IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 2 August 2023, accepted 15 August 2023, date of publication 17 August 2023, date of current version 24 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3306070

RESEARCH ARTICLE

An Actor-Critic Framework for Online Control With Environment Stability Guarantee

PAVEL OSINENKO, GRIGORY YAREMENKO, GEORGIY MALANIYA, AND ANTON BOLYCHEV
Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
Corresponding author: Pavel Osinenko (p.osinenko@gmail.com)

ABSTRACT Online actor-critic reinforcement learning is concerned with training an agent on-the-fly via dynamic interaction with the environment. Due to the specifics of the application, it is not generally possible to perform long pre-training, as it is commonly done in off-line, tabular or Monte-Carlo mode. Such applications may be found more frequently in industry, rather than in pure digital fields, such as cloud services, video games, database management, etc., where reinforcement learning has been demonstrating success. Stability of the closed-loop of the agent plus the environment is a major challenge here, and not only in terms of the environment safety and integrity, but also in terms of sparing resources on failed training episodes. In this paper, we tackle the problem of environment stability under an actor-critic reinforcement learning agent by integration of the Lyapunov stability theory tools. Under the presented approach, the closed-loop stability is secured in all episodes without pre-training. It was observed in a case study with a mobile robot that the suggested agent could always successfully achieve the control goal, while significantly reducing the cost. While many approaches may be exploited for mobile robot control, we suggest that the experiments showed the promising potential of actor-critic reinforcement learning agents based on Lyapunov-like constraints. The presented methodology may be utilized in safety-critical, industrial applications where stability is necessary.

INDEX TERMS Control, stabilization, reinforcement learning, Lyapunov function.

I. INTRODUCTION

Reinforcement learning is an optimal control method that uses imitates leaving beings in environments behaving upon received rewards or punishment [1], [2], [3], [4]. Its applications range from robotics [5], [6], [7], [8], [9] to games such as Go, chess, shogi (also known as Japanese chess) [10], [11], and even complex video games such as StarCraft II [12]. Reinforcement learning bears data-driven character based on experience to infer an optimal policy of actions. The policy seeks to optimize an objective over a (possibly infinite) period of time.

Commonly, reinforcement learning implies extensive training of agents to learn an optimal policy sufficiently precisely. This can be done, for instance, via so-called roll-outs of action-state pairs up to a possibly infinite horizon. Such is the case in some policy gradient methods (see, e. g., [13], [14], [15]). It may in turn mean repeated episodes of environment Monte-Carlo runs [16], [17], [18]. In tabular, dynamic-programming-like reinforcement learning, each value or policy update needs to be done over each node of a mesh in a (compact domain of) state space (see, e. g., [19, Section 3] and [20], [21], [22]). Temporal difference methods, in contrast, can operate without full-episode runs [23], [24], [25], [26], [27], [28], [29].

A. NOTATION

The following table summarizes the notation used throughout the article.

$\mathbb{R}_{\geq 0}$	Set of nonnegative reals
$\mathbb{X}$	State space $\subseteq \mathbb{R}^n$
$\mathbb{U}$	Action space $\subseteq \mathbb{R}^m$
$\mathbb{P}[\cdot]$	Probability measure
$\mathcal{B}_s$	Closed ball of radius s centered at the origin

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

$\mathrm{Lip}_h(s)$	Lipschitz constant of $h(\cdot)$ in $\mathcal{B}_s$ (just Lip_h in global case)
$\|\bullet\|_{\mathrm{op}}$	Operator norm of a matrix
$\mathfrak{J}_{\mathrm{critic}}$	Critic loss
$\mathfrak{J}_{\mathrm{actor}}$	Actor loss
$\hat{J}$	Critic model
J^*	Objective optimum

B. WORK STRUCTURE

The current work is structured as

Section I-C	General control problem of reinforcement learning
Section I-D	Stabilizability assumptions
Section II	Related work in stabilizing reinforcement learning
Section III	Summary of contribution
Section IV	New approach
Section V	Experimental results
Appendix	Proofs, auxiliary materials

C. CONTROL PROBLEM OF REINFORCEMENT LEARNING

Commonly, the starting point in reinforcement learning is the following infinite-horizon optimal control and/or decision problem:

$$\min_{\rho \in \mathcal{U}} J^\rho(x) = \min_{\rho \in \mathcal{U}} \mathbb{E}_{U_t \sim \rho} \left[\int_0^\infty e^{-\gamma t} r(X_t, U_t) \, \mathrm{d}t | X_0 = x \right], \tag{1}$$

where $X_t \in \mathbb{X}$ at a time $t \in \mathbb{T} := \mathbb{R}_{\geq 0}$ is the environment's state with values in the state-space $\mathbb{X}$, r is the running objective (cost or reward in minimization, respectively, maximization problems) function or rate, γ is the discount factor, ρ is the agent's policy of some function class $\mathcal{U}$. The running objective may be taken as a random variable R_t whose probability distribution depends on the state and action. It is a function hereafter for simplicity. The policy may be taken as a probability distribution or a plain function (hence a Markov policy). We will assume the latter from now on. Next, we consider the agent-environment[1] loop dynamics to be modeled via the following transition law:

$$\mathrm{d}X_t = f(X_t, U_t)\,\mathrm{d}t + \sigma(X_t, U_t)\,\mathrm{d}Z_t, \quad \mathbb{P}[X_0 = \mathrm{const}] = 1, \tag{2}$$

where an unknown noise $Z_t \in \mathbb{R}^d$ is modeled by an arbitrary Lipschitz-continuous random process with a fixed Lipschitz constant Lip_Z, and $\sigma : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^{n \times d}$ is a bounded mixing matrix, i. e., $\|\sigma(x, u)\|_{\mathrm{op}} \leq \sigma_{\max}, \sigma_{\max} > 0$. Here, $X_0 = \mathrm{const}$ implies that X_0 has a degenerate distribution.

In the following, when claims are made about sections of random processes, the respective conditions are meant to hold almost surely. We assume that f and σ are locally Lipschitz continuous with respect to X_t and U_t. Notice requiring Z_t to be Lipschitz-continuous with $\mathrm{Lip}_Z = \mathrm{const}$ is equivalent to asserting:

$$\mathrm{d}Z_t \equiv H_t \,\mathrm{d}t, \text{ where } \operatorname*{ess\,sup}_{t \in \mathbb{T}} \|H_t\| \leq \mathrm{const}. \tag{3}$$

Thus, $\mathrm{d}Z_t$ corresponds to an arbitrary perturbation of magnitude no greater than Lip_Z.

For the problem (1), one can state an important recursive property of the objective optimum $J^*(x)$ in the form of the Hamilton-Jacobi-Bellman (HJB) equation as follows:

$$\min_{u \in \mathbb{U}} \{\mathcal{A}^u J^*(x) + r(x, u) - \gamma J^*(x)\} = 0, \quad \forall x \in \mathbb{X}, \tag{4}$$

where $\mathcal{A}^u J^*(x) := \nabla J^*(x)^T \left(f(x, u) + \sigma^T(x, u) H_t\right)$. The common approaches to (1) are dynamic programming [2], [30] and model-predictive control [31], [32], [33], [34]. The latter cuts the infinite horizon to some finite value $T > 0$ thus considering effectively a finite-time optimal control problem. Dynamic programming aims directly at the HJB (4) and solves it iteratively over a mesh in the state space $\mathbb{X}$ and thus belongs to the category of tabular methods. The most significant problem with such a discretization is the curse of dimensionality, since the number of nodes in the said mesh grows exponentially with the dimension of the state space. Evidently, dynamic programming is in general only applicable when the state-space is compact. Furthermore, state-space discretization should be fine enough to avoid undesirable effects that may lead to a loss of stability of the agent-environment closed loop.

Reinforcement learning essentially approximates the optimum objective J^* via a (deep) neural network. The core problem with such an approach is that one cannot know how well the chosen neural network topology is capable of approximating the optimum objective. Although it is known that the extremizer (the optimal policy) has nice properties, e. g., it keeps the environment stable, an extremizer resulting from an approximate optimum objective has in general no such guarantees.

We consider the policy to be implemented in a digital, sampled manner, i. e., the state is measured every δ seconds and an action is computed at every $k\delta$, $k \in \mathbb{Z}_{\geq 0}$ seconds and held for δ seconds accordingly:

Definition 1 (Sampled Policy): The agent-environment (2) is said to obey a sampled policy $\rho : \mathbb{X} \times \mathbb{T} \to \mathbb{U}$ with a sampling time δ if:

$$U_t \equiv \rho(X_{t \,-\, t \bmod \delta}). \tag{5}$$

The sampled setting reflects the discrete-time limitations of reinforcement learning. Although ways of training continuous-time policies are known [35], [36], a digital device, e. g., a microcontroller or a computer, is required anyway to implement the policy, hence the above description. Thus, we can formulate the task of reinforcement learning as finding a sampled policy ρ^* that minimizes the expected total

[1] In classical terminology, "controller-system" loop.

cost

$$J^\rho = \mathbb{E}\left[\int_0^\infty r(X_t, U_t)\,\mathrm{d}t\right] \text{ under } U_t \equiv \rho(X_{t\,-\,t \bmod \delta}). \quad (6)$$

The discount factor was omitted for simplicity as it is non-essential for future derivations. Speaking of the environment, we may require such a policy to achieve the property that the state of the environment be driven into a neighborhood $\mathbb{O}$ of the origin:

$$\mathbb{P}\left[\lim_{t\to\infty} \inf_{s\in\mathbb{O}} \|X_t - s\| = 0\right] = 1. \quad (7)$$

D. STABILIZABILITY

In should be noted that many reinforcement learning analyses require that (2) be stabilizable [3], [37], [38], [39], [40]. We consider here the following stabilizability property.

Definition 2 (Asymptotic Stabilizability Under Sampled Policy): Let $\mathbb{O}$ (an *attractor*) be a compact neighborhood of the origin and let $\mathbb{B}$ (a *basin*) be a compact set containing $\mathbb{O}$. The origin of the environment is said to be stabilizable under a sample policy if for any such $\mathbb{O}$ and $\mathbb{B}$, there exists $\eta : \mathbb{X} \to \mathbb{U}$ (a *stabilizer*) that uniformly drives the states in $\mathbb{B}$ to $\mathbb{O}$ given that $\delta > 0, \sigma_{\max} > 0$ are sufficiently small, i. e.,

$$\begin{aligned} &U_t \equiv \eta(X_{t\,-\,t \bmod \delta}) \\ &\quad\Longrightarrow \exists\, \delta > 0, \sigma_{\max} > 0 \;: \\ &\quad\lim_{\tau\to\infty} \sup_{X_0\in\mathbb{B}, Z_{(\cdot)}} \inf_{s\in\mathbb{O}} \|X_\tau - s\| = 0. \end{aligned} \quad (8)$$

In equation (8), X_t is a part of $\{X_t\}_t$, the environment trajectory, that depends on X_0 and $Z_{(\cdot)}$. We will assume, following [3], [37], [38], [39], and [40]:

Assumption 1: The origin of (2) is asymptotically stabilizable under a sampled policy.

There are numerous ways to design a stabilizer: PID, sliding-mode, flatness-based, energy-based control etc. Such stabilizers explicitly do not take any account of the cost. The latter is addressed by optimal control specifically. Thus, reinforcement learning falls into that category. However, a learned policy is not necessarily a stabilizer. Measures must be taken for this sake. In the next section, we overview selected existing approaches that tackle the environment stability.

II. RELATED WORK IN STABILIZING REINFORCEMENT LEARNING

For various safety-critical applications it is important that the environment eventually reach some neighborhood of the target state or avoids leaving said neighborhood. Though, there are other kinds of problems that regard enforcing safe behavior [41], this paper focuses on those that pertain to stability.

