Guaranteeing Control Requirements via Reward Shaping in Reinforcement Learning

Francesco De Lellis, *Member, IEEE*, Marco Coraggio, *Member, IEEE*, Giovanni Russo, *Senior Member, IEEE*, Mirco Musolesi, *Member, IEEE*, and Mario di Bernardo, *Fellow, IEEE*

***Abstract*— In addressing control problems such as regulation and tracking through reinforcement learning (RL), it is often required to guarantee that the acquired policy meets essential performance and stability criteria such as a desired settling time and steady-state error before deployment. Motivated by this, we present a set of results and a systematic reward-shaping procedure that: 1) ensures the optimal policy generates trajectories that align with specified control requirements and 2) allows to assess whether any given policy satisfies them. We validate our approach through comprehensive numerical experiments conducted in two representative environments from OpenAI Gym: the Pendulum swing-up problem and the Lunar Lander. Utilizing both tabular and deep RL methods, our experiments consistently affirm the efficacy of our proposed framework, highlighting its effectiveness in ensuring policy adherence to the prescribed control requirements.**

***Index Terms*— Computational control, deep reinforcement learning (RL), learning-based control, policy validation, reward shaping.**

I. Introduction

THE paradigm of using reinforcement learning (RL) for control system design has gained substantial traction due to its ability to autonomously learn policies that effectively address complex control problems, relying solely on data and employing a reward maximization process. This approach finds diverse applications, spanning from attitude control [1] and wind farm management [2] to autonomous car-driving [3] and the regulation of plasma using high-fidelity simulators [4]. However, a significant challenge in this domain revolves around ensuring that the learned control policy demonstrates the desired closed-loop performance and steady-state error, posing a crucial open question in control system design.

It is often argued that accurate knowledge of system dynamics is necessary to provide analytical guarantees of stability and performance, which is crucial for industrial applications [5], [6]. In fact, in this article, we introduce a set of analytical results and a constructive procedure for shaping the reward function of approaches based on RL (tabular and function approximation methods that rely on deep learning). The goal is to derive a learned policy that is obtained without the use of a mathematical model of the system dynamics, able to verifiably meet predetermined control requirements in terms of desired settling time and steady-state errors.

In the literature, *reward shaping*, consisting of modifying the reward function to improve learning or control performance, has mostly been used to increase sample efficiency [7], [8], [9], rather than providing guarantees on the learned policy. An early example was presented in [7], where an agent was trained to ride a bicycle exploiting a reward-shaping mechanism. More recently, reduced sample complexity was demonstrated in [9] for a modified Upper Confidence Bound algorithm using shaped rewards. In [8], it was shown that adding a function of the state to the reward keeps the optimal policy unchanged if and only if the function is potential-based. A method to select potential-based functions is presented and validated analytically in [10], requiring knowledge of an appropriate Lyapunov function, to guarantee convergence to a state under the optimal policy. While this result can be used to solve regulation problems, it does not ensure a specific settling time. Moreover, finding a Lyapunov function is often cumbersome for many real-world problems.

When guarantees are given on RL control [6], they are typically provided in terms of reachability of certain subsets of the state space [11], [12], or in terms of *safety* during learning and/or for the learned policy. Namely, in [11], RL is used to select a control law among a set of candidates, using Lyapunov functions to ensure a system enters a goal region with unitary probability, under certain conditions on the controllers. In [12], a partially known system model is

Manuscript received 14 March 2024; accepted 19 April 2024. This work was supported in part by the Research Project "SHARESPACE" funded by the European Union (EU HORIZON-CL4-2022-HUMAN-01-14. SHARESPACE. GA 101092889—http://sharespace.eu) and in part by the Research Project PRIN 2022 "Machine-learning based control of complex multiagent systems for search and rescue operations in natural disasters (MENTOR)" funded by the Italian Ministry of University and Research (2023–2025). The work of Giovanni Russo was supported by the MOST–Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU [PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR)–MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4–D.D. 1033 17/06/2022] under Grant CN00000023. Recommended by Associate Editor M. Abbaszadeh. *(Corresponding author: Mario di Bernardo.)*

Francesco De Lellis is with the Department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125 Naples, Italy (e-mail: francesco.delellis@unina.it).

Marco Coraggio is with the School for Advanced Studies, Scuola Superiore Meridionale, 80138 Naples, Italy (e-mail: marco.coraggio@unina.it).

Giovanni Russo is with the Department of Computer and Electrical Engineering and Applied Mathematics, DIEM, University of Salerno, 840484 Salerno, Italy (e-mail: giovarusso@unisa.it).

Mirco Musolesi is with the Department of Computer Science, University College London, WC1E 6BT London, U.K., and also with the Department of Informatics—Science and Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: m.musolesi@ucl.ac.uk).

Mario di Bernardo is with the Department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125 Naples, Italy, and also with the School for Advanced Studies, Scuola Superiore Meridionale, 80138 Naples, Italy (e-mail: mario.dibernardo@unina.it).

Digital Object Identifier 10.1109/TCST.2024.3393210

used to improve a safe starting policy through RL, avoiding actions that bring the system out of the basin of attraction of a desired equilibrium. Both approaches [11], [12] do not provide guarantees on the time required to reach the desired regions. Safety for RL control has been extensively explored in the literature using various frameworks, such as *constrained Markov decision processes* [13], "shields" [14], *control barrier functions* [15], and a combination of model predictive control and RL [16]. Although these techniques ensure the avoidance of unsafe subsets of the state space, they generally do not provide guarantees on reaching a specific goal region or on control performance metrics, such as settling time.

The problem of synthesizing rewards for control tasks is also the subject of *inverse optimal control* (IOC) [17], focusing on estimating the rewards associated with given observations of states and control inputs, assuming closed-loop stability and/or policy optimality. Initially aimed at determining control functions producing observed outputs [18], IOC has since been connected to RL [19], applied in nonlinear, stochastic environments [20], and its framework has been used to investigate the cost design problem [21]. However, to the best of our knowledge, IOC has not been used specifically to design reward functions that, when optimized, can guarantee specific control performance.

Given a regulation or tracking problem with predetermined stability and performance requirements on steady-state errors and settling time, we advance the state of the art as follows.

1) We introduce a model-free sufficient condition on the discounted return associated with a trajectory to determine if it is *acceptable* (i.e., it satisfies the control requirements).
2) We give a sufficient condition to assess whether a learned policy leads to an acceptable closed-loop trajectory.
3) We provide a procedure to shape the reward function so that the above conditions can be applied to a system of interest and that the optimal policy is acceptable.
4) We successfully validate the approach through two representative control problems from OpenAI Gym [22]: the stabilization of a pendulum [23], and performing landing in the Lunar Lander environment [24].

For reproducibility, the code is available on GitHub [25].

The rest of this article is organized as follows. In Section II, we formalize the problem of constructing a reward function for learning-based control. The main results of our approach are then presented in Section III and validated via numerical simulations on two representative application examples in Section IV. Concluding remarks are given in Section V.

II. Problem Statement

A. Problem Setup

We consider a discrete-time dynamical system of the form

$$x_{k+1} = f(x_k, u_k), \quad x_0 = \tilde{x}_0 \tag{1}$$

where $k \in \mathbb{N}_{\geq 0}$ is the discrete time, $x_k \in \mathcal{X}$ is the *state* at time k, $\mathcal{X}$ is the *state space*, $\tilde{x}_0 \in \mathcal{X}$ is an *initial condition*, $u_k \in \mathcal{U}$ is the *control input* (or *action*) at time k, $\mathcal{U}$ is the *set of feasible inputs*, and $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the system *dynamics*.

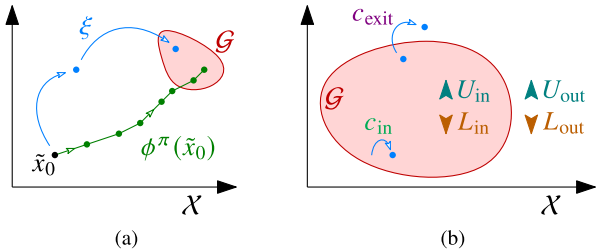


Fig. 1. (a) State-space sequence ξ, a trajectory $\phi^\pi(\tilde{x}_0)$, and a goal region $\mathcal{G}$ (see Section II), while a state-space sequence is simply a sequence of points in the state space $\mathcal{X}$, a trajectory is generated by applying a policy to the dynamics in (1). (b) Terms of the reward structure in Assumption III.1.

Furthermore, we let $\pi : \mathcal{X} \to \mathcal{U}$ be a *control policy* and $u_k = \pi(x_k)$. Let also $\mathcal{X}^\infty := \mathcal{X} \times \cdots \times \mathcal{X}$, with the Cartesian product being applied an infinite number of times. We denote by $\phi^\pi(\tilde{x}_0) \in \mathcal{X}^\infty$ the *trajectory* obtained by applying policy π to system (1) starting from $\tilde{x}_0$ as initial state.

We are interested in finding a policy such that the trajectory generated by it (starting from a given $\tilde{x}_0$) reaches a desired *goal region* $\mathcal{G} \subset \mathcal{X}$ before some *desired settling time* $k_s \in \mathbb{N}_{>0}$ and remains in this region for at least a *desired permanence time* $k_p \in \mathbb{N}_{>0}$ (see Definition II.3 for the rigorous statements). For example, $\mathcal{G}$ could be an arbitrarily small neighborhood of a reference state, with a radius equal to the admitted steady-state error. In our main results, we assume $\mathcal{G}$, k_s, k_p are given; nonetheless, in Section III (see Remark III.8), we will observe that k_p can be arbitrarily large, and in Proposition A.1 (in the Appendix), we give a criterion to assess the feasibility of the settling time constraint when limited knowledge about the system to control is available.

B. Acceptable State-Space Sequences

We will now introduce concepts that will be used for the formalization of the proposed approach.

Definition II.1 (State-Space Sequences)**.** *A* state-space sequence *is a sequence* $\xi = (x_k)_{k\in[0,+\infty)} \in \mathcal{X}^\infty$.

Note that all trajectories are state-space sequences, but the converse is not true. As a matter of fact, given a state-space sequence ξ with $x_0 = \tilde{x}_0 \in \mathcal{X}$, there is no guarantee that there exists a policy π such that $\phi^\pi(\tilde{x}_0) = \xi$. A graphical representation of these concepts is reported in Fig. 1(a).

Definition II.2 (First Exit Instant)**.** *The* first exit instant $k_{\text{exit}}(\xi) \in \mathbb{N}_{>0}$ *of a state-space sequence* $\xi = (x_k)_{k\in[0,+\infty)}$ *is the smallest time instant such that, in* ξ*, we have* $x_{k_{\text{exit}}(\xi)-1} \in \mathcal{G}$ *and* $x_{k_{\text{exit}}(\xi)} \notin \mathcal{G}$*; if this condition never occurs in* ξ*, we set* $k_{\text{exit}}(\xi) = \infty$.

Next, we define the set of acceptable state-space sequences, trajectories, and policies, that is, those that satisfy the performance and steady-state specifications.

