A framework for online, stabilizing reinforcement learning

Grigory Yaremenko, Georgy Malaniya, Pavel Osinenko

Abstract—Online reinforcement learning is concerned with training an agent on-the-fly via dynamic interaction with the environment. Here, 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, model-free approaches, which are akin to dynamic programming. 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. Online reinforcement learning, in contrast, is more akin to classical control, which utilizes some model knowledge about the environment. Stability of the closed-loop (agent plus the environment) is a major challenge for such online approaches. In this paper, we tackle this problem by a special fusion of online reinforcement learning with elements of classical control, namely, based on the Lyapunov theory of stability. The idea is to start the agent at once, without pre-training, and learn approximately optimal policy under specially designed constraints, which guarantee stability. The resulting approach was tested in an extensive experimental study with a mobile robot. A nominal parking controller was used as a baseline. It was observed that the suggested agent could always successfully park the robot, while significantly improving the cost. While many approaches may be exploited for mobile robot control, we suggest that the experiments showed the promising potential of online 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—Nonlinear systems, reinforcement learning, stability

I. INTRODUCTION

A. Summary of stabilizing reinforcement learning

Reinforcement learning is an optimal control method that uses adaptation imitating living beings in their natural environments [1]–[3]. Its applications range from robotics [4], [5] to games such as Go, chess, shogi (also known as Japanese chess) [6], even complex ones such as StarCraft II [7].

Reinforcement learning can be model-based or model-free, online or offline etc. Online here means being able to learn during exploitation, as opposed running training procedures beforehand. Offline, on contrary, refers to learning by prior optimization of an agent that is only deployed once the learning is finalized.

Contemporary reinforcement learning methods are usually understood in the context of approximating (learning) the cost-to-go function $J$ (or Q-function, or advantage function $A$ or something else related to the cost-to-go) via (deep) neural networks. The core problem with such an approach is that one cannot know how good the chosen neural network topology is capable of approximating the cost-to-go function.

Although it is known that the optimizer (the optimal policy) has certain desirable properties, e.g., it keeps the system stable, an optimizer resulting from an approximate cost-to-go function has in general no guarantees for the closed loop, first and foremost in terms of stability.

Measures were and are taken to provide the said guarantees. These roughly go in the following three directions:

1) introduction of a filter to discard unsuitable actions. Such a filter may be human-based [8] or designed on the grounds of formal verification [9];
2) merging of reinforcement learning with (stabilizing) model-predictive control [10]–[15];
3) Lyapunov-based reinforcement learning [16]–[18].

A detailed survey on these methods can be found in [19].

There also exists an approach that intends to utilize the cost-to-go function itself as an effective Lyapunov function for the closed loop. This approach seems only viable if the learned policy is close enough to the optimal one, although some claim an actor-critic controller to be stabilizing under usage of model-free tools such as robustifying terms from the adaptive control field [20]. Unfortunately, the latter paper contains erroneous results – see the detailed analysis and a counter-example in [21].

We focus on the Lyapunov-based stabilizing reinforcement learning as per the third direction as listed above. Some approaches of this category, such as [16], [22], [23], can be regarded as offline. Our interest here lies in online control methods though.

In previous works by the author [18], [24]–[26], a framework was developed for stabilizing, online, Lyapunov-based reinforcement learning. The results in [18] were enabled by the techniques of sample-and-hold stabilization analyses [27]–[29], which were recently extended to the case of stochastic systems [30], [31].
B. Stabilizing reinforcement learning via Lyapunov techniques

Lyapunov theory and its stochastic generalization due to R. Khasminskii [32] are the recognized standard when it comes to stability analyses. Already in the beginning of the 2000s, Lyapunov theory was used to design stabilizing reinforcement learning by Perkins and Barto [16], [22]. The latter work meant to design or restrict the action choices of the agent so that on of the Lyapunov theorems applies. A variety such approaches were developed since. For instance, [17] suggested a stability-preserving Bellman operator ensure the current cost-to-go function estimate satisfy a Lyapunov condition. The approach required iterations over a region in the state space and also that the baseline (stabilizing) policy be sufficiently close to the optimal one. Berkenkamp et al. [33] used state space splitting to verify a Lyapunov condition in each cell. The approach assumed uniformly bounded. An overview of such techniques, unknown parameters and the input-coupling function was taken at discrete moments in time. The inter-sample behavior underlying dynamics be continuous, whereas the actions are disturbance in a sample-and-hold setting, i.e., we assume the under the respective conditions are meant to hold almost surely. We assume that \( f \) and \( \sigma \) are locally Lipschitz continuous with respect to \( X_t \). Particular models for such a bounded noise as in the above description are given in Section II of the appendix.

Consider an agent-environment loop that assumes a state \( X_t \in \mathbb{X} \subset \mathbb{R}^n \) at a time \( t \in \mathcal{T} = \mathbb{R}_+ \), whose dynamics are modeled by the following transition law:

\[
\begin{align*}
\text{d}X_t &= f(X_t, U_t) \text{d}t + \sigma(X_t, U_t) \text{d}Z_t, \\
\mathbb{P} [X_0 = \text{const}] &= 1
\end{align*}
\]

where an unknown noise \( Z_t \in \mathbb{R}^d \) is modelled by an arbitrary Lipschitz-continuous random process with a fixed Lipschitz constant \( \text{Lip}_Z \), and \( \sigma : \mathbb{X} \times \mathcal{U} \rightarrow \mathbb{R}^{n \times d} \) is a bounded mixing matrix, i.e., \( \|\sigma(x, u)\|_{\text{op}} \leq \sigma_{\text{max}} \), \( \sigma_{\text{max}} > 0 \). Here and after, when claims are made about sections of random processes, the respective conditions are meant to hold almost surely. We assume that \( f \) and \( \sigma \) are locally Lipschitz continuous with respect to \( X_t \). Particular models for such a bounded noise as in the above description are given in Section II of the appendix.

Let a policy \( \rho : \mathbb{X} \times \mathcal{T} \rightarrow \mathcal{U} \) determine the agent’s action \( U_t \in \mathcal{U} \) at time \( t \) in the following way:

\[
U_t = \rho(X_t - t \text{ mod } \delta),
\]

where \( \delta \) is the digital controller sampling time of the present (such a setting is also known as “sample-and-hold”) or S&H. Under such a digital control, at best only practical stability in the sense of the following definition can be achieved.

**Definition 1:** A policy \( \rho [X_0](\cdot) \) is said to practically semi-globally stabilize \( \mathbb{1} \) from \( X_0 \) until \( s^* \) if, given \( s^* < r < \infty \), there exists a \( \delta > 0 \) s.t. under (2) a trajectory \( x(t) \) with a sampling time \( \delta \leq \delta \), starting at \( X_0 \) is bounded, enters \( \mathcal{B}_r \) after a time \( t^* \), which depends uniformly on \( \|X_0\|, r \), and remains there for all \( t \geq t^* \).

In the following, for brevity, the wording “semi-globally” is omitted.

We consider infinite horizon optimal control in the sense of cost-to-go rather than accumulated reward, although the two are just different and equipollent formulations. Let the running cost be \( \rho : \mathbb{X} \times \mathcal{U} \rightarrow \mathbb{R} \). The aim of reinforcement learning

Unlike some of the existing Lyapunov-based approaches, the currently presented one is purely online. 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 control-affine, nor does it need the input-coupling to be bounded or bounded away from zero.