Three general groups of approaches to stabilizing reinforcement learning can be distinguished: shield-based reinforcement learning, fusion of model-predictive control and reinforcement learning and Lyapunov-based reinforcement learning. The first group of approaches suggests to directly discard unsafe actions via, generally speaking, a filter which, depending on the context, may be called a shield, an overseer, a supervisor etc. These algorithms vary in the way of how exactly the shield checks actions for safety and issues an emergency solution. The actual design of the shield may be as simple as human-based, i. e., manual [42], or as complex as using formal verification software [43], [44], [45], [46]. Unlike a human overseer, formal-logic-based supervisors are correct by definition and thus are not prone to errors. Yet, their design is highly application-specific and might turn tedious (some hints may be found in [47]). Extensions and applications of supervisory safe reinforcement learning are known. For instance, probabilistic shielding was done in [47], and [48]. Masking unsafe actions via an anticipative supervisor was tried in autonomous driving [49]. The agent was trained on a real car data and tested in a T-junction. Robotic manipulation under safe emergency policies (also called recovery policies) was studied in [50]. Unfortunately, human-based overseers in reinforcement learning are inevitably subjective which may harm their correctness. Shields based on formal logic may be difficult to design and highly specialized whence it is not straightforward to judge their inference property.

Fusion with model-predictive control is an active subfield in stabilizing reinforcement learning (see, e. g., [51], [52], [53], [54], [55]). Various ways of fusion are present, some of which concentrate more on the model learning whereas others look into safety constraints specifically [56], [57], [58], [59], [60], [61], [62], [63]. The famous reinforcement dreamer effectively utilizes the predictive philosophy of model-predictive control [64] (with a term "imagination" used instead of the classical "prediction"). In fact, the dreamer can be seen as a variation of adaptive model-predictive control. It is important to highlight the reliance on a local backup policy here [65], [66]. There is no wonder that such a fusion has potential since model-predictive control possesses an established machinery for guaranteeing stability of the closed loop, such as terminal costs, terminal constraints, contraction tools etc. For instance, Zanon and Gros [51], [52] suggested to use robust model-predictive control to ensure safety of reinforcement learning. The policy parameters were updated to optimize a finite-horizon objective subject to safety constraints. The work [51] implemented the approach only in linear systems though. Some issues remained open, such as safe exploration, model estimation and temporary loss of safety. A methodology more focused on the model-predictive control side was studied in [53], and [54]. This one can rather be classified as learning-based predictive control. Starting with a safe set and a safe policy, this work constructs a predictive agent that solves a model-predictive control optimization problem, if it is feasible, or, otherwise, resorts to the safe policy. The optimization itself was suggested via interior-point methods.

The overall method required a well-calibrated model and solving a generalized eigenvalue problem at each time step to function. A hybrid approach based on learning stage costs in model-predictive control was studied in [67], [68], [69], [70], and [71]. Conditions on learning so as to guarantee the closed-loop stability were derived. Generally, the approach of fusion of reinforcement learning with model-predictive control offers correctness via the established safety and stability theory of model-predictive control. On the other hand, the analyses of infinite-horizon performance [72], [73], [74], [75], [76], [77], [78] can help assess interference of the stabilizing tools with learning. Yet, the field of safe reinforcement learning based on model-predictive control needs future work, especially in terms of elaboration of methods for continuous-time, stochastic environments.

The Lyapunov stability theory was recognized in reinforcement learning already by Perkins and Barto [79], [80]. Advancements were made since [54], [81], although such approaches to stabilizing reinforcement learning were offline and required checking a Lyapunov decay property over a region in the state space during Monte-Carlo agent runs. For instance, [81] suggested a safe Bellman operator to ensure the current optimum objective estimate satisfy a Lyapunov condition. Berkenkamp et al. [54] used state space splitting to verify a Lyapunov condition in each cell. The approach needed certain confidence intervals on the statistical model of the environment. It should be noted that offline Monte-Carlo runs are not always feasible in industrial applications, such as robotics. To this Lyapunov-based stabilizing reinforcement learning, agent design based on stabilization techniques of adaptive control can be categorized (approaches can be traced back to the early 2010s, e. g., [82]). An elaborate approach was suggested by [83], whose elements found application in later works , e. g., [84], [85], [86], [87], [88], [89], [90], [91]. The key idea was to use robustifying controls inspired by [92]. Unfortunately, robustifying controls interfere with learning and do not guarantee stability unless some suitable property such boundedness away from zero of input coupling is satisfied. In [83] and [93], the optimum objective was effectively used as the Lyapunov function candidate for the state. The drift was assumed linear in unknown parameters and the input-coupling function was assumed uniformly bounded. An overview of such techniques, that employ adaptive control in reinforcement learning, can be found in [94]. Early works of (robustly) stabilizing reinforcement learning include, e. g., [95] in which integral quadratic constraints (IQCs) were introduced to inspect closed loop stability properties. Integral quadratic constraints, that allow learning errors in the closed loop as bounded (input) uncertainty, found application in early robustly stabilizing reinforcement learning approaches [95]. Such constraints appeared in recent stability analyses [96]. It should be said that also control barrier functions can be integrated along with (control) Lyapunov functions into reinforcement learning. So, e. g., [97] combined those in a fusion of a safety-critical controller with a learning agent. The approach was tested in a simulation with a bipedal robot in full episodes until success or failure. Another application of control barrier functions is due to [98], where a safe controller was coupled with a model-free reinforcement learning agent. The learning was also episode-based. Comparing a Lyapunov function involved with the optimum objective might give a hint to sub-optimality bounds, which allows assessing interference. Correctness may in turn be achieved by the rigor of the stochastic stability theory [99]. Unfortunately, existing approaches of this category are rarely capable of online functioning without pre-training or Monte-Carlo agent runs. Furthermore, oftentimes, certain assumptions on the environment dynamics are posed in Lyapunov-based stabilizing reinforcement learning such as, e. g., second-order differentiability [100], linearity [101] or global Lipschitzness [4] of the drift. For a detailed survey on the state of the safe and stabilizing reinforcement learning, refer, e. g., to [102].

III. CONTRIBUTION

From the analysis of the existing approaches to reinforcement learning with stability guarantees, a conclusion may be drawn that classical control techniques, such as model-predictive control, Lyapunov-based control and adaptive control, are currently more technically elaborate than supervisor-based safe approaches. This indicates that the connection between classical control theory and vanguard machine learning strengthens. Yet, reinforcement learning with guarantees is rather in its emerging phase and the variety of concrete approaches needs to grow before a unifying and widely recognized framework can be formulated. This paper adds to the state of the art of Lyapunov techniques in stabilizing reinforcement learning. The proposed method introduces specially designed Lyapunov-like constraints on the agent learning so that the environment be stabilized in a suitable sense. It is shown that the critic becomes effectively a (time-varying) Lyapunov function for the closed loop with a suitable decay rate. The environment is considered influenced by bounded stochastic disturbance under sampled policies, i. e., we assume the environment dynamics continuous, whereas the actions are taken at discrete moments in time. The inter-sample behavior of the environment is addressed explicitly in the stability analysis. The reason to assume a bounded noise is because, otherwise, no guarantees can be made on the inter-sample behavior of the state trajectories and this is independent of the agent design (a detailed analysis and justification may be found in the recent stochastic stabilization results under sampled policies [103], [104]). Also, a bounded noise is physically meaningful and some respective stochastic models are provided in the appendix. Unlike many existing Lyapunov-based approaches, the currently presented one is purely online and needs no agent pre-training to achieve environment stability. Also, it should be noted that the derived stability guarantees do not require persistence of excitation. The proposed approach applies to environments with general nonlinear dynamics, not necessarily

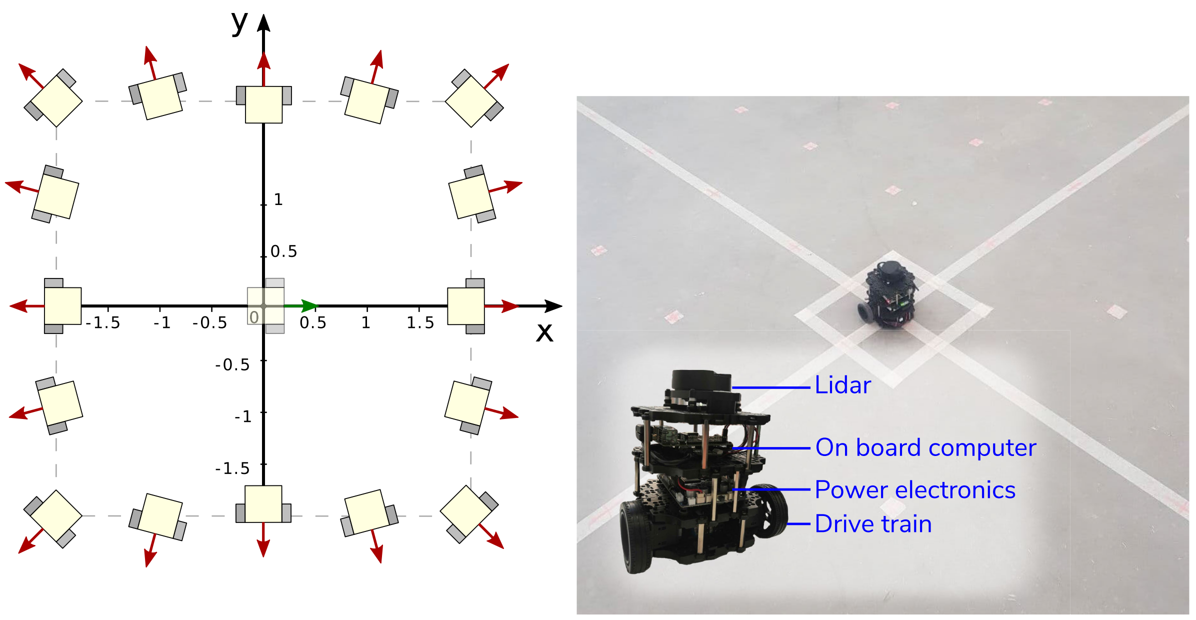

FIGURE 1. Left: target positions and orientations (marked red) for the robot. The initial orientation is marked green. The units are meters. Right: Robotis TurtleBot3 and its key components on the experiment polygon.

control-affine, nor does it need the input-coupling to be bounded or bounded away from zero. Experiments with a mobile robot showed failure of plain actor-critic to stabilize the environment online, whereas the suggested approach beat a baseline stabilizer by far under different running objective functions.

IV. APPROACH

We consider first a state valued critic model $\hat{J} : \mathbb{X} \times \mathbb{W} \rightarrow \mathbb{R}$ as a (deep) neural network with weights $w \in \mathbb{W}$, where $\mathbb{W}$ is compact and $0 \notin \mathbb{W}$. The presented approach does not restrict the user in the choice of a model. Thus, the common recommendations of not-too-rich and not-too-poor structures apply to achieve a trade-off between complexity and avoidance of over-fitting. The environment stability guarantee presented is independent of the critic structure. The ambient idea of the presented approach is motivated by the fact that an ideal critic, which captures the optimum objective sufficiently close, is a actually a Lyapunov function under a policy, which is in turn sufficiently close to the optimizer. The approach resembles this by Lyapunov-like constraints. We fix $\bar{\nu} > 0$, a desirable Lyapunov decay rate. This parameter is arbitrary and independent of the environment, although one might want to set it so that the effective decay do not exceed the running objective. Notice that by the virtue of the HJB, the decay rate of the optimum objective is precisely the running objective. Furthermore, one may pick a learning rate $\beta > 0$. A scheduler, momentum optimization routine parameters or other sophisticated hyper-parameters, found in such methods as ADAM, are omitted as non-essential for the derivations that follow. The user is free to configure the optimization routine however desired. Moreover, we stress here that an optimization routine needs not to actually respect the posed stabilizing constraints. The fact is, that if they happen to be satisfied, the computed critic weights are accepted, otherwise they are not. Explicit account of the constraints may help the actor fire more frequently though. Now, let $\mathfrak{J}_{\text{critic}}$ and $\mathfrak{J}_{\text{actor}}$ denote the critic and actor loss, respectively. The optimization of the critic is performed subject to stabilizing constraints as discussed further. The said optimization problem will not necessarily be feasible, whence a stabilizer is invoked in case of a failure. If a suitable minimizer is found, the actor fires instead. The essential part here is to record a successful state-weight pair x°, w°. Notice that an arbitrary switching between two stabilizers does not necessarily result in closed-loop stability. A careful addressing of the switching mechanism is needed. This is precisely what is done in the presented approach. The resulting critic of the recorded state x° becomes effectively a multi-step Lyapunov function (details are in the appendix).