Definition II.3 (Acceptable State-Space Sequences, Trajectories, and Policies)**.** *Given the desired goal region* $\mathcal{G}$*, the desired settling time* k_s*, and the desired permanence time* k_p*, a state-space sequence* $\xi = (x_0, x_1, x_2, \ldots)$ *or equivalently*

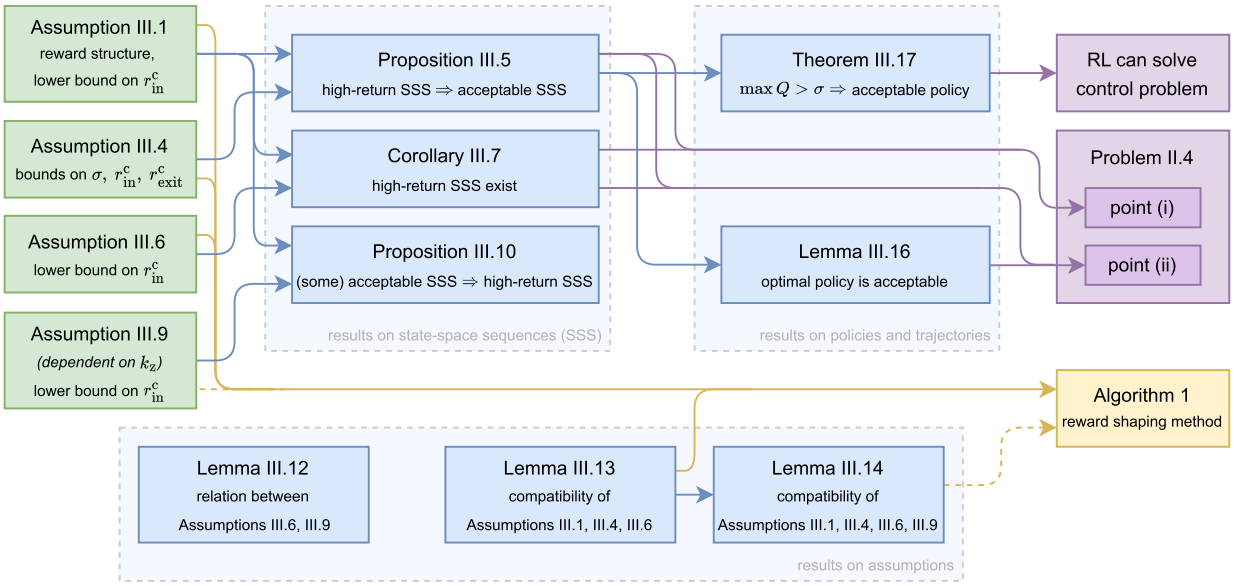


Fig. 2. Schematic representation of the main assumptions and results in Section III. Green blocks denote assumptions, blue blocks indicate analytical findings, yellow blocks denote algorithms, and purple blocks refer to the problems being studied. Dashed arrows denote optional steps in the control design. "SSS" means "state-space sequence"; the symbols in the figure are defined in Section III.

a trajectory $\phi^\pi(\tilde{x}_0) = (\tilde{x}_0, x_1, x_2, \ldots)$ *is* acceptable *if the following holds.*

1) $\exists k \leq k_s : x_k \in \mathcal{G}$ *(i.e., the state is in* $\mathcal{G}$ *not later than time* k_s*).*
2) $k_{exit}(\xi) > k_p$ *(i.e., the state does not exit* $\mathcal{G}$ *before time* k_p *included).*

A policy π *is* acceptable from $\tilde{x}_0$ *if* $\phi^\pi(\tilde{x}_0)$ *is acceptable.*

It can be immediately verified that there exists at least one acceptable state-space sequence provided that $\mathcal{G} \neq \varnothing$. Indeed, this state-space sequence is $\xi = (x_0, x_1, \ldots)$ with $x_k \in \mathcal{G}$ for all k, which can be verified to be acceptable by checking the two conditions in Definition II.3.

C. Using Reinforcement Learning to Find Acceptable Control Policies

Following [26] and [27], we employ an RL solution to automatically identify an acceptable policy for a given initial condition $\tilde{x}_0$ and to do so without the need of knowing the dynamics f. Namely, let $r : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ be a *reward function*, so that $r(x', x, u)$ is the reward obtained by the agent when taking action u in state x and arriving at the new state x' at the next time instant. Let also $J^\pi : \mathcal{X}^\infty \to \mathbb{R}$ be the *(discounted) return function* defined as

$$J^\pi(\xi) := \sum_{k=1}^{\infty} \gamma^{k-1} r(x_k, x_{k-1}, u_{k-1}) \tag{2}$$

where $\xi \in \mathcal{X}^\infty$ is a state-space sequence, $u_k = \pi(x_k)$, and $\gamma \in [0, 1]$ is a given *discount factor*.[1] To find an acceptable policy, we set the following optimization problem and solve it via RL:

$$\max_{\pi} \; J^\pi(\phi^\pi(\tilde{x}_0)) \tag{3a}$$
$$\text{s.t. } x_{k+1} = f(x_k, u_k), \quad k \in \{0, 1, 2, \ldots\} \tag{3b}$$
$$u_k = \pi(x_k), \quad k \in \{0, 1, 2, \ldots\} \tag{3c}$$
$$x_0 = \tilde{x}_0 \in \mathcal{X}. \tag{3d}$$

Thus, the problem we aim to solve can be stated as follows.

Problem II.4. *Shape the reward function* r *so that: 1) it is possible to determine that a trajectory* $\phi^\pi(\tilde{x}_0)$ *is acceptable by assessing the value of* $J^\pi(\phi^\pi(\tilde{x}_0))$ *and 2) an acceptable policy from* $\tilde{x}_0$ *(provided it exists) can be found by solving (3).*

III. Main Results

In Section III-A, we relate acceptable state-space sequences and their return (solving point 1) in Problem II.4), in Section III-C, we embed the theory in a constructive procedure to shape rewards, in Section III-D, we give analogous results for trajectories, and finally in Section III-E, we show that acceptable policies can be found using RL algorithms. The assumptions we make and how they are related are schematically summarized in Fig. 2.

A. Assessing Acceptable State-Space Sequences

We start by defining the structure of the shaped reward.

Assumption III.1 (Reward Structure)**.** *The reward function can be written as*

$$r(x', x, u) = r^b(x', x, u) + r^c(x', x) \tag{4}$$

where the following holds.

[1] According to this formulation, it is possible to evaluate J^π on a state-space sequence that is not a trajectory (which is needed for the theoretical results presented in Section III); in this case, even though the value of the states are not generated following policy π, in general, it is still necessary to specify π to obtain the values of the inputs u_k used for the computation of the reward r. When J^π is evaluated on a trajectory, for example, $J^{\pi_1}(\phi^{\pi_2})$, we will only consider the case in which $\pi_1 = \pi_2$.

1) $r^{\mathrm{b}} : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ *is a* bounded reward term, *that is, such that there exist finite* $U_{\mathrm{out}}, U_{\mathrm{in}}, L_{\mathrm{out}}, L_{\mathrm{in}} \in \mathbb{R}$ *such that*

$$\sup_{x' \in \mathcal{X} \setminus \mathcal{G},\ x \in \mathcal{X},\ u \in \mathcal{U}} r^{\mathrm{b}}(x', x, u) \leq U_{\mathrm{out}} \tag{5a}$$

$$\sup_{x' \in \mathcal{G},\ x \in \mathcal{X},\ u \in \mathcal{U}} r^{\mathrm{b}}(x', x, u) \leq U_{\mathrm{in}} \tag{5b}$$

$$\inf_{x' \in \mathcal{X} \setminus \mathcal{G},\ x \in \mathcal{X},\ u \in \mathcal{U}} r^{\mathrm{b}}(x', x, u) \geq L_{\mathrm{out}} \tag{5c}$$

$$\inf_{x' \in \mathcal{G},\ x \in \mathcal{X},\ u \in \mathcal{U}} r^{\mathrm{b}}(x', x, u) \geq L_{\mathrm{in}}. \tag{5d}$$

2) $r^{\mathrm{c}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is a* correction term *given by*

$$r^{\mathrm{c}}(x', x) = \begin{cases} r^{\mathrm{c}}_{\mathrm{in}}, & \text{if } x' \in \mathcal{G} \\ r^{\mathrm{c}}_{\mathrm{exit}}, & \text{if } x \in \mathcal{G} \text{ and } x' \notin \mathcal{G} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

with $r^{\mathrm{c}}_{\mathrm{in}}, r^{\mathrm{c}}_{\mathrm{exit}} \in \mathbb{R}$. *Moreover, it holds that*

$$r^{\mathrm{c}}_{\mathrm{in}} \geq U_{\mathrm{out}} - L_{\mathrm{in}}. \tag{7}$$

In practice, $r^{\mathrm{c}}_{\mathrm{in}}$ will typically be a positive reward for being inside the goal region, while $r^{\mathrm{c}}_{\mathrm{exit}}$ will normally be a negative reward for having left the goal region—refer to Fig. 1(b) for a diagrammatic representation.

Remark III.2 (Generality of Assumption III.1)**.** *Assumption III.1 is not too restrictive. Indeed, if one wants to use a preexisting reward, it is only required it is bounded [see (5)]. It can then be shaped by adding the correction term* r^{c} *to it.*

We also define the differences

$$\Delta_{\mathrm{in}} \coloneqq U_{\mathrm{in}} - L_{\mathrm{in}} \geq 0 \tag{8a}$$

$$\Delta_{\mathrm{out}} \coloneqq U_{\mathrm{out}} - L_{\mathrm{out}} \geq 0. \tag{8b}$$

To assess properties of state-space sequences, trajectories, and policies from their associated return, we define the *return threshold* $\sigma \in \mathbb{R}$ and introduce the following definition.

Definition III.3. *(High-Return State-Space Sequences, Trajectories, and Policies): A state-space sequence* ξ *is* high-return *if* $J^{\pi}(\xi) > \sigma$ *for any policy* π. *A trajectory* $\phi^{\pi}(\tilde{x}_0)$ *is high-return if* $J^{\pi}(\phi^{\pi}(\tilde{x}_0)) > \sigma$. *A policy* π *is* high-return from $\tilde{x}_0$ *if* $\phi^{\pi}(\tilde{x}_0)$ *is high-return.*

Of the quantities introduced so far, those that we assume to be given (i.e., fixed) are $\mathcal{G}$, k_{s}, k_{p}, γ, U_{in}, U_{out}, L_{in}, and L_{out}; conversely, the quantities to be designed are σ, $r^{\mathrm{c}}_{\mathrm{in}}$, and $r^{\mathrm{c}}_{\mathrm{exit}}$.

Next, we introduce an assumption on the correction terms in the reward.