The proofs of all the theorems, as well as the extra graphical material on the simulation studies, are to be found in the appendix.
is to find a policy that optimizes a cost-to-go $J_\rho : \mathbb{X} \to \mathbb{R}$, where

$$J_\rho(X_0) := E \left[ \int_0^\infty r(X_t, U_t) \, dt \right], \quad X_0 = \text{const} \text{ almost surely},$$

and $U_t$ satisfies (2). The minimum of the cost-to-go is the optimal cost-to-go function $J_\rho$.

Note, that despite the fact that $t$ appears in (2), $J_\rho$ need not depend on time, since $\rho$ is interpreted as a stationary policy in discrete time.

**Remark 1:** It must be noted that the proposed approach is also applicable to the case, when one is dealing with maximization of total reward rather than minimization of total cost. In the former case however running costs and cost-to-go function $J$ are replaced with running rewards and value function $V$ respectively, while the minima of total cost are obviously replaced with maxima of total reward.

The goal of the present paper is to determine an algorithm capable of constructing a practically stabilizing policy on the fly, given the agent’s transition law, while also addressing the above stated aim via reinforcement learning methods. In other words, the proposed method is intended to generate a quasi-optimal practically stabilizing control policy $\rho[X_0]$ for an arbitrary initial state $X_0$ given the state transition function $f(\cdot, \cdot)$. Notably, the generation of such policies should not require any pre-training, i.e., the algorithm is supposed to immediately provide a viable policy given only $f(\cdot, \cdot)$ and $X_0$.

In the next section, we give the central assumptions in succeeding at this problem statement.

### A. Assumptions

The starting point in many reinforcement learning analyses is the requirement that $L_\nu$ be stabilizable [36]–[39]. In this work, we make the following assumptions that are both implied by stabilizability [40].

**Assumption 1:** There exists a locally Lipschitz continuous, positive-definite function $L : \mathbb{R}^n \to \mathbb{R}^+$, a diverging continuous positive definite function $\nu : \mathbb{R}^n \to \mathbb{R}^+$ and $\alpha_{\text{low}}, \alpha_{\text{up}} \in K_{\infty}$ s.t. for any compact $\mathbb{X} \subset \mathbb{X}$, there exists a compact set $\bar{U} \subset U$ and it holds that, for any $x \in \mathbb{X}$,

- $L$ has a decay rate satisfying

$$\inf_{u \in \bar{U}} \mathcal{L}_{\nu}(x,u) \leq -\nu_0(x),$$

- $\alpha_{\text{low}}(\|x\|) \leq L(x) \leq \alpha_{\text{up}}(\|x\|).$

In the above, $\mathcal{L}_{\nu}(x,u) L(x)$ is a suitable generalized directional differential operator of $L(x)$ in the direction of $f(x, u)$, e.g., directional subderivative [41].

**Assumption 2:** There exists a stabilizing sample-and-hold policy $\eta : \mathbb{X} \to \mathbb{U}$ associated with the latter Lyapunov function $L$, i.e.,

$$\forall \delta \in (0, \bar{\delta}) : \quad U_t := \eta(X_t \mod \delta) \implies L(X_{(k+1)\delta}) - L(X_k \delta) \leq -\delta \nu(X_k \delta),$$

where $\nu(x) > 0$ whenever $S_{\sigma_{\max}, \delta} > \|x\| > \bar{S}_{\sigma_{\max}, \delta}$.

A recent stochastic stability analysis [31] has shown that as long as the noiseless system

$$dX_t^{\text{noiseless}} = f(X_t^{\text{noiseless}}, U_t) \, dt$$

is stabilizable, assumptions 1-2 hold. Furthermore it is true that

$$\bar{s}_{\sigma_{\max}, \delta} \to 0, \quad \sigma_{\max} \to 0 \text{ and } \delta \to 0,$$

$$\bar{S}_{\sigma_{\max}, \delta} \to +\infty, \quad \sigma_{\max} \to 0 \text{ and } \delta \to 0.$$

**Remark 2:** Assumption [1] is weaker than in some reinforcement learning schemes with guarantees, such as [39] where $L$ needs actually to satisfy a Hamilton-Bellman-Jacobi inequality, but is still somewhat strong (however, see [33]). This is, as discussed above, the price to pay for ensuring practical stabilizability property of the policy resulting from learning. However, already controllability, as required in convergence analyses of reinforcement learning (see, e.g., [37], [39]), implies stabilizability [27], [42]. If the closed-loop is stable, there is, e.g., a constructive way to find a Lyapunov function (see [43]). Also, there are numerical techniques to build control Lyapunov functions [44]. Furthermore, they can be constructed from physical considerations, e.g., kinetic and potential energy [33]. Then, there are systematic techniques such as augmented Lyapunov functions in adaptive control; backstepping; stabilizing model-predictive control whose finite-horizon cost function can be shown to be a Lyapunov function etc.

**Remark 3:** Also note that the existence of such $\alpha_{\text{low}}(\cdot)$ and $\alpha_{\text{up}}(\cdot)$ are equivalent to radial unboundedness and positive-definiteness respectively [45]. Furthermore, a constructive proof of the latter properties would have to involve a construction of such functions. Thus if a $L(\cdot)$ was constructively proven to be a Lyapunov function, then the respective $\alpha_{\text{low}}(\cdot)$ and $\alpha_{\text{up}}(\cdot)$ can be extracted from the latter proof.

Let the critic $\hat{J} : \mathbb{X} \times \mathbb{W} \to \mathbb{R}$ be an approximator, e.g., a deep neural network, with weights $w \in \mathbb{W}$, where $\mathbb{W}$ is compact and $0 \notin \mathbb{W}$. It is up to the user of the proposed approach to choose their critic structure. A flexible critic, capable of closely approximating the cost-to-go function will likely yield a better performance. However, the stabilization itself will occur regardless of such nuances as long as the critic is chosen so as to satisfy the following assumption: The actor will be considered also in a parametric form $\rho^\eta : \mathbb{X} \times \Theta \to \mathbb{U}$, where $\Theta$ is a compact actor weight set.

**Assumption 3:** For each $\zeta > 0$ there exists $w^\eta_\zeta \in \mathbb{W}$ s.t. $\zeta L(x) = \hat{J}^\eta_\zeta(x)$, for all $x \in \mathbb{X}$. Note that if $\zeta > 0$ then $\zeta L(x)$ is also a control-Lyapunov function that corresponds to the same stabilizing policy as $L(x)$.

**Remark 4:** The simplest way to satisfy Assumption [3] is to choose $\hat{J}^\eta_\zeta(x) = w_0 \varphi(x, w^{[2:N_c+1]}_\zeta) + w_1 L(x)$, where $\varphi$ is a (deep) neural network with some $N_c$ weights $w^{[2:N_c+1]}_\zeta$, and $w_0, w_1 \in [0, \infty)$. 


Remark 5: Assumption 3 means that the critic structure be rich enough so as to capture the structure of the Lyapunov function as per Assumption 1. We use it in such an exact format only for brevity. A relaxed form with some discrepancy between $\hat{J}^w(x)$ and $\zeta L(x)$ would also do the job, one would only need to account for it in the constraints (C1)-(C4). A weaker, local condition of the kind $\forall \alpha \exists w^c \forall x \in X' \hat{J}^w(x) = \zeta L(x)$ would also fit, whereas one would just need to ensure that such an $\alpha'$ that the current and predicted state lie in. Again, we choose the simplest form for the ease of exposition.