Fix a pair $\hat{\alpha}_{\text{low}}$, $\hat{\alpha}_{\text{up}}$ of $\mathcal{K}_\infty$ functions, i. e., monotone increasing, unbounded functions. In other words, $\hat{\alpha}_{\text{low}}$, $\hat{\alpha}_{\text{up}}$ satisfy:

$$\hat{\alpha}_{\text{low}}(0) = \hat{\alpha}_{\text{up}}(0) = 0,$$
$$\lim_{s\to\infty} \hat{\alpha}_{\text{low}}(s) = \lim_{s\to\infty} \hat{\alpha}_{\text{up}}(s) = \infty.$$

It should be noted that the stabilizing properties of the proposed approach will hold regardless of which $\hat{\alpha}_{\text{low}}$, $\hat{\alpha}_{\text{up}}$ are chosen as long as they belong to $\mathcal{K}_\infty$. For instance, a reasonable choice of $\hat{\alpha}_{\text{low}}$, $\hat{\alpha}_{\text{up}}$ would be

$$\hat{\alpha}_{\text{low}}(s) = a_{\text{low}} s^2,\ \hat{\alpha}_{\text{up}}(s) = a_{\text{up}} s^2,\ \ 0 < a_{\text{low}} < a_{\text{up}}. \tag{9}$$

The hyper-parameters a_{low}, a_{up} and $\bar{\nu}$ determine a trade-off between freedom of learning and worst-case-scenario

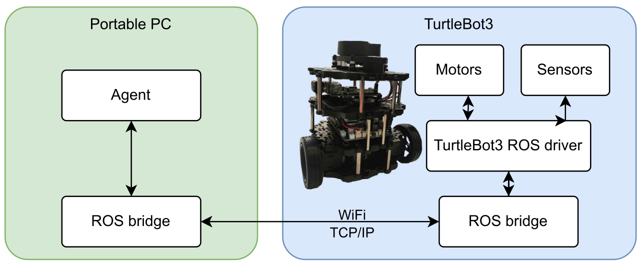

FIGURE 2. **Communication flowchart of the agent in Python and Robotis TurtleBot 3 via ROS.**

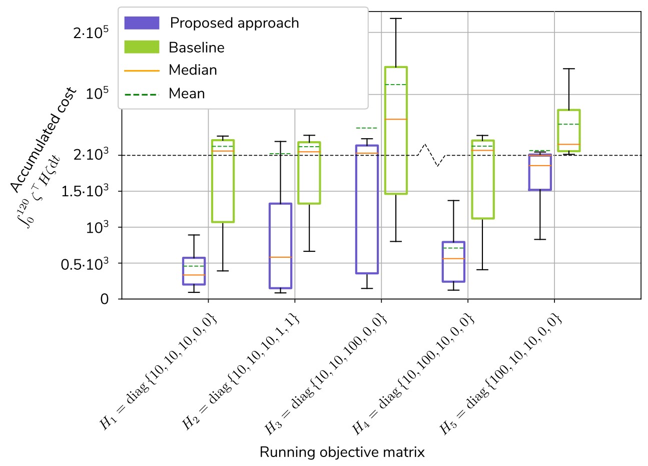

FIGURE 3. **The accumulated running objective of the agent vs. the stabilizer, i. e., $\int_0^{120} \zeta^\top H \zeta \, dt$. The box bounds are, respectively, the first and third quartiles Q1, Q3. The whiskers are the same quartiles plus/minus one and a half interquartile range Q3-Q1. The y-axis scale is split for better reading.**

reaching time of the attractor. If $\frac{a_{\text{low}}}{a_{\text{up}}}$ and $\bar{\nu}$ are chosen to be sufficiently small, the weights of the critic will not be prevented from converging to their ideal values, however if the critic is underfitted, smaller values of $\frac{a_{\text{low}}}{a_{\text{up}}}$, $\bar{\nu}$ may entail slower stabilization accordingly. Summarizing, the stabilizing constraints read at time step k:

$$\hat{J}^w(x_k) - \hat{J}^{w^\circ}(x^\circ) \le -\bar{\nu}\delta,$$
$$\hat{\alpha}_{\text{low}}(\|x_k\|) \le \hat{J}^w(x_k) \le \hat{\alpha}_{\text{up}}(\|x_k\|). \tag{10}$$

The critic loss $\mathfrak{J}_{\text{critic}}$ may be taken in various forms. In this regard, the presented approach does not restrict the user. For instance, consider the following "deferred temporal difference loss" at time step k:

$$\mathfrak{J}_{\text{critic}}(w) = \left(\hat{J}^w(x^\circ) - \sum_{k'=1}^{N^\circ} r(x_{k-k'}, u_{k-k'}) - \hat{J}^{w^\circ}(x_k)\right)^2 + \beta^{-2} \left\| w - w^\circ \right\|^2, \tag{11}$$

where N° denotes the number of steps since the last successful critic update and β is a regularization coefficient that is roughly equivalent to the learning rate in gradient descent based minimization. Alternatively, instead of performing a TD(N)-like update, one may do the update via TD(1) with a batch of size N:

$$\mathfrak{J}_{\text{critic}}(w) = \sum_{k'=1}^{N} \left(\hat{J}^w(x_{k-k'}) - r(x_{k-k'}, u_{k-k'}) - \hat{J}^{w^\circ}(x_{k-k'+1})\right)^2 + \beta^{-2} \left\| w - w^\circ \right\|^2. \tag{12}$$

The regularization term $\beta^{-2} \|w - w^\circ\|^2$ is redundant if gradient descent based minimization is used, since one could simply set a learning rate as opposed to penalizing the displacement of weights. Notice the choice of the critic loss (or learning rate) does not prevent environment stabilization, although the quality of the learning may be affected. Another possible choice of the critic model is state-action valued $\hat{Q} : \mathbb{X} \times \mathbb{U} \times \mathbb{W} \to \mathbb{R}$. In this case, besides x°, w°, the last successful action u° is also stored. The stabilizing constraints read:

$$\hat{Q}^w(x_k, u_k) - \hat{Q}^{w^\circ}(x^\circ, u^\circ) \le -\bar{\nu}\delta,$$
$$\hat{\alpha}_{\text{low}}(\|x_k\|) \le \hat{Q}^w(x_k, u_k) \le \hat{\alpha}_{\text{up}}(\|x_k\|). \tag{13}$$

The actor loss at time step k may be taken as usual in reinforcement learning, considering last successful critic updates, e. g.,

$$\mathfrak{J}_{\text{actor}}(u) = r(x^\circ, u) + \hat{J}_+^{w^\circ}, \tag{14}$$

Algorithm 1 Online Stabilizing Actor-Critic

```
1:  Input: $\bar{\nu} > 0$, $\beta > 0$, $\eta(\cdot)$ – stabilizer
2:  $x_0 \leftarrow X_0$;
3:  $w_0 \leftarrow$ arbitrary in $\mathbb{W}$;
4:  $u_0 \leftarrow \eta(x_0)$;
5:  $w^\circ \leftarrow w_0$;
6:  $x^\circ \leftarrow x_0$;
7:  for $k := 1 \dots \infty$ do
8:      perform $u_{k-1}$;
9:      $x_k \leftarrow$ observed $X_{k\delta}$;
10:     Try critic update:
            $w^* \leftarrow \arg\min_{w \in \mathbb{W}} \mathfrak{J}_{\text{critic}}(w)$
            s. t. $\hat{J}^w(x_k) - \hat{J}^{w^\circ}(x^\circ) \le -\bar{\nu}\delta$,
            $\hat{\alpha}_{\text{low}}(\|x_k\|) \le \hat{J}^w(x_k) \le \hat{\alpha}_{\text{up}}(\|x_k\|)$;
11:     if solution $w^*$ found then
12:         $x^\circ \leftarrow x_k$;
13:         $w^\circ \leftarrow w^*$;
14:         Update action:
                $u_k \leftarrow \arg\min_{u \in \mathbb{U}} \mathfrak{J}_{\text{actor}}(u)$;
15:     else
16:         Call stabilizer: $u_k \leftarrow \eta(x_k)$;
17:     end if
18: end for
```

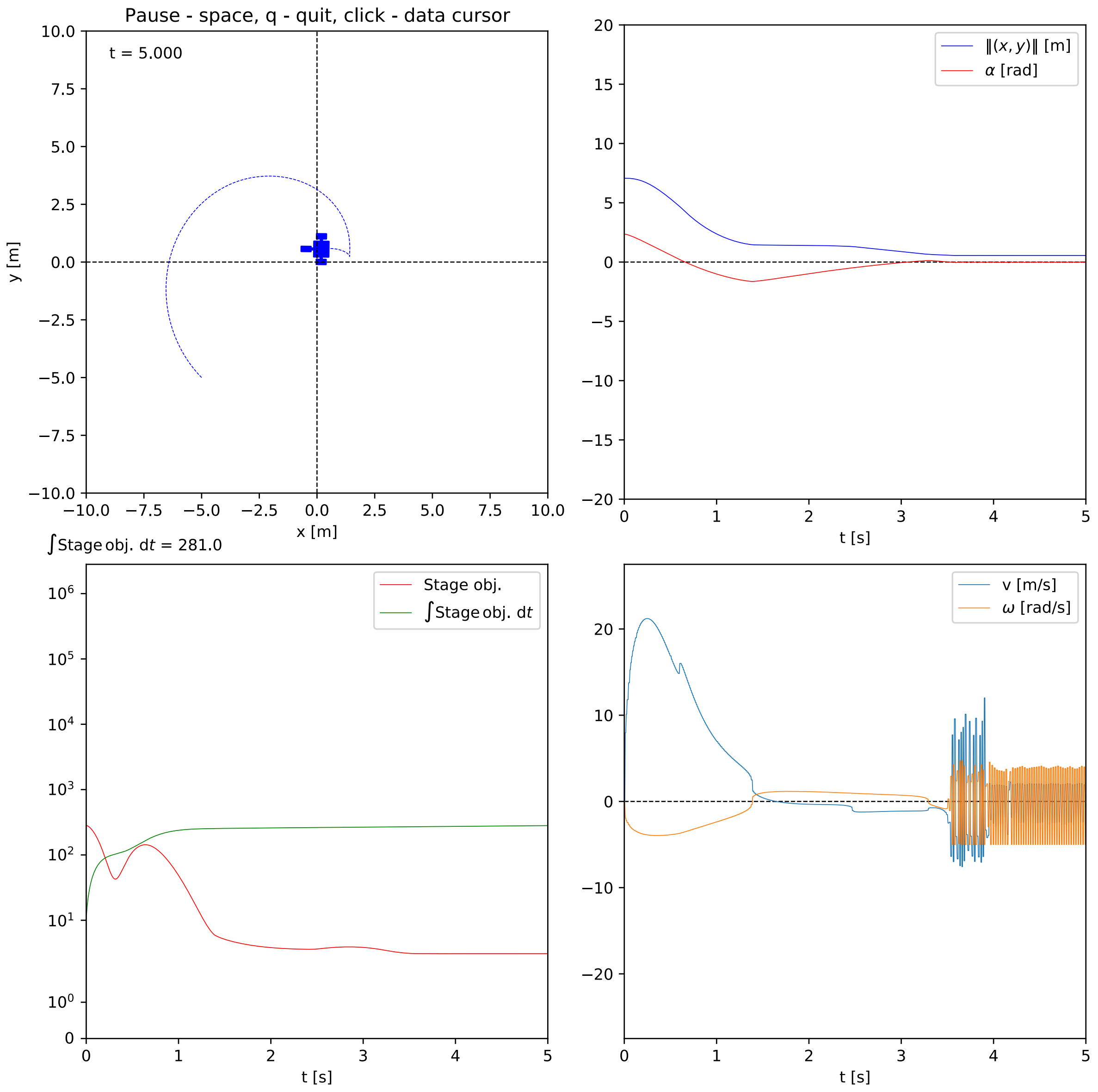

FIGURE 4. **Robot parking using the nominal stabilizer. The dither in control near the origin is caused by controller sampling.**

or

$$\mathfrak{J}_{\text{actor}}(u) = \hat{Q}^{w^\circ}(x^\circ, u). \tag{15}$$

Here, $\hat{J}_{+}^{w^\circ}$ is, e. g., the state valued critic or a statistic thereof, e. g., expectation, at the next state under the action u. An exploration action may be introduced. A basic variant of the algorithm is given in the listing 1.

