TU Delft

Delft University of Technology

Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee

Han, Minghao; Tian, Yuan; Zhang, Lixian; Wang, Jun; Pan, Wei

DOI
10.1016/j.automatica.2021.109689

Publication date
2021

Document Version
Final published version

Published in
Automatica

Citation (APA)
Han, M., Tian, Y., Zhang, L., Wang, J., & Pan, W. (2021). Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee. *Automatica*, *129*, Article 109689. https://doi.org/10.1016/j.automatica.2021.109689

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Automatica 129 (2021) 109689

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee☆

Minghao Han [a], Yuan Tian [b], Lixian Zhang [a], Jun Wang [c], Wei Pan [b,*]

[a] Department of Control Science and Engineering, Harbin Institute of Technology, China
[b] Department of Cognitive Robotics, Delft University of Technology, Netherlands
[c] Department of Computer Science, University College London, UK

ARTICLE INFO

Article history:
Received 28 April 2020
Received in revised form 18 November 2020
Accepted 9 April 2021
Available online 8 May 2021

Keywords:
Data-based control
Reinforcement learning
Constrained dynamic system
Uniformly ultimate boundedness stability
Lyapunov's method

ABSTRACT

Reinforcement learning (RL) is promising for complicated stochastic nonlinear control problems. Without using a mathematical model, an optimal controller can be learned from data evaluated by certain performance criteria through trial-and-error. However, the data-based learning approach is notorious for not guaranteeing stability, which is the most fundamental property for any control system. In this paper, the classic Lyapunov's method is explored to analyze the uniformly ultimate boundedness stability (UUB) solely based on data without using a mathematical model. It is further shown how RL with UUB guarantee can be applied to control dynamic systems with safety constraints. Based on the theoretical results, both off-policy and on-policy learning algorithms are proposed respectively. As a result, optimal controllers can be learned to guarantee UUB of the closed-loop system both at convergence and during learning. The proposed algorithms are evaluated on a series of robotic continuous control tasks with safety constraints. In comparison with the existing RL algorithms, the proposed method can achieve superior performance in terms of maintaining safety. As a qualitative evaluation of stability, our method shows impressive resilience even in the presence of external disturbances.

© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

The recent progress in reinforcement learning (RL) (Sutton, Barto, & Williams, 1992) has produced many interesting and impressive results in control problems and proves to be effective in finding optimal controllers for nonlinear stochastic systems modeled by Markov decision process (MDP), for which the traditional control methods are hardly applicable. However, the learning methods are notorious for not guaranteeing stability. Given a control system, stability is one of the most important properties, because an unstable system is typically useless and potentially dangerous. This presents a major bottleneck for the broad control engineering applications. Stability analysis has a long history in control engineering, in which Lyapunov's method plays a central role (Sastry, 2013; Slotine, Li, et al., 1991; Vidyasagar, 2002). However, the classical control methods rely on the full or partial knowledge of the system dynamics to design controllers and are largely limited to systems with simple dynamics. Thus, it is a natural move to combine RL with control theory to develop learning control methods with a stability guarantee (Busoniu, de Bruin, Tolic, Kober, & Palunko, 2018; Han, Tian, Zhang, Wang, & Pan, 2019; Han, Zhang, Wang, & Pan, 2020).

Among various definitions of stability, uniformly ultimate boundedness stability (UUB) has been extensively studied for dynamic systems (Corless & Leitmann, 1981; Jain & Bhasin, 2017). UUB generally says that the trajectories will enter the neighborhood of the equilibrium within finite time and never escape from this set thereafter (see the trajectory in Fig. 1). Intuitively, this property is consistent with the requirement of many control tasks with constraints on the states, where the states are required to stay in a certain region which is safe. Thus in this paper, as an application scenario, the controller with UUB guarantee is learned to solve control tasks with safety constraints. In terms of learning a controller, UUB is well known in the context of adaptive dynamic programming (ADP) on (1) UUB of the states or tracking error in uncertain systems (Mu, Ni, Sun, & He, 2016; Shih, Kaul, Jagannathan, & Drallmeier, 2007; Yang, Liu, Wei, & Wang, 2016) and (2) UUB of the estimation error of critic function and controller (Luo, Yang, Liu, & Wu, 2019; Shih et al., 2007; Wang, Liao, & Dong, 2018). However, in ADP, the model structure, not necessarily the model parameters, should be known a prior. And the input structure of the nonlinear system is often assumed

☆ M. Han and L. Zhang were supported in part by the National Natural Science Foundation of China under Grant 12072088, 62003118 and 62003117, in part by the Natural Science Foundation of Heilongjiang Province of China under Grant ZD2020F001, in part by the China Scholarship Council under Grant 202006120085. W. Pan was supported by Huawei and AnKobot. The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Alessandro Abate under the direction of Editor Ian R. Petersen.

* Corresponding author.
E-mail address: wei.pan@tudelft.nl (W. Pan).

https://doi.org/10.1016/j.automatica.2021.109689
0005-1098/© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

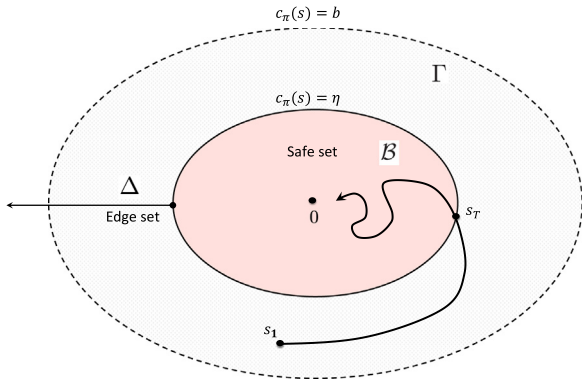


Fig. 1. Conceptual illustration of UUB in the state space and the relation among the starting set Γ, the edge set Δ and the inner set $\mathcal{B}$. For trajectories starting from the set $\Gamma : \{s|\|s\| \leq b\}$, the state will eventually enter and stay inside the inner set $\mathcal{B} : \{s|\|s\| \leq \eta\}$ after T time steps..

to be affine. However, the UUB analysis for the general class of nonlinear stochastic systems by solely using data has not been addressed and remains as an open problem (Busoniu et al., 2018; Gorges, 2017).

The control tasks with safety constraints on the state have been extensively studied in model predictive control (MPC) literature (Mayne, 2001; Mayne, Rawlings, Rao, & Scokaert, 2000) and the results are applied in various industrial processes (Garcia, Prett, & Morari, 1989; Scattolini, 2009). In the context of RL, control problems with safety constraints are also well studied. In Achiam, Held, Tamar, and Abbeel (2017), the authors proposed a safety constrained policy optimization (CPO) approach based on the trust region method, which guarantees the constraint satisfaction with a safe initial policy, but it is restricted to the on-policy algorithms and suffers from the low sample efficiency. In Chow, Nachum, Duenez-Guzman, and Ghavamzadeh (2018), a Lyapunov-based approach for solving constrained control tasks is proposed with a novel way of constructing the Lyapunov function through linear programming. In Chow, Nachum, Faust, Ghavamzadeh, and Duenez-Guzman (2019), the above result is further generalized to continuous control tasks. However, the above results can only guarantee that the cumulative sum of a designed constraint function being kept under a threshold. Moreover, none of these results provides stability guarantees of any kind. Since the property of attraction is missing, simply satisfying the safety constraints does not imply stability, and the agent may easily violate constraints in the presence of slight disturbances.

To apply machine learning algorithms to control constrained dynamic systems is advancing recently. In Berkenkamp, Turchetta, Schoellig, and Krause (2017), a model-based RL method is proposed to deal with Lipschitz continuous deterministic nonlinear systems. Nevertheless, safety is ensured by validating the stability condition on discretized points in the subset of state space with the help of a learned model, limiting its application to rather simple and low-dimensional systems. The combination between RL with control barrier functions (CBF) has raised many attentions in recent years (Cheng, Orosz, Murray, & Burdick, 2019; Choi, Castañeda, Tomlin, & Sreenath, 2020). The general idea is to incorporate the RL controller with a model-based controller using CBFs. Cheng et al. (2019) exploit the nominal model to design a CBF-based controller to ensure safety, then the unknown dynamic is learned using the Gaussian process and the RL controller further improves the return performance. In Choi et al. (2020), based on the controller designed using the nominal model, RL is exploited to solve the safe control problems in control affine nonlinear systems under model uncertainty. Another common way of solving constrained control tasks is to incorporate MPC with online model-learning techniques, such as Ostafew, Schoellig, and Barfoot (2016), Thananjeyan et al. (2020) and Zanon and Gros (2020). Saunders, Sastry, Stuhlmueller, and Evans (2018) propose an RL framework that can exploit expert knowledge to safely improve control performance without violating safety constraints. Different from these approaches, in this paper, neither the nominal model nor expert knowledge is needed to learn a safe controller.

In this paper, a novel data-based UUB theorem without using a mathematical model is proposed. Based on the theoretical result, an off-policy based on an actor–critic algorithm and a policy optimization algorithm are developed respectively to learn controllers with the UUB guarantee. The contributions of this paper can be summarized as follows:

- A novel and principled method is proposed to construct Lyapunov functions based on data to analyze the closed-loop stability of stochastic nonlinear systems characterized by MDP.
- The classical definition of UUB is generalized to deal with control tasks with safety constraints on the states.
- Practical algorithms are designed to search for the optimal safe policy with UUB guarantee while safety is guaranteed both during learning and exploitation.

In a series of high-dimensional continuous control tasks with safety constraints such as locomotion for legged robots and manipulators, as well as a quadrotor, the proposed algorithms outperform the existing (safe) RL algorithms (Achiam et al., 2017; Chow et al., 2019) in terms of both performance and safety. Besides, it is empirically shown that the controller with the UUB guarantee is more capable of dealing with perturbations and disturbances in comparison with those without such guarantees.

The remainder of this paper is organized as follows: the preliminaries and problem formulation are introduced in Section 2; the main theoretical result is presented in Section 3; an off-policy algorithm and a policy optimization algorithm with the UUB guarantee are described in Section 4; the experiments that validate the proposed algorithms are presented in Section 5; Section 6 concludes this work.

