Lyapunov Design for Robust and Efficient Robotic Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Recent advances in the reinforcement learning (RL) literature have enabled roboticists to automatically train complex policies in simulated environments. However, due to the poor sample complexity of these methods, solving reinforcement learning problems using real-world data remains a challenging problem. This paper introduces a novel cost-shaping method which aims to reduce the number of samples needed to learn a stabilizing controller. The method adds a term involving a control Lyapunov function (CLF) – an 'energy-like' function from the model-based control literature – to typical cost formulations. Theoretical results demonstrate the new costs lead to stabilizing controllers when smaller discount factors are used, which is well-known to reduce sample complexity. Moreover, the addition of the CLF term 'robustifies' the search for a stabilizing controller in the low-data regime. We demonstrate our approach with two hardware examples where we learn stabilizing controllers for a cartpole and an A1 quadruped with only seconds and a few minutes of fine-tuning data, respectively.

Figure 1: We learn precise policies on hardware for the `Quanser` cartpole [1] (top) and the `Unitree A1` quadruped [2] (bottom) using only seconds and a few minutes of real-world data, respectively.

1 Introduction

Recent years have witnessed many success stories for reinforcement learning (RL)-based controllers which are trained using massive amounts of simulation data. The primary obstacle for controller design methods which are based on a nominal dynamics model (including traditional 'model-based' methods) is the 'reality gap' between the simulation environment and the real-world system. This discrepancy occurs because it is generally impractical to model every facet of a real robotic system. This has led to the use of techniques such as domain randomization [3, 4], which attempt to robustify the trained controller by adding random perturbations to the simulated dynamics. Another approach is domain adaptation [5], which trains controllers over a distribution of unknown dynamics parameters and then attempts to identify the 'true' parameters online. In the last few years, these methods have been key ingredients in a number of impressive hardware results [6, 7, 8, 3, 9, 10].

However, despite these successes, there has been a renewed effort to incorporate data from hardware experiments into RL pipelines [11]. These two approaches are not at odds, as 'fine-tuning'

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

[12, 13, 14, 15] approaches can be used to improve a policy based on a simulation model. This is an ideal middle ground which enables engineers to incorporate known features of the system into the model while still allowing model-free methods to improve performance by leveraging complex unknown nonlinearities in the environment. However, the poor sample complexity of RL methods still presents an obstacle, as learning optimal polices can require significant amounts of data.

One of the key parameters in RL problems is the *discount factor*, which controls the effect that long-horizon behaviors have on the overall cost function. Theoretical and empirical results [16] have demonstrated that problems with large discount factors are significantly more challenging to solve. The standard approach to synthesizing stabilizing control policies with RL (and also model-based optimal control approaches [17]) is to simply penalize the distance of the state from the desired operating point. Unfortunately, costs of this form typically require using a high discount factor to learn a stabilizing policy [18, 17]. This is because stabilization is an inherently long-horizon behavior and, when costs of this form are used, the agent must plan over long horizons to see the benefits of stabilizing the system (rather than simply minimizing control effort).

This paper introduces a cost-shaping framework which aims to reduce the sample complexity of learning stabilizing control policies. Our method adds a term involving an (approximate) *control Lyapunov function* (CLF) for the system to the traditional cost formulations discussed above. CLFs are tools from the model-based control literature which are generalized 'energy functions' for the system. Any control law which decreases the value of the CLF at each time-step can be guaranteed to stabilize the system. Thus, CLFs reduce the long-horizon objective of stabilizing the system to a simple one-step condition. The CLF term we add to the cost function rewards actions which decrease the CLF over time, and thus incentivizes policies which stabilize the system. We provide a formal stability analysis which demonstrates that the new cost functions lead to stabilizing control policies, even when small discount factors are used and the learned policy is highly suboptimal.

We illustrate the effectiveness of our cost-shaping technique in three distinct contexts. In our hardware experiments for the cartpole and A1, we fine-tune control policies which are based on a nominal simulation model for the real world system. For the A1 experiment, the nominal controller is a model-based control architecture [19] and we use a CLF which is hand-desired around the desired trotting gait. For the cartpole swing-up task, it is much more difficult to hand-design a CLF for the desired task. Thus, we first train a RL policy for the task using the soft actor-critic algorithm [20] in simulation. We then use the value function learned during the initial training as a CLF for our fine tuning experiments. We discuss why the learned value function is a reasonable CLF for the system in Section 2.2. For both of our hardware experiments we rapidly learn stabilizing controllers by solving problems with small discount factors. Finally, with a toy simulation example of an inverted pendulum we demonstrate how our method can still accelerate learning by using moderate values of discount factors even when the proposed CLF used in our formulation is very crude.

1.1 Related Work

We outline how our approach departs from related work; Appendix A contains further discussion.