The main environment stability result is formulated in Theorem 1.

Theorem 1: Let $\mathbb{O}, \mathbb{B}, \eta$ be an attractor, a basin and a sampled stabilizer, respectively. Then, the sampled policy generated by Algorithm 1 drives the state of (2) to $\mathbb{O}$ given that the basin $\mathbb{B}$ is sufficiently large and $\delta > 0, \sigma_{\max} > 0$ are sufficiently small, i. e.,

$$X_0 \in \mathbb{B} \implies \exists\, \delta > 0, \sigma_{\max} > 0 \text{s. t.}$$
$$\mathbb{P}\left[\lim_{k\to\infty} \inf_{s\in\mathbb{O}} \|X_{k\delta} - s\| = 0 \mid U_{k\delta} \text{ set by Algorithm 1}\right] = 1. \tag{16}$$

Proof: See Appendix. ■

The equation (16) basically implies that X_t will eventually reach any open superset of $\mathbb{O}$ and stay there permanently. In general, under noise and finite sampling rate, it is only possible to ensure stabilization up to a certain vicinity of the equilibrium, which is why the theoretical result in this paper is formulated in terms of asymptotic stability of neighborhoods

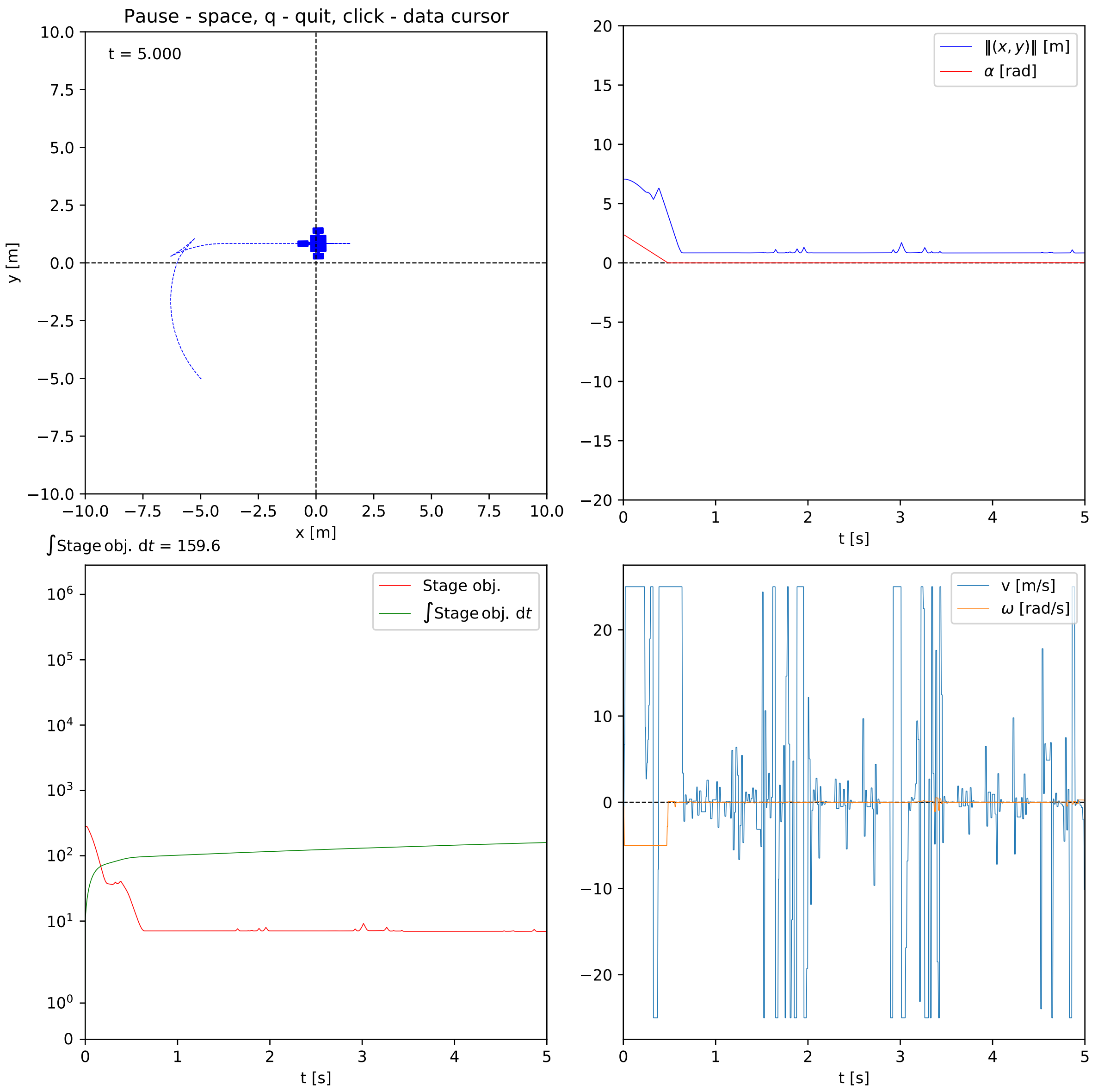

FIGURE 5. **Robot parking using a benchmarking actor-critic agent without stabilizing constraints. One can observe a badly missed target parking pose.**

rather than asymptotic stability of equilibria themselves. This is due to the fact that a neighborhood of the equilibrium exists by the nature of noise, where it is no longer possible to reliably reduce the distance to the equilibrium. While disturbance attenuation methods could potentially be used to reduce the noise effect, this is out of scope. Notice how Algorithm 1 retains the basin and the attractor of the stabilizer.

In simpler terms Theorem 1 states that Algorithm 1 computes a policy that is guaranteed to stabilize the environment. This result is valuable, because it indicates that even a poorly trained reinforcement learning agent is still guaranteed to stabilize the environment as long as it is applied according to Algorithm 1.

V. EXPERIMENTAL STUDY

In this section, we present an evaluation of the presented approach in an optimal mobile robot parking control problem.[2]

A. EXPERIMENTAL SETUP

The experiments were performed with a mobile robot under Algorithm 1. The Robotis TurtlBot3 was used as the experimental medium. Here, a pair of actuated wheels and a ball-wheel are attached to a platform that moves on a

[2]The authors thank Ksenia Makarova and Dmitrii Dobriborsci for supporting the experiments.

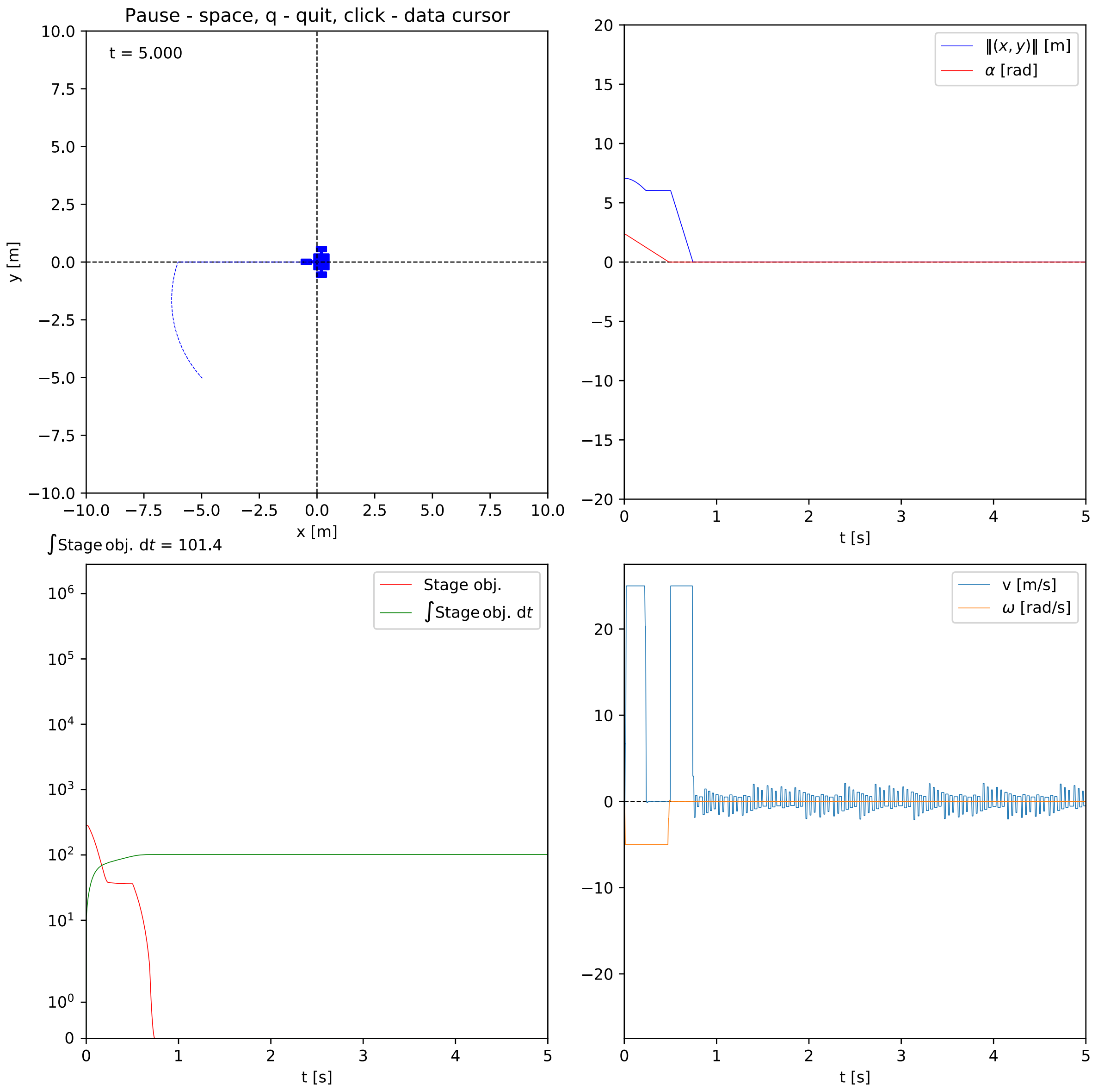

FIGURE 6. **Robot parking using Algorithm 1. The parking goal is achieved well, the cost is reduced significantly.**

the surface. Robotis TurtleBot 3 was equipped with a lidar, an inertia measurement unit (IMU) for implementation of the linear and angular speed control, as well as for dead reckoning, which is also fused with the lidar data for position determination. The experiments were run on a test polygon with concrete floor with markings of coordinate axes and 20 cm step nodes. The robot started in the center with a fixed orientation and drove to one of the target positions as shown in Fig. 1 (a).

