Reinforcement Learning Policies With Local LQR Guarantees For Nonlinear Discrete-Time Systems

Samuele Zoboli, Vincent Andrieu, Daniele Astolfi, Giacomo Casadei, Jilles S Dibangoye, Madiha Nadri

Reinforcement Learning Policies With Local LQR Guarantees For Nonlinear Discrete-Time Systems

Samuele Zoboli[a,b], Vincent Andrieu[a], Daniele Astolfi[a], Giacomo Casadei[c], Jilles S. Dibangoye[b], Madiha Nadri[a]

***Abstract*—Optimal control of nonlinear systems is a difficult problem which has been addressed by both the Control Theory (CT) and Reinforcement Learning (RL) communities. Frequently, the former relies on the linearization of the system thus obtaining only local guarantees. The latter relies on data to build model-free controllers, focused solely on performances. In this paper we propose a methodology to combine the advantages of both approaches, casting the formulation of an optimal local Linear Quadratic Regulator (LQR) into a Deep RL problem. Our solution builds on the linear framework to derive a learnt nonlinear controller showing local stability properties and global performances.**

I. Introduction

In recent years Deep Reinforcement Learning (RL) received considerable attention due to its ability to autonomously learn how to solve complex high-dimensional control tasks, see, e.g., [21], [15], [20]. However, most of state-of-the-art algorithms focus on control performances and very few of them study the stability properties in a Lyapunov sense [9]. This led to a limited use of these methods in real-world applications. Safety requirements or strong uncertainties that cannot be learnt during the training phase eventually reduced their reliability outside of lab environment, due to the unpredictable nature of the learnt controllers.

On the flip side, Control Theory (CT) has a long history of solutions guaranteeing both performances and stability in the linear framework (see, e.g., [10]). Unfortunately, it is hard to extend the theory to unstructured nonlinear systems. A common approach is to design a controller for the linearized model around an equilibrium point. However, this technique restricts the stability and performance guarantees to a neighbourhood of the equilibrium point, denoted as domain of attraction.

The last decades saw increasing interest in linking the fields of CT and RL (see, e.g. [16], [19], [13]), and the inclusion of Lyapunov theory in learning algorithms received increasing attention. A notable model-based solution was proposed in [2], following the work in [1]. However the authors assume a suitable initial safe policy and Lyapunov function are given. In [6], the authors claim near-constraint satisfaction via policy gradient methods by projecting either the policy parameters or the action onto a feasible set described by a Lyapunov constraint. Last, [8] uses a Lyapunov function as critic for guaranteeing stability in the mean cost of the learnt policy.

In this paper we propose a methodology to integrate basic model knowledge in standard model-free RL algorithms. By enforcing a predefined structure to the control policy we provide theoretical guarantees concerning the local stability of the system via Lyapunov indirect theorem. We focus on the optimal control of nonlinear systems under Linear Quadratic cost functions. The approach enhances the capabilities of the linear controller by learning to operate outside its domain of attraction while maintaining its local properties.

The paper is organized as follows: Section II introduces some preliminary notions and states the problem we aim at solving. A solution is proposed in Section III and the learning behavior is discussed in Section IV. An illustration of an inverted pendulum is given in Section V. Conclusions are drawn in Section VI.

II. Preliminaries

A. Dynamic programming concepts

Deterministic dynamic programming deals with discrete-time dynamical systems generating a sequence of states $(x_t)_{t\in\mathbb{N}}$, $x \in \mathbb{R}^n$ under the influence of control inputs $(u_t)_{t\in\mathbb{N}}$, $u \in \mathbb{R}^m$. The objective is to find a sequence of control inputs, often denoted as *control policy* $u = (u_t)_{t\in\mathbb{N}}$, which leads to the minimum of a γ-discounted function (also known as *state-value function* under u), $\mathbf{J}^u : \mathbb{R}^n \to \mathbb{R}$, defined as $\mathbf{J}^u(x) = \sum_{t=0}^{\infty} \gamma^t \mathbf{r}(x_t, u_t)$, where $\gamma \in (0, 1)$. See, e.g., [4]. Hence, the state-value function under an optimal policy is the optimal state-value function $\mathbf{J}^\star$, i.e. the one that provides the minimum cost for each state. In the RL framework, the *action-value function*, $\mathbf{Q}^u : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $\mathbf{Q}^u(x_t, u_t) = \mathbf{r}(x_t, u_t) + \gamma \mathbf{J}^u(x_{t+1})$, turns out to be more practical when dealing with unknown dynamics.

B. Infinite horizon LQR

Consider a deterministic discrete-time linear system

$$x(t+1) = Ax(t) + Bu(t), \qquad t \in \mathbb{N},$$

with state $x \in \mathbb{R}^n$ and control input $u \in \mathbb{R}^m$. For the sake of convenience, we denote $x_t := x(t)$ the state x at discrete-time t. The same notation will adopted for all variables and inputs, obtaining the compact notation

$$x_{t+1} = Ax_t + Bu_t. \tag{1}$$

[a]Université Claude Bernard Lyon 1, CNRS, LAGEPP UMR 5007, 43 boulevard du 11 novembre 1918, F-69100, Villeurbanne, France.

[b]INRIA Chroma team, CITI Lab. INSA Lyon, Villeubanne, France.

[c]Laboratoire Ampere Dpt. EEA of the École Centrale de Lyon, Université de Lyon, 69134 Ecully, France.

This research was partially supported by the French Grant ANR DELICIO (ANR-19-CE23-0006).