2. Preliminaries

In RL, a dynamical system is often characterized by a Markov decision process (MDP) in which the next state only depends on the current state and action. In MDPs, $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$ is the state vector at time t, $\mathcal{S}$ denotes the state space. The agent then takes an action $a_t \in \mathcal{A} \subseteq \mathbb{R}^m$ according to a stochastic policy/controller[1] $\pi(a_t|s_t)$, resulting in the next state s_{t+1}. The transition of the state is dominated by the transition probability density function $p(s_{t+1}|s_t, a_t)$, which denotes the probability density of the next state s_{t+1}. In MDP, a reward function $r(s_t, a_t)$ is used to measure the immediate performance of a state–action pair (s_t, a_t). The goal is to find π which can maximize the objective function/return $J(\pi) \triangleq \sum_{t=1}^{\infty} \mathbb{E}_{s_t,a_t} \gamma^t r(s_t, a_t)$. Additionally, for control problems with safety constraints, a continuous nonnegative constraint function $c(s_t, a_t)$ is introduced to measure how safe a state–action pair is. The state–action pair can be viewed as safe if $c(s_t, a_t)$ is lower than a designed threshold.

Next, the notation of this paper will be introduced in two parts. First, the concept and definition of UUB studied in this paper are introduced. Then the constrained control problems or so-called constrained Markov decision process (CMDP) are described as a particular application scene of UUB.

[1] We use controller throughout the paper.

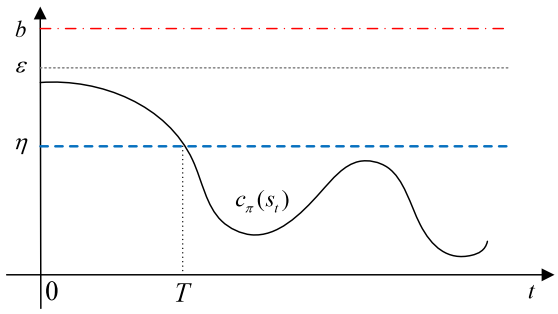


Fig. 2. An illustration of the concept of UUB in time domain.

2.1. Uniformly Ultimate Boundedness (UUB) stability

First, the classical definition of UUB stability is given as follows.

Definition 1 (*Thowsen, 1983*)**.** A system is said to be uniformly ultimately bounded with ultimate bound η, if there exist positive constants b, η, $\forall \epsilon < b$, there exists $T(\epsilon, \eta)$, such that $\|s_{t_0}\| < \epsilon \implies \|s_t\| < \eta, \forall t > t_0 + T$. If this holds for arbitrary large ϵ, then it is globally uniformly ultimately bounded.

The classical definition of UUB generally says that for trajectories starting from a point where the norm of the state less than b, the state will eventually enter and stay inside the set where $\|s\| \leq \eta$ after T time steps.

However, the above definition is of limited use for the general class of control tasks with safety constraints, where the constraint functions $c(s_t, a_t)$ are not necessarily the norm of the state, i.e., $\|s\|$. For example, safety-critical applications like autonomous vehicles may use the distance from the working area or central lane as the safety constraint; altitude control of a drone may require that sinusoid of an angle to be less than a certain threshold. Therefore, the classical definition of UUB is extended to the more general case as follows.

Definition 2. A system is said to be uniformly ultimately bounded with respect to $c_\pi(\cdot)$, if there exist positive constants η, b, $\forall \epsilon < b$, there exists $T(\epsilon, \eta)$, such that $c_\pi(s_1) \leq \epsilon \implies c_\pi(s_t) \leq \eta, \forall t \geq T$.

Where $c_\pi(s) \triangleq \mathbb{E}_{a\sim\pi} c(s, a)$ denotes the constraint function under the controller π. In the rest of this paper, UUB refers to the property defined above.

The difference between Definition 1 and Definition 2 is merely on the substitution of the norm of states with a constraint function. The general idea of UUB in Definition 2 is demonstrated in Figs. 1 and 2 in the state space and time domain respectively. With the proper choice of $c(s, a)$, UUB in terms of the constraint function implies recoverability from danger within finite time. For example, a vehicle will recover to the road within finite time if it is accidentally disturbed and run into a risky area; a motor can recover to normal status within finite time if the kinetic energy or torque accidentally exceeds a dangerous threshold.

2.2. Control of constrained dynamic system

In this section, the control problems with safety constraints will be formulated. It will be further shown how safety constraints can be ensured as a consequence of the UUB guarantee.

The safety constraints are measured by a continuous non-negative constraint function $c(s_t, a_t)$. In the safety constrained control tasks, the objective is to find a controller π which not only maximizes $J(\pi)$ but also satisfy the expectation of the safety constraint $\mathbb{E}_{s_t} c_\pi(s_t) \leq d, \forall t \in [1, \infty)$, i.e.,

$$\max_\pi J(\pi) \text{ s.t. } \mathbb{E}_{s_t} c_\pi(s_t) \leq d, \forall t \in [1, \infty). \tag{1}$$

An MDP with such safety constraints is called constrained Markov decision process (CMDP) (Altman, 1999).

If the system is UUB in terms of the constraint function with an ultimate bound $\eta < d$, the value of the constraint function is guaranteed to converge and stay under η after T time steps. To further ensure safety during the T time steps, it is also needed to ensure that the expectation of the constraint function to be lower than d. Later it will be shown that this is an inherent property of the system being UUB. Thus solving the control problem (1) is equivalent to finding an optimal controller that ensures the closed-loop system is UUB.

Before proceeding, some notations are introduced. The action-value function $Q_\pi(s, a) \triangleq \sum_{t=1}^{\infty} \mathbb{E}_{s_t, a_t}[\gamma^t r(s_t, a_t)|s_1 = s, a_1 = a]$ denotes the subsequent return under the controller π after taking action a at the state s. The value function with respect to constraint function $V_\pi^c(s) \triangleq \sum_{t=1}^{\infty} \mathbb{E}_{s_t, a_t}[\gamma^t c(s_t, a_t)|s_1 = s]$ denotes the discounted sum of constraint function starting from the state s under the controller π. $\rho(s_1)$ denotes the probability density function of starting states, which is a continuous function and takes positive values on the starting set $\Gamma \triangleq \{s|c_\pi(s) \leq b\}$. The closed-loop transition probability is denoted as $p_\pi(s'|s) \triangleq \int_{\mathcal{A}} \pi(a|s)p(s'|s, a)\mathrm{d}a$. Also note that the closed-loop state distribution at a certain instant t as $p(s|\rho, \pi, t)$, which can be defined iteratively: $p(s'|\rho, \pi, t+1) = \int_{\mathcal{S}} p_\pi(s'|s)p(s|\rho, \pi, t)\mathrm{d}s, \forall t \in \mathbb{Z}_{[1,\infty)}$ and $p(s|\rho, \pi, 1) = \rho(s)$.

Two important sets are exploited in this paper, as shown in Fig. 1. First, the edge set $\Delta \triangleq \{s|c_\pi(s) \geq \eta\}$ is composed of states that are unsafe. Conversely, the inner set $\mathcal{B} \triangleq \{s|c_\pi(s) < \eta\}$ is the set of absolutely safe states and $\mathcal{B} \cup \Delta = \mathcal{S}$.

3. Theoretical results

In this section, the main assumptions and a novel data-based UUB theorem for stochastic systems are proposed. The UUB theorem enables analyzing the UUB of the system through a data-based manner, which further enables policy learning in the next section.

First, we made the following assumption:

Assumption 1. The constraint function $c(s, a)$ is non-negative and there exists an integrable function $g(s, a)$ such that $c(s, a) \leq g(s, a), \forall (s, a) \in \mathcal{S} \times \mathcal{A}$.

This assumption easily holds with the typical choices of constraint function in practice, such as kinetic energy, altitude, etc. It is also assumed that the Markov chain induced by policy π is ergodic with a unique stationary distribution q_π,

$$q_\pi(s) = \lim_{t\to\infty} p(s|\rho, \pi, t),$$

which is a common assumption for the optimal control of a Markov decision process (Bhandari, Russo, & Singal, 2018; Korda & La, 2015; Sutton, Maei, & Szepesvári, 2009; Zou, Xu, & Liang, 2019).

Our approach utilizes the Lyapunov function to prove the stability condition. Lyapunov's method has long been used in control theory for stability analysis and controller design (Boukas & Liu, 2000), but mostly exploited along with a known model of a dynamic system, whether deterministic or probabilistic (Corless & Leitmann, 1981; Huang, Han, Cai, & Liu, 2011; Thowsen, 1983).

In this paper, instead of using a model, we will present the following theorem as a sufficient condition for UUB solely based on samples.

Theorem 1. *If there exist a function $L(s): \mathcal{S} \to \mathbb{R}_+$ and positive constants α_1, α_2, α_3, η, such that*

$$\alpha_1 c_\pi(s) \le L(s) \le \alpha_2 c_\pi(s), \forall s \in \mathcal{S} \tag{2}$$

and

$$\begin{aligned} &\mathbb{E}_{s\sim\mu_N}\left(\mathbb{E}_{s'\sim P_\pi} L(s')\mathbb{1}_\Delta(s') - L(s)\mathbb{1}_\Delta(s)\right) \\ &< -\alpha_3 \mathbb{E}_{s\sim\mu_N} c_\pi(s)\mathbb{1}_\Delta(s) \end{aligned} \tag{3}$$

where $\mu_N(s)$ denotes the average distribution of s over the finite N time steps,

$$\mu_N(s) \doteq \frac{1}{N}\sum_{t=1}^{N} p(s|\rho,\pi,t)$$

N is the maximum instant that the probability of being in the edge set is greater than zero, $N = \max\{t : \mathbb{P}(s \in \Delta|\rho,\pi,t) > 0\}$; $N = \infty$ if for any δ there exists an instant $t > \delta$ such that $\mathbb{P}(s \in \Delta|\rho,\pi,t) > 0$. $\mathbb{1}_\Delta(s)$ denotes the function

$$\mathbb{1}_\Delta(s) = \begin{cases} 1 & s \in \Delta \\ 0 & s \notin \Delta \end{cases}$$

where $\Delta = \{s|c_\pi(s) \ge \eta\}$.

Then one has the following holds: (i) the system is uniformly ultimately bounded with ultimate bound η; (ii) the expectation $\mathbb{E}_{p(s|\rho,\pi,t)} c_\pi(s)$ is bounded during the N time steps, $\mathbb{E}_{p(s|\rho,\pi,t)} c_\pi(s) < \frac{\alpha_2 b}{\alpha_3} + \eta, \forall t \in [1, N]$.

Proof. The proof can be divided into two steps. First, we will prove that N is finite based on the conditions and assumptions, then prove the boundedness on the expectation of c_π during the N steps. To show this, we will assume that N is infinity and prove by contradiction.