The pair of control signals for the respective actuators is decomposed into the forward velocity U_t^1 that is aligned with the direction in which the robot is facing and the angular velocity U_t^2 applied to the center of mass of the robot and directed perpendicular to the platform. The dynamics of the robot read:

$$\mathrm{d}X_t = \mathrm{d}\begin{pmatrix} x_{\text{rob}} \\ y_{\text{rob}} \\ \theta_{\text{rob}} \end{pmatrix}_t = \begin{pmatrix} U_t^1 \cos\theta_{\text{rob},t} \\ U_t^1 \sin\theta_{\text{rob},t} \\ U_t^2 \end{pmatrix} \mathrm{d}t. \tag{17}$$

We took the running objective in the following form:

$$r(x, u) = \zeta^\top H \zeta, \tag{18}$$

where $\zeta := [x - x^*, u]$, and H is a diagonal, positive-definite, running objective matrix and x^* is the target position. The running objective matrices were taken in several variants, to differently prioritize the state and action components of the cost, namely:

$$\begin{aligned} H_1 &= \operatorname{diag}(10, 10, 10, 0, 0), \\ H_2 &= \operatorname{diag}(10, 10, 10, 1, 1), \\ H_3 &= \operatorname{diag}(10, 10, 100, 0, 0), \\ H_4 &= \operatorname{diag}(10, 100, 10, 0, 0), \\ H_5 &= \operatorname{diag}(100, 10, 10, 0, 0). \end{aligned} \tag{19}$$

A full-feedback proportional controller was taken for a stabilizer. The agent was implemented in Python using PyTorch and CasADi modules (see the flowchart in Fig. 2). The goal of the experimental validation was to physically demonstrate the capabilities of Algorithm 1 to achieve superiority in the accumulated cost compared to the stabilizer under various H matrices. For each running objective matrix H, 32 runs were performed, i. e., for each target position and, respectively, the stabilizer and the benchmarked agent. The agent performance was evaluated by the accumulated running cost of 120 seconds, the total run time. The agent sampling frequency was set to 20 Hz which was fairly sufficient for the real-time robot control. The results are presented in the next section.

B. RESULTS AND DISCUSSION

The statistics of the accumulated running cost as per $\int_0^{120} \zeta^\top H \zeta \, \mathrm{d}t$ for different running objective matrices are shown in Fig. 3. Each statistic was computed over multiple runs for each agent type and running cost matrix (see Fig. 1). We observed that the policy by Algorithm 1 yielded considerably lower accumulated running cost showing statistically significant improvement under all selected running cost matrices. In our experiments, we tested several actor-critic baselines but none could park the robot in a meaningful number of episodes. We did not consider policy gradient agents specifically due to their Monte-Carlo learning nature whence no stabilization could have been achieved within few episodes either. For demonstration purposes, we present an exemplary simulation run of parking into the origin using the stabilizer, Algorithm 1 and a plain actor-critic agent without stabilizing constraints. One can see that, as expected, the plain actor-critic agent missed the stabilization goal, whereas the stabilizer fulfilled this task significantly better (see Fig. 4, 5, 6). However, it was the policy by Algorithm 1 that could both achieve the stabilization goal and improve the cost of the stabilizer (see Table 1).

To conclude, the outcomes of the experiment are consistent with the theoretical findings manifested by Theorem 1, thus verifying that the formally derived guarantees hold in practice. About four-fold reduction in average accumulated cost demonstrated the significant impact of the proposed approach on performance. Furthermore, as seen in Table 1, the score of the stabilizing baseline is strongly inferior to both actor-critic and the proposed algorithm, which indicates that the superior performance is not merely due to "stealing" a known stabilizer, but rather is the consequence of knowledge being extracted from the stabilizing baseline during learning.

TABLE 1. Average accumulated cost.

Policy	Accumulated cost
Algorithm 1	88.0
Actor-critic without stabilizing constraints	152.0
Stabilizer (priorly known)	252.0

VI. CONCLUSION

A reinforcement learning algorithm was proposed (Algorithm 1) that ensures environment stability. The stability guarantee was formalized and proven (Theorem 1). A set of experiments was performed that involved running the algorithm on a real robot. The outcomes of the experiments were consistent with the claim of stability that was earlier formally proven in Theorem 1. Additionally, during the experiments Algorithm 1 was observed to produce a significant improvement in performance over the baseline. A prospective research may involve investigating the nuances of how the proposed approach applies to multi-agent systems, such as the ones described in [105].

APPENDIX A
PROOFS

Lemma 1: Let $Z_{[k]}$ denote Z_t restricted to domain $[k\delta, (k+1)\delta]$ and let $F(x_k, u_k, z_k) := X_{(k+1)\delta}$ if $X_{k\delta} = x_k, U_{k\delta} = u_k$, $\mathrm{d}Z_{[k]}(t) = z_k(t)\,\mathrm{d}t$. Notice this latter assumption is valid, because Z_t was assumed to be Lipschitz continuous and thus it is analytical. Then, F is locally Lipschitz-continuous with respect to x_k, u_k and z_k.

Proof: Let $\operatorname{Lip}_\bullet(a, b)$ denote the local Lipschitz constant of $\bullet$ on a segment connecting a and b. Without loss of generality let us assume that $k = 0$. Now let X_t denote the solution of

$$\mathrm{d}X_t = f(X_t, u_0)\,\mathrm{d}t + \sigma(X_t, u_0) z_0(t)\,\mathrm{d}t, \quad X_0 = x_0, \tag{20}$$

and let X'_t denote the solution of

$$\mathrm{d}X'_t = f(X'_t, u_0 + \Delta u)\,\mathrm{d}t + \sigma(X'_t, u_0 + \Delta u)(z_0(t) + \Delta z(t))\,\mathrm{d}t, \quad \text{where } X_0 = x_0 + \Delta x. \tag{21}$$

Now, subtracting the two equations leads to the following upper bound:

$$\begin{aligned} \mathrm{d}\left\|X'_t - X_t\right\| \le{} & \operatorname{Lip}_f(x, x + \Delta x)\left(\left\|X'_t - X_t\right\| + \|\Delta u\|\right)\mathrm{d}t \\ & + \operatorname{Lip}_\sigma(x, x + \Delta x)\operatorname{Lip}_Z \\ & \times \left(\left\|X'_t - X_t\right\| + \|\Delta u\|\right)\mathrm{d}t + \sigma_{\max}\left\|\Delta z(t)\right\|_1 \mathrm{d}t, \end{aligned} \tag{22}$$

which simplifies to

$$\left\|X_t' - X_t\right\| \leq \|\Delta x\| + \sigma_{\max} \|\Delta z(\cdot)\|_1 + \zeta(x, \Delta x)\left(\|\Delta u\| t + \int_0^t \left\|X_t' - X_t\right\| \mathrm{d}\tau\right). \tag{23}$$

where we denoted

$$\zeta(x, \Delta x) := \mathrm{Lip}_f(x, x + \Delta x) + \mathrm{Lip}_\sigma(x, x + \Delta x)\mathrm{Lip}_Z. \tag{24}$$

In turn, using the Grönwall inequality, we obtain the following:

$$\left\|X_t' - X_t\right\| \leq (\sigma_{\max} \|\Delta z(\cdot)\|_1 + \|\Delta x\| + \zeta(x, \Delta x) \|\Delta u\| t)e^{\zeta(x,\Delta x)t}. \tag{25}$$

Thus,

$$\begin{aligned}&\|F(x_0 + \Delta x, u_0 + \Delta u, z_0 + \Delta z) - F(x_0, u_0, z_0)\| \\ &\leq (\sigma_{\max} \|\Delta z(\cdot)\|_1 + \|\Delta x\| + \zeta(x, \Delta x) \|\Delta u\| \delta)\, e^{\zeta(x,\Delta x)\delta} \\ &\leq \mathrm{Lip}_F\, \underbrace{(\|\Delta z(\cdot)\|_1 + \|\Delta x\| + \|\Delta u\|)}_{\text{Norm of displacement.}},\end{aligned} \tag{26}$$

where $\mathrm{Lip}_F = \max(\sigma_{\max}, 1, \delta\zeta(x, \Delta x)) \exp(\zeta(x, \Delta x)\delta)$. ■

Let $h(x) := \inf_{s\in\mathbb{O}} \|x - s\|$ and let $\mathcal{Z}$ denote the set of all possible $z_k(\cdot)$, i. e., $\mathcal{Z} := \{z : [0, \delta] \to \mathbb{R}^d \mid z(\cdot) \text{ is measurable}, \|z(\cdot)\|_\infty < \mathrm{Lip}_Z\}$.

Lemma 2: If a sampled stabilizer is employed, then $F(x_k, u_k, z_k)$ is Lipschitz continuous (with respect to x_k, u_k, z_k) over the set of states spanned by all possible sampled (discrete time) trajectories starting with $X_0 \in \mathbb{B}$.

Proof: Let us denote the said union of states within trajectories with $\mathcal{S}$. Equation (8) implies that

$$\begin{aligned}&\forall\, \varepsilon > 0,\ \exists\, T \geq 0\, \forall\, t \geq T, \\ &\quad \mathbb{P}\left[h(X_t) < \varepsilon \mid X_0 \in \mathbb{B},\ U_\tau \equiv \eta(X_{\tau\, -\, \tau \bmod \delta})\right] = 1,\end{aligned} \tag{27}$$

which in turn leads to

$$\begin{aligned}&\exists\, K \in \mathbb{N}_0\, \forall k \geq K \\ &\quad \mathbb{P}\left[X_{k\delta} \in \mathbb{B} \mid X_0 \in \mathbb{B},\ U_\tau \equiv \eta(X_{\tau\, -\, \tau \bmod \delta})\right] = 1.\end{aligned} \tag{28}$$

As a consequence, the following inclusion holds:

$$\mathcal{S} \subset \left(\mathbb{B} \cup \bigcup_{i=0}^{K} \underbrace{F(\cdot, \mathbb{U}, \mathcal{Z}) \circ \ \ldots\ \circ F(\cdot, \mathbb{U}, \mathcal{Z}) \circ F(\mathbb{B}, \mathbb{U}, \mathcal{Z})}_{\text{Image of } \mathbb{B} \text{ at step } i.}\right) \tag{29}$$

Notice that since $\mathcal{Z}$ is bounded, $\mathbb{B} \times \mathbb{U} \times \mathcal{Z}$ is bounded too. By Lemma 1, F is locally Lipschitz-continuous, which implies that it maps bounded sets to bounded sets. Therefore, each element of the above union is a bounded set. Consequently, the right-hand side is a finite union of bounded sets and is therefore bounded itself.

Evidently $\mathcal{S}$ is bounded, since it is a subset of a bounded set. Finally, the latter implies that the Lipschitz constant of $F(\cdot, \cdot, \cdot)$ can be considered fixed over $\mathcal{S}$ and thus $F(\cdot, \cdot, \cdot)$ is Lipschitz-continuous over $\mathcal{S}$. ■

Definition 3: A continuous function $\beta : [0, +\infty) \times [0, +\infty) \to [0, +\infty)$ is said to belong to class $\mathcal{KL}$ if

- $\beta(s, r)$ is a class $\mathcal{K}_\infty$ function with respect to s for each fixed r.
- $\beta(s, r)$ a decreasing function with respect to r for each fixed s.
- For each fixed s it holds that $\lim_{r\to\infty} \beta(s, r) = 0$.