We suppose that the pair (A, B) is stabilizable. The infinite-horizon LQR problem asks for a stabilizing controller optimal with respect to a quadratic cost function. By letting $\gamma \in (0, 1]$ be a discount factor, the discounted problem considers the following cost function

$$J_\gamma(x, u) = \sum_{t=0}^{\infty} \gamma^t (x_t^\top Q_\gamma x_t + u_t^\top R_\gamma u_t), \tag{2}$$

where $Q_\gamma = {Q_\gamma}^\top \geq 0$ and $R_\gamma = {R_\gamma}^\top > 0$. In [3, Chapter 4.3] it is shown that, for a given discount factor γ, the optimal controller u^* for (1) with cost (2) is given as

$$u_t^\star = K_\gamma^\star x_t, \quad K_\gamma^\star = -\gamma(R_\gamma + \gamma B^\top P_\gamma B)^{-1} B^\top P_\gamma A, \tag{3}$$

where $K_\gamma^\star$ is the optimal discounted gain and P_γ is the unique solution of the discounted Discrete-time Algebraic Riccati Equation (DARE)

$$P_\gamma = Q_\gamma + \gamma A^\top (P_\gamma - \gamma P_\gamma B (R_\gamma + \gamma B^\top P_\gamma B)^{-1} B^\top P_\gamma) A. \tag{4}$$

Equation (4) can be rewritten and solved as a standard DARE for the new matrices $\widetilde{A} = \sqrt{\gamma} A$, $\widetilde{B} = \sqrt{\gamma} B$. From [5, Section 3], the value functions for the discounted LQR problem under the optimal controller are

$$\mathbf{J}_\gamma^\star(x_t) = x_t^\top P_\gamma x_t, \qquad \mathbf{Q}_\gamma^\star(x_t, u_t) = z_t^\top \mathbf{H}_\gamma z_t, \tag{5}$$

with $z_t := \mathrm{col}(x_t, u_t) \in \mathbb{R}^{n+m}$ and $\mathbf{H}_\gamma$ given by

$$\mathbf{H}_\gamma = \begin{pmatrix} Q_\gamma + \gamma A^\top P_\gamma A & \gamma A^\top P_\gamma B \\ \gamma B^\top P_\gamma A & R_\gamma + \gamma B^\top P_\gamma B \end{pmatrix}.$$

In the discounted framework, closed-loop stability is dependent on the choice of the discount factor γ. In [17, Section IV, Corollary 3], the authors defined a conservative lower bound $\gamma^* \in (0, 1]$, depending on Q_γ and R_γ, such that for any $\gamma \in (\gamma^*, 1]$ the origin of the closed-loop (1), (3) is exponentially stable. Note that for $\gamma = 1$, we recover the so-called "undiscounted LQR" problem in which the cost function reads

$$J(x, u) := J_1(x, u) = \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t, \tag{6}$$

with optimal solution $u^\star$ given by

$$u_t^\star = K^\star x_t, \quad K^\star = -(R + B^\top P B)^{-1} B^\top P A, \tag{7}$$

with P the symmetric positive definite matrix solution of the DARE

$$P = A^\top P A - A^\top P B (R + B^\top P B)^{-1} B^\top P A + Q. \tag{8}$$

For the undiscounted problem, stability of the closed-loop system is always guaranteed.

C. Actor-Critic algorithms

Many state-of-the-art RL algorithms use Actor-Critic architectures [11] (see Figure 1). A key feature of the RL framework is that the environment (plant) returns a *reward* $\mathbf{r} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, which embeds information about the quality of the chosen action. In Actor-Critic algorithms, this signal is used to update the parameter vector $\phi \in \mathbb{R}^q$ of the

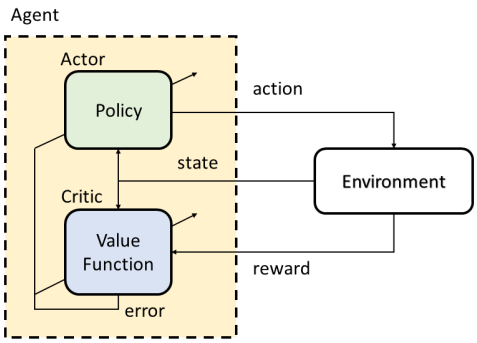


Fig. 1: Actor-Critic structure

critic value function estimator $\hat{\mathbf{Q}}^\phi$, according to a criterion $J^\phi : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q \to \mathbb{R}$. The parameter vector $\theta \in \mathbb{R}^p$ of the actor policy $\boldsymbol{\pi}^\theta$ are subsequently updated according to a different cost function $J^\theta : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ depending on the new critic estimation. We can then define the class of Actor-Critic algorithms $\mathbb{A}$ as the set of operators $\mathbb{A} := \{\mathbf{a} : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^p \times \mathbb{R}^q\}$ with input (θ_0, ϕ_0) and output $(\theta_{\bar{\kappa}}, \phi_{\bar{\kappa}})$ such that, for any $\kappa \in [0, \bar{\kappa}] \subseteq \mathbb{N}$, the parameters ϕ and θ are updated via Stochastic Gradient Descent [23, Chapter 9.3]. The classic formulation imposes $\theta_{\kappa+1} = \theta_\kappa - \lambda_\theta \nabla_\theta J^\theta$ and $\phi_{\kappa+1} = \phi_\kappa - \lambda_\phi \nabla_\phi J^\phi$, where $\lambda_\phi, \lambda_\theta$ are the positive learning rates, subscript κ identifies a parameter vector at iteration κ, ∇_θ is the gradient with respect to the policy parameters, ∇_ϕ is the gradient with respect to the critic value function parameters and $\bar{\kappa} \in \mathbb{N}$ is the iteration index such that $\nabla_\theta J^\theta \leq \epsilon$ and $\nabla_\phi J^\phi \leq \epsilon$ for some small $\epsilon > 0$ and for all $\kappa \geq \bar{\kappa}$. One of the most common objective for updating the critic parameters asks for the minimization of

$$J^\phi(x_t, u_t) = \sum_{t=0}^{\infty} \Big(\mathbf{r}(x_t, u_t) + \hat{\mathbf{Q}}^\phi(x_{t+1}, u_{t+1}) - \hat{\mathbf{Q}}^\phi(x_t, u_t) \Big)^2. \tag{9}$$

Most update procedures for algorithms learning a deterministic policy exploit deterministic policy gradient theorem [22]. Given a parametrized deterministic policy $\boldsymbol{\pi}^\theta$ and a set $\mathcal{S} \subset \mathbb{R}^n$, deterministic policy gradient updates take the form

$$\nabla_\theta J^\theta(x_t, u_t) = \int_{\mathcal{S}} \mathbf{d}^{\boldsymbol{\pi}}(s) \nabla_\theta \boldsymbol{\pi}^\theta(s) \nabla_a \hat{\mathbf{Q}}^\phi(s, a) ds, \tag{10}$$

where ∇_a the gradient with respect to the actions $a = \boldsymbol{\pi}^\theta(s)$ and $\mathbf{d}^{\boldsymbol{\pi}}(s) := \int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} \Pr(x_0) \Pr(x_0 \to s, t, \boldsymbol{\pi}^\theta) ds$ is the discounted state distribution, being $\Pr(x_0 \to s, t, \boldsymbol{\pi}^\theta)$ the probability of reaching the state $s \in \mathcal{S}$ after transitioning for t time steps with initial condition x_0, following the policy $\boldsymbol{\pi}^\theta$.