Discount Factors, Sample Complexity and Reward Shaping: It is well-understood that the discount factor has a significant effect on the size of the data set that RL algorithms need to achieve a desired level of performance. Specifically, it has been shown in numerous contexts [21, 22, 23, 24] that smaller discount factors lead to problems which can be solved more efficiently. This has led to a number of works which explicitly treat the discount factor as a parameter which can be used to control the complexity of the problem alongside reward shaping techniques [25, 26, 16, 27, 28, 29]. Compared to these works, our primary contribution is to demonstrate how CLFs can be combined with model-free algorithms to rapidly learn stabilizing controllers for robotic systems.

Fine-tuning with Real World Data: Recently, there has been much interest in using RL to fine-tune policies which have been pre-trained in simulation [12, 13, 14, 15]. These methods typically optimize the same cost function with a large discount factor in both simulation and on the real robot. In contrast, using our cost reshaping techniques, we solve a different problem with a smaller discount factor on hardware which can be solved more efficiently. In Appendix D, we show that our method outperforms typical fine-tuning approaches under moderate perturbations to the dynamics model.

Learning with Control Lyapunov Functions: A number of recent works have also tried to overcome the reality gap by using data-driven methods to improve CLF-based controllers [30, 31, 32, 33, 34, 35]. While these methods work very well when a true CLF for the real world system is available, our method is more general as we can still efficiently learn stabilizing controllers when only an approximate CLF is available by modulating the discount factor used to optimize our cost.

2 Background and Problem Setting

Throughout the paper we will consider deterministic discrete-time systems of the form:

$$x_{k+1} = F(x_k, u_k), \tag{1}$$

where $x_k \in \mathcal{X} \subset \mathbb{R}^n$ is the state at time k, $u_k \in \mathcal{U} \subset \mathbb{R}^m$ is the input applied to the system at that time, and $F\colon \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$ is the transition function for the system. This general nonlinear model is broad enough to cover many important continuous control tasks for robotics. We will let Π denote the space of all control polices $\pi\colon \mathcal{X} \to \mathcal{U}$ for the system.

2.1 Control Lyapunov Functions

Control Lyapunov functions [36, 37, 38, 39] are 'energy-like' functions for the dynamics (1). When a CLF is available it greatly simplifies the search for an asymptotically stabilizing policy for the system. This is because, while asymptotic stability is an infinite-horizon property, CLFs reduce the search for a stabilizing policy to a set of greedy one-step criteria. We review the notion of asymptotic stability in Appendix B for unfamiliar readers.

Definition 1. *We say that a positive definite function $W\colon \mathbb{R}^n \to \mathbb{R}$ is a* control Lyapunov function *(CLF) for* (1) *if the following condition holds for each $x \in \mathcal{X}\backslash\{0\}$:*

$$\min_{u \in \mathcal{U}} W(F(x,u)) - W(x) < 0. \tag{2}$$

The condition (2) ensures that for each $x \in \mathcal{X}$ there exists a choice of input which decreases the 'energy' $W(x)$. Any policy which satisfies the one-step condition $W(F(x, \pi(x))) - W(x) < 0$ can be guaranteed to asymptotically stabilize the system [40]. Given a CLF for the system, model-based methods constructively synthesize a controller which satisfies this property using either closed-form equations [37] or by solving an online (convex) optimization problem [41, 39] to satisfy (2) .

Remark 1. *(Designing Control Lyapunov Functions) While there is no general procedure for designing CLFs by hand for general nonlinear systems, there do exist constructive procedures for designing CLFs for many important classes of robotic systems, such as manipulator arms [38] and robotic walkers [39] using structural properties of the system. However, even in these cases, the CLF is typically designed ignoring constraints on the inputs and thus is only a local CLF (such as in the inverted pendulum example we consider in Section 4). In cases where it is too difficult to design a CLF by hand for the desired task, such as in the cartpole swing-up example we consider in Section 4, we demonstrate how to learn a CLF using a simulation environment.*

Remark 2. *(Effects of Model Uncertainty) An additional complication arises when there is a discrepancy between the dynamics of the real robot and the model used to design the CLF. Fortunately, CLFs are known to be robust under moderate dynamics perturbations [42, 30].*