Assumption III.4. *Assume that*

$$\sigma \geq \frac{U_{\mathrm{out}}}{1-\gamma} \tag{9}$$

and, given the desired settling time k_{s} *and the desired permanence time* k_{p}, *assume that*

$$r^{\mathrm{c}}_{\mathrm{in}} \leq -U_{\mathrm{in}} - U_{\mathrm{out}} \frac{1-\gamma^{k_{\mathrm{s}}}}{\gamma^{k_{\mathrm{s}}}} + \sigma \frac{1-\gamma}{\gamma^{k_{\mathrm{s}}}} \tag{10}$$

$$r^{\mathrm{c}}_{\mathrm{exit}} \leq -U_{\mathrm{out}} - \frac{1}{\gamma^{k_{\mathrm{p}}-1}} \left[\left(U_{\mathrm{in}} + r^{\mathrm{c}}_{\mathrm{in}}\right) \frac{1+\gamma^{k_{\mathrm{p}}-1}(\gamma-1)}{1-\gamma} - \sigma \right]. \tag{11}$$

In the following proposition, we state a key result that solves point 1) in Problem II.4.

Proposition III.5. *Let Assumptions III.1 and III.4 hold. Then, high-return state-space sequences are acceptable.*

Proof: We will show that, for any policy π, if a state-space sequence ξ is not acceptable, then it is not high-return (consequently, if ξ is high-return, then it is acceptable).

ξ can be not acceptable if and only if one of the following three scenarios occurs (see Definition II.3).

1) ξ is never in the goal region $\mathcal{G}$.
2) ξ is in $\mathcal{G}$ for the first time at a time later than k_{s}.
3) ξ exits from $\mathcal{G}$ at time $k_{\mathrm{exit}}(\xi) \leq k_{\mathrm{p}}$.

We now consider the three cases one by one and show that, for any π, if any of them occurs then it must hold that ξ is not high-return, that is, $J^{\pi}(\xi) \leq \sigma$.

Case 1: In this case, the state-space sequence is never in the goal region, that is, $\forall k \in [0, \infty), x_k \notin \mathcal{G}$. Therefore, only the third case in (6) is fulfilled, for all k, and we obtain $r^{\mathrm{c}}(x_k, x_{k-1}) = 0$ for all k. For any policy π, exploiting (2), (4), (5a), and (9), we obtain[2]

$$\begin{aligned} J^{\pi}(\xi) &= \sum_{k=1}^{+\infty} \gamma^{k-1} r^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) \\ &\leq U_{\mathrm{out}} \sum_{k=1}^{+\infty} \gamma^{k-1} = \frac{U_{\mathrm{out}}}{1-\gamma} \leq \sigma. \end{aligned} \tag{12}$$

Note that, in (12) and in the rest of the proof, the dependency of J^{π} on the specific policy π is made irrelevant by using the bounds in (5).

Case 2: Defining $k_{\mathrm{enter}} \coloneqq (\min k \text{ s.t. } x_k \in \mathcal{G})$, we have that $k_{\mathrm{enter}} > k_{\mathrm{s}}$. For the sake of simplicity and without loss of generality, assume that the state is always in the region $\mathcal{G}$ after k_{enter} (i.e., $x_k \in \mathcal{G}, \forall k \geq k_{\mathrm{enter}}$).[3] For any policy π, from (2), (4), and (6), we obtain

$$\begin{aligned} J^{\pi}(\xi) = &\sum_{k=1}^{k_{\mathrm{enter}}-1} \gamma^{k-1} r^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) \\ &+ \sum_{k=k_{\mathrm{enter}}}^{+\infty} \gamma^{k-1} \left[r^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) + r^{\mathrm{c}}_{\mathrm{in}} \right]. \end{aligned} \tag{13}$$

Exploiting (7), and recalling that $k_{\mathrm{enter}} > k_{\mathrm{s}}$, from (13), we obtain

$$\begin{aligned} J^{\pi}(\xi) \leq &\sum_{k=1}^{k_{\mathrm{s}}} \gamma^{k-1} r^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) \\ &+ \sum_{k=k_{\mathrm{s}}+1}^{+\infty} \gamma^{k-1} \left[r^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) + r^{\mathrm{c}}_{\mathrm{in}} \right]. \end{aligned} \tag{14}$$

[2]Recall that, for $|\gamma| < 1$, the geometric series is $\sum_{k=0}^{+\infty} \gamma^k = 1/(1-\gamma)$ and the truncated geometric series is $\sum_{k=0}^{n-1} \gamma^k = (1-\gamma^n)/(1-\gamma)$.

[3]The reason why we do not lose generality is that we are interested in upper bounding $J^{\pi}(\xi)$ with σ, and the simplifying assumption makes $J^{\pi}(\xi)$ the largest possible, because the smallest reward obtainable inside $\mathcal{G}$ (i.e., $r^{\mathrm{c}}_{\mathrm{in}} + L_{\mathrm{in}}$) is at least equal to the largest reward obtainable outside $\mathcal{G}$ (i.e., U_{out}), because of (7).

Then, from (14) and exploiting (5), we obtain

$$\begin{aligned} J^\pi(\xi) &\le U_{\text{out}} \sum_{k=1}^{k_\text{s}} \gamma^{k-1} + (U_{\text{in}} + r_{\text{in}}^\text{c}) \sum_{k_\text{s}+1}^{\infty} \gamma^{k-1} \\ &= U_{\text{out}} \sum_{k=0}^{k_\text{s}-1} \gamma^{k} + (U_{\text{in}} + r_{\text{in}}^\text{c}) \gamma^{k_\text{s}} \sum_{k=0}^{+\infty} \gamma^{k} \\ &= U_{\text{out}} \frac{1-\gamma^{k_\text{s}}}{1-\gamma} + (U_{\text{in}} + r_{\text{in}}^\text{c}) \frac{\gamma^{k_\text{s}}}{1-\gamma}. \end{aligned}$$

Exploiting (10), it is immediate to see that $J^\pi(\xi) \le \sigma$.

Case 3: From the definition of k_{exit} (see Section II), we have $x_{k_{\text{exit}}(\xi)-1} \in \mathcal{G}$ and $x_{k_{\text{exit}}(\xi)} \notin \mathcal{G}$. From (7), the largest $J^\pi(\xi)$ is obtained when the state-space sequence ξ is such that $x_k \in \mathcal{G}$, $\forall k \in [1, k_{\text{exit}}(\xi))$, ξ then exits the region $\mathcal{G}$ at $k_{\text{exit}}(\xi) = k_\text{p}$, and enters again at time $k_\text{p}+1$. Thus, without loss of generality, we assume this is the case. Then, we have

$$\begin{aligned} J^\pi(\xi) &\le \sum_{k=1}^{k_\text{p}-1} \gamma^{k-1} \left[r^\text{b}(x_k, x_{k-1}, u_{k-1}) + r_{\text{in}}^\text{c} \right] \\ &\quad + \gamma^{k_\text{p}-1} (r^\text{b}(x_{k_\text{p}}, x_{k_\text{p}-1}, u_{k_\text{p}-1}) + r_{\text{exit}}^\text{c}) \\ &\quad + \sum_{k=k_\text{p}+1}^{\infty} \gamma^{k-1} \left[r^\text{b}(x_k, x_{k-1}, u_{k-1}) + r_{\text{in}}^\text{c} \right] \\ &\le (U_{\text{in}} + r_{\text{in}}^\text{c}) \left[\sum_{k=0}^{k_\text{p}-2} \gamma^{k} + \gamma^{k_\text{p}} \sum_{k=0}^{\infty} \gamma^{k} \right] \\ &\quad + \gamma^{k_\text{p}-1} \left(U_{\text{out}} + r_{\text{exit}}^\text{c} \right) \\ &= (U_{\text{in}} + r_{\text{in}}^\text{c}) \frac{1-\gamma^{k_\text{p}-1}+\gamma^{k_\text{p}}}{1-\gamma} + \gamma^{k_\text{p}-1} \left(U_{\text{out}} + r_{\text{exit}}^\text{c} \right). \end{aligned}$$

Exploiting (11), we immediately verify that $J^\pi(\xi) \le \sigma$. □

Notably, Proposition III.5 does not guarantee the existence of any high-return state-space sequence. The existence of the latter is instead guaranteed by Corollary III.7.

Assumption III.6. *Let*

$$r_{\text{in}}^\text{c} > \sigma(1-\gamma) - L_{\text{in}}. \tag{15}$$

Corollary III.7. *Let Assumption III.1 hold. A sufficient condition for the existence of high-return state-space sequences is that Assumption III.6 holds. Moreover, a necessary condition for the existence of high-return state-space sequences is that*

$$r_{\text{in}}^\text{c} > \sigma(1-\gamma) - U_{\text{in}}. \tag{16}$$

Proof: Let P be the proposition "$\exists \xi \in \mathcal{X}^\infty : \forall \pi$, $J^\pi(\xi) > \sigma$."

Assumption III.6 $\Rightarrow P$*:* Consider a state-space sequence $\xi^\diamond = (x_0, x_1, \ldots)$ with all $x_k \in \mathcal{G}$. Then, for any π, from (2), (6), and (7), it holds that $J^\pi(\xi^\diamond) \ge (r_{\text{in}}^\text{c} + L_{\text{in}})/(1-\gamma)$. Exploiting Assumption III.6, we derive that $J^\pi(\xi^\diamond) > \sigma$.

(16) $\Leftarrow P$*:* To demonstrate this result, we show equivalently that $\neg$(16) $\Rightarrow \neg P$. Using again (2), (6), and (7), for any policy π, we derive that $J^\pi(\xi) \le (r_{\text{in}}^\text{c} + U_{\text{in}})/(1-\gamma)$ for all the state-space sequences $\xi \in \mathcal{X}^\infty$. If (16) does not hold, we have $J^\pi(\xi) \le \sigma, \forall \xi \in \mathcal{X}^\infty$. □

We remark that although Assumption III.6 is not a necessary condition itself for the existence of high-return state-space sequences, it implies (16) [because of (8a)], which is one.

Remark III.8 (Selection of k_p)**.** *Equation (11) captures the only assumption that depends on k_p. Given this assumption, it is possible to observe that k_p can be set to any arbitrarily large value, thus not limiting the variety of problems that can be addressed using the present theoretical framework.*

To summarize, we demonstrated that it is possible to check if a state-space sequence is acceptable by verifying that it is high-return. Conversely, there may exist acceptable state-space sequences that are not high-return, for example, those that exit (and reenter) $\mathcal{G}$ before k_s, or those that enter $\mathcal{G}$ before k_s but not early enough to collect sufficient rewards to be high-return. In some cases though, it is possible to prove that acceptable state-space sequences are high-return, such as those that enter the goal region not later than a certain time instant (k_z) and never exit it, as formalized by the next proposition.