Remark. In the paper we will focus on algorithms learning the parameter vector θ of a deterministic parametrized policy $\boldsymbol{\pi}^\theta : \mathbb{R}^p \to \mathbb{R}^m$ via (10). However the proposed results can be directly extended to include methods exploiting stochastic policy gradient updates since the proposed solution is independent from the choice of the RL algorithm.

D. Problem statement

Consider a deterministic discrete-time nonlinear system

$$x_{t+1} = f(x_t, u_t), \tag{11}$$

with state solution $(x_t)_{t\in\mathbb{N}}$ taking values in $\mathbb{R}^n$, control input $(u_t)_{t\in\mathbb{N}}$ taking values in $\mathbb{R}$ ($m = 1$), and $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ being a continuously differentiable function in a neighbourhood of the origin. Without loss of generality we suppose that $f(0, 0) = 0$ and we denote $A := \frac{\partial f}{\partial x}(0, 0)$ and $B := \frac{\partial f}{\partial u}(0, 0)$.

Problem 1. *Let J be an undiscounted cost function of the form* (6) *and assume a linearized model of the form* (1) *for* (11) *is known, with (A, B) stabilizable. The goal is to learn an optimal parametrized control policy $\boldsymbol{\pi}^\theta : \mathbb{R}^n \times \mathbb{R}^p \to \mathcal{U} \subseteq \mathbb{R}$ with parameters $\theta \in \mathbb{R}^p$ such that the origin of the closed-loop system* (11) *with $u_t = \boldsymbol{\pi}^\theta(x_t)$ is locally asymptotically stable for all $\theta \in \mathbb{R}^p$, namely*

$$\frac{\partial \boldsymbol{\pi}^\theta}{\partial x}(0) = K^\star, \qquad \forall\, \theta \in \mathbb{R}^p, \tag{12}$$

where $K^\star \in \mathbb{R}^n$ is the LQR optimal gain given in (7)*.*

Remark. Solving the optimal problem implies requirement (12) since $K^\star$ is the unique optimal solution to the local problem. Moreover (12) ensures local stability around the origin by Lyapunov indirect theorem, being $K^\star$ stabilizing for the linearized system. ◁

III. Algorithm design

To solve the problem we rely on deep reinforcement learning algorithms. Hence, we design a suitable cost setting the learning objective. Afterwards, we enforce a specific structure on the learnt policy to guarantee that the approximation constraint (12) is satisfied.

A. Reward shaping

Rewards play a fundamental role in RL. Most RL algorithms require the value function $\mathbf{J}^{\boldsymbol{\pi}}$ to have a finite value in order to converge. However, when defined as an infinite sum over time, boundedness of $\mathbf{J}^{\boldsymbol{\pi}}$ may not be always ensured. Hence, starting from J in (6) we look for a reward function $\mathbf{r} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ and a discount factor $\gamma \in (0, 1)$ defining a suitable γ-discounted function

$$J_{\gamma,\mathrm{RL}}(x, u) = \sum_{t=0}^{\infty} \gamma^t \mathbf{r}(x_t, u_t) \tag{13}$$

which sets the learning objective for the agent.

In the following Lemma, we show that given any undiscounted problem of the form (6)-(7), we can always redefine an associated discounted problem of the form (2)-(3) so that the optimal gains (7) and (3) coincide.

Lemma 1. *Consider system* (1) *and an associated undiscounted optimal control problem of the form* (6)*. Then, for any $\gamma \in (0, 1]$, the optimal gain $K^\star$ defined in* (7) *is the optimal solution of the discounted problem* (2) *with Q_γ, R_γ defined as*

$$Q_\gamma = \gamma Q + (1 - \gamma)P, \qquad R_\gamma = \gamma R. \tag{14}$$

Moreover, the state-value function (5) *of the discounted problem is finite.*

Proof. See Appendix A. □

Based on Lemma 1, the undiscounted cost J in (6) can be replaced by its discounted version (2) by using the weights (14). Thus, we obtain now a suitable form of the cost $J_{\gamma,\mathrm{RL}}$ for RL algorithms without affecting the local optimal solution to the problem. Consequently, the reward $\mathbf{r}$ from (13) can be set as

$$\mathbf{r}(x_t, u_t) = x_t^\top Q_\gamma x_t + u_t^\top R_\gamma u_t, \tag{15}$$

with Q_γ and R_γ defined as in (14). The value of γ is a free parameter to be chosen as the most suitable one for the algorithm convergence.