In that case, the finite-horizon sampling distribution $\mu_N(s)$ turns into the infinite-horizon sampling distribution

$$\mu(s) = \lim_{N\to\infty} \mu_N(s) = \lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} p(s|\rho,\pi,t)$$

The existence of $\mu(s)$ is guaranteed by the existence of $q_\pi(s)$. Since the sequence $\{p(s|\rho,\pi,t), t \in \mathbb{Z}_+\}$ converges to $q_\pi(s)$ as t approaches ∞, then by the Abelian theorem, the sequence $\{\frac{1}{T}\sum_{t=1}^{T} p(s|\rho,\pi,t), T \in \mathbb{Z}_+\}$ also converges and $\mu(s) = q_\pi(s)$. Then one naturally has that the sequence $\{\mu_N(s)L(s)\mathbb{1}_\Delta(s), T \in \mathbb{Z}_+\}$ converges pointwise to $q_\pi(s)L(s)\mathbb{1}_\Delta(s)$.

Let $g_\pi(s)$ denote $\mathbb{E}_{a\sim\pi} g(s,a)$. According to Assumption 1 and (2), $L(s) \le \alpha_2 c_\pi(s) \le \alpha_2 g_\pi(s)$, which follows that

$$\mu_N(s)L(s)\mathbb{1}_\Delta(s) \le \alpha_2 \mu_N(s) g_\pi(s)\mathbb{1}_\Delta(s)$$

According to the Lebesgue's Dominated convergence theorem (Royden, 1968), if a sequence $f_n(s)$ converges pointwise to a function f and is dominated by some integrable function g in the sense that,

$$|f_n(s)| \le g(s), \forall s \in \mathcal{S}, \forall n$$

then one has

$$\lim_{n\to\infty}\int_{\mathcal{S}} f_n(s)\mathrm{d}s = \int_{\mathcal{S}} \lim_{n\to\infty} f_n(s)\mathrm{d}s$$

Applying this theorem to the first term in the left-hand-side of (3)

$$\begin{aligned} &\mathbb{E}_{s\sim\mu}\mathbb{E}_{s'\sim p_\pi} L(s')\mathbb{1}_\Delta(s') \\ &= \int_{\mathcal{S}} \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N} p(s|\rho,\pi,t)\Big(\int_{\mathcal{S}} p_\pi(s'|s)L(s')\mathbb{1}_\Delta(s')\mathrm{d}s'\Big)\mathrm{d}s \\ &= \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}\int_{\mathcal{S}} L(s')\mathbb{1}_\Delta(s')\int_{\mathcal{S}} p_\pi(s'|s)p(s|\rho,\pi,t)\mathrm{d}s\mathrm{d}s' \\ &= \lim_{N\to\infty}\frac{1}{N}\sum_{t=2}^{N+1}\mathbb{E}_{p(s|\rho,\pi,t)}L(s)\mathbb{1}_\Delta(s) \end{aligned}$$

Similarly, $\mathbb{E}_{s\sim\mu}L(s) = \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}\mathbb{E}_{p(s|\rho,\pi,t)}L(s)$. It follows that on the left-hand-side of (3),

$$\begin{aligned} &\mathbb{E}_{s\sim\mu}(\mathbb{E}_{s'\sim p_\pi} L(s')\mathbb{1}_\Delta(s') - L(s)\mathbb{1}_\Delta(s)) \\ &= \lim_{N\to\infty}\frac{1}{N}\Big(\sum_{t=2}^{N+1}\mathbb{E}_{p(s|\rho,\pi,t)}L(s)\mathbb{1}_\Delta(s) - \sum_{t=1}^{N}\mathbb{E}_{p(s|\rho,\pi,t)}L(s)\mathbb{1}_\Delta(s)\Big) \\ &= \lim_{N\to\infty}\frac{1}{N}(\mathbb{E}_{p(s|\rho,\pi,N+1)}L(s)\mathbb{1}_\Delta(s) - \mathbb{E}_{\rho(s)}L(s)\mathbb{1}_\Delta(s)) \end{aligned}$$

Since $\mathbb{E}_{\rho(s)}L(s)\mathbb{1}_\Delta(s)$ is finite, thus the limitation value $\lim_{N\to\infty}\frac{1}{N}(\mathbb{E}_{\rho(s)}L(s)\mathbb{1}_\Delta(s)) = 0$. The above equation equals

$$\begin{aligned} &\lim_{N\to\infty}\frac{1}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}L(s)\mathbb{1}_\Delta(s) \\ &\ge \lim_{N\to\infty}\frac{\alpha_1}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}c_\pi(s)\mathbb{1}_\Delta(s) \end{aligned}$$

Note that $c_\pi(s)\mathbb{1}_\Delta(s)$ is greater than η if $s \in \Delta$ and equals zero if $s \notin \Delta$, which can be summarized as $c_\pi(s)\mathbb{1}_\Delta(s) \ge \eta\mathbb{1}_\Delta(s)$. Thus

$$\begin{aligned} &\lim_{N\to\infty}\frac{1}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}L(s)\mathbb{1}_\Delta(s) \\ &\ge \lim_{N\to\infty}\frac{\alpha_1\eta}{N}\mathbb{E}_{p(s|\rho,\pi,N+1)}\mathbb{1}_\Delta(s) \\ &= \lim_{N\to\infty}\frac{\alpha_1\eta}{N}\mathbb{P}(s \in \Delta|\rho,\pi,N+1) \\ &= 0 \end{aligned} \tag{4}$$

Now let us look into the right-hand-side of (3). Since $\mu(s) = q_\pi(s)$, the right-hand-side of (3) equals

$$\begin{aligned} &-\alpha_3\mathbb{E}_{s\sim q_\pi} c_\pi(s)\mathbb{1}_\Delta(s) \\ &\le -\alpha_3\mathbb{E}_{s\sim q_\pi}\eta\mathbb{1}_\Delta(s) \\ &= -\alpha_3\eta\lim_{t\to\infty}\mathbb{P}(s \in \Delta|\rho,\pi,t) \end{aligned}$$

Combining the above inequality with (3) and (4), one has that $\lim_{t\to\infty}\mathbb{P}(s \in \Delta|\rho,\pi,t) < 0$, which is contradictory with the fact that $\mathbb{P}(s \in \Delta|\rho,\pi,t)$ is non-negative. Thus there exists a finite N such that $\mathbb{P}(s \in \Delta|\rho,\pi,t) = 0$ for all $t > N$, which concludes the proof of UUB.

Additionally, it will be shown that the expectation of $c_\pi(s)$ is bounded by a finite value during the N time steps. As N is a finite value, (3) implies that

$$\begin{aligned} &\mathbb{E}_{p(s|\rho,\pi,N+1)}L(s)\mathbb{1}_\Delta(s) - \mathbb{E}_{\rho(s)}L(s)\mathbb{1}_\Delta(s) \\ &< -\alpha_3\sum_{t=1}^{N}\mathbb{E}_{p(s|\rho,\pi,t)}c_\pi(s)\mathbb{1}_\Delta(s) \end{aligned}$$

Then for any instant $n \in [1, N]$, one has the following holds

$$\begin{aligned} &\mathbb{E}_{p(s|\rho,\pi,n)}c_\pi(s)\mathbb{1}_\Delta(s) \\ &\le \frac{\alpha_2}{\alpha_3}\mathbb{E}_{\rho(s)}c_\pi(s) - \sum_{t=n+1}^{N}\mathbb{E}_{p(s|\rho,\pi,t)}c_\pi(s)\mathbb{1}_\Delta(s) \\ &\quad - \sum_{t=1}^{n-1}\mathbb{E}_{p(s|\rho,\pi,t)}c_\pi(s)\mathbb{1}_\Delta(s) \end{aligned}$$

Note that the expectation of $c_\pi(s)$ at instant n equals

$$\begin{aligned}&\int_{\mathcal{S}} p(s|\rho,\pi,n)c_\pi(s)\mathrm{d}s\\ =&\int_{\Delta} p(s|\rho,\pi,n)c_\pi(s)\mathrm{d}s+\int_{\mathcal{B}} p(s|\rho,\pi,n)c_\pi(s)\mathrm{d}s\\ =&\mathbb{E}_{p(s|\rho,\pi,n)}c_\pi(s)\mathbb{1}_\Delta(s)+\int_{\mathcal{B}} p(s|\rho,\pi,n)c_\pi(s)\mathrm{d}s\end{aligned}$$

Then the bound of the expectation of $c_\pi(s)$ at instant n is derived as follows,

$$\begin{aligned}&\mathbb{E}_{p(s|\rho,\pi,t)}c_\pi(s)\\ \le&\frac{\alpha_2}{\alpha_3}\mathbb{E}_{\rho(s)}c_\pi(s)-\sum_{t=n+1}^{N}\mathbb{E}_{p(s|\rho,\pi,t)}c_\pi(s)\mathbb{1}_\Delta(s)\\ &-\sum_{t=1}^{n-1}\mathbb{E}_{p(s|\rho,\pi,t)}c_\pi(s)\mathbb{1}_\Delta(s)+\int_{\mathcal{B}} p(s|\rho,\pi,n)c_\pi(s)\mathrm{d}s\\ \le&\frac{\alpha_2 b}{\alpha_3}-\eta\sum_{t=n+1}^{N}\mathbb{P}(s\in\Delta|\rho,\pi,t)\\ &-\eta\sum_{t=1}^{n-1}\mathbb{P}(s\in\Delta|\rho,\pi,t)+\eta\mathbb{P}(s\in\mathcal{B}|\rho,\pi,n)\\ <&\frac{\alpha_2 b}{\alpha_3}+\eta\end{aligned}$$

which concludes the proof. □

Some discussion and explanations are needed for the above theorem. First, (2) confines the property that the Lyapunov function needs to satisfy. (3) is the data-based energy decreasing condition, of which the evaluation requires sampling data according to sampling distribution $\mu_N(s)$. Although $\mu_N(s)$ is defined on the state space $\mathcal{S}$, the indication function $\mathbb{1}_\Delta(s)$ only takes the non-zero value on the edge set Δ. Thus (3) requires the Lyapunov value to be decreasing on large in the edge set Δ and eventually entering the inner set $\mathcal{B}$.

Remark 1. While results on UUB of various systems are well-known (Corless & Leitmann, 1981; Huang et al., 2011; Thowsen, 1983), however, both of these results require the full knowledge of the dynamic model of the system. On the contrary, the proposed UUB theorem enables a data-based approach to analyze the stability of the system, i.e. collecting lots of state transition pairs and evaluate the value of (3) through the Monte-Carlo method. In the data-based stability analysis, the system can be a complete black-box, as long as its dynamic satisfies the Markov property.