Assumption 4: The critic $\hat{J}^w(\cdot)$ is locally Lipschitz continuous for every $w \in \mathbb{W}$. Thus the notation $\Lambda_j$ implies $\forall w \in \mathbb{W} | \hat{J}^w(x) - \hat{J}^w(x') \leq \text{Lip}_j \|x - x'\|$. Though, since $\hat{J}(\cdot)$ is locally Lipschitz continuous, its Lipschitz constant $\text{Lip}_j$ is ambiguous until a bounded domain is specified.

IV. PRELIMINARIES

Let $\Lambda_{U_j}(X_t)$ be the Euler estimate of $X_{t+\delta}$ for the noiseless system:

$$\Lambda_{U_j}(X_t) = X_t + \delta f(X_t, U_t).$$

Remark 6: The Euler prediction scheme is chosen for the ease of exposition of the forthcoming analyses, but other choices, including Runge-Kutta methods would work.

Online reinforcement learning implies that at each time step $k$ an update is being carried out for both the control input $u$ and the critic weights $w$:

$$u_k, w_k \leftarrow F(x_k, w_{k-1}, \ldots)$$

The present paper discusses how online stability can be ensured by enforcing certain stabilizing constraints when performing the latter update. Let the stabilizing constraint function $G^w_j(\cdot, \cdot, \cdot, \cdot)$ be defined as follows:

$$G^w_j(w, w_{\text{prev}}, x_{\text{next}}, x) := \begin{cases} \hat{J}^w(x) - \hat{J}^w_{\text{prev}}(x) - \delta \varepsilon \\ L(x_{\text{next}}) - \hat{J}^w(x_{\text{next}}) \\ \hat{J}^w(x_{\text{next}}) - \hat{J}^w(x) + \delta \nu \\ \alpha_{\text{low}}(\|x\|) - \hat{J}^w(x) \\ \hat{J}^w(x) - \alpha_{\text{up}}(\|x\|) \end{cases}$$

where $\alpha_{1,2} \in \mathbb{K}_{\infty}$ are chosen s.t. $\forall x \alpha_{\text{low}}(\|x\|) \leq \alpha_{\text{low}}(\|x\|), \alpha_{\text{up}}(\|x\|) \leq \alpha_{\text{up}}(\|x\|)$.

Note that ensuring $G^w_j(w, w_{\text{prev}}, x_{\text{next}}, x) \leq 0$ is equivalent to imposing the following four scalar constraints:

$$\hat{J}^w(x) \leq \hat{J}^w_{\text{prev}}(x) + \delta \varepsilon,$$

$$L(x_{\text{next}}) \leq \hat{J}^w(x_{\text{next}}),$$

$$\hat{J}^w(x_{\text{next}}) - \hat{J}^w(x) \leq -\delta \nu,$$

$$\alpha_{\text{low}}(\|x\|) \leq \hat{J}^w(x) \leq \alpha_{\text{up}}(\|x\|),$$

(V. ALGORITHM)

It will be shown that enforcing either $G^w_j(w_k, w_{k-1}, x_k, x_{k-1}) \leq 0$ or $G^w_j(w_k, w_{k-1}, \Lambda_u(x_k), x_k) \leq 0$ is sufficient to guarantee a certain kind of stability. At the same time the latter constraints do not prevent the critic from converging to the optimal cost-to-go function.

Below are examples of particular algorithms that involve these constraints.

A SARSALike approach can be used if the model of the noise is not priorly known is shown in Algorithm 1.

Algorithm 1: CALF as SARSA

Data: $\varepsilon > 0$, $\nu > 0$, $\zeta > 0$

Result: $U_0, U_\delta, U_{2\delta}, \ldots$

$$x_0 \leftarrow X_0;$$

$$w_0 \leftarrow w^\circ;$$

for $k = 0 \ldots \infty$

$$\left( \theta_k \right) \leftarrow \arg\min \text{s.t. } G^\nu_{\text{up}}(w, w_k, \Lambda_{\text{up}}(x_k), x_k) \geq 0$$

where

$$J_{\varepsilon}(\theta, w) = r(x_k, \rho(\theta(x_k))) + \hat{J}^w_{\text{up}}(\Lambda_{\text{up}}(x_k))$$

perform $U_{\delta}$

$$x_{k+1} \leftarrow \text{observe } X_{(k+1)\delta}$$

$$w_{k+1} \leftarrow \arg\min \text{s.t. } G^\nu_{\text{up}}(w, w_k, x_{k+1}, x_k) \leq 0$$

where

$$J_{\varepsilon}(w) = \left( \hat{J}^w(x_k) - r(x_k, U_{\delta}) - \hat{J}^w_{\text{up}}(x_{k+1}) \right)^2$$

end

A sort of an actor-critic can be employed provided that one has the means to estimate the expected value (for instance by computing the mean of an artificial sample). It is shown in Algorithm 2.

Although the above examples each present a case of single step temporal difference learning, the proposed approach would not in any way inhibit a different choice of temporal difference updates, such as for instance TD($N$) or TD($\lambda$). In fact the stabilizing properties of an algorithm of this kind are not affected by how the updates are performed as long as the said updates do not violate the stabilizing constraints. Though, of course, these choices do affect the resulting total cost.

Remark 7: One can introduce regularization terms into the actor and critic loss functions with some tuning parameters $\beta_{1,2,3} > 0$ as follows:

$$\mathbb{J}_a(\theta, w) + \beta_1 \|w - w_k\|^2 + \beta_2 \|\theta - \theta_{k-1}\|^2$$

$$\mathbb{J}_c(w) + \beta_3 \|w - w_k\|^2.$$

These regularization terms may be employed to regulate weight change thus imitating a learning rate of a gradient-like update rules.

Remark 8: One can formulate the loss, e.g., in terms of an experience replay of size $M$:

$$w_{k+1} \leftarrow \arg\min \text{ s.t. } \sum_{j=0}^{M-1} e_j(w) \leq \hat{J}^w_{\text{up}}(x_{k+1})$$

where $e_j(w) = (\hat{J}^w(x_{k-j}) - r(x_{k-j}, U_{\delta}) - \hat{J}^w_{\text{up}}(x_{k+1-j}))^2$ and the buffer count is backwards relative to $k$. 
Algorithm 2: CALF as stabilizing actor-critic

Data: $\varepsilon > 0$, $\vartheta > 0$, $\zeta > 0$
Result: $U_0, U_\delta, U_{2\delta}, \ldots$

\[ w_0 \leftarrow w_0^\vartheta; \]
for $k = 0 \ldots \infty$ do

\[ x_k \leftarrow \text{observe } X_k \delta \]

\[ \begin{align*}
& \theta_{k-1} \leftarrow \arg \min_{\theta} \mathcal{J}_a(\theta, w) \\
& \text{where } \\
& \mathcal{J}_a(\theta, w) = r(x_k, \rho^\theta(x_k)) + \\
& \mathbb{E}\left[ J_{w_k}(X_{(k+1)\delta}) | X_{k\delta} = x_k, U_{k\delta} = \rho^\theta(x_k) \right] \\
& w_{k+1} \leftarrow \arg \min_{w} \mathcal{J}_c(w) \\
& \text{where } \\
& \mathcal{J}_c(w) = \left( J_{w_k}(x_k) - r(x_k, U_{k\delta}) - \\
& \mathbb{E}\left[ J_{w_k}(X_{(k+1)\delta}) | X_{k\delta} = x_k, U_{k\delta} = \rho_{\theta_k}(x_k) \right] \right)^2 \\
& U_{k\delta} \leftarrow \rho_{\theta_k}(x_k) \\
\end{align*} \]

end

Remark 9: The above described loss functions are formulated inspired by the Bellman optimality principle for undiscounted problems. Introducing a $\gamma \in (0, 1)$ in front of the predicted $\hat{J}$ would make it to address a discounted problem instead.