B. Control policy design

Our goal is to ensure local stability of the closed-loop system (11) with $u_t = \boldsymbol{\pi}^\theta(x_t)$, i.e. to satisfy the local approximation constraint (12) independently from the parameter vector θ. To this end, we enforce a specific structure in our control policy. Given any $k \in \mathbb{N}$, we define $\mathcal{H}_k$ as the set of continuously differentiable functions h satisfying the following

$$\mathcal{H}_k := \Big\{ h \in C^{k+1} : \mathbb{R}^n \to \mathbb{R} : \lim_{s\to 0} \tfrac{h(sx)}{s^j |x|^j} = 0,\ j \in [0, k], \\ \lim_{s\to 0} \tfrac{h(sx)}{s^{k+1}|x|^{k+1}} \neq 0,\ \lim_{|s|\to\infty} h(sx) = 1,\ \forall x \in \mathbb{R}^n \Big\}.$$

The proposed control law is then designed as

$$\begin{aligned} \boldsymbol{\pi}^\theta(x_t) &= \boldsymbol{\pi}_{\mathrm{loc}}(x_t) + \boldsymbol{\pi}^\theta_{\mathrm{learn}}(x_t) \\ \boldsymbol{\pi}_{\mathrm{loc}}(x_t) &= K^\star x_t \\ \boldsymbol{\pi}^\theta_{\mathrm{learn}}(x_t) &= h_1(x_t)\left(\boldsymbol{\mu}^\theta(x_t) - \boldsymbol{\pi}_{\mathrm{loc}}(x_t)\right), \end{aligned} \tag{16}$$

where $h_1 \in \mathcal{H}_1$ and $\boldsymbol{\mu}^\theta : \mathbb{R}^n \to \mathbb{R}$ is a scalar function to be learnt by the RL agent which is parametrized by the set of parameters $\theta \in \mathbb{R}^p$ and satisfies the following assumption:

Assumption 1. *The function $\boldsymbol{\mu}^\theta$ is locally Lipschitz.*

Remark. Since we consider deterministic policy gradient methods (e.g. DDPG [14] and TD3 [7]), the function $\boldsymbol{\mu}^\theta$ denotes the parametrized approximator to be trained. Note that most NNs are locally Lipschitz since they are compositions of locally Lipschitz functions. ◁

Remark. The size of the guaranteed domain of attraction for policy (16) can be controlled by shaping the function h_1. The effect of the learnt component in a neighbourhood of the origin is scaled by such a function. Hence, it is possible to strongly reduce the contribution of $\boldsymbol{\pi}^\theta_{\mathrm{learn}}(x_t)$ in regions where we trust the LQR controller $\boldsymbol{\pi}_{\mathrm{loc}}$ to stabilize the system. ◁

In order to justify our choice for the control policy, in the following Proposition we prove that, by enforcing structure (16) in the control policy and assuming $\boldsymbol{\mu}^\theta$ satisfies Assumption 1, we can learn an arbitrary C^2 policy satisfying the problem constraint (12).

Proposition 1. *The following statements hold.*

1) *Given any* $K^\star \in \mathbb{R}^n$ *and any function* $\boldsymbol{\pi}^\theta : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{\pi}^\theta \in C^2$ *satisfying the local approximation constraint* (12)*, for any function* $h_1 \in \mathcal{H}_1$ *there always exists a locally Lipschitz function* $\boldsymbol{\mu}^\theta : \mathbb{R}^n \to \mathbb{R}$ *satisfying the equality* (16) *for all* $x \in \mathbb{R}^n$.
2) *Let* $\boldsymbol{\mu}^\theta$ *satisfy Assumption 1,* $h_1 \in \mathcal{H}_1$ *and* $\boldsymbol{\pi}^\theta : \mathbb{R}^n \to \mathbb{R}$ *as in* (16)*. Then* (12) *is satisfied.*

Proof. See Appendix B. □

Remark. Due to (16) being always enforced, the local guaranteed properties of classical LQR are ensured even for the untrained policy. ◁

C. Main Result

In this section, we established how to design the objective for the RL agent and how to structure the policy to be learnt. Now we present our solution to Problem 1. It can be solved via Lemma 1 and Proposition 1 by letting Assumption 1 hold.

Theorem 1. *Let be given any algorithm* $\mathbf{a} \in \mathbb{A}$*. Consider the cost* (13) *with the reward function* $\mathbf{r}(x_t, u_t)$ *given by* (15)*. Then by selecting the the control policy* $\boldsymbol{\pi}^\theta$ *as in* (16) *and determining its parameter vector* θ *via* $\mathbf{a}$*, Problem 1 is solved.*

Proof. The reward function defined in (15) allows us to use the discounted cost (13) without affecting the local solution $K^\star$ thanks to Lemma 1, and without restricting our choice of $\gamma \in (0, 1]$. Then (16) ensures constraint (12) is always satisfied via Proposition 1 and Assumption 1. Finally $\mathbf{a} \in \mathbb{A}$ allows the learning of the locally optimal policy parameters. □

The proposed solution is independent from the choice of the deep reinforcement learning algorithm and it can be applied to a wide variety of existing model-free solutions. However we focused on Actor-Critic algorithms, since many of state-of-the art methods exploit the Actor-Critic architecture. Appendix C presents the procedure for learning the controller via an Actor-Critic algorithm.

IV. Learning the policy

In the following section we study the behavior of the proposed solution during the training process. We also propose to structure the value function estimator in order to ensure the correctness of the estimation in the origin.

A. Training behavior

For exploring the training behavior it is sufficient to combine the update law (10) and the policy (16). Plugging the equality (16) in the deterministic policy gradient equation (10) from Section II highlights that

$$\nabla_\theta \boldsymbol{\pi}^\theta(x_t) = h_1(x_t) \nabla_\theta \boldsymbol{\mu}^\theta(x_t), \tag{17}$$

being $\boldsymbol{\mu}^\theta(x_t)$ the only term depending on the parameters. Equation (17) shows that the closer the system gets to the equilibrium point the smaller the updates become, since $h_1 \in \mathcal{H}_1$. Note that one can substitute the policy (16) in the nonlinear model (11) and obtain a new system under the learnt input $\boldsymbol{\mu}^\theta(x_t)$. Hence, the exploration noise which is typically added to deterministic policies can be applied directly on $\boldsymbol{\mu}^\theta(x_t)$. By doing so, during training $\boldsymbol{\pi}^\theta(x_t)$ is a random variable with

$$\begin{aligned} \mathrm{E}\left[\boldsymbol{\pi}^\theta(x_t)\right] &= (1 - h_1(x_t))K^\star x_t + h_1(x_t)\mathrm{E}\left[\boldsymbol{\mu}^\theta(x_t)\right], \\ \mathrm{Var}\left[\boldsymbol{\pi}^\theta(x_t)\right] &= h_1(x_t)^2 \mathrm{Var}\left[\boldsymbol{\mu}^\theta(x_t)\right], \end{aligned}$$

being $\boldsymbol{\mu}^\theta(x_t)$ the only random component. Since $h_1 \in \mathcal{H}_1$, the variance decreases the closer the system is to the equilibrium. Finally, the study suggests that each training episode should be concluded once the equilibrium of the state-space (i.e. an arbitrarily small neighbourhood of it) is reached. This is in accordance with the fact that the agent is actually learning only how to steer the system to the equilibrium point and not how to keep it there, being the solution to the latter problem already provided by the local controller $\boldsymbol{\pi}_{\mathrm{loc}}$.

