policy* (5.21) *is K-step δ-safe for all states $\mathbf{s} \in \mathcal{S}_{\text{safe}}$.*

Proof. Since π_{safe} satisfies the conditions of Proposition 2, the trivial solution $\mathbf{a} = \pi_{\text{safe}}(\mathbf{s})$ is guaranteed to ensure condition (5.20b). Therefore, the safety filter (5.20) is feasible for all states $\mathbf{s} \in \mathcal{S}$ with $\hat{C}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \pi_{\text{safe}}, \eta_{\text{adv}}; \mathbf{s}) \leq \xi$, such that the K-step δ-safety of $\tilde{\pi}$ follows directly from Proposition 2. □

BIBLIOGRAPHY

1. Abbasi-Yadkori, Y. & Szepesvári, C. *Regret bounds for the adaptive control of linear quadratic systems* in *Proceedings of the 24th Annual Conference on Learning Theory* (2011), 1.
2. Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N. & Riedmiller, M. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920* (2018).
3. Abeille, M. & Lazaric, A. Efficient Optimistic Exploration in Linear-Quadratic Regulators via Lagrangian Relaxation. *arXiv preprint arXiv:2007.06482* (2020).
4. Achiam, J., Held, D., Tamar, A. & Abbeel, P. *Constrained policy optimization* in *International Conference on Machine Learning* (2017), 30.
5. Altman, E. *Constrained Markov Decision Processes* (CRC Press, 1999).
6. Alvarez, M. A., Rosasco, L., Lawrence, N. D., *et al.* Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning* **4**, 195 (2012).
7. Antos, A., Szepesvári, C. & Munos, R. *Fitted Q-iteration in continuous action-space MDPs* in *Conference on Neural Information Processing Systems (NeurIPS)* (2008), 9.
8. Ayoub, A., Jia, Z., Szepesvari, C., Wang, M. & Yang, L. F. Model-Based Reinforcement Learning with Value-Targeted Regression. *arXiv preprint arXiv:2006.01107* (2020).
9. Azar, M. G., Osband, I. & Munos, R. *Minimax Regret Bounds for Reinforcement Learning* in *International Conference on Machine Learning* (2017), 263.
10. Bai, Y. & Jin, C. *Provable self-play algorithms for competitive reinforcement learning* in *International Conference on Machine Learning (ICML)* (2020), 551.
11. Başar, T. & Bernhard, P. *H-infinity optimal control and related minimax design problems: a dynamic game approach* (Springer Science & Business Media, 2008).
12. Bastani, O. *Safe Reinforcement Learning with Nonlinear Dynamics via Model Predictive Shielding* in *American Control Conference* (2021), 3488.

13. Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S. & Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature* **588**, 77 (2020).

14. Bemporad, A., Borrelli, F. & Morari, M. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Transactions on automatic control* **48**, 1600 (2003).

15. Berkenkamp, F. *Safe Exploration in Reinforcement Learning: Theory and Applications in Robotics* PhD thesis (ETH Zurich, 2019).

16. Berkenkamp, F., Turchetta, M., Schoellig, A. P. & Krause, A. *Safe Model-based Reinforcement Learning with Stability Guarantees* in *Advances in Neural Information Processing Systems* (2017), 908.

17. Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P. & Bertsekas, D. P. *Dynamic programming and optimal control* (Athena scientific Belmont, MA, 1995).

18. Bogunovic, I., Scarlett, J., Jegelka, S. & Cevher, V. *Adversarially robust optimization with Gaussian processes* in *Conference on Neural Information Processing Systems (NeurIPS)* (2018), 5760.

19. Botev, Z. I., Kroese, D. P., Rubinstein, R. Y. & L'Ecuyer, P. in *Handbook of statistics* 35 (Elsevier, 2013).

20. Botev, Z. I., Kroese, D. P., Rubinstein, R. Y. & L'Ecuyer, P. in *Handbook of Statistics* 35 (Elsevier, 2013).

21. Brafman, R. I. & Tennenholtz, M. R-max - a General Polynomial Time Algorithm for Near-optimal Reinforcement Learning. *J. Mach. Learn. Res.* **3**, 213 (2003).

22. Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. & Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* **4**, 1 (2012).

23. Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J. & Schoellig, A. P. Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning (2021).

24. Buckman, J., Hafner, D., Tucker, G., Brevdo, E. & Lee, H. *Sample-efficient reinforcement learning with stochastic ensemble value expansion* in *Advances in Neural Information Processing Systems* (2018), 8224.

25. Camacho, E. F. & Alba, C. B. *Model predictive control* (Springer science & business media, 2013).

26. Chowdhury, S. R. & Gopalan, A. *Online learning in kernelized Markov decision processes* in *Conference on Artificial Intelligence and Statistics (AISTATS)* (2019), 3197.
27. Chua, K., Calandra, R., McAllister, R. & Levine, S. *Deep reinforcement learning in a handful of trials using probabilistic dynamics models* in *Conference on Neural Information Processing Systems (NeurIPS)* (2018), 4754.
28. Clavera, I., Fu, V. & Abbeel, P. Model-Augmented Actor-Critic: Backpropagating through Paths. *arXiv preprint arXiv:2005.08068* (2020).
29. Cover, T. M. & Thomas, J. A. Entropy, relative entropy and mutual information. *Elements of information theory* **2**, 12 (1991).
30. Curi, S., Berkenkamp, F. & Krause, A. *Efficient model-based reinforcement learning through optimistic policy search and planning* in. **33** (2020), 14156.
31. Curi, S., Bogunovic, I. & Krause, A. *Combining Pessimism with Optimism for Robust and Efficient Model-Based Deep Reinforcement Learning* in *International Conference on Machine Learning* (2021), 2254.
32. Curi, S., Melchior, S., Berkenkamp, F. & Krause, A. *Structured Variational Inference in Partially Observable Unstable Gaussian Process State Space Models* in *Learning for Dynamics and Control* (2020), 147.
33. Dani, V., Hayes, T. P. & Kakade, S. M. Stochastic linear optimization under bandit feedback (2008).
34. Dean, S., Tu, S., Matni, N. & Recht, B. *Safely Learning to Control the Constrained Linear Quadratic Regulator* in *American Control Conference* (2019), 5582.
35. Dearden, R., Friedman, N. & Andre, D. *Model based Bayesian exploration* in *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence (UAI), 1999* (1999), 150.
36. Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., *et al.* Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **602**, 414 (2022).
37. Degris, T., White, M. & Sutton, R. S. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839* (2012).
38. Deisenroth, M. & Rasmussen, C. E. *PILCO: A model-based and data-efficient approach to policy search* in *International Conference on machine learning (ICML)* (2011), 465.

39. Deisenroth, M. P., Neumann, G. & Peters, J. *A survey on policy search for robotics* (now publishers, 2013).

40. Der Kiureghian, A. & Ditlevsen, O. Aleatory or epistemic? Does it matter? *Structural Safety* **31**, 105 (2009).

41. Desautels, T., Krause, A. & Burdick, J. W. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research (JMLR)* **15**, 3873 (2014).

42. Ding, D., Wei, X., Yang, Z., Wang, Z. & Jovanović, M. *Provably Efficient Safe Exploration via Primal-Dual Policy Optimization* in *International Conference on Artificial Intelligence and Statistics* (2021), 3304.

43. Domingues, O. D., Ménard, P., Pirotta, M., Kaufmann, E. & Valko, M. Regret bounds for kernel-based reinforcement learning. *arXiv preprint arXiv:2004.05599* (2020).

44. Dulac-Arnold, G., Mankowitz, D. & Hester, T. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901* (2019).

45. Durand, A., Maillard, O.-A. & Pineau, J. Streaming kernel regression with provably adaptive mean, variance, and regularization. *The Journal of Machine Learning Research (JMLR)* **19**, 650 (2018).

46. Efroni, Y., Merlis, N., Ghavamzadeh, M. & Mannor, S. *Tight regret bounds for model-based reinforcement learning with greedy policies* in *Advances in Neural Information Processing Systems* (2019), 12203.

47. Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E. & Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101* (2018).

48. Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J. & Tomlin, C. J. A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems. *IEEE Transactions on Automatic Control* **64**, 2737 (2019).

49. Fujimoto, S., Van Hoof, H. & Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477* (2018).

50. Gaitsgory, V., Grüne, L., Höger, M., Kellett, C. M. & Weller, S. R. Stabilization of strictly dissipative discrete time systems with discounted optimal control. *Automatica* **93**, 311 (2018).

51. Gal, Y. *Uncertainty in deep learning* PhD Thesis (PhD thesis, University of Cambridge, 2016).

52. García, J. & Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* **16**, 1437 (2015).

53. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor* in *International Conference on Machine Learning (ICML)* (2018), 1861.

54. Hafner, R. & Riedmiller, M. Reinforcement learning in feedback control. *Machine learning* **84**, 137 (2011).

55. Hargraves, C. R. & Paris, S. W. Direct trajectory optimization using nonlinear programming and collocation. *Journal of guidance, control, and dynamics* **10**, 338 (1987).

56. Hewing, L., Arcari, E., Fröhlich, L. P. & Zeilinger, M. N. On Simulation and Trajectory Prediction with Gaussian Process Dynamics. *arXiv preprint arXiv:1912.10900* (2019).

57. Hong, Z.-W., Pajarinen, J. & Peters, J. Model-based Lookahead Reinforcement Learning. *arXiv preprint arXiv:1908.06012* (2019).

58. Iyengar, G. N. Robust dynamic programming. *Mathematics of Operations Research* **30**, 257 (2005).

59. Jacobson, D. H. New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach. *Journal of Optimization Theory and Applications* **2**, 411 (1968).

60. Jaksch, T., Ortner, R. & Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research* **11**, 1563 (2010).

61. Jiang, N., Kulesza, A., Singh, S. & Lewis, R. *The dependence of effective planning horizon on model accuracy* in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (2015), 1181.

62. Jin, C., Yang, Z., Wang, Z. & Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. *arXiv preprint arXiv:1907.05388* (2019).

63. Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., *et al.* Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374* (2019).

64. Kakade, S., Krishnamurthy, A., Lowrey, K., Ohnishi, M. & Sun, W. Information Theoretic Regret Bounds for Online Nonlinear Control. *arXiv preprint arXiv:2006.12466* (2020).

65. Kakade, S. & Langford, J. *Approximately optimal approximate reinforcement learning* in *In Proc. 19th International Conference on Machine Learning* (2002).

66. Kalweit, G. & Boedecker, J. *Uncertainty-driven imagination for continuous deep reinforcement learning* in *Conference on Robot Learning* (2017), 195.

67. Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C. & Cevher, V. Robust Reinforcement Learning via Adversarial training with Langevin Dynamics. *Conference on Neural Information Processing Systems (NeurIPS)* **33** (2020).

68. Kamthe, S. & Deisenroth, M. *Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control* in *International Conference on Artificial Intelligence and Statistics* (2018), 1701.

69. Kamthe, S. & Deisenroth, M. *Data-efficient reinforcement learning with probabilistic model predictive control* in *Conference on Artificial Intelligence and Statistics (AISTATS)* (2018), 1701.

70. Kim, K. & Braatz, R. Generalised polynomial chaos expansion approaches to approximate stochastic model predictive control. *International Journal of Control* **86**, 1324 (2013).

71. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

72. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]* (2013).

73. Kirschner, J., Bogunovic, I., Jegelka, S. & Krause, A. *Distributionally Robust Bayesian Optimization* in *Conference on Artificial Intelligence and Statistics (AISTATS)* (2020), 2174.

74. Kirschner, J. & Krause, A. *Information directed sampling and bandits with heteroscedastic noise* in *Proceedings of the 31st Conference On Learning Theory* **75** (PMLR, 2018), 358.

75. Kocsis, L. & Szepesvári, C. *Bandit based monte-carlo planning* in *European conference on machine learning* (2006), 282.