Let $u^i := \{u_j\}_{j=0}^i$, $z^i := \{z_j\}_{j=0}^i$ and let

$$F^i(x_0, u^i, z^i) := F(\cdot, u_i, z_i) \circ\ \ldots\ \circ F(\cdot, u_1, z_1) \circ F(x_0, u_0, z_0).$$

It is known (see Proposition 2.2 in [106]) that stability induced by η implies that there exists a $\mathcal{KL}$ function β such, that

$$\forall i \in \mathbb{N}_0, x_0 \in \mathbb{B}\ \ \beta(h(x_0), i) > h(F^i(x_0, u^i, z^i)). \tag{30}$$

Furthermore, by Proposition 7 in [107], for any $\mathcal{KL}_\infty$ function β there exist $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$, such that

$$\beta(s, r) = \rho_1(\rho_2(s)e^{-r})\ \ \forall\, s > 0,\ r > 0. \tag{31}$$

Lemma 3: $F^i(x_0, u^i, \cdot)$ is Lipschitz continuous with respect to $z^\infty \in \mathcal{Z}^\infty$.

Proof: Although the domain of $F^i(x_0, u^i, \cdot)$ was initially defined as $\mathcal{Z}^i$, it could nonetheless be considered as function over the extended domain $\mathcal{Z}^\infty$ that is constant with respect to $\{z_j\}_{j=i+1}^\infty$. F^i is obtained by composing Lipschitz-continuous functions and is thus Lipschitz continuous w.r.t. $\mathcal{Z}^i$. It is not hard to show that when an abstract Lipschitz-continuous function $\varphi(a)$ is complemented with a redundant variable b, it remains Lipschitz continuous and preserves the same lipschitz constant:

$$\begin{aligned}\varphi(a + \Delta a, b + \Delta b) - \varphi(a, b) &= \varphi(a + \Delta a) - \varphi(a) \\ &\leq \mathrm{Lip}_\varphi \|\Delta a\| \\ &\leq \mathrm{Lip}_\varphi \|\Delta a\| + \mathrm{Lip}_\varphi \|\Delta b\| .\end{aligned} \tag{32}$$

■

It is well known that the existence of a Lyapunov function can be used to infer stability. However, to do so it is necessary to prove the following converse result first:

Lemma 4: There exists a locally Lipschitz-continuous function $L : \mathbb{R}^n \to \mathbb{R}_+$, a continuous function $\nu : \mathbb{R}^n \to \mathbb{R}_+$ that is strictly positive outside of $\mathbb{O}$ and $\alpha_{1,2} \in \mathcal{K}_\infty$ s. t.

i) L has a decay rate satisfying

$$\begin{aligned}&X_{k\delta} \notin \mathbb{O},\ X_{k\delta} \in \mathcal{S},\ \ U_t \equiv \eta(X_{t\, -\, t \bmod \delta}) \\ &\quad \Longrightarrow L(X_{(k+1)\delta}) - L(X_{k\delta}) < -\nu(\|X_{k\delta}\|),\end{aligned} \tag{33}$$

ii) $\forall x \in \mathbb{X}\ \ \alpha_{\text{low}}(h(x)) \leq L(x) \leq \alpha_{\text{up}}(h(x))$.

Proof: Using (30) and (31) we obtain

$$h(F^i(x_0, u^i, z^i)) \leq \beta(h(x_0), i) \leq \rho_1(\rho_2(h(x_0))e^{-i}), \tag{34}$$

which in turn implies

$$\omega(h(F^i(x_0, u^i, z^i))) \leq \rho_2(h(x_0))e^{-i}. \tag{35}$$

Define $L_0 : \mathbb{X} \times \mathcal{Z} \to \mathbb{R}_+$ by

$$L_0(x_0, z^\infty) = \sum_{i=0}^{\infty} \omega(h(F^i(x_0, u^i, z^i))). \tag{36}$$

It follows from (35) that

$$\begin{aligned}\alpha_{\text{low}}(h(x_0)) &:= \omega(h(x_0)) \leq L_0(x_0, z^\infty) \\ &\leq \sum_{k=0}^{\infty} \rho_2(h(x_0))e^{-k} \leq \frac{e}{e-1}\rho_2(h(x_0)) \\ &=: \alpha_{\text{up}}(h(x_0)).\end{aligned} \tag{37}$$

The above also shows that the series in (36) converges uniformly w.r.t. x_0, z^∞, because the tail sums can be bounded in the following way:

$$\begin{aligned}0 &\leq \sum_{i=n}^{\infty} \omega(h(F^i(x_0, u^i, z^i))) \\ &\leq \sum_{k=n}^{\infty} \sup_{x\in\mathcal{S}} \rho_2(h(x))e^{-k} \leq \frac{1}{1-e^{-1}} \sup_{x\in\mathcal{S}} \rho_2(h(x)) \\ &\quad - \frac{1-e^{-n}}{1-e^{-1}} \sup_{x\in\mathcal{S}} \rho_2(h(x)) \xrightarrow{n\to\infty} 0.\end{aligned} \tag{38}$$

Thus, the series in (36) converges uniformly with respect to z^∞, while the elements of the sequence are Lipschitz continuous with respect to z^∞ by Lemma 3, thus by uniform convergence theorem the series converges to a function that is uniformly continuous with respect to z^∞. Finally, define $L(x) = \sup_{z^\infty\in\mathcal{Z}} L_0(x, z^\infty)$. Evidently:

$$\begin{aligned}\alpha_{\text{low}}(\cdot) &:= \omega(\cdot), \\ \alpha_{\text{up}}(\cdot) &:= \frac{e}{e-1}\rho_2(\cdot), \\ \nu(\cdot) &:= \omega(h(\cdot)).\end{aligned} \tag{39}$$

■

Lemma 5: Let $\mathbb{O}'$ be the closure of an open superset of $\mathbb{O}$, then under 1

$$\exists\, T > 0, : X_0 \in \mathbb{B} \implies \exists \delta > 0, \sigma_{\max} > 0$$
$$\mathbb{P}\left[\sup_{t\in[T,+\infty)} \inf_{s'\in\mathbb{O}'} \left\|X_t - s'\right\| > 0\right] = 0. \tag{40}$$

The above proposition enables us to construct an upper bound on the reaching time of $\mathbb{O}$.

Consider 1. Let us denote, for brevity:

$$\begin{aligned}x^\circ &:= x_{\text{prev}}, \\ \hat{J}^\circ &:= \hat{J}^{w_{\text{prev}}}(x^\circ).\end{aligned} \tag{41}$$

Also, let us denote:

$$\begin{aligned}\hat{\mathbb{K}} &:= \{k \in \mathbb{N}_0 : \text{critic finds a solution } w^*\}, \\ \mathbb{K}_L &:= \{k \in \mathbb{N}_0 : \eta \text{ is invoked}\},\end{aligned} \tag{42}$$

the sets of time step indices where the critic succeeds to find a solution w^* and, respectively, where it does not and so the stabilizing policy η is invoked.

Now, define:

$$x_k^\circ := \begin{cases} x_k, & k \in \hat{\mathbb{K}}, \\ x_{k-1}^\circ, & k \in \mathbb{K}_L, \end{cases}$$

and, by the same token, $\hat{J}_k^\circ$.

Let L_k denote $L(X_{k\delta})$. When running 1, we expect the behavior of L_k and $\hat{J}_k^\circ$ to look schematically like 7 depicts.

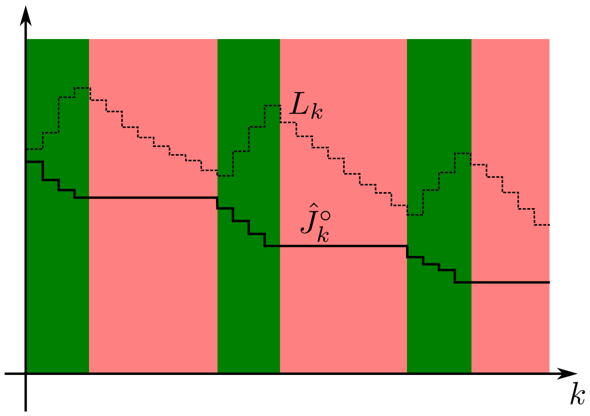

FIGURE 7. Schematic of the critic and a Lyapunov function behavior under Algorithm 1.

It holds that

$$\forall k \in \mathbb{N}_0 \quad \left\|x_k^\circ\right\| \leq \hat{\alpha}_{\text{low}}^{-1}(\hat{J}_k^\circ) \leq \hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ).$$

Observe that

$$\forall k \in \mathbb{N}_0$$
$$L(x_k^\circ) \leq \alpha_{\text{up}}(h(x_k^\circ)) \leq \alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_k^\circ)) \leq \alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ)).$$

Since for all $k \in \mathbb{K}_L$, L_k is non-increasing, we conclude that

$$\forall k \in \mathbb{N}_0 \; L_k \leq \alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ)),$$

whence

$$\forall k \in \mathbb{N}_0 \; \|X_{k\delta}\| \leq \alpha_{\text{low}}^{-1}(\alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ))),$$

Since $\mathbb{B} \subset \mathcal{S}$ and $\mathbb{B}$ can be chosen arbitrarily large given that $\delta > 0$, $\sigma_{\max} > 0$ are sufficiently small, we can thus without loss of generality assume:

$$\inf_{b\notin\mathcal{S}} \|b\| > \alpha_{\text{low}}^{-1}(\alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ))). \tag{43}$$

Now, if $X_{k\delta} \notin \mathbb{O}$, then

$$\nu(X_{k\delta}) \geq \inf_{x\notin\mathbb{O}'} \nu(x). \tag{44}$$

We have, denoting $\Delta\bullet_k \triangleq \bullet_{k+1} - \bullet_k$:

$$\begin{aligned}\forall k \in \hat{\mathbb{K}} \; \Delta\hat{J}_k^\circ &\leq \inf_{x\notin\mathbb{O}'} \nu(x), \\ \forall k \in \mathbb{K}_L \; \Delta L_k &\leq \inf_{x\notin\mathbb{O}'} \nu(x).\end{aligned} \tag{45}$$

If the critic always succeeded, then $\mathbb{O}$ would be reached after no more than

$$\hat{T} := \frac{\hat{J}_0^\circ - \hat{\alpha}_{\text{low}}(\inf_{x \notin \mathbb{O}'} \|x\|)}{\inf_{x \notin \mathbb{O}'} \nu(x)} \tag{46}$$

steps. If the stabilizing policy were always invoked instead, $\mathbb{O}'$ would be reached after no more than

$$T_L := \frac{L_0 - \alpha_{\text{low}}(\inf_{x \notin \mathbb{O}'} \|x\|)}{\inf_{x \notin \mathbb{O}'} \nu(x)}$$

steps.

But, unlike $\hat{J}_0^\circ$, the Lyapunov function L_k may step-wise increase when $k \in \hat{\mathbb{K}}$ (see 7). However, L_k never exceeds $\alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ))$ as shown above. Thus, let us define

$$T_L := \frac{\alpha_{\text{up}}(\hat{\alpha}_{\text{low}}^{-1}(\hat{J}_0^\circ)) - \alpha_{\text{low}}(\inf_{x \notin \mathbb{O}'} \|x\|)}{\inf_{x \notin \mathbb{O}'} \nu(x)}. \tag{47}$$

This is a reaching time of $\mathbb{O}$ if only η were invoked after a worst growth of L_k after a successful critic update. We say "a reaching time" to stress that it is effectively just a bound. Now, let $T^* := \max\{\hat{T}, T_L\}$.

There are several possible scenarios that can precede the event of reaching $\mathbb{O}$. The two limiting cases, when only the critic succeeds or η is involved, are clear from the reaching time T^*. That is, $\mathbb{O}$ is reached within T^* steps. We need thus to argue about the mixed case. It is easy to see that the worst case is when per every critic success, there is $T^* - 1$ invocations of η until $\mathbb{O}'$ is "almost" reached, but not quite, followed by another critic success and so on. Thus, the overall reaching time of $\mathbb{O}'$ under 1 is

$$T^*(T^* - 1). \tag{48}$$

Now that an upper bound on the reaching time of an arbitrary proximity to the attractor $\mathbb{O}$ is obtained, whence the proof is concluded.