B. Improving learning by reshaping the value function estimation

For designing critics, model-free RL algorithms usually rely on value functions in order to drive the policy towards the optimal solution. We propose a value function estimator built on the knowledge of the policy (16) and of the linear framework. Suppose we are interested in the action-value function under the control policy (16). Let us denote the its estimator by $\hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}}(x_t, u_t)$. Due to the structure of $\boldsymbol{\pi}^\theta$, equation (5) provides a suitable local approximation. Consequently, we impose the following local constraint $\forall z_t \in \mathcal{Z}$, $\mathcal{Z} = \{z_t \in \mathbb{R}^{n+1} : z_t = (0, u_t), \forall u_t \in \mathcal{U}\}$

$$D^i \hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}}(z_t) = D^i \mathbf{Q}^\star_\gamma(z_t),\ |i| \leq 2\,, \tag{18}$$

with D^i denoting the derivative of order i with the multi-index notation for multi-variable functions (see, Appendix D). Similarly to the policy equation (16), the estimated action-value function $\hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}}(x_t, u_t)$ for the nonlinear system (11) under (16) is modeled as

$$\begin{aligned} \hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}}(x_t, u_t) &= \mathbf{Q}_{\mathrm{loc}}(x_t, u_t) + \hat{\mathbf{Q}}^\phi_{\mathrm{learn}}(x_t, u_t) \\ \mathbf{Q}_{\mathrm{loc}}(x_t, u_t) &= \mathbf{Q}^\star_\gamma(x_t, u_t) \\ \hat{\mathbf{Q}}^\phi_{\mathrm{learn}}(x_t, u_t) &= h_2(x_t)(\boldsymbol{\Omega}^\phi(x_t, u_t) - \mathbf{Q}_{\mathrm{loc}}(x_t, u_t)), \end{aligned} \tag{19}$$

where $h_2 \in \mathcal{H}_2$, $\boldsymbol{\Omega}^\phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is a parametrized function whose parameter vector $\phi \in \mathbb{R}^q$ is learnt by the RL agent and satisfying the following assumption:

Assumption 2. *The function* $\boldsymbol{\Omega}^\phi$ *is locally Lipschitz.*

As for the control policy, we justify the design choice (19) via the following Proposition, showing that (19) and Assumption 2 allow learning a generic C^3 function satisfying the local constraint (18).

Proposition 2. *The following statements hold.*

1) *Given any* $\mathbf{Q}_{\mathrm{loc}} \in: \mathbb{R}^{n+1} \to \mathbb{R}, \mathbf{Q}_{\mathrm{loc}} \in C^3$ *and any function* $\hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}} : \mathbb{R}^{n+1} \to \mathbb{R}, \hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}} \in C^3$ *satisfying the local approximation constraint* (18) *for any* $h_2 \in \mathcal{H}_2$*, there always exists a locally Lipschitz function* $\boldsymbol{\Omega}^\phi : \mathbb{R}^{n+1} \to \mathbb{R}$ *satisfying the equality* (19) *for all* $x \in \mathbb{R}^n, \forall u \in \mathcal{U} \subseteq \mathbb{R}$.

2) *Let* $\mathbf{\Omega}^{\phi}$ *satisfy Assumption 2,* $h_2 \in \mathcal{H}_2$ *and* $\hat{\mathbf{Q}}^{\phi}_{\boldsymbol{\pi}} : \mathbb{R}^{n+1} \to \mathbb{R}$ *as in* (19)*. Then* (18) *is satisfied.*

Proof. The proof follows the same steps performed in the proof of Proposition 1. Note that (18) holds for all points in $\mathcal{Z}$ instead of the single point scenario of Proposition 1. Moreover, we match the second order approximation via $h_2 \in \mathcal{H}_2$. □

V. Simulation Results

We run simulations in a frictionless inverted pendulum environment. The system nonlinear model is

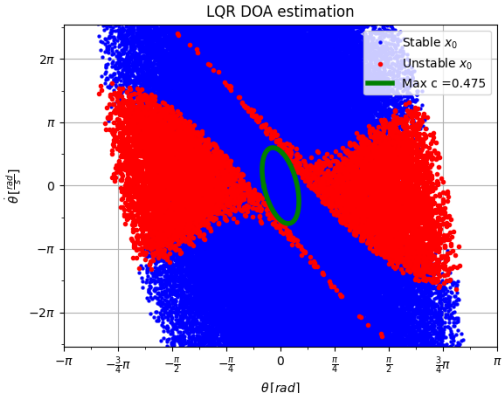
$$\alpha_{t+1} = \alpha_t + \omega_t \, \Delta t$$
$$\omega_{t+1} = \omega_t - \frac{3g}{2\ell} \sin(\alpha_t + \pi)\Delta t + \frac{3}{2m\ell^2}[\mathrm{sat}(u_t) + d_t]\Delta t,$$

where $\alpha \in [-\pi, \pi)$ is the angle between the position of the pendulum and the top vertical one, $\omega \in \mathbb{R}$ is its rate of change, $\mathrm{sat}(\cdot)$ is a saturation function limiting the control input torque u_t in $[-2, 2]$, d_t is wind disturbance affecting the system, $\Delta t = 0.05$ is the discretization step, $g = 10$ is the gravitational acceleration, $\ell = 1$ is the length of the pendulum and $m = 1$ is its mass. We denote $x_t := (\alpha_t, \omega_t)^\top$. The goal is to stabilize the pendulum at the unstable equilibrium $(x^\star, u^\star) = (0, 0)$, corresponding to the top vertical position, starting from any random initial condition. The parameters of the discounted LQR problem for the linearized system are