76. Koller, T., Berkenkamp, F., Turchetta, M. & Krause, A. *Learning-based Model Predictive Control for Safe Exploration* in *IEEE Conference on Decision and Control* (2018), 6059.

77. Krause, A. & Ong, C. S. *Contextual Gaussian Process Bandit Optimization.* in *Conference on Neural Information Processing Systems (NeurIPS)* (2011), 2447.

78. Lagoudakis, M. G. & Parr, R. *Value function approximation in zero-sum Markov games* in *Conference on Uncertainty in artificial intelligence (UAI)* (2002), 283.

79. Lakshminarayanan, B., Pritzel, A. & Blundell, C. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles* in *Advances in Neural Information Processing Systems* (2017), 6405.

80. Lakshminarayanan, B., Pritzel, A. & Blundell, C. in *Advances in Neural Information Processing Systems 30* (eds Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S. & Garnett, R.) 6402 (Curran Associates, Inc., 2017).

81. Lattimore, T. & Szepesvári, C. Bandit algorithms. *preprint* (2018).

82. Lederer, A., Umlauft, J. & Hirche, S. Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control. *arXiv:1906.01376 [cs, stat]* (2019).

83. Li, W. & Todorov, E. *Iterative linear quadratic regulator design for nonlinear biological movement systems.* in *ICINCO (1)* (2004), 222.

84. Li, Y., Zhang, W. & Liu, X. Stability of Nonlinear Stochastic Discrete-Time Systems. *Journal of Applied Mathematics* **2013** (2013).

85. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

86. Littman, M. L. in *Machine learning proceedings* 157 (Elsevier, 1994).

87. Littman, M. L. & Szepesvári, C. *A generalized reinforcement-learning model: Convergence and applications* in *International Conference on Machine Learning (ICML)* (1996), 310.

88. Liu, Z., Zhou, H., Chen, B., Zhong, S., Hebert, M. & Zhao, D. Constrained Model-based Reinforcement Learning with Robust Cross-Entropy Method. *arXiv preprint arXiv:2010.07968* (2020).

89. Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E. & Mordatch, I. *Plan online, learn offline: Efficient learning and exploration via model-based control* in *International Conference on Learning Representations (ICLR)* (2019).

90. Lu, X. & Van Roy, B. *Ensemble sampling* in *Advances in neural information processing systems* (2017), 3258.

91. Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T. & Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858* (2018).

92. Malik, A., Kuleshov, V., Song, J., Nemer, D., Seymour, H. & Ermon, S. *Calibrated model-based deep reinforcement learning* in *International Conference on Machine Learning (ICML)* (2019), 4314.

93. Mania, H., Tu, S. & Recht, B. *Certainty Equivalence is Efficient for Linear Quadratic Control* in *Neural Information Processing Systems* (2019), 10154.

94. McHutchon, A. *Modelling nonlinear dynamical systems with Gaussian Processes* PhD thesis (PhD thesis, University of Cambridge, 2014).

95. Mesbah, A. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems* **36**, 30 (2016).

96. Meyn, S. & Tweedie, R. *Markov Chains and Stochastic Stability* (1993).

97. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).

98. Mohamed, S., Rosca, M., Figurnov, M. & Mnih, A. Monte Carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652* (2019).

99. Moldovan, T. M., Levine, S., Jordan, M. I. & Abbeel, P. *Optimism-driven exploration for nonlinear systems* in *Robotics and Automation (ICRA), 2015 IEEE International Conference on* (IEEE, 2015), 3239.

100. Morari, M. & H. Lee, J. Model predictive control: past, present and future. *Computers & Chemical Engineering* **23**, 667 (1999).

101. Munos, R., Stepleton, T., Harutyunyan, A. & Bellemare, M. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems* **29** (2016).

102. Munos, R. & Szepesvári, C. Finite-Time Bounds for Fitted Value Iteration. *Journal of Machine Learning Research* **9** (2008).