Assumption III.9 (Dependent on the Choice of k_z)**.** *Given some $k_\text{z} \in \mathbb{N}_{\ge 0}$, with $k_\text{z} \le k_\text{s}$, let*

$$r_{\text{in}}^\text{c} > -L_{\text{in}} - L_{\text{out}} \frac{1-\gamma^{k_\text{z}-1}}{\gamma^{k_\text{z}-1}} + \sigma \frac{1-\gamma}{\gamma^{k_\text{z}-1}}. \tag{17}$$

Proposition III.10. *Let $k_\text{z} \in \mathbb{N}_{\ge 0}$ such that $k_\text{z} \le k_\text{s}$. If Assumptions III.1 and III.9 hold, then state-space sequences that are in $\mathcal{G}$ for the first time at time k_z or earlier and have $k_{\text{exit}} = \infty$ are high-return.*

Proof: According to the hypothesis, let $\xi = (x_0, x_1, \ldots)$ be a state-space sequence such that $x_k \notin \mathcal{G}$ for $k < k_\text{z}$ and $x_k \in \mathcal{G}$ for $k \ge k_\text{z}$. For all policies π, from (2), (4), and (6), we obtain

$$\begin{aligned} J^\pi(\xi) &= \sum_{k=1}^{k_\text{z}-1} \gamma^{k-1} r^\text{b}(x_k, x_{k-1}, u_{k-1}) \\ &\quad + \sum_{k=k_\text{z}}^{+\infty} \gamma^{k-1} \left[r^\text{b}(x_k, x_{k-1}, u_{k-1}) + r_{\text{in}}^\text{c} \right]. \end{aligned}$$

Exploiting (5d) and (5c) yields

$$\begin{aligned} J^\pi(\xi) &\ge L_{\text{out}} \sum_{k=1}^{k_\text{z}-1} \gamma^{k-1} + \left(L_{\text{in}} + r_{\text{in}}^\text{c} \right) \sum_{k=k_\text{z}}^{+\infty} \gamma^{k-1} \\ &= L_{\text{out}} \sum_{k=0}^{k_\text{z}-2} \gamma^{k} + \left(L_{\text{in}} + r_{\text{in}}^\text{c} \right) \gamma^{k_\text{z}-1} \sum_{k=0}^{+\infty} \gamma^{k} \\ &= L_{\text{out}} \frac{1-\gamma^{k_\text{z}-1}}{1-\gamma} + \left(L_{\text{in}} + r_{\text{in}}^\text{c} \right) \frac{\gamma^{k_\text{z}-1}}{1-\gamma}. \end{aligned}$$

Given (17), it follows that $J^\pi(\xi) > \sigma$. Moreover, say ξ' a state-space sequence that is in $\mathcal{G}$ for the first time at some time $k_\text{z}' < k_\text{z}$, that is, with $x_k \notin \mathcal{G}$ for $k < k_\text{z}'$ and $x_k \in \mathcal{G}$ for $k \ge k_\text{z}'$. For all policies π, exploiting (7) and the fact that $J^\pi(\xi) > \sigma$, we have

$$\begin{aligned} J^\pi(\xi') &\ge L_{\text{out}} \sum_{k=1}^{k_\text{z}'-1} \gamma^{k-1} + (L_{\text{in}} + r_{\text{in}}^\text{c}) \sum_{k=k_\text{z}'}^{+\infty} \gamma^{k-1} \\ &\ge L_{\text{out}} \sum_{k=1}^{k_\text{z}-1} \gamma^{k-1} + (L_{\text{in}} + r_{\text{in}}^\text{c}) \sum_{k=k_\text{z}}^{+\infty} \gamma^{k-1} > \sigma. \end{aligned}$$

□

From (17), we notice that the larger $r^{\mathrm{c}}_{\mathrm{in}}$ is, the later state-space sequences are required to be in $\mathcal{G}$ to be high-return. Moreover, it is again important to remark that while state-space sequences that enter $\mathcal{G}$ within k_{z} time steps always exist, the same is not necessarily true for trajectories: this depends on the dynamics of the system being controlled.

Remark III.11 (Tracking)**.** *In the results in Section III-A, it was never assumed that $\mathcal{G}$ is a fixed region in the state space. Indeed, it is possible to carry out the same analysis by considering a time-dependent goal region $\mathcal{G}_k$ and, for simplicity of computation, the quantities*

$$U \coloneqq \sup_{x'\in\mathcal{X},\ x\in\mathcal{X},\ u\in\mathcal{U}} r^{\mathrm{b}}(x',x,u)$$
$$L \coloneqq \inf_{x'\in\mathcal{X},\ x\in\mathcal{X},\ u\in\mathcal{U}} r^{\mathrm{b}}(x',x,u)$$

rather than U_{out}, U_{in}, and L_{out}, L_{in} should be used in (5), respectively. This reformulation can be used to address tracking control problems.

Reviewing the findings derived so far, a shaped reward r needs to satisfy Assumptions III.1, III.4, and III.6 (to exploits Proposition III.5 and Corollary III.7) and optionally Assumption III.9 (with some chosen k_{z}, to exploit Proposition III.10); see also Fig. 2. Next, we characterize the relation between these assumptions and show that they can hold simultaneously.

B. Compatibility of the Assumptions

First, we give the following lemma to aid the selection of $r^{\mathrm{c}}_{\mathrm{in}}$.

Lemma III.12. *Given some $k_{\mathrm{z}} \leq k_{\mathrm{s}}$, if Assumption III.9 holds, Assumption III.6 also holds.*

Proof: See the Appendix. □

We say that two or more assumptions are *compatible* if they can hold simultaneously. To guarantee that high-return state-space sequences are acceptable (see Proposition III.5) and that such state-space sequences exist (see Corollary III.7), we need Assumptions III.1, III.4, and III.6, whose compatibility is ensured by the following lemma.

Lemma III.13. *Assumptions III.1, III.4, and III.6 are compatible if*

$$\sigma > \frac{U_{\mathrm{out}}}{1-\gamma} + \frac{\Delta_{\mathrm{in}}\gamma^{k_{\mathrm{s}}}}{(1-\gamma)(1-\gamma^{k_{\mathrm{s}}})}. \tag{18}$$

Proof: See the Appendix. □

To guarantee that a class of acceptable state-space sequences are high-return, we need Assumption III.9 (see Proposition III.10), whose compatibility with previous ones is ensured by the following lemma.

Lemma III.14. *Given some $k_{\mathrm{z}} \leq k_{\mathrm{s}}$, Assumptions III.1, III.4, III.6, and III.9 are compatible if*

$$\sigma > \frac{\gamma^{k_{\mathrm{s}}}}{(1-\gamma)(1-\gamma^{k_{\mathrm{s}}})}\Delta_{\mathrm{in}} + \frac{U_{\mathrm{out}}}{1-\gamma} + \frac{\gamma^{k_{\mathrm{s}}}(1-\gamma^{k_{\mathrm{z}}-1})}{(1-\gamma)(\gamma^{k_{\mathrm{z}}-1}-\gamma^{k_{\mathrm{s}}})}\left(\frac{\gamma^{k_{\mathrm{s}}}\Delta_{\mathrm{in}}}{(1-\gamma^{k_{\mathrm{s}}})} + \Delta_{\mathrm{out}}\right). \tag{19}$$

Proof: See the Appendix. □

C. Constructive Procedure for Reward Shaping

In Algorithm 1, we propose a constructive procedure that can be applied to shape the reward functions used in Section III. To provide more flexibility, the procedure takes a preexisting reward r^{b} as input, bounded according to (5). If no r^{b} is available, it is possible to set $r^{\mathrm{b}} = 0$. As Lemma III.13 ensures set $\mathcal{I}$ in the algorithm is not empty, the latter always terminates successfully. Once Algorithm 1 has been used to obtain a shaped reward r (thus fixing $r^{\mathrm{c}}_{\mathrm{in}}$, $r^{\mathrm{c}}_{\mathrm{in}}$, σ, which remain constant), it is possible to run an RL algorithm to learn a suitable control policy, as explained below in Section III-E.

Algorithm 1 Reward Shaping

Input: A goal region $\mathcal{G}$, a desired settling time k_{s}, and a desired permanence time k_{p}; a bounded reward function r^{b}, and discount factor γ.

Output: A reward function r satisfying to Assumptions III.1, III.4, III.6.

1 $U_{\mathrm{out}} \leftarrow \sup_{x'\in\mathcal{X}\setminus\mathcal{G},x\in\mathcal{X},u\in\mathcal{U}} r^{\mathrm{b}}(x',x,u)$; ▷ c.f. Eq. (5a)

2 $U_{\mathrm{in}} \leftarrow \sup_{x'\in\mathcal{G},x\in\mathcal{X},u\in\mathcal{U}} r^{\mathrm{b}}(x',x,u)$; ▷ c.f. Eq. (5b)

3 $L_{\mathrm{in}} \leftarrow \inf_{x'\in\mathcal{G},x\in\mathcal{X},u\in\mathcal{U}} r^{\mathrm{b}}(x',x,u)$; ▷ c.f. Eq. (5d)

4 $\sigma \leftarrow \mathrm{rand}\left(\frac{U_{\mathrm{out}}}{1-\gamma} + \frac{(U_{\mathrm{in}}-L_{\mathrm{in}})\gamma^{k_{\mathrm{s}}}}{(1-\gamma)(1-\gamma^{k_{\mathrm{s}}})},\ \infty\right)$; ▷ from Lemma III.13

5 $\mathcal{I} \leftarrow \left(\sigma(1-\gamma) - L_{\mathrm{in}},\ -U_{\mathrm{in}} - U_{\mathrm{out}}\frac{1-\gamma^{k_{\mathrm{s}}}}{\gamma^{k_{\mathrm{s}}}} + \sigma\frac{1-\gamma}{\gamma^{k_{\mathrm{s}}}}\right]$; ▷ c.f. Eqs. (7), (10), (15)

6 $r^{\mathrm{c}}_{\mathrm{in}} \leftarrow \mathrm{rand}(\mathcal{I})$;

7 $r^{\mathrm{c}}_{\mathrm{exit}} \leftarrow \mathrm{rand}\Big(-\infty,\ -U_{\mathrm{out}} - \frac{1}{\gamma^{k_{\mathrm{p}}-1}}\Big[(U_{\mathrm{in}} + r^{\mathrm{c}}_{\mathrm{in}})\frac{1+\gamma^{k_{\mathrm{p}}-1}(\gamma-1)}{1-\gamma} - \sigma\Big]\Big)$; ▷ c.f. Eq. (11)

8 build r^{c} as in (6);

9 $r \leftarrow r^{\mathrm{b}} + c$;

It is to be noted that in some cases the values of $r^{\mathrm{c}}_{\mathrm{in}}$ and $r^{\mathrm{c}}_{\mathrm{exit}}$ resulting from Algorithm 1 might be significantly larger in absolute value when compared to those in r^{b}. This can lead to a relatively *sparse* reward function r, that is, one where relatively large values (in absolute value) are present but infrequent in the state-action space. Notoriously, this lack of frequent feedback information can make learning more difficult, especially when deep RL algorithms are used (see [28], [29] and references therein). To mitigate this issue, it is possible to select $r^{\mathrm{c}}_{\mathrm{in}}$ and $r^{\mathrm{c}}_{\mathrm{exit}}$ as small in absolute value as possible, while still complying with Assumptions III.1, III.4, and III.6. Reward-shaping methods that do not make the reward sparse will be the subject of future work.