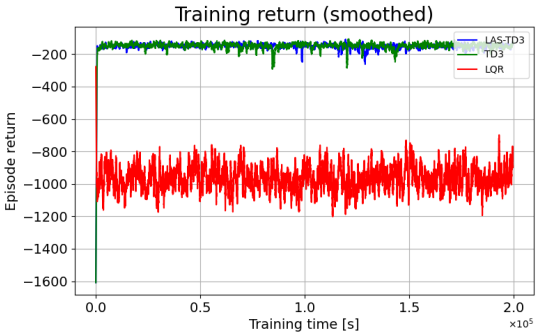
$$Q_\gamma = \begin{pmatrix} 1 & 0 \\ 0 & 0.1 \end{pmatrix}, \quad R_\gamma = 0.001, \quad \gamma = 0.99,$$
$$K^\star = \begin{pmatrix} -19.3006 & -5.9918 \end{pmatrix}, \quad P = \begin{pmatrix} 8.088 & 0.4782 \\ 0.4782 & 0.1624 \end{pmatrix}.$$

We train the proposed policy using a state-of-the-art deep reinforcement learning algorithm, namely TD3 [7]. The nominal version of the learning algorithm comes from Stable Baselines 3 library [18] and it is adapted to include our policy (16) and value function (19), following the steps in Appendix C. Since most of RL algorithms are designed to maximize the expected reward, we simply invert the sign of $\mathbf{r}(x_t, u_t)$ and $\mathbf{Q}_{\mathrm{loc}}(x_t, u_t)$. Each training episode is stopped as soon as the state enters a small circular area of radius 10^{-5} centered in the origin, or their time limit is reached. We use the functions $h_1(x) = \tanh(\tanh^{-1}(0.99)\frac{x^\top P x}{c})$ and $h_2(x) = \tanh(\tanh^{-1}(0.99)(\frac{x^\top P x}{c})^{\frac{3}{2}})$, which saturate outside the Lyapunov level set $\{V(x) = c\}$, with $c \in \mathbb{R}_{>0}$. The value of c is estimated by sampling random initial conditions and testing the convergence to the equilibrium point, see Figure 2a, and we select $c = 0.47$.

We evaluate the performances and stability of our solution by comparing the results with the simple LQR and the nominal version of the algorithm. Standard TD3 is trained with the same reward function (15). In order to allow good training, episodes are not stopped as soon as a neighbourhood of the origin is reached, but at end of their time length limit. We use the same hyperparameters for both the standard and the locally asymptotically stable version of the algorithm: 3-layers fully-connected NNs with ReLU activation functions,



(a) Estimated level set for saturated LQR



(b) Episode return during training

64 units for the first hidden layer and 32 units as the second one, both for the critic and for the actor, and a learning rate of $\lambda = 0.00371$.

We first analyze the learning performances. For each training episode $d_t = 0$, $\forall t \in \mathbb{R}$ and the initial condition is randomly sampled. From Figure 2b we can infer that classical LQR succeeds only if the initial state lies inside it's domain of attraction. Its performances strongly fluctuates since the magnitude of the return of an episode (i.e. the final cost of the episode) is very big (bad) if the initial condition lies outside of its domain of attraction or quite small (good) if the system starts close to the equilibrium. On the other hand, our solution behaves comparably to the nominal TD3 algorithm.

We also study the stability of the closed loop system. During training, we periodically evaluate the policy by running an experiment in a different environment corrupted by uncertainties. We simulate measurement noise $u_t = \boldsymbol{\pi}^\theta(x_t + w_t)$, $w_t \sim \mathcal{N}(0, 0.03)$ and mass mismatch $m_{\mathrm{real}} = 1.2 m_{\mathrm{train}}$. Moreover, the input is perturbed by external sinusoidal wind $d_t = 0.36 \sin(\frac{\pi}{50} t)$. A "stability score" is extracted as the maximum of the norm of the error vector at steady state. We run the stability evaluation episodes twice, at first with initial condition $x_0 = (0.945\pi, 0)^\top$, close to the position "down" of the pendulum, then with $x_0 = (0, 0)^\top$, corresponding to the desired equilibrium point. Figure 3 clearly shows that the RL approach focuses solely on performances. Moreover, we note that as $h_1(x) \to 1$, the learnt component becomes predominant, possibly affecting stability.

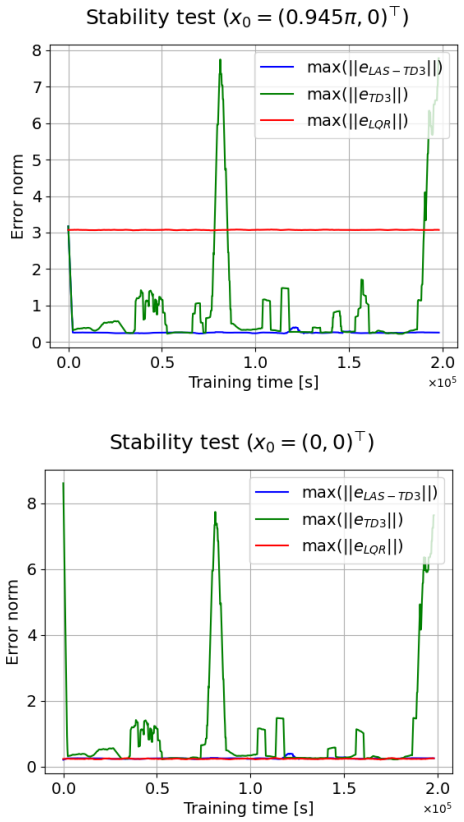


Fig. 3: Stability of policies during training (smoothed)

VI. Conclusions

We presented a solution addressing the design of LQR controller for nonlinear system. By uniting basic knowledge on the system behavior around an equilibrium point and deep reinforcement learning techniques we provided a controller ensuring classical local LQR guarantees and experimental global attractiveness. The proposed method is directly implementable starting from standard RL algorithms. Even if the procedure is proposed for single input systems, it can be straightforwardly expanded to higher dimensions. Simulations proved our solution improves both the classical control via linearization and standard RL algorithms.