Remark 10: Notice that Algorithms [1] and [2] explicitly involve $L$, if it is available. Another variant of CALF presented in Algorithm [3] lifts this requirement, although both require knowledge of a stabilizing policy.

Remark 11: An immediate hint on the practical implementation of the approach should be made. As Theorem 2 shows, the problem 3 is feasible at every step, up until the system’s state reaches a certain neighbourhood of the origin. Thus, one does not necessarily have to run constrained optimization algorithms. Instead, one might perform unconstrained optimization and check constraint (C4) satisfaction afterwards and reset to the initial admissible solution in case of violation. In this case, the scheme is somewhat reminiscent of the supervisor-based stabilizing reinforcement learning. In fact, one can just add penalty terms involving the constraints to the loss function. These considerations can help speed up the calculations.

Remark 12: A remark should be made about the minimal inference aspect. The constraints (C1)–(C4) in general restrict the learning of the agent, which is evident. However, it must be pointed out that if $\varepsilon_4, \vartheta, \alpha_{\text{low}}$ and $\alpha_{\text{up}}$ are appropriately configured, the optimal critic weights will not be excluded from the set of attainable critic weights. Such a configuration, however may fail to exist if the magnitude of noise is too large, i.e., $\sigma_{\text{max}} \geq \min_{x \in S^*} r(x, \rho^*(x)) / \text{Lip}_j \cdot \text{Lip}_y$, where $\rho^*$ is the optimal policy. The latter can however be remedied by introducing a scaling factor for the cost itself, i.e., $r(\cdot, \cdot) \leftarrow ar(\cdot, \cdot)$, where $a > 1$. This would obviously yield an equivalent problem that can be tuned appropriately.

We can now state the major results about the CALF framework. The present paper proves that imposing the aforementioned constraints will ensure that the system’s state $\xi_t$ will eventually reach a certain neighbourhoods of the origin $B_{\xi^*}$. The latter radius $s^*$ is given by:

\[ s^* = \inf \{ s \in \mathbb{R} | \inf_{s \leq \|x\| \leq s^*} \nu(x) > 6 \text{Lip}_j \sigma_{\text{max}} \text{Lip}_y \}, \]

where $S^*$ is chosen in such a way that $\bar{S} > S^* > \alpha_{\text{up}}^{-1}(\alpha_{\text{up}}(\|X_0\|))$ and the respective Lipschitz constant Lip$_j$ is taken over $B_{S^*}$. For sufficiently small $\delta, \sigma_{\text{max}}$ all of the above values are well defined.

**Theorem 1:** For a sufficiently small $\delta > 0$ the following implication holds:

\[ \forall k \in \mathbb{N}_0 G^\varepsilon_{\vartheta}(w_k, w_{k-1}, \Lambda_{U_{k\delta}}(X_{k\delta}), X_{k\delta}) \leq 0 \]
\[ \vee X_{k\delta} \in B_{\xi^*} \]
\[ \implies \inf \{ t \in [0, +\infty) | \xi_t \in B_{\xi^*} \} < +\infty, \]

where $0 < \varepsilon < \vartheta - (1 + \frac{1}{2}) \text{Lip}_j \sigma_{\text{max}} \text{Lip}_y$.

**Theorem 2:** For a sufficiently small $\delta > 0$ there exists such $\vartheta$ that the following implication holds:

\[ \forall k < K G^\varepsilon_{\vartheta}(w_k, w_{K-1}, \Lambda_u(X_{K\delta}), X_{K\delta}) \leq 0 \vee X_{K\delta} \in B_{\xi^*} \]
\[ \implies \exists w \in \mathbb{W}, u \in \mathbb{U} G^\varepsilon_{\vartheta}(w, w_{K-1}, \Lambda_u(X_{K\delta}), X_{K\delta}) \leq 0 \]
\[ \vee X_{K\delta} \in B_{\xi^*}, \]

where $0 < \varepsilon < \vartheta - (1 + \frac{1}{2}) \text{Lip}_j \sigma_{\text{max}} \text{Lip}_y$.

Here, $s^*$ is the distance from the origin within which we can no longer guarantee feasibility of [3]. Naturally, in the presence of noise with some positive amplitude there will be such a neighborhood of the origin, where it is no longer possible to reliably reduce the distance to the origin. It should be noted here that disturbance attenuation methods might be used to augment the resulting policy to reduce the described noise effect, but it is beyond the scope of this work. The next theorem concerns feasibility of the CALF optimization problem.

The next section presents the experimental evaluation.

**VI. EXPERIMENTS**

**A. Preliminaries of experimental validation**

Experiments were performed with a mobile robot under CALF in Algorithm [4] variation with the critic loss defined
via temporal difference over an experience replay as per (10). The noiseless dynamics part of the robot were described as:

\[ dX_t = f_{\text{Car}}(X_t, U_t) \ dt, \]

where \( T_{\text{Car}}(x_1, x_2, x_3, u_1, u_2) = (u_1 \cos x_3, u_1 \sin x_3, u_2)^T, \)
\( x_1, x_2, x_3 \) are the Cartesian coordinates and angle, and \( u_1, u_2 \)
are the linear and angular speed, respectively. Both actor and critic were implemented as neural networks. The running cost was considered in the following form:

\[ \rho = \chi^T H \chi, \]

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

\[
\begin{align*}
H_1 & = \text{diag} \{10, 10, 10, 0, 0\}, H_2 = \text{diag} \{10, 10, 10, 1, 1\}, \\
H_3 & = \text{diag} \{10, 10, 100, 0, 0\}, H_4 = \text{diag} \{10, 100, 10, 0, 0\}, \\
H_5 & = \text{diag} \{100, 10, 10, 0, 0\}.
\end{align*}
\]

(16)

The model of the mobile robot is also known as the nonholonomic integrator (NI) \( (\dot{x}_{NI1}, \dot{x}_{NI2}, \dot{x}_{NI3})^T = (v_{NI1}, v_{NI2}, x_{NI1} v_{NI2} - x_{NI2} v_{NI1})^T, \) for which various stabilization methods exist. We took \( L(x_{NI}) = \min_{\zeta \in [-\rho, \rho]} \left\{ x_{NI1}^4 + x_{NI2}^4 + (x_{NI1}^2 \cos(\zeta) + x_{NI2}^2 \sin(\zeta) + \sqrt{|x_{NI3}|})^2 \right\} \)
and the corresponding nominal parking policy as per [46], while using a transformation between the NI and Cartesian coordinates of (14) was used.

It should be noted that the goal of the experimental validation was to physically demonstrate the capabilities of the CALF algorithm to improve accumulated cost compared to the nominal policy under various \( H \) matrices. The goal here was not to come up with the best parking controller which can be done by numerous ways, including model-predictive control, sliding-mode control etc.

The software core of this work is the specially designed and programmed Python package for hybrid simulation of agents and environments (generally speaking, not necessarily reinforcement learning agents) called rcognita [47]. Its main idea is to have an explicit implementation of sample-and-hold controls with user-defined sampling time specification. For the purposes of this study, rcognita was coupled with the Robot Operating System (ROS) to communicate with a Robotis TurtleBot 3, the hardware platform used (details follow in the next Section). Figure 1 illustrates a flowchart of interconnection of rcognita and ROS.