Some connections are to be drawn between safety constrained control problems and the proposed UUB theorem. If the system is UUB with ultimate bond $\frac{\alpha_2 b}{\alpha_3}+\eta<d$, then it is guaranteed that the system satisfies the safety constraint in (1). These conditions can be satisfied by choosing the hyperparameters α_2, α_3, and η. (3) is the condition that requires training of the control policy, which will be discussed in detail in the following section.

4. Reinforcement learning algorithms with UUB stability guarantee

In this section, combined with the theoretical result in Theorem 1, both an off-policy and an on-policy RL algorithm are proposed respectively. First, based on the result in Theorem 1, an actor–critic RL algorithm called Lyapunov-based soft actor–critic (LSAC) is given, where two critic functions are used. The first is the standard critic function $Q(s,a)$ in the actor–critic RL algorithm, which is used to evaluate the performance in terms of the cumulative return. The other critic function is introduced to evaluate the UUB condition (3). We call the second critic function as Lyapunov critic function $L_c(s,a)$. Then a trust-region policy optimization algorithm, Lyapunov-based constrained policy optimization (LCPO) is developed. LCPO ensures that at each update step the UUB condition is satisfied and increases the cumulative return monotonically so that the safety during training can also be guaranteed. In both LSAC and LCPO, a Lyapunov function is firstly specified to directly learn the controller and Lyapunov function. The controller is then updated to ensure that the energy decreasing UUB condition (3) holds for the learned Lyapunov function.

4.1. Learning a Lyapunov critic function

In Theorem 1, the Lyapunov function $L(s)$ is essential in the stability analysis. However, it is not directly applicable in an existing actor–critic learning framework since the gradient of L with respect to the controller π is unavailable. To enable the actor–critic learning, the Lyapunov critic function $L_c(s,a)$ is introduced to prove the stability theorem therefore making sure the learned controller π can guarantee the stability of the closed-loop system. L_c depends on both the state s and the action a.[2] L_c satisfies $L(s)=\mathbb{E}_{a\sim\pi}L_c(s,a)$, such that it can be exploited by judging the value of (3). In this paper, L_c is constructed by using a fully connected deep neural network (DNN) parameterized by ϕ. A ReLU activation function is used in the output layer of the DNN to ensure positive output.

From a theoretical point of view, some functions, such as the norm of state and value function, naturally satisfy the basic requirement of Lyapunov function (2). These functions are referred to as Lyapunov candidate functions. However, Lyapunov candidate functions are conceptual functions without any parameterization. Since their gradient with respect to the controller is not tractable, they are not directly applicable in an actor–critic learning process. In the proposed framework, the Lyapunov candidate acts as a supervision signal during the training of L_c. L_c is updated to approximate the target function L_{target} related to the chosen Lyapunov candidate, minimizing the following objective function simply using least square algorithm,

$$J(\phi)=\mathbb{E}_{\mathcal{D}}\left[\frac{1}{2}(L_c(s,a)-L_{\text{target}}(s,a))^2\right] \tag{5}$$

where $\mathcal{D}=\{(s,a,s',r,c)\}$ is the set of observed transition tuple under the controller π.

The choice of the Lyapunov candidate plays an important role in learning a controller. In control theory, the sum of quadratic polynomials, e.g., $L(s)=s^\top Ps$ where P is a positive definite matrix, is often used. Such Lyapunov functions can be efficiently discovered by using semi-definite programming (SDP) solvers with certain limited conservatism for control tasks. In the context of RL (Berkenkamp et al., 2017; Chow et al., 2018), the value function V_π^c is proved to be a valid Lyapunov candidate. In the meantime, the constraint function c_π is also a valid Lyapunov candidate due to its nonnegativity. The value function and constraint function are chosen to be the Lyapunov candidate in this paper while other potential choices are left for future study.

With the value function chosen to be the Lyapunov candidate, the target function L_{target} in (5) is

$$L_{\text{target}}(s,a)=c(s,a)+\max_{a'}\gamma L_c'(s',a') \tag{6}$$

[2] It should be noted that the Lyapunov critic function L_c is *not* a proper Lyapunov function since it also depends on action a.

where L_c' is the network that has the same structure as L_c, but parameterized by a different set ϕ'. These parameters of the neural network are updated through exponentially moving average controlled by a hyperparameter $\tau \in \mathbb{R}_{(0,1)}$, $\phi'_{k+1} \leftarrow \tau\phi_k + (1-\tau)\phi'_k$, as typically used in actor–critic algorithms (Lillicrap et al., 2015). It should be noted that when the constraint function is chosen to be the Lyapunov candidate, the target function L_{target} is much simpler by having $L_{\text{target}}(s,a) = c(s,a)$, and the L_c' network and moving average update are not needed.

Remark 2. This remark will collectively summarize the Lyapunov terms used in the section above for clarity. (i) L refers to the Lyapunov function. (ii) Lyapunov candidates are functions that potentially can be used as a Lyapunov function, such as value function and $\|s\|$. (iii) The Lyapunov critic function L_c is a function dependent on the state and action, and satisfies $L = \mathbb{E}_{a\sim\pi}L_c(s,a)$. (iv) The target function L_{target} refers to the supervision signal used in the training of L_c, which takes different forms when different Lyapunov candidates are chosen. (v) L_c' is a network that shares the same structure with L_c, but only the parameters are updated by applying moving average to the parameter of L_c. This is a typical trick used in the RL literature, designed to improve the stability of the learning process.

Remark 3. In this paper, the Lyapunov functions are parameterized using neural networks. It is also possible to choose a parameterization form that can be updated using SDP, such as a quadratic of s and a. However, such a parameterization may result in a large approximation error when the constraint function involves multiple types of nonlinearities that cannot be approximated using a single polynomial, e.g. the constraint functions with non-differentiable nonlinearities that will be used Section 5. Alternatively, neural networks are powerful function approximators that can theoretically approximate any nonlinear functions in desired precision, thus we exploit neural networks to show the general applicability of the proposed method, and leave other parameterizations for future studies.

4.2. Lyapunov-Based safe off-policy RL algorithm

A novel off-policy RL algorithm based on the actor–critic algorithm, i.e., Lyapunov Safe Actor–Critic (LSAC), is proposed to learn the controller π that can maximize the return while guaranteeing the UUB for the closed-loop system.

The controller π_θ is parameterized by a DNN $f_\theta(s,\epsilon)$ depending on s and a Gaussian noise ϵ. The goal is to learn θ that can maximize $J(\pi)$ in (1), while satisfying the UUB condition (3) simultaneously. In this paper, we build the actor–critic algorithm based on the maximum entropy framework (Haarnoja, Zhou, Abbeel, & Levine, 2018), which can enhance the exploration of the controller during learning and has been proven to substantially improve the robustness of the learned controller (Haarnoja, Zhou, Hartikainen, et al., 2018; Ma et al., 2020). The learning problem is summarized as follows,

$$\max_{\theta} \quad \mathbb{E}_{\mathcal{D}} Q_{\pi_\theta}(s,a) \tag{7}$$

$$\text{s.t. } (3) \tag{8}$$

$$-\mathbb{E}_{\mathcal{D}} \log \pi_\theta(a|s) \geq \mathcal{H}_t \tag{9}$$

where (9) sets the entropy of the controller to be larger than a designed threshold $\mathcal{H}_t$. By exploiting the Lagrange method, solving the above constrained optimization problem is equivalent to minimizing the following objective function,

$$\begin{aligned} J(\pi) =& \mathbb{E}_{\mathcal{D}}\left[-Q(s,f_\theta(s,\epsilon)) + \beta \log \pi_\theta(f_\theta(s,\epsilon)|s)\right] \\ &+ \lambda \mathbb{E}_{\mathcal{D}_\Delta}[L_c(s', f_\theta(s',\epsilon))\mathbb{1}_\Delta(s') \\ &- (L_c(s,a) - \alpha_3 c)\mathbb{1}_\Delta(s)] \end{aligned} \tag{10}$$

where β and λ are positive Lagrangian multipliers. Both the values of β and λ are adjusted through the gradient descent/ascent method. $\mathcal{D}_\Delta$ denotes the transition pairs collected from the sampling distribution $\mu_N(s)$.

In our implementation, the double Q-learning technique (Van Hasselt, Guez, & Silver, 2016) is exploited, where two Q-functions $\{Q_1, Q_2\}$ are parameterized by neural networks with parameters ν_1 ν_2. The Q-function with the lower value is exploited in the learning process (Fujimoto, Hoof, & Meger, 2018), which is useful in mitigating performance degradation caused by the bias in the value estimation. Taking these techniques into consideration, the gradient concerning θ is obtained as

$$\begin{aligned} \nabla_\theta J(\pi) =& \mathbb{E}_{\mathcal{D}}[-\min_i Q_i(s,a)\nabla_\theta f_\theta(s',\epsilon) \\ &+ \beta\nabla_\theta \log(\pi_\theta(a|s)) + \beta\nabla_a \log \pi_\theta(a|s)] \\ &+ \lambda \mathbb{E}_{\mathcal{D}_\Delta}[\nabla_{a'} L_c(s',a')\nabla_\theta f_\theta(s',\epsilon)\mathbb{1}_\Delta(s')] \end{aligned} \tag{11}$$

The gradient is composed of two parts: (1) the gradient estimated by the Q-function and the entropy of the controller based on the samples from replay buffer $\mathcal{D}$, and (2) the gradient estimated by the Lyapunov critic based on the samples from the edge buffer $\mathcal{D}_\Delta$. (11) is the basis of the actor–critic algorithm and enables the update of controller with observed transition pairs.

In the actor–critic algorithm, the Q-function is updated by using gradient descent to minimize the following objective function

$$J(Q_i) = \mathbb{E}_{\mathcal{D}} \frac{1}{2}\left[r + \gamma Q_i'(s', f_\theta(s',\epsilon)) - Q_i(s,a)\right]^2, \ i \in \{1,2\}$$

where Q_i' is the target network that has the same structure with Q_i and parameterized by ν_i' but updated through moving average.

Finally, the values of Lagrange multipliers β and λ are adjusted by gradient ascent to maximize the following objectives, respectively,

$$J(\beta) = \beta\mathbb{E}_{\mathcal{D}}[\log \pi_\theta(a|s) + \mathcal{H}_t]$$
$$J(\lambda) = \lambda\mathbb{E}_{\mathcal{D}_\Delta}[L_c(s', f_\theta(s',\epsilon))\mathbb{1}_\Delta(s') - (L_c(s,a) - \alpha_3 c)\mathbb{1}_\Delta(s)]$$

The pseudocode of LSAC is summarized in Algorithm 1.