A. MISCELLANEOUS REMARKS

Remark 1: Since the number of invocations of η is not greater than T^* till the $\mathcal{B}_{s^*}$ is reached, the critic $\hat{J}^\circ$ is effectively a multi-step Lyapunov function, i. e., $\hat{J}^\circ$ is non-increasing and

$$\forall k \in \mathbb{N}_0 \; \hat{J}_{k+T^*}^\circ - \hat{J}_k^\circ < 0.$$

APPENDIX B
STATE-ACTION VERSION OF THE ALGORITHM

See Algorithm 2.

APPENDIX C
BOUNDED NOISE MODELS

This section briefly reflects the key models from the survey [108]. Some Latin and Greek letters here are to be interpreted separately from the rest of the text and thus not to be confused. The easiest way to achieve bounded noise is to apply a saturation function the standard Brownian motion B_t. Such is the case of the sine-Wiener process $Z_t' = \sin\left(\sqrt{\frac{2}{\tau_a}} Z_t'\right)$ with an autocorrelation time parameter τ_a. Another way is to augment the environment description with a dynamical noise model. Thus, the overall description would read, for instance:

$$\begin{aligned} \mathrm{d}X_t &= (f(X_t, U_t) + \sigma(X_t, U_t) Z_t')\,\mathrm{d}t, \\ \mathrm{d}Z_t' &= \xi(Z_t')\,\mathrm{d}t + \chi(Z_t')\,\mathrm{d}B_t, \end{aligned} \tag{49}$$

where $\{Z_t'\}$ is the noise process with an internal model described by the drift function ξ and diffusion function χ, B_t is the standard Wiener process. Particular ways to construct the respective stochastic differential equations include the following [108]:

- The Doering-Cai-Lin (DCL) noise

$$\mathrm{d}Z_t' = -\tfrac{1}{b_1} Z_t'\,\mathrm{d}t + \sqrt{\tfrac{1 - Z_t'^2}{b_1(b_2+1)}}\,\mathrm{d}B_t, \tag{50}$$

with parameters $b_2 > -1$, $b_1 > 0$;

- The Tsallis-Stariolo-Borland (TSB) noise

$$\mathrm{d}Z_t' = -\tfrac{1}{b_1} \tfrac{Z_t'}{1 - Z_t'^2}\,\mathrm{d}t + \sqrt{\tfrac{1 - b_2}{b_1}}\,\mathrm{d}B_t, \tag{51}$$

with $b_1 > 0$, $b_2 < 1$ parameters;

Algorithm 2 Online Stabilizing Actor-Critic. State-Action Critic

```
1:  Input: $\bar{\nu} > 0$, $\beta > 0$, $\eta(\cdot)$ – stabilizer
2:  $x_0 \leftarrow X_0$;
3:  $w_0 \leftarrow$ arbitrary in $\mathbb{W}$;
4:  $u_0 \leftarrow \eta(x_0)$;
5:  $w^\circ \leftarrow w_0$;
6:  $x^\circ \leftarrow x_0$;
7:  $u^\circ \leftarrow u_0$;
8:  for $k := 1 \ldots \infty$ do
9:      perform $u_{k-1}$;
10:     $x_k \leftarrow$ observed $X_{k\delta}$;
11:     Update the action: $u_k \leftarrow \arg\min_{u \in \mathbb{U}} \hat{Q}^{w^\circ}(x_k, u)$;
12:     Try critic update:
            $w^* \leftarrow \arg\min_{w \in \mathbb{W}} \mathfrak{J}_{\text{critic}}(w)$
                s. t. $\hat{Q}^w(x_k, u_k) - \hat{Q}^{w^\circ}(x^\circ, u^\circ) \leq -\bar{\nu}\delta$,
                    $\hat{\alpha}_{\text{low}}(\|x_k\|) \leq \hat{Q}^w(x_k, u_k) \leq \hat{\alpha}_{\text{up}}(\|x_k\|)$;
13:     if solution $w^*$ found then
14:         $x^\circ \leftarrow x_k$;
15:         $u^\circ \leftarrow u^*$;
16:         $w^\circ \leftarrow w^*$;
17:     else
18:         Replace action for stabilizer: $u_k \leftarrow \eta(x_k)$;
19:     end if
20: end for
```

- Kessler–Sørensen (KS) noise

$$\mathrm{d}Z'_t = -\frac{b_3}{\pi b_1}\tan\left(\frac{\pi}{2}Z'_t\right)\mathrm{d}t + \frac{2}{\pi\sqrt{b_1(b_2+1)}}\,\mathrm{d}B_t, \qquad (52)$$

  with $b_1 > 0, b_2 \geq 0, b_3 = \frac{2b_2+1}{b_2+1}$ parameters.

The above models essentially design the drift and/or diffusion functions so as to confine the strong solutions to stay within $(-1, 1)$ (component-wise) almost surely (the unitary bound is chosen for simplicity and may be adjusted according to the application). It should be noted that the corresponding functions ξ, χ do not satisfy Lipschitz conditions in the standard way. Nevertheless, existence and uniqueness of strong solutions can be ensured [108]. So, for instance, in the case of the TSB noise, the drift is at least locally Lipschitz. This fact, together with non-reachability of the boundaries $-1, 1$ (which can be shown) furnishes the strong solution existence and uniqueness.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[2] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Belmont, MA, USA: Athena Scientific, 2019.

[3] F. L. Lewis and D. Liu, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, vol. 17. Hoboken, NJ, USA: Wiley, 2013.

[4] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*. U.K.: IET, 2012.

[5] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 378–383.

[6] M. Al Borno, M. de Lasa, and A. Hertzmann, "Trajectory optimization for full-body movements with complex contacts," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 8, pp. 1405–1414, Aug. 2013.

[7] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4906–4913.

[8] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, "Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments," 2020, *arXiv:2005.13857*.

[9] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's cube with a robot hand," 2019, *arXiv:1910.07113*.

[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[11] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

[12] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[13] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J. Artif. Intell. Res.*, vol. 15, pp. 319–350, Nov. 2001.

[14] S. M. Kakade, "A natural policy gradient," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001.

[15] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 2219–2225.

[16] B. Bouzy and G. Chaslot, "Monte–Carlo go reinforcement learning experiments," in *Proc. IEEE Symp. Comput. Intell. Games*, May 2006, pp. 187–194.

[17] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement learning in continuous action spaces through sequential Monte Carlo methods," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, 2007, pp. 833–840.

[18] T. Vodopivec, S. Samothrakis, and B. Ster, "On Monte Carlo tree search and reinforcement learning," *J. Artif. Intell. Res.*, vol. 60, pp. 881–936, Dec. 2017.

[19] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel, "Approximate value iteration in the reinforcement learning context. Application to electrical power system control," *Int. J. Emerg. Electr. Power Syst.*, vol. 3, no. 1, Aug. 2005.

[20] A. R. Tavares and L. Chaimowicz, "Tabular reinforcement learning in real-time strategy games via options," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2018, pp. 1–8.

[21] X. Zhao, S. Ding, Y. An, and W. Jia, "Asynchronous reinforcement learning algorithms for solving discrete space path planning problems," *Appl. Intell.*, vol. 48, no. 12, pp. 4889–4904, 2018.

[22] M. Yin and Y.-X. Wang, "Asymptotically efficient off-policy evaluation for tabular reinforcement learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 3948–3958.

[23] A. Thomas and S. I. Marcus, "Reinforcement learning for MDPs using temporal difference schemes," in *Proc. 36th IEEE Conf. Decis. Control*, vol. 1, Dec. 1997, pp. 577–583.

[24] H. Van Seijen, A. R. Mahmood, P. M. Pilarski, M. C. Machado, and R. S. Sutton, "True online temporal-difference learning," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 5057–5096, 2016.

[25] R. S. Sutton and B. Tanner, "Temporal-difference networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1377–1384.

[26] D. Silver, R. S. Sutton, and M. Müller, "Temporal-difference search in computer go," *Mach. Learn.*, vol. 87, no. 2, pp. 183–219, May 2012.

[27] M. Grześ and D. Kudenko, "Online learning of shaping rewards in reinforcement learning," *Neural Netw.*, vol. 23, no. 4, pp. 541–550, May 2010.

[28] S. Box and B. Waterson, "An automated signalized junction controller that learns strategies by temporal difference reinforcement learning," *Eng. Appl. Artif. Intell.*, vol. 26, no. 1, pp. 652–659, Jan. 2013.

[29] W. Konen, "Reinforcement learning for board games: The temporal difference algorithm," Res. Center CIOP, Comput. Intell., Optim. Data Mining, TH Köln–Cologne Univ. Appl. Sci., Tech. Rep. TR 03/2015, 2015.

[30] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Aug. 2009.

[31] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[32] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*, vol. 20, Cambridge, U.K.: Cambridge Univ. Press, Feb. 2011.

[33] M. L. Darby and M. Nikolaou, "MPC: Current practice and challenges," *Control Eng. Pract.*, vol. 20, no. 4, pp. 328–342, Apr. 2012.

[34] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014.

[35] Ç. Yıldız, M. Heinonen, and H. Lähdesmäki, "Continuous-time model-based reinforcement learning," 2021, *arXiv:2102.04764*.

[36] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, Jan. 2000.

[37] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[38] A. Heydari, "Revisiting approximate dynamic programming and its convergence," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2733–2743, Dec. 2014.

[39] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[40] Y. Jiang and Z.-P. Jiang, "Global adaptive dynamic programming for continuous-time nonlinear systems," *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 2917–2929, Nov. 2015.

[41] J. Zhang, D. Yang, H. Zhang, Y. Wang, and B. Zhou, "Dynamic event-based tracking control of boiler turbine systems with guaranteed performance," *IEEE Trans. Autom. Sci. Eng.*, early access, Jul. 18, 2023, doi: 10.1109/TASE.2023.3294187.

[42] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," 2017, *arXiv:1707.05173*.

[43] Y. K. Tan and A. Platzer, "Deductive stability proofs for ordinary differential equations," 2020, *arXiv:2010.13096*.

[44] A. Platzer and J.-D. Quesel, "KeYmaera: A hybrid theorem prover for hybrid systems (system description)," in *Automated Reasoning*. Berlin, Germany: Springer, 2008, pp. 171–178.

[45] A. Platzer and J.-D. Quesel, "European train control system: A case study in formal verification," in *Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9–12, 2009. Proceedings*. Springer, 2009, pp. 246–265.

[46] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.

[47] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem, "Shield synthesis for reinforcement learning," in *Proc. Int. Symp. Leveraging Appl. Formal Methods*. Rhodes, Greece: Springer, 2020, pp. 290–306.

[48] B. Könighofer, R. Bloem, S. Junges, N. Jansen, and A. Serban, "Safe reinforcement learning using probabilistic shields," in *Proc. 31st CONCUR Int. Conf. Concurrency Theory*, Vienna, Austria. Wadern, Germany: Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2020.

[49] D. Isele, A. Nakhaei, and K. Fujimura, "Safe reinforcement learning on autonomous vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–6.

[50] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery RL: Safe reinforcement learning with learned recovery zones," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4915–4922, Jul. 2021.

[51] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

[52] M. Zanon, S. Gros, and A. Bemporad, "Practical reinforcement learning of stabilizing economic MPC," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2258–2263.

[53] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 6059–6066.

[54] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017.

[55] F. Berkenkamp, "Safe exploration in reinforcement learning: Theory and applications in robotics," Ph.D. dissertation, ETH Zürich, Zürich, Switzerland, 2019.