B. Hardware and experiment setup

The hardware platform of the current study was a mobile wheeled robot Robotis TurtleBot 3 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.

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. 2 (a).

For each running cost matrix \( H \), 32 runs were performed, i.e., for each target position and, respectively, the nominal and CALF agents. The agent performance was evaluated by the accumulated running cost of 120 s, the total run time. The agent sampling time was set to 0.05 sec which was fairly sufficient for the real-time robot control. The results are presented in the next section.

C. Results

The statistics of the accumulated running cost as per \( \int_0^{120} \chi^T H \chi \ dt \) for different running cost matrices under the CALF and nominal agents are shown in Fig. 3. Each statistic was computed over multiple runs for each agent type and running cost matrix (see Fig. 2). We observed that the CALF agent yielded considerably lower accumulated running cost showing statistically significant improvement under all selected running cost matrices.

VII. DISCUSSION AND EXTENSIONS

It would be fair to point out that in mechanical control engineering it is not uncommon to be aware of a stabilizing policy, while not knowing the corresponding Lyapunov function. However it is not difficult to adapt the proposed approach to a scenario of this kind. For instance, consider Algorithm 3.

The above algorithm relies on similar considerations to those that validate the original results. Though instead of imposing (C2) it instead falls back to the stabilizing policy to maintain decay whenever the critic-based update fails to do so. This way we’re merely performing a kind of on-policy learning, except unlike SARSA the policy that is being evaluated does not perform exploratory actions, but instead sometimes performs stabilizing actions when necessary.

Much like in the case of the original approach, the algorithm ensures a certain kind of stability.

**Theorem 3:** For a sufficiently small \( \delta > 0 \) there exists such \( \nu \), that Algorithm 3 will ensure

\[
\inf \{t \in [0, +\infty) \mid X_t \in B_{\nu^*} \} < +\infty.
\]

(17)
Algorithm 3: Stabilizing-policy-only CALF

**Input:** \( \nu > 0, \zeta > 0, \eta(\cdot) \) – stabilizing policy

\[
x_0 \leftarrow X_0; \\
w_0 \leftarrow \text{arbitrary}; \\
u_0 \leftarrow \eta(x_0); \\
w_{\text{prev}} \leftarrow w_0; \\
x_{\text{prev}} \leftarrow x_0;
\]

for \( k := 1 \ldots \infty \) do

perform \( U_{(k-1)\delta} \); 

\[
x_k \leftarrow \text{observe } X_{k\delta};
\]

Update the critic:

\[
w^* \leftarrow \arg \min_{w \in W} \left( \hat{J}^w(x_{k-1}) - r_{\text{prev}} - \hat{J}^{w_{\text{prev}}}(x_k) \right)^2
\]

s.t. \( \hat{J}^w(x_k) - \hat{J}^{w_{\text{prev}}}(x_{\text{prev}}) \leq -\nu \delta, \\
\alpha_{\text{low}}(\|x_k\|) \leq \hat{J}^w(x_k) \leq \alpha_{\text{up}}(\|x_k\|); \)

if solution \( w^* \) found then

\[
w_k \leftarrow w^*
\]

Update the actor:

\[
U_k \delta \leftarrow \arg \min_{u \in U} \left\{ r(x_k, u) + \hat{J}^{w_k}(A_u(x_k)) \right\};
\]

\[
x_{\text{prev}} \leftarrow x_k \\
w_{\text{prev}} \leftarrow w_k \\
r_{\text{prev}} \leftarrow r(x_k, u_k)
\]

else

Invoke a stabilizing action:

\[
U_k \delta \leftarrow \eta(x_k) \\
r_{\text{prev}} \leftarrow r(x_k, u_k) + r_{\text{prev}};
\]

A brief testing procedure was too performed for Algorithm 3 that involved the same set-up as described in section VI. A brief testing procedure was performed for Algorithm 3 that involved the same set-up as described in section VI. The performance of the algorithm was compared against that of both the nominal policy and the non-stabilizing version of Algorithm 3, i.e., the result of stripping constraints and nominal stabilizing actions from the latter. Table 1 indicates a significant improvement as compared to both baselines.

Table 1: Average total cost

| Algorithm                                      | Total cost |
|-----------------------------------------------|------------|
| Stabilizing-policy-only CALF (Algorithm 3)     | 88.0       |
| Actor-critic without stabilizing constraints   | 152.0      |
| Nominal stabilizing policy                     | 252.0      |
[49] F. Clarke, “Lyapunov functions and discontinuous stabilizing feedback,” *Annual Reviews in Control*, vol. 35, no. 1, pp. 13–33, 2011.

[50] D. Domingo, A. d’Onofrio, and F. Flandoli, “Properties of bounded stochastic processes employed in biophysics,” *Stochastic Analysis and Appl.*, vol. 38, no. 2, pp. 277–306, 2020.

---

**A. Preliminaries**

First note that (1) is in fact equivalent to

\[ dX_t = f(X_t, U_t) \, dt + \sigma(X_t, U_t)Z'_t \, dt, \]

\[ \mathbb{P}[X_0 = \text{const}] = 1, \]

where \( Z' \) is a measurable bounded random process, such that \( \forall t \| Z' \| \leq \text{Lip}_Z \). The latter is motivated by the fact that due to absolute continuity of \( Z \), there always exists such \( Z' \) that \( Z_t = Z_0 + \int_0^t Z'_\tau \, d\tau \), with the last identity implying \( dZ_t = Z'_t \, dt \).

Recall that the statements of the presented theorems concern an arbitrary Lyapunov function \( L \). If a function \( L \) is already known to be a Lyapunov function, then the same applies to \( L' := \zeta^{-1} L \), which makes \( \zeta L = L' \). Thus without loss of generality let’s assume \( \zeta = 1 \) and denote \( w^\# := w^\#_1 \).

Once the system’s state \( X_{k\delta} \) is in \( B_{r'} \subset B \|X_0\| \), which is referred to as the core ball, the setting of \( (U_k, w_k) \) is arbitrary.

This is dictated by the fact that the optimization problem may become infeasible in a small vicinity of the origin due to noise and S&H behavior. Nor is one interested in what happens there as far as S&H-setting is concerned.

Let \( v^* = \hat{\alpha}_{\text{low}}(r) \). At this point, note that for any \( w \in \mathbb{W} \),

\[ \hat{\alpha}_{\text{low}}(\|x\|) \leq \hat{J}^w(x) \leq v^* \implies \|x\| \leq r \]

and also

\[ \hat{\alpha}_{\text{up}}(s^*) \leq \hat{J}^w(x) \leq \hat{\alpha}_{\text{up}}(\|x\|) \implies \|x\| \geq s^* , \]

which relate the value of \( \hat{J} \) to the facts that \( x \in B_r \) or \( s \notin B_{s^*} \), respectively. It can be seen that, among other factors to be detailed later, the bounding functions \( \hat{\alpha}_{\text{low, up}} \) contribute to the radius of the target ball.

Finally, call an actor-critic sequence \( (U_k, w_k)_{k \in \mathbb{N}_0} \) admissible if, for any \( k \in \mathbb{N}_0 \), (C1)–(C4) are satisfied along the corresponding trajectory as long as \( X_{k\delta} \notin B_{s^*} \).

A single element of an actor-critic sequence will be called an actor-critic tuple.

First, let \( S^* > \hat{\alpha}_{\text{low}}^{-1}(\alpha_{\text{up}}(\|X_0\|)) \) and let \( \text{Lip}_f \) be the Lipschitz constant of \( f \) on \( B_{s^*} \).