Algorithm 1 Lyapunov-based Safe Actor–Critic Algorithm (LSAC)

Set iteration index $i \leftarrow 0$ and learning rate δ

repeat

Sample s_1 according to ρ

for each time step **do**

Sample a_t from $\pi(a_t|s_t)$ and step forward

Observe and store $(s_t, a_t, r_t, c_t, s_{t+1})$ in $\mathcal{D}$

end for

Record the largest instant N at which $s_N \in \Delta$

Store all tuples $(s_t, a_t, r_t, c_t, s_{t+1}), t \leq N$ in $\mathcal{D}_\Delta$

for each update step **do**

Sample mini-batches of transitions from $\mathcal{D}$ and $\mathcal{D}_\Delta$ and update parameters with gradients,

$$\theta \leftarrow \theta - \delta\nabla_\theta J(\pi)$$
$$\phi \leftarrow \phi - \delta\nabla_\phi J(L_c)$$
$$\nu_i \leftarrow \nu_i - \delta\nabla_{\nu_i} J(Q_i)$$
$$\lambda \leftarrow \lambda + \delta\nabla_\lambda J(\lambda)$$
$$\beta \leftarrow \beta + \delta\nabla_\beta J(\beta)$$

end for

$i \leftarrow i + 1$

until (3) is satisfied and i exceeds a designed threshold.

4.3. Lyapunov-Based on-policy RL algorithm

The off-policy algorithms can exploit the data collected under a different controller and update the controller much more frequently than the on-policy algorithms, and thus is more favorable in terms of data efficiency and convergence speed. However, safety can be hardly guaranteed during the training process of off-policy algorithms (Chow et al., 2019). When the controller is trained online in the real-world and data are collected directly from the physical systems, safety needs to be guaranteed even during training. To this end, an on-policy algorithm called Lyapunov Constrained Policy Optimization (LCPO) is proposed for these safety-critical scenarios.

In comparison with LSAC (11), instead of approximating the policy gradient using the approximated critics, LCPO is a trust-region style method built upon the linearization of constraint and objective functions locally around the current parameter θ. In trust-region methods, a local constrained optimization problem is solved at each update step and the parameter updates a small step towards improving the objective while satisfying constraints. This determines that the optimization of policy needs more samples than the actor–critic methods to correctly approximate the constrained optimization problem, as well as more computation time in the line-search at each update (Achiam et al., 2017). However, on the other hand, this also guarantees the approximate monotonic improvement of the performance and safety during training with an initial safe policy (Amodei et al., 2016; Moldovan & Abbeel, 2012). Following such a procedure, it is possible to directly train a controller on the real system with safety being assured. The update of the policy parameter θ at the k_{th} iteration can be formulated as follows

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{\substack{s\sim\mathcal{D}\\ a\sim\pi}} Q_{\pi_k}(s,a) \tag{12}$$

$$\text{s.t. } \mathbb{E}_{\substack{s\sim\mathcal{D}_\Delta\\ a\sim\pi}} \left[L(s')\mathbb{1}_\Delta(s') - (L(s) - \alpha_3 c)\,\mathbb{1}_\Delta(s)\right] \le 0 \tag{13}$$

$$\mathbb{E}_{\mathcal{D}} D_{\text{KL}}(\pi_\theta|\pi_k) \le \delta \tag{14}$$

Here, π_k denotes the policy parameterized by θ_k at k_{th} update. In trust-region method, there is a local policy search constraint (14) to prevent the policy from taking unreasonably large update steps, and ensure that post-update policy stays in the neighborhood of the previous policy specified by δ. Here, $D_{KL}(p|q)$ denotes the KL-divergence between two distributions p and q, $D_{KL}(p|q) \doteq \mathbb{E}_p \log(p/q)$. The KL-divergence is a measure of the difference between two distributions and is commonly used in trust-region methods (Achiam et al., 2017; Schulman, Levine, Abbeel, Jordan, & Moritz, 2015). At each update step, the above constrained optimization problem is solved analytically. Since the search of policy is constrained around the previous policy π_k by (14), it is possible to linearize the objective function (12) and the safety constraint (13) around π_k and approximate the local policy search constraint (14) using second-order expansion (Achiam et al., 2017). The approximated optimization problem is as follows,

$$\begin{aligned}\theta_{k+1} = &\arg\max_{\theta} g_Q^\top (\theta - \theta_k)\\ &\text{s.t. } g_L^\top (\theta - \theta_k) + h \le 0\\ &\frac{1}{2}(\theta - \theta_k)^\top H (\theta - \theta_k) \le \delta\end{aligned} \tag{15}$$

where g_Q and g_L are gradients of the objective function and the safety constraint function with respect to θ at θ_k, h is the value of the safety constraint function at θ_k and H is the Hessian of the KL-divergence. Note that the Fisher information matrix H is guaranteed to be positive semi-definite, thus the above optimization problem is convex and its dual is as follows,

$$\max_{\lambda,\beta\ge 0} \frac{1}{2\beta}(g_Q^T H^{-1} g_Q - 2\lambda\mathcal{Z} + \lambda^2\mathcal{N}) + \lambda h - \frac{\beta\delta}{2} \tag{16}$$

where $\mathcal{Z} = g_Q^T H^{-1} g_L$ and $\mathcal{N} = g_L^T H^{-1} g_L$. Suppose that the original problem is feasible and λ^* and β^* are the solutions to (16), then the optimal solution to the primal problem (15) is given by

$$\theta_{k+1} = \theta_k + \frac{1}{\beta^*} H^{-1}(g_Q - \lambda^* g_L) \tag{17}$$

If the optimization problem (15) is not feasible, then a recovery update step is needed. For safety constrained tasks, it is important that the policy π recovers to a set of safe policies as soon as possible. In the meantime, (14) needs to be satisfied as this is the basis of local approximate optimization. The recovery step is equivalent to solving the following optimization problem,

$$\begin{aligned}\theta_{k+1} = &\arg\min_{\theta} g_L^\top (\theta - \theta_k) + h\\ &\text{s.t. } \frac{1}{2}(\theta - \theta_k)^\top H (\theta - \theta_k) \le \delta\end{aligned} \tag{18}$$

The optimal solution to the above recovery optimization problem is

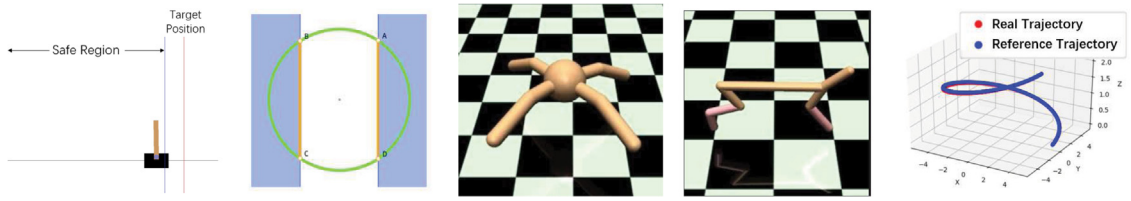
$$\theta^* = \theta_k - \sqrt{\frac{2\delta}{g_L^\top H^{-1} g_L}} H^{-1} g_L \tag{19}$$

In LCPO, the Lyapunov function $L(s)$ is also a DNN parameterized by ϕ. The Lyapunov critic function L_c is not needed since LCPO does not involve critic–actor updates. Meanwhile, the Lyapunov function candidates are still valid following a similar approximation procedure as LSAC. The pseudo-code of LCPO can be found in Algorithm 2.

Algorithm 2 Lyapunov-based Constrained Policy Optimization (LCPO)

for $i = 1, 2, \ldots$ **do**
  Sample s_1 according to ρ
  for each time step **do**
    Sample a_t from $\pi(a_t|s_t)$ and step forward
    Observe and store $(s_t, a_t, r_t, c_t, s_{t+1})$ in $\mathcal{D}$
  end for
  Record the largest instant N at which $s_N \in \Delta$
  Store all tuples $(s_t, a_t, r_t, c_t, s_{t+1})$, $t \le N$ in $\mathcal{D}_\Delta$
  Estimate g_Q, g_L, h, H with $\mathcal{D}$ and $\mathcal{D}_\Delta$
  if (15) is feasible **then**
    Calculate the optimal λ^* and β^*
    Calculate the proposal θ^* with (17)
  else
    Calculate θ^* with (19)
  end if
  Update θ_{k+1} by backtracking linesearch to satisfy the sample estimate (13)
  Clear $\mathcal{D}$ and $\mathcal{D}_\Delta$
end for

Remark 4. The comparison between LCPO and LSAC in terms of data efficiency is to be made. As shown in the pseudo-code Algorithm 1, LSAC updates multiple steps after a trajectory has been sampled. It is even possible to update at every step after observing a new state–action pair, though this is not adopted in LSAC. In comparison, LCPO only proceeds one step after each iteration of observing the trajectory. This is due to the nature of on-policy algorithms: after one update of θ, the collected data become off-policy data, i.e. the data generated by a different controller, and cannot be used by the on-policy algorithms anymore. Thus, at the end of each iteration, LCPO needs to empty the set of transition pairs $\mathcal{D}$ and $\mathcal{D}_\Delta$. On the contrary, LSAC repeatedly makes use of the data collected by different controllers. As a result, LSAC possesses better data efficiency than LCPO.



(a) CartPole (b) Point-Circle (c) Ant-Safe (d) HalfCheetah-Safe (e) Quadrotor

Fig. 3. Demonstration of environments screenshots.

Remark 5. Another distinguishing difference between LCPO and LSAC is the safety guarantee during training, which is a key consequence between off- and on-policy algorithms. LCPO is implemented based on the trust-region optimization, solving an approximated optimization problem locally and assuring that every post-update policy is safe. If the initial policy is safe, the safety will be ensured during learning. Otherwise, LCPO will try to find a safe policy first using the recovery update in Eq. (19). In comparison, LSAC does not hold this property but only can assure safety at the end of the training. From a practical point of view, LSAC is suitable for training a safe policy in a virtual environment and then deployed to a real system; LCPO is directly applicable for online training. Furthermore, a potential choice is to combine the strengths of LSAC and LCPO: use LSAC to learn an initial policy in the virtual environment then transfer to LCPO for further online training.

4.4. Further discussion

Before proceeding to show the effectiveness of LSAC and LCPO for various environments, we would like to further discuss the possible adverse effects of sample-based approximation and UUB analysis.