2.2 Stability of Dynamic Programming and Reinforcement Learning

Here we review how 'typical' choices of cost functions found in the literature can be used to learn stabilizing controllers. These results are from the dynamic-programming literature, and will serve as a point of comparison for the stability and robustness properties of our method. In particular, we first consider a running cost $\ell\colon \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ of the form $\ell(x,u) = Q(x) + R(u)$, where $Q\colon \mathbb{R}^n \to \mathbb{R}$ is the state cost and $R\colon \mathcal{U} \to \mathbb{R}$ is the input. Both Q and R are assumed to be positive definite functions (in practice, both are usually quadratic functions). Given a policy $\pi \in \Pi$, discount factor $\gamma \in [0,1]$, and initial condition $x_0 \in \mathcal{X}$ the associated long-run cost is:

$$\begin{aligned} V_\gamma^\pi(x_0) = &\sum_{k=0}^{\infty} \gamma^k \ell(x_k, \pi(x_k)) \\ &\text{s.t. } x_{k+1} = F(x_k, \pi(x_k)), \end{aligned} \tag{3}$$

where $V_\gamma^\pi\colon \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is the *value function* associated to π. Small discount factors incentivize policies which greedily optimize a small number of time steps into the future, while larger discount factors promote policies which reduce the cost in the long-run. We say that a policy $\pi_\gamma^* \in \Pi$ is *optimal* if it achieves the smallest cost from each $x \in \mathcal{X}$:

$$V_\gamma^{\pi_\gamma^*}(x) = V_\gamma^*(x) := \inf_{\pi \in \Pi} V_\gamma^\pi(x), \quad \forall x \in \mathbb{R}^n, \tag{4}$$

where $V_\gamma^* \colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is the *optimal value function*. Together V_γ^* and π_γ^* capture the 'ideal' behavior induced by the cost function (3). Unfortunately, it is impractical to search over the space of all policies Π. This necessitates the use of function approximation schemes (e.g. feed-forward neural networks) to instead represent a subset of policies $\hat{\Pi} \subset \Pi$ to search over. Indeed, modern RL approaches for robotics randomly sample the space of trajectories to optimize problems of the from:

$$\inf_{\pi \in \hat{\Pi}} \mathbb{E}_{x_0 \sim X_0} \big[V_\gamma^\pi(x_0) \big], \tag{5}$$

where X_0 is a distribution over initial conditions. While this approach enables these methods to optimize high-dimensional policies, they are data-hungry, can display high-variance and thus frequently return highly sub-optimal policies when data is limited. To better understand the effect that this has on the stability of learned policies, for each $\pi \in \hat{\Pi}$ define the *optimality gap*:

$$\epsilon_\gamma^\pi(x) = V_\gamma^\pi(x) - V_\gamma^*(x), \tag{6}$$

and recall that from (3), for each $x \in \mathcal{X}$ the policy satifies the difference equation:

$$V_\gamma^\pi(x) = \gamma V_\gamma^\pi(F(x, \pi(x))) + \ell(x, \pi(x)). \tag{7}$$

From these equations we can obtain:

$$\begin{aligned} V_\gamma^\pi(F(x,\pi(x))) - V_\gamma^\pi(x) &= \frac{1}{\gamma}\big(-\ell(x,\pi(x)) + (1-\gamma)V_\gamma^\pi(x)\big) && (8)\\ &= \frac{1}{\gamma}\big(-\ell(x,\pi(x)) + (1-\gamma)[V_\gamma^*(x) + \epsilon_\gamma^\pi(x)]\big) && (9)\\ &\leq \frac{1}{\gamma}\big(-Q(x) + (1-\gamma)[V_\gamma^*(x) + \epsilon_\gamma^\pi(x)]\big), && (10) \end{aligned}$$

where we have first rearranged (7), then used $V_\gamma^\pi(x) = V_\gamma^*(x) + \epsilon_\gamma^\pi(x)$, and finally we have used $\ell(x, \pi(x)) \leq Q(x)$. Inequalities of this sort are the building block for proving the stability of suboptimal polices in the dynamic programming literature [17, 18].

Remark 3. *(Value Functions as CLFs) By inspecting the cost (3) we see that V_γ^π is positive definite (since Q is positive definite). Thus, if the right-hand side of (10) is negative for each $x \in \mathcal{X} \setminus \{0\}$, this inequality shows that V_γ^π is a CLF for (1), and that π is an asymptotically stabilizing control policy. In other words, V_γ^π is a CLF which is implicitly learned during the training process. Indeed, many RL algorithms directly learn an estimate of the value function, a fact which we later exploit to learn a CLF for the cartpole swing up-task in Section 4 using the nominal simulation environment, which is then used for fine-tuning a learned controller with real-world data.*

Note that the RHS of (10) will only be negative if $V_\gamma^*(x) + \epsilon_\gamma^\pi(x) < \frac{1}{1-\gamma}Q(x)$ for each $x \in X$. Since from (3) we know that $V_\gamma^*(x) > Q(x)$ for each $x \in \mathcal{X}$, even the optimal policy (which has no optimality gap) will only be stabilizing if γ is large enough. On the other hand, for a fixed $\gamma \in (0, 1]$, this inequality also explains how sub-optimal a policy can be while maintaining stability.