Remark III.15 (Advanced Reward-Shaping Algorithm)**.** *For simplicity, in Algorithm 1, we did not include the requirement captured by (17) on $r^{\mathrm{c}}_{\mathrm{in}}$ (used to ensure that a family of acceptable state-space sequences are high-return, according to Proposition III.10), as it depends on the time instant k_{z}, which would be a further parameter to select. This constraint can be incorporated in Algorithm 1 by first selecting $k_{\mathrm{z}} \leq k_{\mathrm{s}}$ (possibly exploiting knowledge of the system to control), enforcing (19) at line 4 (Lemma III.14 ensures $\mathcal{I}$ is not empty), and using the right-hand side of (17) as lower bound of $\mathcal{I}$ at line 5.*

D. Assessing Acceptable Trajectories

In Section III-A, we showed how the value of the return $J^{\pi}(\xi)$ can be used to assess whether ξ is an acceptable

state-space sequence. The same theory applies to trajectories (which, by definition, are state-space sequences).

It is important to remark that, while the existence of high-return state-space sequences is ensured by Proposition III.7, it can be much more difficult to establish if there actually exist policies that generate high-return trajectories. This depends on the dynamics of the system at hand and the performance required and is tightly related to the problem of reachability [30], with the addition of requirements on the settling time and the permanence time.

E. Assessing Acceptable Policies in Value-Based Reinforcement Learning

First, we provide a simple result stating that an acceptable policy (see Definition II.3) can be found by achieving the optimum in (3), thus solving point 2) in Problem II.4.

Lemma III.16. *Let Assumptions III.1 and III.4 hold. If there exists a high-return policy $\pi^\diamond$ from $\tilde{x}_0 \in \mathcal{X}$, then the optimal policy $\pi^\star$ solving the problem objective defined in (3) is acceptable from $\tilde{x}_0$.*

Proof: As $\pi^\star$ maximizes the return in (3), then $J^{\pi^\star}(\phi^{\pi^\star}(x_0)) \geq J^{\pi^\diamond}(\phi^{\pi^\diamond}(x_0)) > \sigma$, exploiting Proposition III.5 (applicable, as Assumptions III.1 and III.4 hold). □

Proposition III.5 allows to detect acceptable state-space sequences by evaluating their return J^π. However, this is not normally known in an RL setting, but it is instead approximated through a value function. In particular, let $Q : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ be the state-action value function associated with the *greedy policy*

$$\pi_g(x) = \arg\max_{u \in \mathcal{U}} Q(x, u). \tag{20}$$

Q is normally updated iteratively with the Bellman operator so that it converges to the value of J^{π_g}, in the sense that $Q(x, u) \approx r(f(x, u), x, u) + \gamma J^{\pi_g}(\phi^{\pi_g}(f(x, u))$ [31, Sec. 3].

In the next theorem, we conclude the analysis by showing how the acceptability of a policy can be evaluated by assessing the value of Q.

Theorem III.17. *Consider a state $x_k \in \mathcal{X}$ at time k. Let Assumptions III.1 and III.4 hold and assume that*

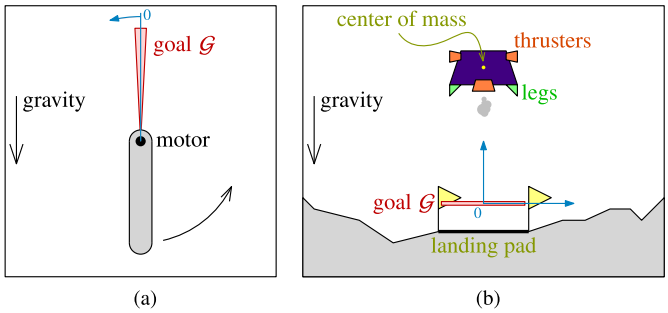
$$\operatorname{sign}\left(\max_{u \in \mathcal{U}} Q(x_k, u) - \sigma\right) = \operatorname{sign}(J^{\pi_g}(\phi^{\pi_g}(x_k)) - \sigma). \tag{21}$$

If $\max_{u \in \mathcal{U}} Q(x_k, u) > \sigma$, then π_g is an acceptable policy from x_k.

Proof: Exploiting (21), $\max_{u \in \mathcal{U}} Q(x_k, u) > \sigma$ implies that $J^{\pi_g}(\phi^{\pi_g}(x_k)) > \sigma$. Thus, it is immediate to apply Proposition III.5 (using Assumptions III.1 and III.4) to obtain that $\phi^{\pi_g}(x_k)$ is acceptable. □

It is important to clarify that $\phi^{\pi_g}(x_k)$ being an acceptable trajectory means that, by following policy π_g: 1) the state-space sequence will be in $\mathcal{G}$ before k_s time instants have passed (i.e., $\exists\, k' \in [k, k + k_s] : x_{k'} \in \mathcal{G}$) and 2) the state will not exit from $\mathcal{G}$ before $k_p + 1$ time instants have passed, (i.e., $\nexists\, k'' \in [k + 1, k + k_p] : x_{k''-1} \in \mathcal{G}, x_{k''} \notin \mathcal{G}$). Moreover, we note that (21) is satisfied if Q is well approximating J^{π_g}, in the sense that

$$\left|\max_{u \in \mathcal{U}} Q(x_k, u) - J^{\pi_g}(\phi^{\pi_g}(x_k))\right| < |J^{\pi_g}(\phi^{\pi_g}(x_k)) - \sigma|. \tag{22}$$



(a) (b)

Fig. 3. Sketch representation of the environments used in the numerical validation in Section IV. (a) Pendulum. (b) Lunar Lander. Both the pendulum and the lander are depicted in their initial states.

Indeed, (22) implies (21) through Lemma A.2 in the Appendix.[4] Equation (22) is fulfilled after a finite number of iterations if the algorithm used to update the value of Q is converging asymptotically to J^{π_g}, which has been proved formally for RL algorithms like state action reward state action (SARSA) and Q-learning [31]. In the latter, the greedy policy and the function Q are guaranteed to converge to the optimal policy and its discounted return J, respectively; hence, if high-return policies exist, Lemma III.16 guarantees that the learned policy is acceptable.

IV. Numerical Results

We validate the theory presented in Section III by means of two representative case studies (and corresponding RL environments, from OpenAI Gym [22]): Pendulum [23] and Lunar Lander [24]. The former is a classic nonlinear benchmark problem in control theory, whereas the latter is a more sophisticated control problem with multiple inputs and outputs. In particular, we first validate Theorem III.17 using Q-learning to learn a policy that stabilizes a pendulum within a predefined time; then, we show that the theory also holds when using a deep RL algorithm, such as Double deep Q network (DQN), to learn a policy able to land a spacecraft fulfilling desired time constraints.

In each scenario, the learning phase and deployment phase are repeated in $S \in \mathbb{N}_{>0}$ independent sessions, which are composed of $E \in \mathbb{N}_{>0}$ episodes. Each episode is a simulation lasting $N \in \mathbb{N}_{>0}$ time steps. Moreover, we always use the ϵ-greedy policy [31] during learning.

For reproducibility, the code is available on GitHub [25].

A. Pendulum

1) Description of the Pendulum Environment: In this environment, the objective is to stabilize a rigid pendulum affected by gravity to the upward position in a certain time, by exploiting a torque applied at the joint. In particular, the pendulum is a rigid rod, having length $l = 1$ m, mass $m = 1$ kg and moment of inertia $I = 1/3$ ml^2; the gravitational acceleration is taken equal to $g = 10$ m/s^2. A graphical depiction of the scenario is given in Fig. 3(a).

[4] Assuming $J^{\pi_g}(\phi^{\pi_g}(x_k)) \neq \sigma$. The case that $J^{\pi_g}(\phi^{\pi_g}(x_k)) = \sigma$ is, however, not of interest, as the trajectory $\phi^{\pi_g}(x_k)$ would not be high return.

a) State: The state at time k is $x_k := [x_{k,1} \quad x_{k,2}]^\mathsf{T}$, where $x_{k,1}$ and $x_{k,2}$ are the angular position and angular velocity of the pendulum, respectively. To apply Q-learning, which is a tabular RL method, we discretize the state space (and the set of acceptable inputs). Namely, the position $x_{k,1}$ takes values in $[-\pi, \pi]$ rad, with $[-\pi, -\pi/9]$ discretized into eight equally spaced values, $(-\pi/9, -\pi/36]$ into seven values, and $(-\pi/36, 0]$ into five values (analogously for $[0, \pi]$). The velocity $x_{k,2}$ takes values in $[-8, 8]$ rad/s, with $[-8, -1]$ discretized into ten values, and $(-1, 0]$ into nine values (analogously for $[0, 8]$). $x_{k,1} = 0$ and $x_{k,1} = \pi$ correspond to the upward and downward positions, respectively. In each simulation, the initial condition is chosen as $x_0 = [\pi \quad 0]^\mathsf{T}$.

b) Control inputs: The control input u_k is a torque applied at the pendulum's rotating joint, with values chosen in $[-2, 2]$ Nm, with the interval $[-2, -0.2]$ discretized into nine values, and $(-0.2, 0]$ into four values (analogously for $[0, 2]$).

c) Control problem: Let $x^{\text{ref}} := [0 \quad 0]^\mathsf{T}$ denote the unstable vertical position. The goal region is $\mathcal{G} := \{x \in \mathcal{X} \mid \|x - x^{\text{ref}}\| < \theta\}$ ($\|\cdot\|$ being the Euclidean norm), with $\theta = 0.42$ amounting to 5% of the maximum distance from the origin, in the state space. We select the desired settling time as $k_\text{s} = 500$ time steps and the desired permanence time as $k_\text{p} = 1000$ time steps (see Section II).

d) Reward: To guarantee the required performance and steady-state specifications, the reward function is chosen as in (4), with r^b being the standard Gym reward, given by

$$r^\text{b}(x_k, x_{k-1}, u_{k-1}) = -x_{1,k}^2 - 0.1\ x_{2,k}^2 - 0.001\ u_{k-1}^2. \quad (23)$$

Following Algorithm 1, from (23), we compute that

$$\begin{aligned}
U_\text{out} &= \max_{x_k \notin \mathcal{G}, x_{k-1} \in \mathcal{X}, u_{k-1} \in \mathcal{U}} r^\text{b} = -0.1\theta^2 \approx -0.018 \\
L_\text{out} &= \min_{x_k \notin \mathcal{G}, x_{k-1} \in \mathcal{X}, u_{k-1} \in \mathcal{U}} r^\text{b} \\
&= -\pi^2 - 0.1 \cdot 8^2 - 0.001 \cdot 2^2 \approx -16.27 \\
U_\text{in} &= \max_{x_k \in \mathcal{G}, x_{k-1} \in \mathcal{X}, u_{k-1} \in \mathcal{U}} r^\text{b} = 0 \\
L_\text{in} &= \min_{x_k \in \mathcal{G}, x_{k-1} \in \mathcal{X}, u_{k-1} \in \mathcal{U}} r^\text{b} = -\theta^2 - 0.001 \cdot 2^2 \approx -0,18.
\end{aligned}$$

Then, given $\gamma = 0.99$ [see (2)], we select $\sigma = 10\,000$ [see (9)], and the correction terms in (6) as

$$\begin{aligned}
r_\text{in}^\text{c} &= -U_\text{in} - U_\text{out} \frac{1-\gamma^{k_\text{s}}}{\gamma^{k_\text{s}}} + \sigma \frac{1-\gamma}{\gamma^{k_\text{s}}} \approx 1.52 \cdot 10^4 \\
r_\text{exit}^\text{c} &= -U_\text{out} - \frac{1}{\gamma^{k_\text{p}-1}} \left[(U_\text{in} + r_\text{in}^\text{c}) \frac{1 + \gamma^{k_\text{p}-1}(\gamma - 1)}{1-\gamma} - \sigma \right] \\
&\approx -3.50 \cdot 10^{10}.
\end{aligned}$$

e) Parameters: We take $S = 5$ sessions, $E = 10\,000$ episodes, and $N = 1000$ time steps. We set the learning rate to 0.8. For the ϵ-greedy policy, we select $\epsilon = 0.05$.