Now, let

\[ f := \sup_{u \in B_{s^*}} \|f(x, u)\| . \]

**Lemma 1:** If for some \( t \) it is true that \( \|X_t\| \leq \hat{\alpha}_{\text{low}}^{-1}(\alpha_{\text{up}}(\|X_0\|)) \), then \( X_{t+i} \in B_{s^*} \) is implied by \( t < \frac{\hat{\alpha}_{\text{low}}^{-1}(\alpha_{\text{up}}(\|X_0\|))}{\text{Lip}_f + \sigma_{\text{max}} \text{Lip}_Z} \).

**Proof:** Let’s assume that by \( t + \hat{t} \), \( \|X_{t+i}\| \) has reached \( S^* \) for the first time. Then, obviously

\[ S^* - \hat{\alpha}_{\text{low}}^{-1}(\alpha_{\text{up}}(\|X_0\|)) \leq \|X_{t+i}\| - \|X_t\| \leq (\hat{f} + \sigma_{\text{max}} \text{Lip}_Z) \hat{t} . \]

This in turn means that \( \hat{t} \) would have to be at least as large as \( \frac{S^* - \hat{\alpha}_{\text{low}}^{-1}(\alpha_{\text{up}}(\|X_0\|))}{\hat{f} + \sigma_{\text{max}} \text{Lip}_Z} \) for the agent to be able to transcend \( S^* \) by that time.

\[ \blacksquare \]
Lemma 2: Let the following constraints be in place:
\begin{align*}
G^0_w(w_k, w_{k-1}, A_{U_k})(X_{k\delta}), X_{k\delta}) & \leq 0, \\
G^0_z(w_{k+1}, w_k, A_{U_{k+1}})(X_{(k+1)\delta}), X_{(k+1)\delta}) & \leq 0,
\end{align*}
(20)
then if \( \delta \) is sufficiently small and \( \varepsilon < \tilde{\nu} - (1 + \frac{1}{3}\text{Lip}_{\max}\text{Lip}_Z, \text{Lip}_f, \text{Lip}_j) \), the following implication holds:
\begin{align*}
\dot{J}^{w_k}(X_{k\delta}) & \leq \alpha_{up}(\|X_0\|) \implies \\
\dot{J}^{w_{k+1}}(X_{(k+1)\delta}) & \leq \alpha_{up}(\|X_0\|).
\end{align*}
(21)
Proof: First, observe that \( \dot{J}^{w_k}(X_{k\delta}) \leq \alpha_{up}(\|X_0\|) \) implies
\begin{equation}
\dot{\alpha}_{low}(\|X_{k\delta}\|) \leq \dot{J}^{w_k}(X_{k\delta}) \leq \alpha_{up}(\|X_0\|),\quad \|X_{k\delta}\| \leq \dot{\alpha}_{low}^{-1}(\alpha_{up}(\|X_0\|)).
\end{equation}
(22)
Together with Lemma 7 the above yields
\begin{equation}
\forall t \in [k\delta, (k+1)\delta] \quad X_t \in B_{S^*}.
\end{equation}
(23)
The latter justifies the usage of earlier mentioned bounds and Lipschitz constants \( f, \text{Lip}_j, \text{Lip}_f \) that assume \( X_t \in B_{S^*} \). This enables us to derive
\begin{equation}
\begin{aligned}
\|X_{(k+1)\delta} - A_{U_k}(X_{k\delta})\| & \leq \\
& \leq \left\| \int_{k\delta}^{(k+1)\delta} f(X_t, U_{k\delta}) \, dt - f(X_{k\delta}, U_{k\delta}) \delta \right\| + \\
& \left\| \int_{k\delta}^{(k+1)\delta} \sigma(X_t, U_{k\delta})Z_t' \, dt \right\| \\
& \leq \left\| \int_{k\delta}^{(k+1)\delta} f(X_t, U_{k\delta}) - f(X_{k\delta}, U_{k\delta}) \, dt \right\| + \\
& \left\| \int_{k\delta}^{(k+1)\delta} \sigma(X_t, U_{k\delta})Z_t' \, dt \right\| \\
& \leq \left\| \int_{k\delta}^{(k+1)\delta} \|f(X_t, U_{k\delta}) - f(X_{k\delta}, U_{k\delta})\| \, dt + \\
& \left\| \int_{k\delta}^{(k+1)\delta} \|\sigma(X_t, U_{k\delta})Z_t'\| \, dt \right\| \\
& \leq \int_{k\delta}^{(k+1)\delta} \text{Lip}_f(\tilde{f} + \sigma_{\max}\text{Lip}_Z)\delta \, dt + \\
& \int_{k\delta}^{(k+1)\delta} \sigma_{\max}\text{Lip}_Z \, dt \\
& = \text{Lip}_f(\tilde{f} + \sigma_{\max}\text{Lip}_Z)\delta^2 + \sigma_{\max}\text{Lip}_Z \delta.
\end{aligned}
\end{equation}
(24)
Due to \( G^0_w(\ldots) \leq 0 \) the latter implies
\begin{equation}
\begin{aligned}
\dot{J}^{w_{k+1}}(X_{(k+1)\delta}) & \leq \dot{J}^{w_k}(X_{k\delta}) + \varepsilon_1 \leq \\
& \dot{J}^{w_k}(A_{U_k}(X_{k\delta})) + \text{Lip}_j\chi_1(\delta) + \varepsilon_1 + \\
& \dot{J}^{w_k}(X_{k\delta}) - \delta\tilde{\nu} + \varepsilon + \text{Lip}_j\chi_1(\delta) \leq \\
& \alpha_{up}(\|X_0\|) - \delta\tilde{\nu} + \varepsilon + \text{Lip}_j\chi_1(\delta) \leq \\
& \alpha_{up}(\|X_0\|) + \varepsilon + \delta(\text{Lip}_j\sigma_{\max}\text{Lip}_Z - \tilde{\nu}) + \\
& \delta^2\text{Lip}_j\text{Lip}_f(\tilde{f} + \sigma_{\max}\text{Lip}_Z).\quad (25)
\end{aligned}
\end{equation}
Now, pick \( \delta \) to be sufficiently small:
\begin{equation}
\delta < \frac{\tilde{\nu}}{2\text{Lip}_j\text{Lip}_f(\tilde{f} + \sigma_{\max}\text{Lip}_Z)}.
\end{equation}
(26)
Substituting (26) into (25) yields
\begin{equation}
\dot{J}^{w_{k+1}}(X_{(k+1)\delta}) \leq \alpha_{up}(\|X_0\|).
\end{equation}
(27)
Lemma 3: Let \( K \in \mathbb{N} \) and \( w_0 = w^\# \). If \( \tilde{\nu} > 2\text{Lip}_j\sigma_{\max}\text{Lip}_Z + 2\frac{\varepsilon}{5} \), then for a sufficiently small \( \delta \) the following implication holds:
\begin{equation}
\forall k \leq K \quad G^0_w(w_k, w_{k-1}, A_{U_k})(X_{k\delta}), X_{k\delta}) \leq 0 \implies \\
\forall t \in [k\delta, (k+1)\delta] \quad X_t \in B_{S^*}.
\end{equation}
(28)
Proof: Note, that
\begin{equation}
\dot{J}^{w_0}(X_0) = L(X_0) \leq \alpha_{up}(\|X_0\|).
\end{equation}
(29)
By performing mathematical induction with (29) as its base and Lemma 2 as its step we obtain:
\begin{equation}
\forall k \leq K \quad \dot{J}^{w_k}(X_{k\delta}) \leq \alpha_{up}(\|X_0\|).
\end{equation}
(30)
Together with (23) the above yields (28).