First, in both LSAC and LCPO, the safety constraint is evaluated using a sample estimate of the inequality (3), which unavoidably introduces estimation error unless the number of samples is infinite. Therefore, a possible research direction is to establish a quantitative relation between the reliability of the safety guarantee and the number of samples. Furthermore, stronger safety constraints can be introduced by considering the estimation error.

Second, as both LSAC and LCPO rely on the sample-based gradient approximation in controller training, the learning algorithms may take update steps in undesirable directions temporarily due to some approximation errors (such as reducing the return or violating the safety constraints). Nevertheless, this effect can be modified by choosing reasonable hyperparameters such as learning rate and batch size such that the undesirable update does not affect the convergence of the learning process.

Finally, different from the classical model-based controller design methods, the proposed method does not need a dynamic model to design the controller. Instead, it is model-free which means only the data from the trial and error will be used to learn the controller until a satisfactory one can be found. This process is undoubtedly time-consuming and hardly applicable to a system in the real world. Thus in practice, it is favorable to train the controller virtually first, then transfer and fine-tune the controller in the real world (Harrison et al., 2020; Tan et al., 2018; Yu, Kumar, Turk, & Liu, 2019).

5. Experiments

In our experiments, we would like to address the following questions:

- How does our approach perform compared with other existing safe RL algorithms for CMDP tasks?
- Can the algorithms converge with different parameter initializations and learn safe control policies in the presence of function approximation error?
- Does the policy with UUB guarantee ensure the system to recover to the inner set under perturbation and disturbance?
- Under what circumstances the Lyapunov-based safe RL algorithms may fail?

5.1. Experiment setups

In the following, five CMDP tasks are set up ranging from simulated robot locomotion in the MuJoCo simulator (Todorov, Erez, & Tassa, 2012) to motion planning of a simulated quadrotor (see Fig. 3): (i) Cartpole-Safe: The agent is rewarded for sustaining the pole vertically at a target position, while limited in a safe region; (ii & iii) HalfCheetah-Safe (Chow et al., 2019) and Ant-safe: The agent is rewarded for running while the speed is limited for safety; (iv) Point-Circle (Achiam et al., 2017): The agent is rewarded for following a circular trajectory while limited to stay in a safe region $|x| < \bar{x}$; (v) Quadrotor-Safe: The agent is rewarded for tracking a spiral trajectory while constrained to stay under a certain altitude. The details of the control tasks are described in the following.

5.1.1. Cartpole-safe

In this experiment, the agent is expected to sustain the pole vertically at a target position $x = 6$. This is a modified version of CartPole with continuous action space (Brockman et al., 2016). The action is the horizontal force F on the cart, $F \in [-20, 20]$. While tracking the target position, the safety constraint $x < 4$ needs to be ensured. The state space of x is $[0, 10]$ and the episodes end if the cart moves outside this region. The agent is initialized randomly as $x \in [0, 4]$. The reward function is given by $r = 20\times\text{sign}(1-|x-6|)\times(1-|x-6|)^2+\text{sign}(\frac{\pi/2-|\theta|}{\pi/2})\times(\frac{\pi/2-|\theta|}{\pi/2})^2$. The constraint function is given by $c = \max(|x| - 3.2, 0)^2/5$. The episodes are set at a length of 250.

It should be noted that the target position is outside the safe region. As a result, the optimal safe behavior is to sustain the pole vertically at the edge of the safe region. This setup is deliberate in order to test whether safety can outweigh the return under the control of the trained agents.

5.1.2. Point-circle

This task is borrowed from Achiam et al. (2017), where the agent controls a mass point to run in a wide circle but limited to stay in a safe region. The agent is initialized at $(0, 0)$. The agent is rewarded for moving the mass point counter-clockwise along a circle of radius 15m, $r = \frac{-y\times v_x+x\times v_y}{1+|\sqrt{(x^2+y^2)}-15|}$. We constrain the x-axis position to be less than 2.4, and the constraint function is: $c = \max(|x| - 2.4, 0)$. The episodes are set at a length of 65.

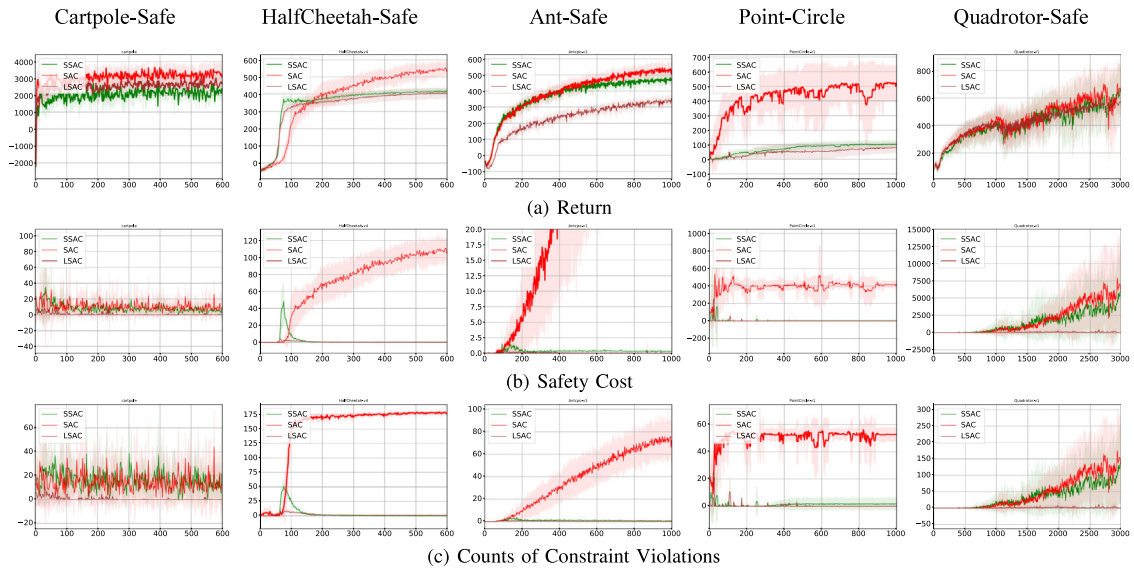


Fig. 4. The average performance of SAC (brown), LSAC (red), SSAC where the shaded areas show the 1-SD confidence intervals over 10 random seeds. The X-axis indicates the total time steps in thousands. Return and the sum of constraint functions are shown in the first and the second row respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1.3. Halfcheetah-safe

HalfCheetah-Safe is taken from Chow et al. (2019). The task is to control a HalfCheetah (a 2-legged simulated robot with 17 degrees of freedom) to run as fast as possible with maximum velocity limited for safety. The reward function is $r = v - 0.1 \times \|a\|^2$ where v is the forward speed of the HalfCheetah and a is the control input. The constraint function is $c = \max(|v| - 2.7, 0)^2$. The episodes are set at a length of 200.

5.1.4. Ant-Safe

In Ant-Safe, the agent controls an Ant (a quadrupedal simulated robot) to run as fast as possible while satisfying the safety constraint on forwarding speed, $v < 2.7$. The reward function is $r = v - 0.5 * \sum a^2 - cost_{\text{contact}} + 1$ where $cost_{\text{contact}}$ equals -1 if the robot touches the ground and 0 otherwise. The constraint function $c = \max(|v| - 2.7, 0)^2$. The episodes are set at a length of 200. Reward function and other settings are the same with the default setting in the OpenAI Gym. The episodes are set at a length of 200.

5.1.5. Quadrotor-safe

This task is to control a drone to track a spiral trajectory in 3D space. In the meantime, the altitude is limited to be under a threshold to avoid hitting the ceiling. The simulation environment is modified from an open-source Crazyflie simulator Matlab code[3] into OpenAI Gym's structure. The simulator has three parts, the CrazyFlies model parameters, the PD controller, and the dynamic equations. The control proceeds as follows: The quadrotor simulator outputs the next state of the quadrotor given the force, torques, and the current state. The controller maps the observation of the current state to the desired next step and the desired state equals the current state adds the desired step. Last, the PD controller converts the current state and desired state to force and torques, and the loop continues. The state includes the quadrotor's position, velocities, attitude, angular velocities and reference trajectory, $s_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}, p, q, r, \dot{p}, \dot{q}, \dot{r}, x_{target}, y_{target}, z_{target}]$. The policy outputs desired relative changes in position and velocity from current state, $a_t = [\Delta x, \Delta y, \Delta z, \Delta \dot{x}, \Delta \dot{y}, \Delta \dot{z}]$. And an episode ends when the current position is too far from the reference trajectory. For this task, the reward is $r = -\|d\| + 1$ where d is the distance to the reference trajectory. The constraint function is $c = 100 * \max(|z| - 0.4, 0)$. For this experiment, the episodes are set at a length of 2000.

5.2. Evaluation and comparison analysis

In this part, the performance of LSAC on the CMDP tasks is evaluated and compared with the existing safe RL algorithm, safe soft actor–critic (SSAC) (Chow et al., 2019) with optimized hyperparameters. SSAC is a safety constrained variant of the original algorithms through the Lagrange relaxation procedure (Bertsekas, 1997). The soft actor–critic (SAC) (Haarnoja, Zhou, Hartikainen, et al., 2018) is also included to show that the optimal behaviors are generally unsafe in the CMDP setting. In the meantime, the performance of LCPO is compared with constrained policy optimization (LCPO) (Achiam et al., 2017), a state-of-the-art trust-region method for CMDP tasks. Since the trust-region methods (LCPO, CPO) require large batch sizes and need more samples than the gradient-based methods to reach convergence, thus, these two classes of methods are compared separately. The sum of the constraint function of episodes (safety cost) and the counts of constraint violations is used as the measure of safety and the return is used to evaluate performance. The goal is to suppress these measures to zero while maximizing the return, i.e., the expectation of constraint function at each time step and the expectation of constraint violations are required to be zero.

5.2.1. Algorithm hyperparameters

The hyperparameters for the LSAC and LCPO can be found in Appendix.