[56] N. Karnchanachari, M. de la Iglesia Valls, D. Hoeller, and M. Hutter, "Practical reinforcement learning for MPC: Learning from sparse objectives in under an hour on a real robot," in *Proc. 2nd Conf. Learn. Dyn. Control*, in Proceedings of Machine Learning Research, vol. 120, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., 2020, pp. 211–224.

[57] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," 2018, *arXiv:1811.01848*.

[58] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 8289–8300.

[59] D. Hoeller, F. Farshidian, and M. Hutter, "Deep value model predictive control," in *Proc. Conf. Robot. Learn.*, in Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100, 2020, pp. 990–1004.

[60] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2020.

[61] S. Reddy, A. D. Dragan, S. Levine, S. Legg, and J. Leike, "Learning human objectives by evaluating hypothetical behavior," in *Proc. Int. Conf. Mach. Learn.*, 2019.

[62] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2786–2793.

[63] K. D. Asis, A. Chan, S. Pitis, R. Sutton, and D. Graves, "Fixed-horizon temporal difference methods for stable reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 3741–3748.

[64] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[65] J. Köhler, R. Soloperto, M. Müller, and F. Allgöwer, "A computationally efficient robust model predictive control framework for uncertain nonlinear systems," *IEEE Trans. Autom. Control*, vol. 66, no. 2, pp. 794–801, Feb. 2021.

[66] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3050–3057, Apr. 2020.

[67] P. Osinenko, G. Yaremenko, D. Dobriborsci, G. Malaniia, and I. Osokin, "A generalized stacked reinforcement learning method for sampled systems," *IEEE Trans. Autom. Control*, pp. 1–8, 2023.

[68] L. Beckenbach, P. Osinenko, and S. Streif, "Model predictive control with stage cost shaping inspired by reinforcement learning," in *Proc. Conf. Decis. Control (CDC)*, 2019, pp. 7110–7115. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9030185&casa_token=OgYQMU1H1k0AAAAA:q8-9yhpEIV5abbiKJswOCi8C_zHR_nTB3tLVV9788EdnoL6GZTl89dOTzojYcXPkNnVLR4SlIlli&tag=1

[69] L. Beckenbach, P. Osinenko, and S. Streif, "Addressing infinite-horizon optimization in MPC via Q-learning," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 60–65, 2018, doi: 10.1016/j.ifacol.2018.10.175.

[70] L. Beckenbach, P. Osinenko, T. Göhrt, and S. Streif, "Constrained and stabilizing stacked adaptive dynamic programming and a comparison with model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 1349–1354, doi: 10.23919/ecc.2018.8550545.

[71] L. Beckenbach, P. Osinenko, and S. Streif, "On closed-loop stability of model predictive controllers with learning costs," in *Proc. Eur. Control Conf. (ECC)*, May 2020, pp. 184–189. [Online]. Available: https://ieeexplore.ieee.org/document/9143704

[72] L. Grune and A. Rantzer, "On the infinite horizon performance of receding horizon controllers," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2100–2111, Oct. 2008.

[73] L. Grüne, "Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems," *SIAM J. Control Optim.*, vol. 48, no. 2, pp. 1206–1228, Jan. 2009.

[74] A. Boccia, L. Grüne, and K. Worthmann, "Stability and feasibility of state constrained MPC without stabilizing terminal constraints," *Syst. Control Lett.*, vol. 72, pp. 14–21, Oct. 2014.

[75] K. Worthmann, "Estimates of the prediction horizon length in MPC: A numerical case study," *IFAC Proc. Volumes*, vol. 45, no. 17, pp. 232–237, 2012.

[76] M. Reble and F. Allgöwer, "Unconstrained model predictive control and suboptimality estimates for nonlinear continuous-time systems," *Automatica*, vol. 48, no. 8, pp. 1812–1817, Aug. 2012.

[77] M. Lorenzen, M. A. Müller, and F. Allgöwer, "Stochastic model predictive control without terminal constraints," *Int. J. Robust Nonlinear Control*, vol. 29, no. 15, pp. 4987–5001, Oct. 2019.

[78] L. Hewing, K. P. Wabersich, and M. N. Zeilinger, "Recursively feasible stochastic model predictive control using indirect feedback," *Automatica*, vol. 119, Sep. 2020, Art. no. 109095.

[79] T. Perkins and A. Barto, "Lyapunov design for safe reinforcement learning control," in *Proc. Safe Learn. Agents, Papers From AAAI Symp.*, 2002, pp. 23–30.

[80] T. J. Perkins and A. G. Barto, "Lyapunov-constrained action sets for reinforcement learning," in *Proc. ICML*, vol. 1, 2001, pp. 409–416.

[81] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018.

[82] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2226–2236, Dec. 2011.

[83] K. G. Vamvoudakis, M. F. Miranda, and J. P. Hespanha, "Asymptotically stable adaptive–optimal control algorithm with saturating actuators and relaxed persistence of excitation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 11, pp. 2386–2398, Nov. 2016.

[84] Z. Yang, Y. Chen, M. Hong, and Z. Wang, "Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8353–8365.

[85] Q. Wei, R. Song, Z. Liao, B. Li, and F. L. Lewis, "Discrete-time impulsive adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4293–4306, Oct. 2020.

[86] H. Su, H. Zhang, H. Jiang, and Y. Wen, "Decentralized event-triggered adaptive control of discrete-time nonzero-sum games over wireless sensor-actuator networks with input constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4254–4266, Oct. 2020.

[87] H. Zhang, K. Zhang, G. Xiao, and H. Jiang, "Robust optimal control scheme for unknown constrained-input nonlinear systems via a plug-n-play event-sampled critic-only algorithm," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 9, pp. 3169–3180, Sep. 2020.

[88] S. K. Jha, S. B. Roy, and S. Bhasin, "Initial excitation-based iterative algorithm for approximate optimal control of completely unknown LTI systems," *IEEE Trans. Autom. Control*, vol. 64, no. 12, pp. 5230–5237, Dec. 2019.

[89] G. P. Kontoudis and K. G. Vamvoudakis, "Kinodynamic motion planning with continuous-time Q-learning: An online, model-free, and safe navigation framework," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3803–3817, Dec. 2019.

[90] S. K. Jha, S. B. Roy, and S. Bhasin, "Direct adaptive optimal control for uncertain continuous-time LTI systems without persistence of excitation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 12, pp. 1993–1997, Dec. 2018.

[91] N. T. Luy, N. T. Dang, D. Q. Minh, and T. H. Vinh, "Machine learning based-distributed optimal control algorithm for multiple nonlinear agents with input constraints," in *Proc. 5th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Nov. 2018, pp. 276–281.

[92] M. Polycarpou, J. Farrell, and M. Sharma, "On-line approximation control of uncertain nonlinear systems: Issues with control input saturation," in *Proc. Amer. Control Conf.*, vol. 1, 2003, pp. 543–548.

[93] R. Kamalapurkar, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for approximate optimal regulation," *Automatica*, vol. 64, pp. 94–104, Feb. 2016.

[94] R. Kamalapurkar, P. Walters, J. Rosenfeld, and W. E. Dixon, *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Berlin, Germany: Springer, 2018.

[95] R. M. Kretchmar, P. M. Young, C. W. Anderson, D. C. Hittle, M. L. Anderson, and C. C. Delnero, "Robust reinforcement learning control with static and dynamic stability," *Int. J. Robust Nonlinear Control*, vol. 11, no. 15, pp. 1469–1500, 2001.

[96] M. Jin and J. Lavaei, "Stability-certified reinforcement learning: A control-theoretic perspective," *IEEE Access*, vol. 8, pp. 229086–229100, 2020.

[97] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control Lyapunov functions and control barrier functions," 2020, *arXiv:2004.07584*.

[98] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 3387–3395.

[99] R. Khasminskii and G. Milstein, *Stochastic Stability of Differential Equations* (Stochastic Modelling and Applied Probability). Springer, 2011.

[100] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon, "A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems," *Automatica*, vol. 49, no. 1, pp. 82–92, Jan. 2013.

[101] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Syst. Control Lett.*, vol. 100, pp. 14–20, Feb. 2017.

[102] P. Osinenko, D. Dobriborsci, and W. Aumer, "Reinforcement learning with guarantees: A review," *IFAC-PapersOnLine*, vol. 55, no. 15, pp. 123–128, 2022.

[103] P. Osinenko and G. Yaremenko, "On stochastic stabilization of sampled systems," in *Proc. Conf. Decis. Control (CDC)*, 2021, pp. 5326–5331. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9682838?casa_token=skSjRnRpXD4AAAAA:mNp0e9PA8OBdV_Q_Bz9sKj-7rJb4Jh9jP65onX9urIp_xm3uYywHdWHj9hVPAsAMQ5Bc3w3gKv2Auw

[104] P. Osinenko, G. Yaremenko, and G. Malaniia, "On stochastic stabilization via non-smooth control Lyapunov functions," *IEEE Trans. Autom. Control*, vol. 68, no. 8, pp. 4925–4931, 2022.

[105] J. Zhang, H. Zhang, Z. Ming, and Y. Mu, "Adaptive event-triggered time-varying output bipartite formation containment of multiagent systems under directed graphs," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 18, 2022, doi: 10.1109/TNNLS.2022.3154028.

[106] Z.-P. Jiang and Y. Wang, "A converse Lyapunov theorem for discrete-time systems with disturbances," *Syst. Control Lett.*, vol. 45, no. 1, pp. 49–58, Jan. 2002.

[107] E. D. Sontag, "Comments on integral variants of ISS," *Syst. Control Lett.*, vol. 34, nos. 1–2, pp. 93–100, May 1998.

[108] D. Domingo, A. d'Onofrio, and F. Flandoli, "Properties of bounded stochastic processes employed in biophysics," *Stochastic Anal. Appl.*, vol. 38, no. 2, pp. 277–306, Mar. 2020.

PAVEL OSINENKO He received the Diploma degree (Hons.) from the Bauman Moscow State Technical University, Russia, in 2009, and the Ph.D. degree (Hons.) in vehicle optimal control and identification from the Dresden University of Technology in 2014. His thesis was dedicated to robust flight control systems and model reduction. He has extensive work experience in the German and Russian academic and industrial sectors. Also, He taught a great number of courses in Germany and Russia on control theory and artificial intelligence. His main expertise is in automation, robotics and reinforcement learning.

GRIGORY YAREMENKO received the bachelor's degree from the Faculty of Computational Mathematics and Cybernetic Faculty, Moscow State University, in 2020, where the main focus of his studies and research were dynamical systems related disciplines and machine learning. His bachelor's dissertation was on Kalman filtering to fluid dynamics and was awarded the first place at the Faculty's annual bachelor thesis competition. During his studies with Moscow State University, he was with Vira Realtime, as an Industrial Scientist and a Software Engineer, from 2018 to 2020, producing data driven solutions for leak detection in oil pipelines. In 2022, he was awarded the master's degree for completing Skoltech's "Data Science" Program. During his time with Skoltech, he conducted theoretical research in stochastic stabilization and reinforceme nt learning.

GEORGIY MALANIYA received the master's degree from the Faculty of Mechanics and Mathematics, Moscow State University, in 2021. He has industrial experience in research and development with Huawei Tech, as a Machine Learning Research Engineer, from 2019 to 2021, while developing time series models based on Bayesian methods. In 2021, he was granted the Ph.D. position with the Skoltech Artificial Intelligence in Dynamic Action Laboratory. His research interests include control theory, symplectic geometry, algebraic topology, machine learning, dynamical systems, reinforcement learning, and manifold learning.

ANTON BOLYCHEV received the master's degree from the Faculty of Mechanics and Mathematics, Moscow State University, in 2020, where he conducted research in advanced probability theory. He has industrial experience in research and development in quantitative research and machine learning. In 2023, he joined the Skoltech Artificial Intelligence in Dynamic Action Laboratory, as the Ph.D. student. His research interest includes reinforcement learning with a focus on computationally efficient implementation.