2) Results of Q-Learning in the Pendulum Environment: After training is completed for all sessions, we test the capability of the learned policies to swing up and stabilize the pendulum within the desired settling time. The results are portrayed in Fig. 4, showing the distance of the trajectories from x^{ref}, position, and velocity in time. We observe that the control problem is solved in all sessions, suggesting that the optimal policy (which would be an acceptable one, according to Lemma III.16) has been found. Interestingly, this might be difficult to detect by looking only at the returns obtained during training, plotted in Fig. 5. Indeed, the discounted returns per episode appear to decrease as training progresses. This happens because, as the agent progressively learns to enter the goal region and to do so earlier in later episodes, the chance of it incurring the penalty r_exit^c for existing the goal region increases as the result of random explorative actions, taken by the ϵ-greedy policy used during learning. Although this does not prevent learning from converging to the optimal policy, in practical implementations, this can be avoided by letting the exploration rate ϵ decay in later episodes. However, tuning the decay rate is highly problem-dependent and no general rule can therefore be given here.

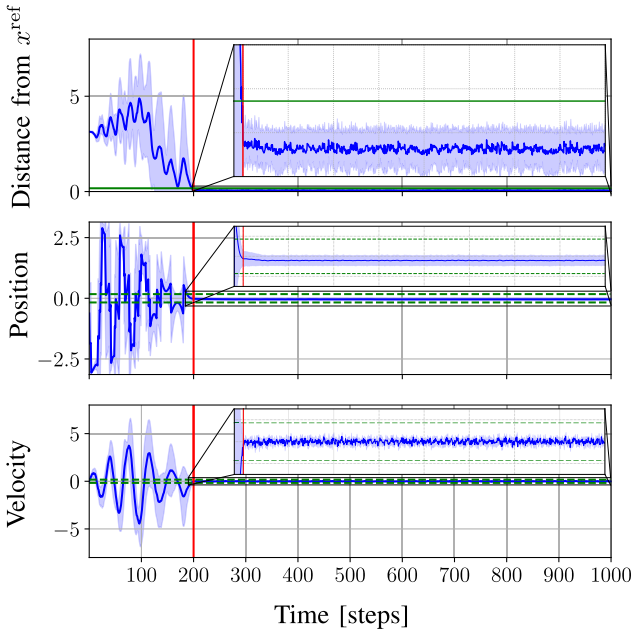


Fig. 4. Average (blue line) plus/minus standard deviation (shaded area) of $\|x_k - x^{\text{ref}}\|$ (top), angular position $x_{k,1}$ (middle), and angular velocity $x_{k,2}$ (bottom), obtained by S policies trained with Q-learning in the pendulum environment. The green solid line (top) indicates the goal region $\mathcal{G}$; the green dashed line (middle and bottom) indicates neighborhoods of width 2θ centered in $x_{1,k} = x_1^{\text{ref}} = 0$ (middle) and in $x_{2,k} = x_2^{\text{ref}} = 0$ (bottom). The red line indicates the time instant when the (averaged) trajectory enters the goal region.

In our experiments, learning ended after the planned episodes. An alternative heuristic method to determine when to terminate the learning stage is to pause training at regular intervals, and simulate using the greedy policy in (20). If the return obtained exceeds σ, the greedy policy is deemed acceptable (see Proposition III.5), ending learning; otherwise, training continues.

B. Lunar Lander

1) Description of the Lunar Lander Environment: In a 2-D space, a stylized spaceship must land at a small speed in a specific area in a predetermined time, in the presence of gravity, and the absence of friction. The spacecraft has three thrusters to guide its descent and two supporting legs at the bottom, as depicted in Fig. 3(b).

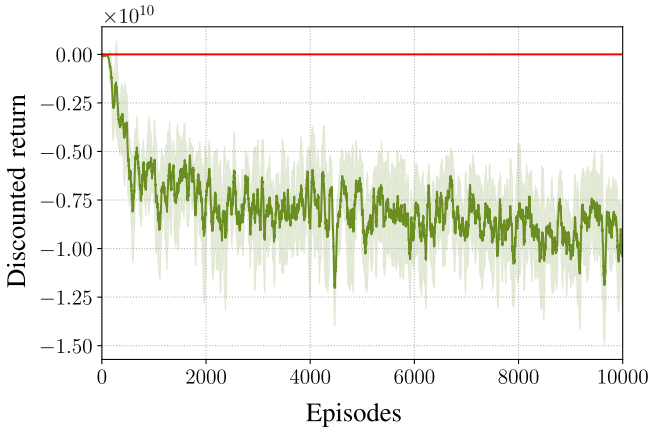


Fig. 5. Average (green line) plus/minus standard deviation (shaded area) of the discounted returns per episode obtained in S training sessions with Q-learning in the pendulum environment. The red line indicates the threshold value σ (see Section III-A). The returns are averaged backward across episodes using a moving window of 50 samples.

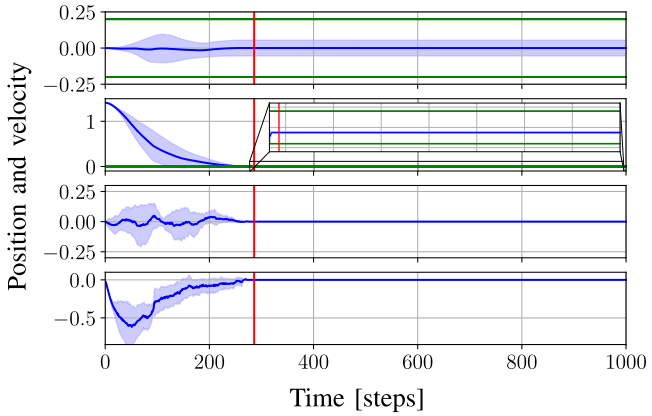


Fig. 6. Average (blue line) plus/minus standard deviation (shaded area) of the trajectory obtained by S policies trained with Double DQN in the Lunar Lander environment. From top to bottom: position on the horizontal axis, position on the vertical axis, velocity on the horizontal axis, velocity on the vertical axis. The green lines define the goal region. The red line indicates when the (averaged) trajectory enters the goal region.

a) State: The state at time k is $x_k = [p_k^\mathsf{T}\ v_k^\mathsf{T}\ \theta_k\ \omega_k\ l_k^l\ l_k^r]^\mathsf{T}$, where $p_k \in \mathbb{R}^2$ is the horizontal and vertical position of the center of mass (arbitrary units; a.u.), $v_k \in \mathbb{R}^2$ is its horizontal and vertical velocity (a.u./s), $\theta_k \in [0, 2\pi)$ rad is the orientation of the lander (with 0 corresponding to the orientation of a correctly landed spacecraft), ω_k is the rate of change of the orientation (rad/s), $l_k^l \in \{0, 1\}$ (resp. l_k^r) is 1 if the left (resp. right) leg is touching the ground. The initial conditions are given by $p_0 = [0\ 1.4]^\mathsf{T}$ (consequently, $l_0^1 = l_0^2 = 0$), $v_0 = [0\ 0]^\mathsf{T}$, $\theta_0 = 0$, and $\omega_0 = 0$. The landing area is the region $[-0.2\ 0.2] \times [-0.001\ 0.001]$ (horizontal and vertical intervals, respectively). The terrain topography (beyond the landing pad) is random in each simulation.

b) Control inputs: At each time step k, the lander can use at most one of its three thrusters. In particular, we let $u_k^m \in \{0, 1\}$ be 1 if at time k the main engine on the bottom of the spacecraft is used at full power or 0 if it is OFF, and define $u_k^l, u_k^r \in \{0, 1\}$ analogously for the left and right thrusters, respectively. Then, the control input at time k is the vector $u_k = [u_k^m\ u_k^l\ u_k^r]^\mathsf{T}$, which has four possible values, depending on which thruster, if any, is used.

c) Control problem: The goal region $\mathcal{G}$ is the set of states where p_k is in the landing pad, $v_k = 0$, $\theta_k = \omega_k = 0$, and $l_k^l = l_k^r = 1$. Additionally, we select the desired settling time as $k_\mathrm{s} = 500$ time steps and the desired permanence time as $k_\mathrm{p} = 1000$ time steps (see Section II). We also remark that the simulation stops if the lander touches the ground beyond the landing pad, or if it lands on the pad at a speed that is too high. During training only, the simulation is also halted if the spacecraft lands correctly. Further detail can be found in [24].

d) Reward: The reward function is in the form introduced in (4), with r^b generated according to the standard environment definition [24]. Namely, let $\hat{r} : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ be a function given by

$$\begin{aligned}\hat{r}(x_k, x_{k-1}, u_{k-1}) \coloneqq\ & 100(\|p_k\| - \|p_{k-1}\|) \\ & + 100(\|v_k\| - \|v_{k-1}\|) \\ & + 100(|\theta_k| - |\theta_{k-1}|) \\ & + 10(l_k^l - l_{k-1}^l) + 10(l_k^r - l_{k-1}^r) \\ & + 0.3\ u_{k-1}^m + 0.03\ u_{k-1}^l + 0.03 u_{k-1}^r. \quad (24)\end{aligned}$$

Then, r^b is given by

$$r^\mathrm{b}(x_k, x_{k-1}, u_{k-1}) = \begin{cases} 100, & \text{if } \beta_1 \text{ is true} \\ -100, & \beta_2 \text{ is true} \\ \hat{r}, & \text{otherwise} \end{cases} \quad (25)$$

where β_1 and β_1 are two mutually exclusive Boolean conditions, namely β_1 is true if the spacecraft lands on the ground and stops, and β_2 becomes true if the lander touches any point of the map with a speed that is too high (i.e., it crashes), or goes beyond the operating area of the environment, that is, $[-1.5, 1.5] \times [-1.5, 1.5]$. Following Algorithm 1, from (25), we derive that $U_\mathrm{out} = 100$, $L_\mathrm{out} = -100$, $U_\mathrm{in} = 100$, $L_\mathrm{in} = 100$. Given $\gamma = 0.99$ [see (2)], we select $\sigma = 12\,000$ [see (9)], and the correction terms in (6) as

$$\begin{aligned} r_\mathrm{in}^\mathrm{c} &= -U_\mathrm{in} - U_\mathrm{out}\frac{1-\gamma^{k_\mathrm{s}}}{\gamma^{k_\mathrm{s}}} + \sigma\frac{1-\gamma}{\gamma^{k_\mathrm{s}}} \approx 3.04 \cdot 10^3 \\ r_\mathrm{exit}^\mathrm{c} &= -U_\mathrm{out} - \frac{1}{\gamma^{k_\mathrm{p}-1}}\left[(U_\mathrm{in} + r_\mathrm{in}^\mathrm{c})\frac{1+\gamma^{k_\mathrm{p}-1}(\gamma-1)}{1-\gamma} - \sigma\right] \\ &\approx -6.93 \cdot 10^9. \end{aligned}$$

e) Parameters: We take $S = 5$ sessions, $E = 1000$ episodes, and $N = 1000$ time steps. For the ϵ-greedy policy, we select $\epsilon = 0.1$. To better stabilize the values of Q and help prevent overestimation, we use a standard Double DQN algorithm (a variation of DQN [32]; see [33] for a detailed description), implemented in TensorFlow 2. For the neural networks, we used an input layer with eight nodes, two hidden layers each composed of 128 nodes with rectified linear unit (ReLU) activation functions, and an output layer with three nodes and linear activation functions. The networks were trained using the Adam optimizer [34], with a learning rate of 0.001. Samples collected during training are stored in a replay buffer and at each training update a batch of 128 samples is used.