For LSAC, there are three networks: the policy network, the Q network, and the Lyapunov network. For the policy network, we use a fully-connected MLP with two hidden layers of 256 units, outputting the mean and standard deviations of a Gaussian distribution. For the Q network and the Lyapunov network, we use a fully-connected MLP with two hidden layers of 256 units, outputting the Q value and the Lyapunov value. All the hidden layers use Relu activation function and we adopt the same invertible squashing function technique as Haarnoja, Zhou, Hartikainen,

[3] https://github.com/yrlu/quadrotor

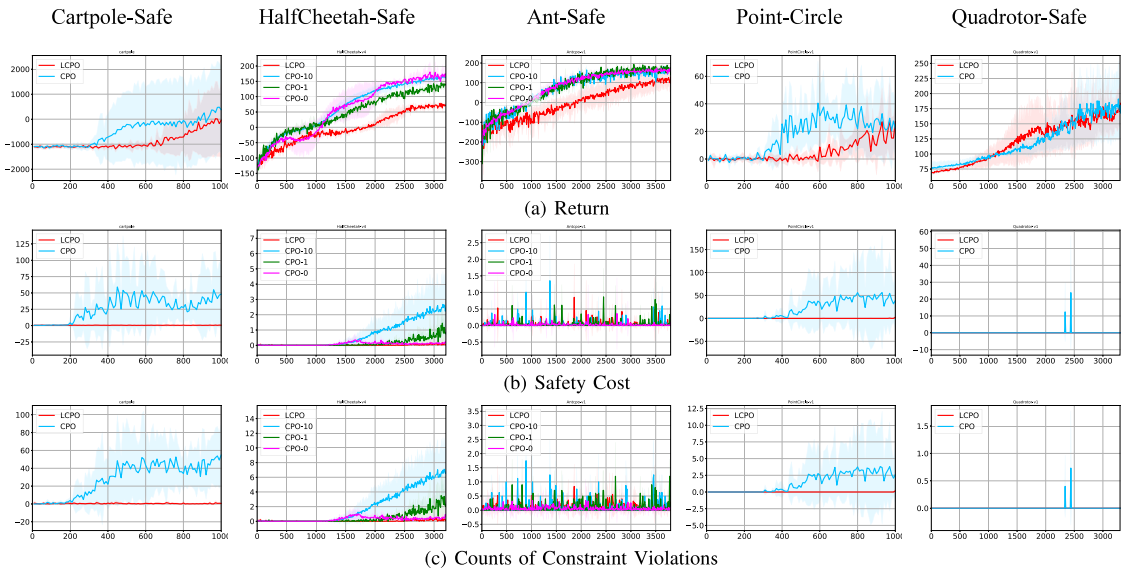


Fig. 5. The average performance of CPO and LCPO, where the shaded areas show the 1-SD confidence intervals over 5 random seeds. The X-axis indicates the total time steps in thousands. The return (a), safety cost (b), and counts on constraint violations (c) are shown in each row respectively.

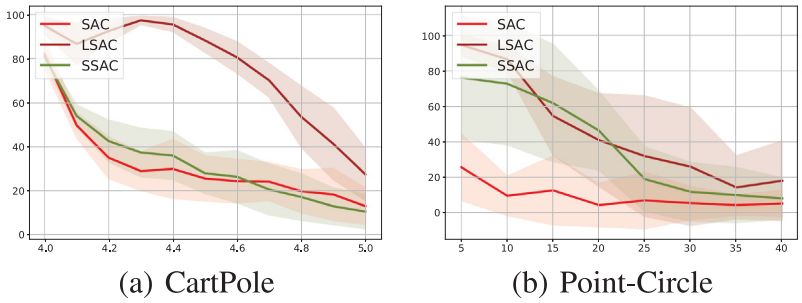


Fig. 6. The recovery rate of agents trained by LSAC, SSAC, and SAC in the presence of impulsive force F with different magnitudes in Cartpole-Safe (a) and Point-Circle (b). The x-axis denotes the magnitude of an instant force. The policies with 10 different initialization are evaluated equally for 500 episodes under each magnitude of a force. The line indicates the average recovery rate of these policies and the shadowed region shows the 1-SD confidence interval.

et al. (2018) to the output layer of the policy network. The hyperparameters can found in Fig. 8.

For LCPO, there are three networks: the policy network, the value network, and the Lyapunov network. For the policy network, it has two hidden layers of sizes (64, 32) with tanh activation functions, outputting the mean and standard deviations of a Gaussian distribution. For the value network, it has two hidden layers of sizes (256, 128) with Relu activation functions, outputting the value. And for the Lyapunov network, it has two hidden layers of sizes (256, 256) with Relu activation functions, outputting the Lyapunov value. The hyperparameters can found in Fig. 9.

The implementation of all the algorithms is based on TensorFlow (Abadi et al., 2016).

5.2.2. Comparison with SSAC

As demonstrated in Fig. 4, though SSAC can maintain state constraint satisfaction on some of the tasks despite some major violations during training (see HalfCheetah-Safe and Ant-Safe), on other tasks it fails to find safe policy both at convergence or during training. On the other hand, our method (LSAC) quickly converges to safe policies across all the tasks (Fig. 4(a)–(e)) while maintaining reasonable return. Besides, LSAC maintains low safety costs (almost zero) and constraint violation times throughout the training with low variance in all the tasks, even though all the policies are randomly initialized.

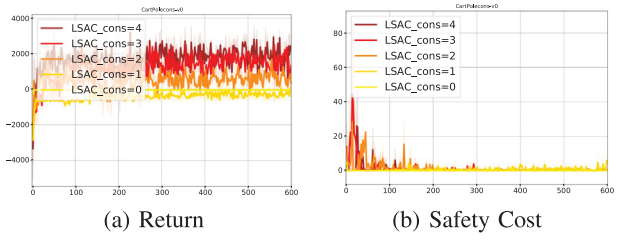


Fig. 7. The average return and constraint function of LSAC in Cartpole-Safe with different constraints $\bar{x}$.

5.2.3. Comparison with CPO

CPO and LCPO are compared on the safe environments as used for the off-policy methods. As shown in Fig. 5, LCPO performs stably in terms of both safety cost and constraint violation counts in all of the environments, approximately achieving zero constraint violation during training and at convergence. In terms of return, LCPO performs comparably with CPO on Point-Circle and Quadrotor-Safe, while in other tasks CPO performs slightly better than LCPO, however, at a cost that CPO violates the safety constraints a lot, as shown in Fig. 5(c). In CartPole and Point-Circle, CPO is not safe both during training and at convergence, while in other experiments it may violate constraints during training. As discussed in Garcıa and Fernández (2015), the discounted-sum

Hyperparameters	Point-Circle	Ant	HalfCheetah	Quadrotor	CartPole-Safe
Lyapunov candidate function	Cost	Value	Value	Value	Value
Minibatch size	256	256	256	256	256
Actor learning rate	1e-4	1e-4	1e-4	1e-4	1e-4
Critic learning rate	3e-4	3e-4	3e-4	3e-4	3e-4
Lyapunove learning rate	3e-4	3e-4	3e-4	3e-4	3e-4
Target entropy	-2	-8	-6	-6	-1
Target smoothing coefficient(τ)	0.005	0.005	0.005	0.005	0.005
Discount(γ)	0.99	0.99	0.99	0.99	0.99
α_3	0.8	1	1	0.8	1

Fig. 8. LSAC Hyperparameters.

Hyperparameters	Ant	HalfCheetah
Lyapunov candidate function	Cost	Cost
Batch size	10000	10000
Critic learning rate	1e-4	1e-4
Lyapunov learning rate	1e-4	1e-4
Rollout length	500	500
Discount(γ)	0.99	0.99
α_3	0.2	0.2
maximum constraint value d	10	10
Stepsize	0.01	0.01
Safety discount	0.5	0.5

Fig. 9. LCPO Hyperparameters.

constrained methods suffer from the tricky tuning of the safety threshold to handle the state constrained problems. To show this, we tested different threshold settings for CPO on Halfcheetah and Ant. In the HalfCheetah-Safe task, CPO swings due to the different settings of the safety threshold. CPO either fails to find the feasible policy or reaches convergence after being unsafe for a long period (more than 500,000 steps). One more thing to note is that CPO requires additional cost shaping, by having a network evaluating the chance of constraint violation to achieve the best performance, while our approach does not need such techniques.

5.3. Recoverability to inner set

As shown in Figs. 1 and 2, UUB assures that the system can recover to the inner set when it is wrongly initialized or accidentally disturbed and appears in the edge set. Now we evaluate the recoverability of the agents trained by LSAC and baselines when interfered by an unseen exogenous disturbance in CartPole-Safe and Point-Circle. Impulsive forces are implemented on the robot to push it outside the inner set and see whether it can recover to the inner set. Their performance is measured by the recovery rate, i.e., the probability of successful recovery after impulsive disturbances. Under different impulse magnitudes, the policies trained by LSAC and baselines are evaluated 500 times, and the results are shown in Fig. 6.

As observed in the figure, LSAC achieves the best performance in terms of recoverability when interfered by forces with different magnitudes, while SSAC is less possible to recover under the same circumstances. Note that the agent cannot recover from arbitrarily large disturbances since only local UUB is assured.

5.4. Ablation on constraints

We want to test how does the proposed algorithm trade-off between performance and safety. In CartPole, the safety constraint is contradictory to the performance, i.e. being safer will decrease the return. Thus, we gradually strengthen the constraint and see how does LSAC reacts and when does it fail to find a feasible policy. Specifically, the size of inner set $\{x|x \in [0, \bar{x}]\}$ is reduced by assigning $\bar{x}$ with $\{0, 1, 2, 3, 4\}$. The performance of LSAC in CartPole-Safe with different sizes of the inner set is compared, see Fig. 7. As $\bar{x}$ approaches zero, the average return of LSAC also decreases while safety is maintained. However, when $\bar{x} = 0$ and only the origin is safe, the agent fails to sustain the pole and dies almost immediately. This implies that LSAC may fail in the case that safety constraints are too strong.

6. Conclusion

In this paper, a novel data-based approach for analyzing the uniformly ultimate bounded stability of a learning control system is proposed. Based on the theoretical results, two model-free reinforcement learning algorithms are developed, i.e., Lyapunov safe actor–critic and Lyapunov constrained policy optimization. The proposed algorithms are evaluated on a series of robotic continuous control tasks with safety constraints. In comparison with the existing RL algorithms, the proposed method can reliably assure safety in various challenging continuous control tasks. As a qualitative evaluation of stability, our method shows impressive resilience even in the presence of external disturbances.

For future work, we would like to explore the following directions: (i) automating the design of constraint function; (ii) extending the Lyapunov-based approach to model-based setting; (iii) optimizing the performance upper bound while assuring the stability guarantee.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation* (pp. 265–283).

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. In *Proceedings of the 34th international conference on machine learning-volume 70* (pp. 22–31). JMLR. org.

Altman, E. (1999). *Constrained markov decision processes (vol. 7)*. CRC Press.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in ai safety. arXiv preprint arXiv:1606.06565.

Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems* (pp. 908–918).

Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society*, *48*(3), 334.

Bhandari, J., Russo, D., & Singal, R. (2018). A finite time analysis of temporal difference learning with linear function approximation. arXiv preprint arXiv:1806.02450.