The latter lemma justifies the choice of Lipschitz constants and bounds such as for instance (19). The said bounds will have been made heavy use of in the proofs of the presented theorems.

Next, notice that the right-hand side of (1) is continuous in \( x \) in each \( k \)th sampling step. Noticing that both \( f \) and \( \sigma \) are Lipschitz continuous in the first argument furnishes the existence of local (path-wise) solutions. Combining this with the fact that \( S^* \) is not exceeded we conclude existence of global solutions.

B. Proof of Theorem

Let the following hold:
\begin{equation}
\forall k \in \mathbb{N}_0 \quad G^0_w(w_k, w_{k-1}, A_{U_k})(X_{k\delta}), X_{k\delta}) \leq 0, \quad X_{k\delta} \in B_{S^*}.
\end{equation}
(31)
If one assumes (26) together with \( \varepsilon < \tilde{\nu} - (1 + \frac{1}{3}\text{Lip}_j\sigma_{\max}\text{Lip}_Z) \), then for \( X_{k\delta} \in B_{S^*} \setminus B_{S^*} \) we get
\begin{equation}
\exists \Delta > 0 \quad \dot{J}^{w_{k+1}}(X_{(k+1)\delta}) - \dot{J}^{w_k}(X_{k\delta}) \leq \\
\leq -\delta\tilde{\nu} + \text{Lip}_j\text{Lip}_f(\tilde{f} + \sigma_{\max}\text{Lip}_Z)\delta^2 + \\
+ \text{Lip}_j\sigma_{\max}\text{Lip}_Z \delta + \varepsilon < \Delta.
\end{equation}
(32)
Now if we assume the opposite of (12) that leads to
\begin{equation}
\forall k \in \mathbb{N}_0 \quad \dot{J}^{w_k}(X_{k\delta}) < \dot{J}^{w_0}(X_0) + k\Delta, \quad \forall k \in \mathbb{N}_0 \quad \dot{J}^{w_k}(X_{k\delta}) < 0.\quad (33)
\end{equation}
which is in turn a contradiction, since $J^u$ is non-negative.

Thus, it can be concluded that with a sufficiently small $\delta > 0$, the state of the system will be driven to $B_{\rho}$. The time it takes to reach $B_{\rho}$ can be determined in a uniform way from the decay rate and the value of $J$ [18], [27], [29], [48].

C. Proof of Theorem

Let $\nu := 4Lip_j \sigma_{\max} Lip_Z$. Consider a current state $X_{k\delta} \in B_{\rho} \setminus B_{\rho}$ at some time step $k \in \mathbb{N}_0$. If $\delta \leq \delta$ and $U_{k\delta} := \eta(X_{k\delta})$, then by Assumption [12] it holds that

i) $J_w^u(X_{k+1}\delta) - J_w^u(X_{k\delta}) \leq -\delta \nu(X_{k\delta})$,

ii) $\alpha_{\text{low}}(\|X_{k\delta}\|) \leq J_w^u(X_{k\delta}) \leq \alpha_{\text{up}}(\|X_{k\delta}\|)$.

Given these properties, it needs to be shown that [C1]–[C4] are well posed to mean that these constraints are feasible for all times where $X_{k\delta} \not\in B_{\rho}$.

First let’s ensure that $\delta > 0$ is sufficiently small and select and adequate $\varepsilon$.

$$\delta < \frac{\sigma_{\max} Lip_Z}{3 Lip_j(f + \sigma_{\max} Lip_Z)},$$

$$\varepsilon := \left(2 + \frac{2}{3}\right) Lip_j \sigma_{\max} Lip_Z = \frac{\nu}{2} + \frac{\nu}{6}. \quad (34)$$

Note, that the latter implies

$$\text{Lip}_j \text{Lip}_f(\hat{f} + \sigma_{\max} Lip_Z) \delta < \frac{\text{Lip}_j \sigma_{\max} Lip_Z}{3} < \frac{\nu}{12}. \quad (35)$$

From i), it follows that

$$\dot{J}_w^u(\Lambda U_{k\delta}(X_{k\delta})) - J_w^u(X_{k\delta}) \leq -\delta \nu(X_{k\delta}) + \text{Lip}_j \text{Lip}_f(\hat{f} + \sigma_{\max} Lip_Z) \delta^2 + \text{Lip}_j \sigma_{\max} Lip_Z \delta$$

$$\leq -\delta (\nu(X_{k\delta}) - \frac{\nu}{12} - \frac{\nu}{4}) \leq -\delta \nu.$$

Hence, (C3) is satisfied.

Now, note that

$$\dot{J}_w^u(\cdot) := L(\cdot),$$

which directly implies that (C2) is satisfied.

Also, (36) implies

$$\alpha_{\text{low}}(\|X_{k\delta}\|) \leq J_w^u(X_{k\delta}) \leq \alpha_{\text{up}}(\|X_{k\delta}\|) \implies$$

$$\alpha_{\text{low}}(\|X_{k\delta}\|) \leq J_w^u(X_{k\delta}) \leq \alpha_{\text{up}}(\|X_{k\delta}\|). \quad (37)$$

The latter indicates that (C4) is satisfied.

Observe that

$$J_w^{k-1}(X_{k\delta}) \geq J_w^{k-1}(\Lambda U_{(k-1)\delta}(X_{(k-1)\delta})) - \text{Lip}_j \|X_{k\delta} - \Lambda U_{(k-1)\delta}(X_{(k-1)\delta})\| \geq$$

$$L(\Lambda U_{(k-1)\delta}(X_{(k-1)\delta})) - \text{Lip}_j \text{Lip}_f(\hat{f} + \sigma_{\max} Lip_Z) \delta^2 - \text{Lip}_j \sigma_{\max} Lip_Z \delta, \quad (38)$$

while at the same time

$$\dot{J}_w^u(X_{k\delta}) - L(\Lambda U_{(k-1)\delta}(X_{(k-1)\delta})) + \text{Lip}_j \text{Lip}_f(\hat{f} + \sigma_{\max} Lip_Z) \delta^2 + \text{Lip}_j \sigma_{\max} Lip_Z \delta \leq$$

$$2\text{Lip}_j \text{Lip}_f(\hat{f} + \sigma_{\max} Lip_Z) \delta^2 + 2\text{Lip}_j \sigma_{\max} Lip_Z \delta \leq \frac{\nu}{2} \delta + \frac{\nu}{6} \delta = \varepsilon \delta. \quad (39)$$

Substituting (39) into (38) yields

$$J_w^u(X_{k\delta}) - J_w^{k-1}(X_{k\delta}) \leq \varepsilon \delta, \quad (40)$$

which is equivalent to (C1).

Hence, feasibility of CALF is shown.

D. Proof of Theorem

The proof of the theorem technically follows that of Theo-

The difference is in invocations of the stabilizing policy.

Consider Algorithm 3. Let us denote, for brevity:

$$x^\circ := x_{prev},$$

$$J^\circ := J^{prev}(x^\circ).$$

Also, let us denote:

$$\mathcal{K} := \{k \in \mathbb{N}_0 : \text{critic finds a solution } w^{*}\},$$

$$\mathcal{K}_L := \{k \in \mathbb{N}_0 : \eta \text{ is invoked}\}, \quad (42)$$

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 $\eta$ is invoked.