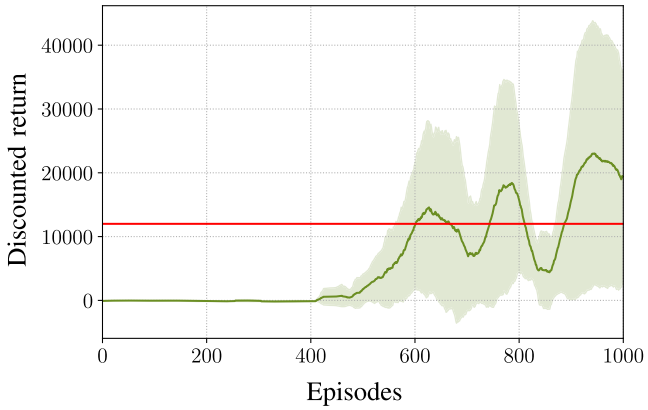


Fig. 7. Average (green line) plus/minus standard deviation (shaded area) of the discounted returns per episode obtained in S training sessions with Double DQN in the Lunar Lander environment. The red line indicates the threshold value σ (see Section III-A). The returns are averaged backward across episodes using a moving window of 50 samples.

2) Results of Double DQN in the Lunar Lander Environment: In this environment, in all sessions, the policies learned with Double DQN can solve the given control problem, fulfilling the given control requirements, as shown in Fig. 6. In Fig. 7, we also report the returns obtained by the learning algorithm. It is possible to observe that we obtain (averaged) returns that are over the threshold value σ. In this case, the large negative returns, which were visible in Fig. 5 for the Pendulum environment, are not present. The reason is that, in the Lunar Lander simulation environment, during training (but not on validation), once the lander stops, the simulation is halted; therefore, in this case, during training the lander never exits the goal region once it has entered it.

V. Conclusion

One of the most significant challenges holding back the use of RL for control applications is the lack of guarantees concerning the performance of the learned policies. In this work, we have presented analytical results that show how a specific shaping of the reward function can ensure that a control problem, such as a regulation problem, is solved with arbitrary precision, within a given settling time. We have validated the proposed theoretical approach on two representative experimental scenarios: the stabilization of a pendulum and the landing of a simplified spacecraft.

One drawback of the present methodology is that the shaped reward might be relatively sparse (as discussed in Section III-C), which could possibly hamper learning when using deep RL algorithms. Future work will focus on integrating existing techniques [35] (and developing new ones) for reward shaping, which can deal with the potential sparse reward problem, on extending the current results to the case of stochastic system dynamics, and on deriving conditions to ensure feasibility of a set of control requirements ($\mathcal{G}$, k_s, k_p) for a given system, which is highly problem-dependent.

Appendix

Let $\delta_\mathcal{G}(x) \coloneqq \min_{y\in\mathcal{G}}\|x-y\|$ be the distance between x and $\mathcal{G}$; let $h(x,u) \coloneqq f(x,u)-x$ and $H \coloneqq \sup_{x\in\mathcal{X},u\in\mathcal{U}}\|g(x,u)\|$.

Proposition A.1. *Consider the system defined in (1), and let H be finite. If $k_\mathrm{s} < \delta_\mathcal{G}(\tilde{x}_0)/H$, there does not exist an acceptable policy π from $\tilde{x}_0$.*

Proof: We will show that there does not exist a policy π such that in $\phi^\pi(\tilde{x}_0)$ there exist some $k' \le k_\mathrm{s}$ such that $x_{k'} \in \mathcal{G}$. Namely, consider some policy π and the trajectory $\phi^\pi(\tilde{x}_0)$; note that

$$\begin{aligned}\delta_\mathcal{G}(\tilde{x}_0) &= \min_{y\in\mathcal{G}}\|\tilde{x}_0-y\| = \min_{y\in\mathcal{G}}\|\tilde{x}_0-x_k+x_k-y\| \\ &\le \min_{y\in\mathcal{G}}\big(\|\tilde{x}_0-x_k\|+\|x_k-y\|\big) \\ &= \|\tilde{x}_0-x_k\| + \min_{y\in\mathcal{G}}\|x_k-y\|. \end{aligned} \tag{26}$$

Rewrite (26) as

$$\min_{y\in\mathcal{G}}\|x_k-y\| = \delta_\mathcal{G}(x_k) \ge \delta_\mathcal{G}(\tilde{x}_0) - \|\tilde{x}_0-x_k\|. \tag{27}$$

A necessary condition for obtaining $x_k \in \mathcal{G}$ is that $\delta_\mathcal{G}(x_k)=0$, which is possible only if

$$\|\tilde{x}_0-x_k\| \ge \delta_\mathcal{G}(\tilde{x}_0). \tag{28}$$

At the same time, it holds that

$$\|\tilde{x}_0-x_k\| = \left\|\tilde{x}_0-\left(\tilde{x}_0+\sum_{j=0}^{k-1} g(x_j,u_j)\right)\right\| \le kH. \tag{29}$$

Thus, to satisfy (28), it is required that $kH \ge \delta_\mathcal{G}(\tilde{x}_0)$. Hence, if $k_\mathrm{s} < \delta_\mathcal{G}(\tilde{x}_0)/H$, then surely $x_k \notin \mathcal{G}$ for all $k \le k_\mathrm{s}$, and thus $\phi^\pi(\tilde{x}_0)$ is not acceptable. □

Proof of Lemma III.12: We rewrite (17) as

$$r_\mathrm{in}^\mathrm{c} > -L_\mathrm{in} + \frac{1-\gamma^{k_\mathrm{z}-1}}{\gamma^{k_\mathrm{z}-1}}[(1-\gamma)\sigma - L_\mathrm{out}] + (1-\gamma)\sigma. \tag{30}$$

Exploiting (8b) and (9), we have $(1-\gamma)\sigma \ge U_\mathrm{out} \ge L_\mathrm{out}$, that is, $(1-\gamma)\sigma - L_\mathrm{out} \ge 0$. Thus, (30) implies (15). □

Proof of Lemma III.13: Assumptions III.1, III.4, and III.6 are compatible if it is possible to select the constants σ, r_in^c, $r_\mathrm{exit}^\mathrm{c}$ in accordance with (9), (7), (10), (11), and (15). It is always possible to select some $r_\mathrm{exit}^\mathrm{c}$ that satisfies to (11); differently, to have (9), (7), (10), and (15) be compatible, the following must hold: 1) (9); 2) $U_\mathrm{out}-L_\mathrm{in} \le -U_\mathrm{in}-U_\mathrm{out}(1-\gamma^{k_\mathrm{s}})/(\gamma^{k_\mathrm{s}})+\sigma(1-\gamma)/(\gamma^{k_\mathrm{s}})$ [from (7) and (10)]; and 3) $\sigma(1-\gamma) - L_\mathrm{in} < -U_\mathrm{in}-U_\mathrm{out}(1-\gamma^{k_\mathrm{s}})/(\gamma^{k_\mathrm{s}})+\sigma(1-\gamma)/(\gamma^{k_\mathrm{s}})$ [from (10) and (15)].

Note that 1) holds if $\sigma \ge U_\mathrm{out}/(1-\gamma)$, 2) holds if $\sigma \ge U_\mathrm{out}/(1-\gamma) + \Delta_\mathrm{in}\gamma^{k_\mathrm{s}}/(1-\gamma)$, and 3) holds if (18) holds, which is the most restrictive of the three. □

Proof of Lemma III.14: From (10) and (17), Assumptions III.4 and III.9 are compatible if

$$\begin{aligned}-L_\mathrm{in} - L_\mathrm{out}\frac{1-\gamma^{k_\mathrm{z}-1}}{\gamma^{k_\mathrm{z}-1}} + \sigma\frac{1-\gamma}{\gamma^{k_\mathrm{z}-1}} \\ < -U_\mathrm{in} - U_\mathrm{out}\frac{1-\gamma^{k_\mathrm{s}}}{\gamma^{k_\mathrm{s}}} + \sigma\frac{1-\gamma}{\gamma^{k_\mathrm{s}}}. \end{aligned} \tag{31}$$

We will show that (31) can be rewritten as (19). Then, recalling that $\gamma \in [0,1]$, $\Delta_\mathrm{in}, \Delta_\mathrm{out} \ge 0$, and $k_\mathrm{z} \le k_\mathrm{s}$, it is clear that (19) is stricter than (18) in Lemma III.13 (and thus implies it), hence proving the thesis that all Assumptions III.1, III.4, III.6, and III.9 are compatible.