Boukas, E., & Liu, Z. (2000). Robust stability and h/sub/spl infin//control of discrete-time jump linear systems with time-delay: An lmi approach. In *Decision and control, 2000. Proceedings of the 39th IEEE conference on (vol. 2)* (pp. 1527–1532). IEEE.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

Busoniu, L., de Bruin, T., Tolic, D., Kober, J., & Palunko, I. (2018). Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*.

Cheng, R., Orosz, G., Murray, R. M., & Burdick, J. W. (2019). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. arXiv preprint arXiv:1903.08792.

Choi, J., Castañeda, F., Tomlin, C. J., & Sreenath, K. (2020). Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. arXiv preprint arXiv:2004.07584.

Chow, Y., Nachum, O., Duenez-Guzman, E., & Ghavamzadeh, M. (2018). A lyapunov-based approach to safe reinforcement learning. arXiv preprint arXiv:1805.07708.

Chow, Y., Nachum, O., Faust, A., Ghavamzadeh, M., & Duenez-Guzman, E. (2019). Lyapunov-based safe policy optimization for continuous control. arXiv preprint arXiv:1901.10031.

Corless, M., & Leitmann, G. (1981). Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems. *IEEE Transactions on Automatic Control*, *26*(5), 1139–1144.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587–1596).

Garcıa, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, *16*(1), 1437–1480.

Garcia, C. E., Prett, D. M., & Morari, M. (1989). Model predictive control: Theory and practice—a survey. *Automatica*, *25*(3), 335–348.

Gorges, D. (2017). Relations between model predictive control and reinforcement learning. *IFAC-PapersOnLine*, *50*(1), 4920–4928.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning* (pp. 1861–1870).

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., et al. (2018). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

Han, M., Tian, Y., Zhang, L., Wang, J., & Pan, W. (2019). H_∞ model-free reinforcement learning with robust stability guarantee. arXiv preprint arXiv:1911.02875.

Han, M., Zhang, L., Wang, J., & Pan, W. (2020). Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*.

Harrison, J., Garg, A., Ivanovic, B., Zhu, Y., Savarese, S., Fei-Fei, L., et al. (2020). Adapt: Zero-shot adaptive policy transfer for stochastic dynamical systems. In *Robotics research* (pp. 437–453). Springer.

Huang, J., Han, Z., Cai, X., & Liu, L. (2011). Uniformly ultimately bounded tracking control of linear differential inclusions with stochastic disturbance. *Mathematics and Computers in Simulation*, *81*(12), 2662–2672.

Jain, A. K., & Bhasin, S. (2017). Uniformly ultimately bounded tracking for uncertain euler-lagrange systems with unknown time-varying input delay. *IFAC-PapersOnLine*, *50*(1), 1415–1420.

Korda, N., & La, P. (2015). On td (0) with function approximation: Concentration bounds and a centered variant with exponential convergence. In *International conference on machine learning*. (pp. 626–634).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Luo, B., Yang, Y., Liu, D., & Wu, H.-N. (2019). Event-triggered optimal control with performance guarantees using adaptive dynamic programming. *IEEE Transactions on Neural Networks and Learning Systems*.

Ma, X., Zhang, Q., Xia, L., Zhou, Z., Yang, J., & Zhao, Q. (2020). Distributional soft actor critic for risk sensitive learning. arXiv preprint arXiv:2004.14547.

Mayne, D. Q. (2001). Control of constrained dynamic systems. *European Journal of Control*, *7*(2–3), 87–99.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), 789–814.

Moldovan, T. M., & Abbeel, P. (2012). Safe exploration in markov decision processes. In *ICML*.

Mu, C., Ni, Z., Sun, C., & He, H. (2016). Data-driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems. *IEEE Transactions on Cybernetics*, *47*(6), 1460–1470.

Ostafew, C. J., Schoellig, A. P., & Barfoot, T. D. (2016). Robust constrained learning-based nmpc enabling reliable mobile robot path tracking. *International Journal of Robotics Research*, *35*(13), 1547–1563.

Royden, H. L. (1968). *Real analysis*. Krishna Prakashan Media.

Sastry, S. (2013). *Nonlinear systems: Analysis, stability, and control (vol. 10)*. Springer Science & Business Media.

Saunders, W., Sastry, G., Stuhlmueller, A., & Evans, O. (2018). Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th international conference on autonomous agents and multiAgent systems* (pp. 2067–2069). International Foundation for Autonomous Agents and Multiagent Systems.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control–a review. *Journal of Process Control*, *19*(5), 723–731.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*. (pp. 1889–1897).

Shih, P., Kaul, B., Jagannathan, S., & Drallmeier, J. (2007). Near optimal output-feedback control of nonlinear discrete-time systems in nonstrict feedback form with application to engines. In *2007 international joint conference on neural networks*. IEEE, 396–401.

Slotine, J.-J. E., Li, W., et al. (1991). *Applied nonlinear control (vol. 199)*. NJ: Prentice Hall Englewood Cliffs.

Sutton, R. S., Barto, A. G., & Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, *12*(2), 19–22.

Sutton, R. S., Maei, H. R., & Szepesvári, C. (2009). A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in neural information processing systems* (pp. 1609–1616).

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., et al. (2018). Sim-to-real: Learning agile locomotion for quadruped robots. arXiv preprint arXiv:1804.10332.

Thananjeyan, B., Balakrishna, A., Rosolia, U., Li, F., McAllister, R., Gonzalez, J. E., et al. (2020). Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *IEEE Robotics and Automation Letters*, *5*(2), 3612–3619.

Thowsen, A. (1983). Uniform ultimate boundedness of the solutions of uncertain dynamic delay systems with state-dependent and memoryless feedback control. *International Journal of Control*, *37*(5), 1135–1143.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033). IEEE.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Vidyasagar, M. (2002). *Nonlinear systems analysis (vol. 42)*. SIAM.

Wang, A., Liao, X., & Dong, T. (2018). Event-driven optimal control for uncertain nonlinear systems with external disturbance via adaptive dynamic programming. *Neurocomputing*, *281*, 188–195.

Yang, X., Liu, D., Wei, Q., & Wang, D. (2016). Guaranteed cost neural tracking control for a class of uncertain nonlinear systems using adaptive dynamic programming. *Neurocomputing*, *198*, 80–90.

Yu, W., Kumar, V. C., Turk, G., & Liu, C. K. (2019). Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3503–3510). IEEE.

Zanon, M., & Gros, S. (2020). Safe reinforcement learning using robust mpc. *IEEE Transactions on Automatic Control*.

Zou, S., Xu, T., & Liang, Y. (2019). Finite-sample analysis for sarsa with linear function approximation. In *Advances in neural information processing systems* (pp. 8665–8675).

Minghao Han received the B.S. degree in the Department of Control Science and Engineering from Harbin Institution of Technology, China, in 2017.

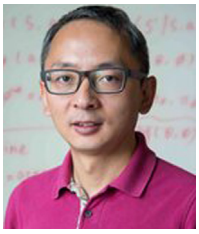
In 2016, he was a visiting student in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, from January to May. In 2017, he started pursuing the Ph.D. degree in control science and engineering at the Research Institute of Intelligence Control and Systems, Harbin Institute of Technology. From January to June in 2020, he was a visiting student at the Department of Cognitive Robotics, Delft University of Technology. He is now a visiting scholar at the

Soft Robotics Lab, ETH Zurich since January 2021. His research interests include control theory, reinforcement learning, and their application in robotics.

Yuan Tian received his B.S. degree from Beijing Institute of Technology, Beijing, China in 2017 and received his M.S. degree from TU Delft in 2019. He is currently a Ph.D. student in ETH Zurich. He is focusing on reinforcement learning applications including prescriptive maintenance, AutoML, portfolio management, etc.

Lixian Zhang received the Ph.D. degree in control science and engineering from Harbin Institute of Technology, Harbin, China, in 2006. From January 2007 to September 2008, he was a Postdoctoral Fellow in the Department Mechanical Engineering at the Ecole Polytechnique de Montreal, Canada. He was a Visiting Professor at the Process Systems Engineering Laboratory, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, from February 2012 to March 2013. Since January 2009, he has been with Harbin Institute of Technology, where he is currently full professor and vice director in the Research Institute of Intelligent Control and Systems, School of Astronautics. His research interests include nondeterministic switched systems, networked control systems, model predictive control and their applications.

Prof. Zhang serves as an Associate Editor for various peer-reviewed journals, including IEEE Transactions on Automatic Control, IEEE Transactions on Cybernetics, etc., and was a leading Guest Editor for the Special Section on Advances in Theories and Industrial Applications of Networked Control Systems in the IEEE Transactions on Industrial Informatics. He was named to the list of Thomson Reuters Highly Cited Researchers in 2014–2019. He is a Fellow of IEEE.



Jun Wang received the Ph.D. degree in Computer Science from Delft University of Technology, the Netherlands. He is currently a Chair Professor of Computer Science, University College London, and Founding Director of M.Sc. Web Science and Big Data Analytics. He is also Co-founder and Chief Scientist in MediaGamma Ltd, a UCL start-up company focusing on AI for intelligent audience decision making. His main research interests are in the areas of AI and intelligent systems, including (multiagent) reinforcement learning, deep generative models, and their diverse applications on information retrieval, recommender systems and personalization, data mining, smart cities, bot planning, computational advertising etc.

Prof. Wang was a recipient of the Beyond Search-Semantic Computing and Internet Economics award by Microsoft Research and also received Yahoo! FREP Faculty award. He has served as an Area Chair in ACM CIKM and ACM SIGIR. His recent service includes co-chair of Artificial Intelligence, Semantics, and Dialog in ACM SIGIR 2018. MediaGamma has received the UCLB One-to-Watch award 2016. He was a technical advisor for startups such as Last.Fm, Passiv Systems, Massive Analytic, Context Scout, and Polecat, and had various projects with Huawei, BT, Microsoft, Yahoo!, Alibaba, Didi etc.

Wei Pan received the Ph.D. degree in Bioengineering from Imperial College London. He is currently an Assistant Professor at Department of Cognitive Robotics, Delft University of Technology. Until May 2018, he was a Project Leader at DJI, Shenzhen, China, responsible for machine learning research for DJI drones and AI accelerator. He is the recipient of Dorothy Hodgkin's Postgraduate Awards, Microsoft Research Ph.D. Scholarship and Chinese Government Award for Outstanding Students Abroad. Dr. Pan serves as an Area Chair of CoRL, Associate Editor of IROS. His research interests include machine learning and control theory and robotics.