References

[1] F. Berkenkamp et al. "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 4661–4666.

[2] F. Berkenkamp et al. "Safe model-based reinforcement learning with stability guarantees". In: *arXiv preprint arXiv:1705.08551* (2017).

[3] D. P. Bertsekas. "Dynamic Programming and Optimal Control 4th Edition, Volume II Chapter 4 Noncontractive Total Cost Problems". In: (2018).

[4] D. P. Bertsekas. *Reinforcement Learning and Optimal Control*. 2nd Printing. Athena scientific, 2018.

[5] S-J. Bradtke. "Reinforcement learning applied to linear quadratic regulation". In: *Advances in neural information processing systems*. 1993, pp. 295–302.

[6] Y. Chow et al. "Lyapunov-based safe policy optimization for continuous control". In: *arXiv preprint arXiv:1901.10031* (2019).

[7] S. Fujimoto, H. Hoof, and D. Meger. "Addressing function approximation error in actor-critic methods". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1587–1596.

[8] M. Han et al. "Actor-critic reinforcement learning for control with stability guarantee". In: *arXiv preprint arXiv:2004.14288* (2020).

[9] H-K. Khalil. *Nonlynear systems*. 3rd ed. 2002.

[10] D-E. Kirk. *Optimal Control Theory: An Introduction*. First Printing. Prentice-Hall networks series. Prentice Hall, 1970.

[11] V. R. Konda and J. N. Tsitsiklis. "Actor-critic algorithms". In: (2000), pp. 1008–1014.

[12] K. Königsberger. *Analysis 2*. Springer-Verlag, 2013.

[13] V. CV Kumar, S. Ha, and K. Yamane. "Improving model-based balance controllers using reinforcement learning and adaptive sampling". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7541–7547.

[14] T. P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[15] V. Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[16] T. J. Perkins and A. G. Barto. "Lyapunov design for safe reinforcement learning". In: *Journal of Machine Learning Research* 3.Dec (2002), pp. 803–832.

[17] R. Postoyan et al. "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost". In: *IEEE Transactions on Automatic Control* 62.6 (2016), pp. 2736–2749.

[18] A. Raffin et al. *Stable Baselines3*. https://github.com/DLR-RM/stable-baselines3. 2019.

[19] J. Randløv, A. Barto, and M. Rosenstein. "Combining reinforcement learning with a local control algorithm". In: *Proc. Seventeenth Intl. Conf. on Machine Learning* (2000), pp. 775–782.

[20] J. Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[21] D. Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144.

[22] D. Silver et al. "Deterministic policy gradient algorithms". In: *International conference on machine learning*. PMLR. 2014, pp. 387–395.

[23] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

APPENDIX

A. Proof of Lemma 1

If $\gamma = 1$ the proof is trivial, hence we will focus on $\gamma \in (0,1)$. First, note that since Q, P and R are symmetric positive definite matrices, so are Q_γ and R_γ for any $\gamma \in (0,1)$. By multiplying both sides of the DARE (8) by γ and by using (14), we get

$$P = Q_\gamma + \gamma A^\top PA - \gamma^2 A^\top PB(R_\gamma + \gamma B^\top PB)^{-1}B^\top PA.$$

Under standard LQR problem assumptions on stabilizability and observability of (1), the solution to the Riccati equation is unique. Hence, from the definition of the discounted DARE (4) we can conclude that $P = P_\gamma$. By inserting the latter and (14) in the optimal gain (7) we obtain

$$K^\star = -\gamma(R_\gamma + \gamma B^\top P_\gamma B)^{-1}B^\top P_\gamma A = K^\star_\gamma,$$

which shows that (7) is also solution to the discounted problem defined by (2) when considering weights in the form of (14).

The state-value function for a given policy is defined as the *cost-to-go* starting from an initial state and following the policy. Given a quadratic cost function as in (6), it holds that

$$\mathbf{J}^u(x_{t+1}) - \mathbf{J}^u(x_t) = -x_t^\top Q x_t - u_t^\top R u_t. \tag{20}$$

By plugging the matrices (14) in (20) and considering the value function (5) with (8) for the undiscounted problem under the optimal input (7), yields

$$\mathbf{J}^\star(x_{t+1}) = \gamma^{-1}\mathbf{J}^\star(x_t) - \gamma^{-1}(x_t^\top Q_\gamma x_t + u_t^{\star\top} R_\gamma u_t^\star). \tag{21}$$

Then, by solving (21) we have

$$\gamma^t \mathbf{J}^\star(x_t) = \mathbf{J}^\star(x_0) - \sum_{j=0}^{t-1} \gamma^j (x_j^\top Q_\gamma x_j + u_j^{\star\top} R_\gamma u_j^\star), \tag{22}$$

where $\mathbf{J}^\star(x_0) \in \mathbb{R}$ denotes the initial condition. Since $\gamma \in (0,1)$ by letting $t \to \infty$ equation (22) simplifies in

$$\sum_{j=0}^{\infty} \gamma^j (x_j^\top Q_\gamma x_j + u_j^{\star\top} R_\gamma u_j^\star) = \mathbf{J}^\star(x_0). \tag{23}$$

Being $\mathbf{J}^\star(x_0)$ finite for the linear system (1) under the optimal controller (7), the state-value function for the discounted problem under the undiscounted solution (7) is also finite and this concludes the proof.