Now, define:

$$x^\circ_k := \begin{cases} x_k, & k \in \mathcal{K}, \\ x^\circ_{k-1}, & k \in \mathcal{K}_L, \end{cases}$$

and, by the same token, $J^\circ_k$.

Notice that, although the user of Algorithm 3 might not be aware of a Lyapunov function associated with a known stabilizing policy, such a Lyapunov function always exists [49]. By Assumption 2, L is a Lyapunov function associated with $\eta$ and $\nu$ is the corresponding sample-wise decay rate.

Notice realizations of CALF as per Algorithms 2 [11] require knowledge of such an $L$, whereas Algorithm 4 only assumes its existence which is justified by converse theorems [49].

Let $L_k$ denote $L(X_{k\delta})$. When running Algorithm 3 we expect the behavior of $L_k$ and $J^\circ_k$ to look schematically like Figure 4 depicts.

It holds that

$$\forall k \in \mathbb{N}_0 \|x^\circ_k\| \leq \frac{\alpha_{low}(J^\circ_k)}{\alpha_{low}(J^\circ_0)}.$$
Fig. 4: Schematic of the critic and a Lyapunov function behavior under Algorithm 3.

whence

\[ \forall k \in \mathbb{N}_0 \| X_{k+1} \| \leq \alpha_{\text{up}}^{-1}(\alpha_{\text{up}}^{-1}(\hat{J}_0^0)), \]

Thus, an overshoot bound \( S^* \) may be set so as to satisfy:

\[ S^* > \alpha_{\text{up}}^{-1}(\alpha_{\text{up}}^{-1}(\hat{J}_0^0)). \tag{43} \]

Now, if \( X_{k+1} \notin B_{s^*} \), then

\[ \nu(X_{k+1}) > 6\text{Lip}_j \sigma_{\text{max}} \text{Lip}_Z. \tag{44} \]

We have, denoting \( \Delta \mathbf{x}_k \equiv \mathbf{x}_{k+1} - \mathbf{x}_k: \)

\[ \forall k \in \mathbb{N}, \Delta \hat{J}_k^0 = \leq -6\text{Lip}_j \sigma_{\text{max}} \text{Lip}_Z \delta, \]

\[ \forall k \in \mathbb{N}, \Delta L_k \leq -6\text{Lip}_j \sigma_{\text{max}} \text{Lip}_Z \delta. \tag{45} \]

If the critic always succeeds, \( B_{s^*} \) would be reached after no more than

\[ \hat{T} := \frac{\hat{J}_0^0 - \alpha_{\text{low}}(s^*)}{6\text{Lip}_j \sigma_{\text{max}} \text{Lip}_Z \delta} \tag{46} \]

steps. If the stabilizing policy were always invoked instead, \( B_{s^*} \) would be reached after no more than

\[ T_L := \frac{L_0 - \alpha_{\text{low}}(s^*)}{6\text{Lip}_j \sigma_{\text{max}} \text{Lip}_Z \delta} \]

steps. But, unlike \( \hat{J}_0^0 \), the Lyapunov function \( L_k \) may step-wise increase when \( k \in \mathbb{N} \) (see Figure 4). However, \( L_k \) never exceeds \( \alpha_{\text{up}}(\hat{J}_0^0) \) as shown above. Thus, let us define

\[ T_L := \frac{\alpha_{\text{up}}(\hat{J}_0^0) - \alpha_{\text{low}}(s^*)}{6\text{Lip}_j \sigma_{\text{max}} \text{Lip}_Z \delta}. \tag{47} \]

This is a reaching time of \( B_{s^*} \) if only \( \eta \) 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 \( T^* := \max\{\hat{T}, T_L\} \).

We now argue what can happen till the ball \( B_{s^*} \) is reached. The two limiting cases, when only the critic succeeds or \( \eta \) is involved, are clear from the reaching time \( T^* \). That is, \( B_{s^*} \) 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 \( \eta \) until \( B_{s^*} \) is “almost” reached, but not quite, followed by another critic success and so on. Thus, the overall reaching time of \( B_{s^*} \) under Algorithm 3 is

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

Remark 13: Since the number of invocations of \( \eta \) is not greater than \( T^* \) till the \( B_{s^*} \) is reached, the critic \( \hat{J}_k^0 \) is effectively a multi-step Lyapunov function, i.e., \( \hat{J}_k^0 \) is non-increasing and

\[ \forall k \in \mathbb{N}, \hat{J}_k^0(T^* + k) < 0. \]

Remark 14: One may ask why not to always use Algorithm 3 if it lifts the requirement on the knowledge of a Lyapunov function. The answer is that the reaching time of Algorithm 3 is much more conservative than that of realizations of CALF that do use a Lyapunov function like Algorithm 2. In fact, there are classes of systems for which Lyapunov functions can be effectively constructed. For instance, the physical energy can be sometimes be used for a Lyapunov function candidate. Furthermore, there is systematic procedures like backstepping which help extend a Lyapunov function to higher dimensions etc. Therefore, it is recommended to use a CALF realization like Algorithm 2 when a Lyapunov function is known, and a realization like Algorithm 3 otherwise.

E. Extra simulation study

This section provides graphical material on some of the results obtained with Algorithm 3 obtained using rcognito.

F. Bounded noise models

This section briefly reflects the key models from the survey [50]. 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 \( Z_t = \sin \left( \frac{2}{\tau_s} Z_t \right) \) with an autocorrelation time parameter \( \tau_s \). Another way is to augment the system description with a dynamical noise model. Thus, the overall description would read, for instance:

\[ dX_t = (f(X_t, U_t) + \sigma(X_t, U_t) Z_t) \, dt, \]

\[ dZ_t = \zeta(Z_t) \, dt + \chi(Z_t) \, dB_t, \tag{49} \]

where \( \{Z_t\} \) is the noise process with an internal model described by the drift function \( \zeta \) and diffusion function \( \chi \). Particular ways to construct the respective stochastic differential equations include the following [50]:

- The Doering-Cai-Lin (DCL) noise

\[ dZ_t = -\frac{1}{\nu} Z_t \, dt + \sqrt{1 - \frac{Z_t^2}{b_1(b_2 + 1)}} \, dB_t, \tag{50} \]

with parameters \( b_2 > -1, b_1 > 0 \);

- The Tsallis-Stariolo-Borland (TSB) noise

\[ dZ_t = -\frac{1}{\nu} Z_t \, dt + \sqrt{\frac{1 - b_2}{b_1}} \, dB_t, \tag{51} \]
with $b_1 > 0$, $b_2 < 1$ parameters;

- Kessler–Sørensen (KS) noise

$$dZ'_t = -\frac{b_3}{\pi b_1} \tan \left( \frac{\pi}{2} Z'_t \right) \, dt + \frac{2}{\pi \sqrt{b_1(b_2+1)}} \, dB_t, \quad (52)$$

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 $\zeta, \Lambda$ do not satisfy Lipschitz conditions in the standard way. Nevertheless, existence and uniqueness of strong solutions can be ensured [50]. 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.

**Fig. 5:** Robot parking using the nominal stabilizing policy. The dither in control near the origin is caused by controller sampling. The accumulated cost is given just for a reference – the stabilizing controller is per se not concerned with any optimization.
Fig. 6: Robot parking using the benchmarking reinforcement learning agent without stabilizing constraints. The parking precision is worse than with the stabilizing controller.
Fig. 7: Robot parking using CALF as in Algorithm 3. The combination of the stabilizing policy with learning allowed for a significantly better stabilization than in both other cases. The accumulated cost is much lower than with the benchmarking agent.