103. Mutny, M. & Krause, A. *Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features* in *Advances in Neural Information Processing Systems* (2018), 9005.

104. Neu, G. & Pike-Burke, C. A unifying view of optimism in episodic reinforcement learning. *arXiv preprint arXiv:2007.01891* (2020).

105. Nilim, A. & El Ghaoui, L. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* **53**, 780 (2005).

106. Osband, I., Blundell, C., Pritzel, A. & Roy, B. V. Deep exploration via bootstrapped DQN. *Conference on Neural Information Processing Systems (NeurIPS)*, 4033 (2016).

107. Osband, I., Russo, D. & Van Roy, B. in *Advances in Neural Information Processing Systems 26* (eds Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z. & Weinberger, K. Q.) 3003 (Curran Associates, Inc., 2013).

108. Osband, I., Van Roy, B. & Wen, Z. Generalization and Exploration via Randomized Value Functions. *arXiv:1402.0635 [cs, stat]* (2014).

109. Parmas, P., Rasmussen, C. E., Peters, J. & Doya, K. *PIPPS: Flexible Model-Based Policy Search Robust to the Curse of Chaos* in *International Conference on Machine Learning* (2018), 4065.

110. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. *Automatic differentiation in pytorch* 2017.

111. Paternain, S., Chamon, L. F., Calvo-Fullana, M. & Ribeiro, A. *Constrained reinforcement learning has zero duality gap* in *Advances in Neural Information Processing Systems* **32** (2019), 7555.

112. Peng, X. B., Andrychowicz, M., Zaremba, W. & Abbeel, P. *Sim-to-real transfer of robotic control with dynamics randomization* in *IEEE international conference on robotics and automation (ICRA)* (2018), 1.

113. Perolat, J., Scherrer, B., Piot, B. & Pietquin, O. *Approximate dynamic programming for two-player zero-sum Markov games* in *International Conference on Machine Learning (ICML)* (2015), 1321.

114. Phan, M., Yadkori, Y. A. & Domke, J. *Thompson sampling and approximate inference* in *Advances in Neural Information Processing Systems* (2019), 8804.

115. Pinto, L., Davidson, J., Sukthankar, R. & Gupta, A. *Robust Adversarial Reinforcement Learning* in *International Conference on Machine Learning (ICML)* (2017), 2817.

116. Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming* (John Wiley & Sons, 2014).

117. Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., *et al. Imagination-augmented agents for deep reinforcement learning* in *Advances in neural information processing systems* (2017), 5690.

118. Rahimi, A. & Recht, B. *Random features for large-scale kernel machines* in *Advances in neural information processing systems* (2008), 1177.

119. Rajeswaran, A., Ghotra, S., Ravindran, B. & Levine, S. *EPOpt: Learning Robust Neural Network Policies Using Model Ensembles* in *International Conference on Learning Representations (ICLR)* (2017).

120. Richards, A. & How, J. P. Robust variable horizon model predictive control for vehicle maneuvering. *International Journal of Robust and Nonlinear Control* **16**, 333 (2006).

121. Rosolia, U. & Borrelli, F. Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Transactions on Automatic Control* **63**, 1883 (2018).

122. Rummery, G. A. & Niranjan, M. *On-line Q-learning using connectionist systems* (Citeseer, 1994).

123. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., *et al.* Mastering atari, go, chess and shogi by planning with a learned model. *Nature* **588**, 604 (2020).

124. Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. *Trust region policy optimization* in *International conference on machine learning* (2015), 1889.

125. Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. *Trust region policy optimization* in *International conference on machine learning* (2015), 1889.

126. Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

127. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

128. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. & Riedmiller, M. *Deterministic Policy Gradient Algorithms* in *International Conference on Machine Learning (ICML)* (2014), 387.

129. Srinivas, N., Krause, A., Kakade, S. & Seeger, M. *Gaussian process optimization in the bandit setting: no regret and experimental design* in *International Conference on Machine Learning (ICML)* (2010), 1015.

130. Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. Gaussian process optimization in the bandit setting: no regret and experimental design. *IEEE Transactions on Information Theory* **58**, 3250 (2012).

131. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).

132. Sutton, R. S. in *Machine Learning Proceedings 1990* (eds Porter, B. & Mooney, R.) 216 (Morgan Kaufmann, San Francisco (CA), 1990).

133. Sutton, R. S. & Barto, A. G. *Reinforcement learning: an introduction* (MIT press, 1998).

134. Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y., *et al. Policy gradient methods for reinforcement learning with function approximation.* in *NIPS* **99** (1999), 1057.

135. Szepesvári, C. Algorithms for Reinforcement Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* **4**, 1 (2010).

136. Tamar, A., Mannor, S. & Xu, H. *Scaling Up Robust MDPs by Reinforcement Learning* in *Proc. of the International Conference on Machine Learning (ICML)* (2014).

137. Tassa, Y., Erez, T. & Todorov, E. *Synthesis and stabilization of complex behaviors through online trajectory optimization* in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), 4906.

138. Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., *et al.* Deepmind control suite. *arXiv preprint arXiv:1801.00690* (2018).

139. Taylor, A., Singletary, A., Yue, Y. & Ames, A. *Learning for Safety-Critical Control with Control Barrier Functions* in *Learning for Dynamics & Control* (2019), 708.

140. Tesauro, G. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation* **6**, 215 (1994).

141. Tessler, C., Efroni, Y. & Mannor, S. *Action Robust Reinforcement Learning and Applications in Continuous Control* in *International Conference on Machine Learning (ICML)* (2019), 6215.

142. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. & Abbeel, P. *Domain randomization for transferring deep neural networks from simulation to the real world* in *International Conference on Intelligent Robots and Systems (IROS)* (2017), 23.

143. Todorov, E., Erez, T. & Tassa, Y. *Mujoco: A physics engine for model-based control* in *Conference on Intelligent Robots and Systems* (2012), 5026.

144. Todorov, E. & Li, W. *A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems* in *Proceedings of the 2005, American Control Conference, 2005.* (2005), 300.

145. Valko, M., Korda, N., Munos, R., Flaounas, I. & Cristianini, N. *Finite-Time Analysis of Kernelised Contextual Bandits* in *Uncertainty in Artificial Intelligence (UAI)* (2013), 654.

146. Van Hasselt, H., Guez, A. & Silver, D. *Deep reinforcement learning with double q-learning* in *Proceedings of the AAAI conference on artificial intelligence* **30** (2016).

147. Van Hasselt, H. P., Hessel, M. & Aslanides, J. *When to use parametric models in reinforcement learning?* in *Advances in Neural Information Processing Systems* (2019), 14322.

148. Venkatraman, A., Capobianco, R., Pinto, L., Hebert, M., Nardi, D. & Bagnell, J. A. *Improved learning of dynamics models for control* in *International Symposium on Experimental Robotics* (2016), 703.

149. Vinitsky, E., Du, Y., Parvate, K., Jang, K., Abbeel, P. & Bayen, A. Robust Reinforcement Learning using Adversarial Populations. *arXiv preprint arXiv:2008.01825* (2020).

150. Wabersich, K., Hewing, L., Carron, A. & Zeilinger, M. Probabilistic Model Predictive Safety Certification for Learning-Based Control. *IEEE Transactions on Automatic Control* **76**, 176 (2021).

151. Wang, T. & Ba, J. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649* (2019).

152. Wang, Z., Gehring, C., Kohli, P. & Jegelka, S. *Batched Large-scale Bayesian Optimization in High-dimensional Spaces* in *International Conference on Artificial Intelligence and Statistics* (2018), 745.

153. Watkins, C. J. & Dayan, P. Q-learning. *Machine learning* **8**, 279 (1992).

154. Wiesemann, W., Kuhn, D. & Rustem, B. Robust Markov decision processes. *Mathematics of Operations Research* **38**, 153 (2013).

155. Williams, G., Drews, P., Goldfain, B., Rehg, J. M. & Theodorou, E. A. *Aggressive driving with model predictive path integral control* in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), 1433.

156. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8**, 229 (1992).

157. Zanette, A. & Brunskill, E. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. *arXiv preprint arXiv:1901.00210* (2019).

158. Zeilinger, M. N., Jones, C. N. & Morari, M. Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization. *IEEE Transactions on Automatic Control* **56**, 1524 (2011).

159. Zhang, K., Kakade, S., Basar, T. & Yang, L. Model-Based Multi-Agent RL in Zero-Sum Markov Games with Near-Optimal Sample Complexity. *Conference on Neural Information Processing Systems (NeurIPS)* **33** (2020).

CURRICULUM VITAE

PERSONAL DATA

Name	Sebastian Martin Curi
Date of Birth	July 2, 1990
Place of Birth	Buenos Aires, Argentina
Citizen of	Germany and Argentina

EDUCATION

2017 – 2022	Eidgenössische Technische Hochschule, Zürich, Switzerland *Final degree:* Doctor in Science
2015 – 2017	Eidgenössische Technische Hochschule, Zürich, Switzerland *Final degree:* Master in Science
2009 – 2014	Instituto Tecnológico de Buenos Aires, Buenos Aires, Argentina *Final degree:* Ingeniero Mecánico
1995– 2008	Escuela Escocesa San Andrés (school) Buenos Aires, Argentina

EMPLOYMENT

2022 –	Machine Learning Engineer *Apple*, Zürich, Switzerland
Summer 2021	Research Intern *Google LLC*, Zürich, Switzerland
2014 – 2015	Research Engineer *Procielo S.A.*, Buenos Aires, Argentina

PUBLICATIONS

1. As, Y., Usmanova, I., Curi, S. & Krause, A. *Constrained Policy Optimization via Bayesian World Models* in *International Conference on Learning Representations* (2022).
2. Bas-Serrano, J., Curi, S., Krause, A. & Neu, G. *Logistic Q-learning* in *International Conference on Artificial Intelligence and Statistics* (2021), 3610.
3. Borsos, Z., Curi, S., Levy, K. Y. & Krause, A. *Online Variance Reduction with Mixtures* in *Proc. International Conference on Machine Learning (ICML)* (2019).
4. Curi, S., Berkenkamp, F. & Krause, A. *Efficient model-based reinforcement learning through optimistic policy search and planning* in. **33** (2020), 14156.
5. Curi, S., Bogunovic, I. & Krause, A. *Combining Pessimism with Optimism for Robust and Efficient Model-Based Deep Reinforcement Learning* in *International Conference on Machine Learning* (2021), 2254.
6. Curi, S., Lederer, A., Hirche, S. & Krause, A. *Adaptive Input Estimation in Linear Dynamical Systems with Applications to Learning-from-Observations* in *2022 IEEE 61st Conference on Decision and Control (CDC)* (2022).
7. Curi, S., Levy, K. Y., Jegelka, S. & Krause, A. *Adaptive sampling for stochastic risk-averse learning* in. **33** (2020), 1036.
8. Curi, S., Levy, K. Y. & Krause, A. *Adaptive Input Estimation in Linear Dynamical Systems with Applications to Learning-from-Observations* in *2019 IEEE 58th Conference on Decision and Control (CDC)* (2019), 4115.
9. Curi, S., Melchior, S., Berkenkamp, F. & Krause, A. *Structured Variational Inference in Partially Observable Unstable Gaussian Process State Space Models* in *Learning for Dynamics and Control* (2020), 147.
10. Fiducioso, M., Curi, S., Schumacher, B., Gwerder, M. & Krause, A. *Safe contextual Bayesian optimization for sustainable room temperature PID control tuning* in *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (2019), 5850.

11. Treven, L., Curi, S., Mutny, M. & Krause, A. *Learning stabilizing controllers for unstable linear quadratic regulators from a single trajectory* in *Learning for Dynamics and Control* (2021), 664.

12. Urpi, N. A., Curi, S. & Krause, A. *Risk-Averse Offline Reinforcement Learning* in *International Conference on Learning Representations* (2020).