B. Proof of Proposition 1

Let us address point 1. By keeping in mind the multi-index properties provided in Appendix D, we consider a multi-index $i \in \mathbb{N}^n$. Constraint (12) implies

$$D^i \boldsymbol{\pi}^\theta(0) = D^i \boldsymbol{\pi}_{\text{loc}}(0), \qquad |i| \leq 1. \tag{24}$$

Select $x_0 = 0$. Since $\boldsymbol{\pi}_{\text{loc}} \in C^2$, if $\boldsymbol{\pi}^\theta \in C^2$ by Lemma 2 in Appendix D, in a neighbourhood of the equilibrium we obtain $\boldsymbol{\pi}^\theta(x_t) = \boldsymbol{\pi}_{\text{loc}}(x_t) + o(x_t^3)$. It is possible to find a suitable definition of $\boldsymbol{\pi}^\theta_{\text{learn}}$ satisfying equation (16) by recalling that $h_1 \in \mathcal{H}_1$. This shows the first item of the proposition. We prove now the second item. If (16) is satisfied $\forall x_t \in \mathbb{R}^n$, constraint (12) is verified if and only if

$$\lim_{s\to 0} \frac{h_1(sx_t)\boldsymbol{\pi}^\theta_{\text{learn}}(sx_t)}{s|x_t|} = 0, \qquad \forall x_t \in \mathbb{R}^n, \tag{25}$$

where $s \in \mathbb{R}$. If $\boldsymbol{\pi}^\theta_{\text{learn}}$ is a locally Lipschitz function, then $\lim_{s\to 0} \left|\boldsymbol{\pi}^\theta_{\text{learn}}(sx_t)\right| \leq \omega_\pi$ for all $x_t \in \mathbb{R}^n$, being $\omega_\pi \in \mathbb{R}_{\geq 0}$. This implies that (25) holds if $h_1 \in \mathcal{H}_1$ and this concludes the proof.

C. Locally Asymptotically Stable Actor-Critic

Algorithm 1 presents the procedure to implement the proposed solution with an Actor-Critic algorithm. It takes as input the undiscounted problem formulation, the linearized matrices, an Actor-Critic algorithm and a discount factor. Successively, it computes the linear optimal solution and the associated discounted problem and sets the saturation functions enforcing the local stability. Finally, it sets up the structured policy and value function(s) and runs the Actor-Critic algorithm for learning the parameters. The output is a learnt optimal locally asymptotically stable deterministic policy.

Algorithm 1: LAS-Actor-Critic

Input: (A,B), Q, R, γ, $\mathbf{a} \in \mathbb{A}$;
Compute P, $K^\star$, $\mathbf{H}_\gamma$, Q_γ, R_γ;
Pick $h_1 \in \mathcal{H}_1$, $h_2 \in \mathcal{H}_2$;
Set $\mathbf{r}$ as in (15);
Set $\boldsymbol{\pi}^\theta$ as in (16) with parameters θ_0;
Set $\hat{\mathbf{Q}}^\phi_{\boldsymbol{\pi}}$ as in (19) with parameters ϕ_0;
Run $\mathbf{a}(\theta_0, \phi_0)$;
Result: LAS control policy $\boldsymbol{\pi}^\theta$

D. Matching approximations

In this section we will identify by x_i the i^{th} component of vector $x \in \mathbb{R}^n$, instead of identifying a time index. Firstly, let us recall some multi-index definitions. For a general multi-index $\alpha \in \mathbb{N}^n$ we denote

$$|\alpha| = \alpha_1 + \cdots + \alpha_n,\ \alpha! = \alpha_1! \ldots \alpha_n!,\ x^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n},$$

for any $x \in \mathbb{R}^n$. Then, given a function $\Phi : \mathbb{R}^n \to \mathbb{R}$ whose l-th order partial derivatives are continuous, it is possible to define its derivative of order l as

$$D^\alpha \Phi = \frac{\partial^{|\alpha|}\Phi}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}} \qquad |\alpha| = l\,.$$

Finally, we define with $o(x^{\nu+1})$ the standard little-o notation for functions of order smaller than $|x|^\nu$. We state now the following lemma.

Lemma 2. *For a given point $y \in \mathbb{R}^n$ and two real-valued functions $\Psi_1, \Psi_2 \in C^{\nu+1} : \mathbb{R}^n \to \mathbb{R}$ such that $D^i\Psi_1(y) = D^i\Psi_2(y)$, $|i| \leq \nu$ where $i \in \mathbb{N}^n$ is a multi index, it holds that $\Psi_1(x) = \Psi_2(x) + o(x^{\nu+2})$.*

Proof. Introduce the multi-indices $i, j \in \mathbb{N}^n$, $i = (i_1, i_2, \ldots, i_n)$, $j = (j_1, j_2, \ldots, j_n)$. Consider an arbitrary

real-valued function $\Phi : \mathbb{R}^n \to \mathbb{R}, \Phi \in C^{l+1}, l \in \mathbb{N}$. Given a point $y \in \mathbb{R}^n$, by Taylor's theorem for multivariate functions [12] it holds that

$$\Phi(x) = \sum_{|i| \leq l} \frac{D^i \Phi(y)}{i!}(x-y)^i + \sum_{|j|=l+1} \Upsilon^{\Phi}(x)(x-y)^j,$$

where $\Upsilon^{\Phi}(x) = \frac{|j|}{j!} \int_0^1 (1-t)^{|j|-1} D^j \Phi(y+t(x-y))dt$. Then, if $\Psi_1, \Psi_2 \in C^{\nu+1}$ they can be equivalently expressed as

$$\Psi_1(x) = \sum_{|i| \leq \nu} \frac{D^i \Psi_1(y)}{i!}(x-y)^i + o(x^{\nu+2}),$$

$$\Psi_2(x) = \sum_{|i| \leq \nu} \frac{D^i \Psi_2(y)}{i!}(x-y)^i + o(x^{\nu+2}).$$

If $D^i \Psi_1(y) = D^i \Psi_2(y)$, $|i| \leq \nu$ we can rearrange the last identity to obtain

$$\sum_{|i| \leq \nu} \frac{D^i \Psi_1(y)}{i!}(x-y)^i = \Psi_2(x) - o(x^{\nu+2}).$$

Finally, by combining all previous equation we obtain $\Psi_1(x) = \Psi_2(x) + o(x^{\nu+2})$, thus completing the proof. $\square$