To show that (31) can be rewritten as (19), rewrite (31) as

$$\sigma > \frac{\gamma^{k_s}\gamma^{k_z-1}\Delta_{\text{in}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})} + \frac{\gamma^{k_z-1}(1-\gamma^{k_s})U_{\text{out}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})} - \frac{\gamma^{k_s}(1-\gamma^{k_z-1})L_{\text{out}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})}. \tag{32}$$

Note that $\gamma^{k_z-1}(1-\gamma^{k_s}) = (\gamma^{k_z-1}-\gamma^{k_s}) + \gamma^{k_s}(1-\gamma^{k_z-1})$. Hence, we have

$$\frac{\gamma^{k_z-1}(1-\gamma^{k_s})U_{\text{out}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})} = \frac{U_{\text{out}}}{1-\gamma} + \frac{\gamma^{k_s}(1-\gamma^{k_z-1})U_{\text{out}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})}$$

and we rewrite (32) as

$$\sigma > \frac{\gamma^{k_s}\gamma^{k_z-1}\Delta_{\text{in}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})} + \frac{U_{\text{out}}}{1-\gamma} - \frac{\gamma^{k_s}(1-\gamma^{k_z-1})\Delta_{\text{out}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})}. \tag{33}$$

Now, note that

$$\begin{aligned}\frac{\gamma^{k_z-1}}{\gamma^{k_z-1}-\gamma^{k_s}} &= \frac{\gamma^{k_z-1}(1-\gamma^{k_s})-(\gamma^{k_z-1}-\gamma^{k_s})}{(\gamma^{k_z-1}-\gamma^{k_s})(1-\gamma^{k_s})} + \frac{1}{1-\gamma^{k_s}} \\ &= \frac{\gamma^{k_s}(1-\gamma^{k_z-1})}{(\gamma^{k_z-1}-\gamma^{k_s})(1-\gamma^{k_s})} + \frac{1}{1-\gamma^{k_s}}\end{aligned}$$

and rewrite (33) as

$$\sigma > \frac{\gamma^{k_s}}{(1-\gamma)(1-\gamma^{k_s})}\Delta_{\text{in}} + \frac{U_{\text{out}}}{1-\gamma} + \frac{\gamma^{2k_s}(1-\gamma^{k_z-1})\Delta_{\text{in}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})(1-\gamma^{k_s})} + \frac{\gamma^{k_s}(1-\gamma^{k_z-1})\Delta_{\text{out}}}{(1-\gamma)(\gamma^{k_z-1}-\gamma^{k_s})}$$

which is immediate to rewrite as (19). □

Lemma A.2. *Given two scalars $a, b \in \mathbb{R}$, with $b \neq 0$, if $|a-b| < |b|$, then* $\text{sign}(a) = \text{sign}(b)$.

Proof: We analyze separately the cases given by the combinations of the signs on $a-b$ and b.

1) Let $b > 0$, $a-b \geq 0$; we have $a \geq b > 0$, that is, $a > 0$.
2) Let $b > 0$, $a-b < 0$; we have $-(a-b) < b$, which simplifies to $a > 0$.
3) Let $b < 0$, $a-b \geq 0$; we have $a-b < -b$ and thus $a < 0$.
4) Let $b < 0$, $a-b < 0$; we have $a < b < 0$, that is, $a < 0$.

□

ACKNOWLEDGMENT

This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

REFERENCES

[1] H. Dong, X. Zhao, and H. Yang, "Reinforcement learning-based approximate optimal control for attitude reorientation under state constraints," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 4, pp. 1664–1673, Jul. 2021.

[2] H. Dong and X. Zhao, "Data-driven wind farm control via multiplayer deep reinforcement learning," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 3, pp. 1468–1475, May 2023.

[3] B. R. Kiran et al., "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, Jun. 2022.

[4] J. Degrave et al., "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, Feb. 2022.

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

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

[7] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 463–471.

[8] AY. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 278–287.

[9] A. Gupta, A. Pacchiano, Y. Zhai, S. Kakade, and S. Levine, "Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 15281–15295.

[10] Y. Dong, X. Tang, and Y. Yuan, "Principled reward shaping for reinforcement learning via Lyapunov stability theory," *Neurocomputing*, vol. 393, pp. 83–90, Jun. 2020.

[11] T. J. Perkins and A. G. Barto, "Lyapunov design for safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, nos. 4–5, pp. 803–832, 2002.

[12] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–9.

[13] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–10.

[14] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.

[15] Z. Marvi and B. Kiumarsi, "Safe reinforcement learning: A control barrier function optimization approach," *Int. J. Robust Nonlinear Control*, vol. 31, no. 6, pp. 1923–1940, 2021.

[16] L. Beckenbach, P. Osinenko, and S. Streif, "A Q-learning predictive control scheme with guaranteed stability," *Eur. J. Control*, vol. 56, pp. 167–178, Nov. 2020.

[17] R. E. Kalman, "When is a linear control system optimal?" *J. Basic Eng.*, vol. 86, no. 1, pp. 51–60, Mar. 1964.

[18] A. E. Bryson, "Optimal control-1950 to 1985," *IEEE Control Syst.*, vol. 16, no. 3, pp. 26–33, Jun. 1996.

[19] L. Rodrigues, "Inverse optimal control with discount factor for continuous and discrete-time control-affine systems and reinforcement learning," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 5783–5788.

[20] É. Garrabé, H. Jesawada, C. Del Vecchio, and G. Russo, "On a probabilistic approach for inverse data-driven optimal control," in *Proc. 62nd IEEE Conf. Decis. Control (CDC)*, Dec. 2023, pp. 4411–4416.

[21] T. Jouini and A. Rantzer, "On cost design in applications of optimal control," *IEEE Control Syst. Lett.*, vol. 6, pp. 452–457, 2022.

[22] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

[23] OpenAI. (2022). *OpenAI Gym Pendulum Online Documentation*. [Online]. Available: https://www.gymlibrary.dev/environments/classic_control/pendulum/

[24] OpenAI. (2022). *OpenAI Gym Lunar Lander Online Documentation*. [Online]. Available: https://www.gymlibrary.dev/environments/box2d/lunar_lander/

[25] F. De Lellis. (2023). *Reward Shaping for RL Based Control*. [Online]. Available: https://github.com/FrancescoDeLellis/Reward-shaping-for-RL-based-control

[26] F. De Lellis, M. Coraggio, G. Russo, M. Musolesi, and M. di Bernardo, "CT-DQN: Control-tutored deep reinforcement learning," in *Proc. 5th Annu. Learn. Dyn. Control Conf.*, vol. 211, 2023, pp. 941–953.

[27] F. De Lellis, M. Coraggio, G. Russo, M. Musolesi, and M. di Bernardo, "Control-tutored reinforcement learning: Towards the integration of data-driven and model-based control," in *Proc. 4th Annu. Learn. Dyn. Control Conf.*, vol. 168, 2022, pp. 1048–1059.

[28] M. Riedmiller et al., "Learning by playing solving sparse reward tasks from scratch," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 4344–4353.

[29] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–21.

[30] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Princeton, NJ, USA: Princeton Univ. Press, 2021.
[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[32] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[33] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th Conf. Artif. Intell. (AAAI)*, Mar. 2016, pp. 2094–2100.
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[35] F. Memarian, W. Goo, R. Lioutikov, S. Niekum, and U. Topcu, "Self-supervised online reward shaping in sparse-reward environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 2369–2375.

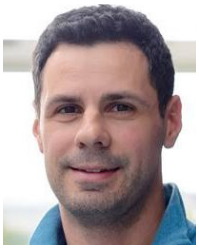
Francesco De Lellis (Member, IEEE) received the Ph.D. degree in information technology and electrical engineering from the University of Naples Federico II, Naples, Italy, in May 2023.

He was a Visiting Researcher at University College Dublin, Dublin, Ireland, in 2020, and University College London, London, U.K., in 2022. He is currently a Post-Doctoral Fellow with the University of Naples Federico II. His research interests include the application of control theory, reinforcement learning, and supervised learning for the development of new methodologies for the control of multiagent complex systems.

Marco Coraggio (Member, IEEE) received the Ph.D. degree in information technology and electrical engineering from the University of Naples Federico II, Naples, Italy, in 2020.

He was a Post-Doctoral Fellow with the University of Naples Federico II from 2020 to 2021. He has been a Post-Doctoral Fellow with the Scuola Superiore Meridionale, School for Advanced Studies, Naples, since 2021. He was a Visiting Student with the University of Bristol, Bristol, U.K., in 2016, and a Visiting Scholar at the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2019, and Linköping University, Linköping, Sweden, in 2023. His current research interests include complex networks and applications, data-driven control, and piecewise smooth and hybrid dynamical systems.

Dr. Coraggio was the finalist, in 2022, and a recipient, in 2023, of the IEEE CSS Italy Young Author Best Paper Award.



Giovanni Russo (Senior Member, IEEE) received the Ph.D. degree from the University of Naples Federico II, Naples, Italy, in 2010.

He was a System Engineer/Integrator of the Honolulu Rail Transit Project with Ansaldo STS, Genova, Italy, from 2012 to 2015; a Research Staff Member in Optimization, Control and Decision Science with IBM Research Ireland, Dublin, Ireland, from 2015 to 2018; and with University College Dublin, Dublin, from 2018 to 2020. He is currently an Associate Professor of automatic control at the University of Salerno, Salerno, Italy. His research interests include contraction theory, analysis/control of nonlinear and complex systems, data-driven sequential decision-making under uncertainty, and control in the space of densities.

Dr. Russo has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS from 2016 to 2019 and the IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS from 2017 to 2023. Since January 2024, he has been serving as a Senior Editor for the IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS. Personal page: https://tinyurl.com/2p8zfpme.

Mirco Musolesi (Member, IEEE) held research and teaching positions at Dartmouth College, Hanover, NH, USA; the University of Cambridge, Cambridge, U.K.; the University of St Andrews, St Andrews, U.K.; and the University of Birmingham, Birmingham, U.K. He is currently a Full Professor of computer science at the Department of Computer Science, University College London, London, U.K., where he leads the Machine Intelligence Laboratory, as part of the Autonomous Systems Research Group. He is also a Full Professor of computer science at the University of Bologna, Bologna, Italy. He has broad research interests spanning several traditional and emerging areas of computer science and beyond. The focus of his laboratory is on machine learning/artificial intelligence and their applications to a variety of theoretical and practical problems and domains. More information about his profile can be found at https://www.mircomusolesi.org.

Mario di Bernardo (Fellow, IEEE) is currently a Professor of automatic control at the University of Naples Federico II, Naples, Italy, and a Visiting Professor of nonlinear systems and control at the University of Bristol, Bristol, U.K. He also serves as a Rector's Delegate for Internationalization at the University of Naples Federico II and coordinates the research area and Ph.D. program on Modeling and Engineering Risk and Complexity of the Scuola Superiore Meridionale, Naples. He authored or coauthored more than 220 international scientific publications, including more than 150 articles in scientific journals, a research monograph, and two edited books.

Dr. di Bernardo was elevated to the grade of fellow of the IEEE in January 2012 for his contributions to the analysis, control, and applications of nonlinear systems and complex networks. In 2017, he received the IEEE George N. Saridis Best Transactions Paper Award for Outstanding Research. On 28th February 2007, he was bestowed the title of Cavaliere of the Order of Merit of the Italian Republic for scientific merits from the President of Italy. He was the President of the Italian Society for Chaos and Complexity from 2010 to 2017, a member of the Board of Governors from 2006 to 2011, and the Vice President for Financial Activities of the IEEE Circuits and Systems Society from 2011 to 2014. In 2015, he was appointed to the Board of Governors of the IEEE Control Systems Society, where he was elected as a member for the term 2023–2025. He was a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2016 to 2017. He is regularly invited as a Plenary Speaker in Italy and abroad. He has been an organizer and a co-organizer for several scientific initiatives and events and received funding from several funding agencies and industries, including the European Union, the U.K. research councils' and the Italian Ministry of Research and University. According to the international database SCOPUS (September 2023), his H-index is 53 and his publications received over 12 000 citations by other authors. He was the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS, a Senior Editor of the IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, and an Associate Editor of the IEEE CONTROL SYSTEMS LETTERS, *Nonlinear Analysis: Hybrid Systems*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II.

Open Access funding provided by 'Università degli Studi di Napoli "Federico II"' within the CRUI CARE Agreement