To make these observations more quantitative we make the following assumptions:

Assumption 1. *There exists $C \geq 1$ such that $V_\gamma^*(x) \leq CQ(x)$ for each $\gamma \in [0, 1]$ and $x \in \mathcal{X}$.*

Assumption 2. *There exists $\delta > 0$ such that the optimality gap for the policy $\pi \in \hat{\Pi}$ satisfies the bound $\epsilon_\gamma^\pi(x) \leq \delta Q(x)$ for each $x \in \mathcal{X}$ and a desired discount factor $\gamma \in (0, 1]$.*

Growth conditions of this form are standard in the literature on the stability of approximate dynamic programming [43, 18, 17, 44]. Noting that $V_\gamma^*(x) \leq V_{\gamma=1}^*(x)$ for each $\gamma \in [0, 1)$, the constant $C \geq 1$ upper-bounds the ratio between the one-step cost and the optimal undiscounted value function. This bound is related to how contractive the associated optimal policy is [43], as if C is smaller then the optimal state trajectory must converge to the origin more rapidly. Assumption 2 provides a measure of how suboptimal the chosen policy is when compared to one-step of the state-cost. The following result is essentially a specialization of the main result from [44]:

Proposition 1. *Let Assumption 1 hold. Let $\pi \in \hat{\Pi}$ be a policy which satisfies Assumption 2 for discount factor $\gamma \in (0, 1]$. Then, π asymptotically stabilizes (1) if $C + \delta < \frac{1}{1-\gamma}$.*

Proof. Combining Assumptions 1 and 2 with equation (10) yields:

$$\gamma V_\gamma^\pi(F(x,\pi(x))) - \gamma V_\gamma^\pi(x) \leq \big(-1 + (1-\gamma)[C+\delta]\big)Q(x) \tag{11}$$

Note that if $C + \delta < \frac{1}{1-\gamma}$ then the right hand side of (11) will be negative definite, which establishes that π asymptotically stabilizes the system. □

Remark 4. *(Stability Properties of Cost Function) In the following section we will derive an analogous result to Proposition 1 for the novel reshaped cost function we propose below. When comparing these results we will primarily focus on the effect of the constant $C \geq 1$ (and the equivalent constant for the new setting). This constant provides an estimate for the size of discount factor that is needed to stabilize (1) when the cost (3) is used. In particular, the discount factor must satisfy $\frac{1}{1-\gamma} > C$ to guarantee stability. The constant C also characterizes how 'robust' the cost function is to suboptimal policies. In particular, for a fixed discount factor, the policy will stabilize the system if $\delta < \frac{1}{1-\gamma} + C$, thus smaller values of C permit more suboptimal policies.*

3 Lyapunov Design for Infinite Horizon Reinforcement Learning

Our method uses a candidate control Lyapunov function $W\colon \mathbb{R}^n \to \mathbb{R}$ for the nonlinear dynamics (1), and reshapes (3) to our proposed new long horizon cost $\tilde{V}_\gamma^\pi\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$:

$$\tilde{V}_\gamma^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k \Big(\big[W\big(F(x_k,\pi(x_k))\big) - W(x_k)\big] + \ell(x_k,\pi(x_k)) \Big) \tag{12}$$
$$\text{s.t. } x_{k+1} = F(x_k,\pi(x_k)).$$

Here the term 'candidate' means that we guess that W is a CLF for the system. As we shall see below, our method works best when W is in fact a CLF for the system, but still provides benefits when it is only an 'approximate' CLF for the system (in a sense we will make precise later).

Given the reshaped cost, we then solve the following policy optimization problem:

$$\inf_{\pi \in \hat{\Pi}} \mathbb{E}_{x_0 \sim X_0}\big[\tilde{V}_\gamma^\pi(x_0)\big] \tag{13}$$

The new cost function (12) includes the amount that W changes at each time step, and thus encourages choices of inputs which decrease W over time. For very small discount factors, the new cost biases the policy search in (13) towards policies which myopically decrease the value of W at each step. To see this, note that in the extreme case where $\gamma = 0$ we can simplify the cost to:

$$\tilde{V}_0^\pi(x_0) = W(F(x_0,\pi(x_0))) - W(x_0) + Q(x_0) + R(\pi(x_0)). \tag{14}$$

Since the policy cannot affect the initial condition $x_0 \in \mathbb{R}$, the only terms in (14) which affect the policy optimization (13) are $W(F(x,\pi(x)))$ and $R(\pi(x))$. Thus, optimal policies will disregard the original state cost Q, and myopically balance decreasing W against the input cost R. On the other hand, when γ is large, optimal polices can increase the value of W in the short run, if it means that W and Q can both be decreased in the long-run. The new optimal value function is given by:

$$\tilde{V}_\gamma^*(x) = \inf_{\pi \in \Pi} \tilde{V}_\gamma^\pi(x), \tag{15}$$

and each policy must now satisfy the new difference equation:

$$\tilde{V}_\gamma^\pi(x) = \gamma \tilde{V}_\gamma^\pi\big(F(x,\pi(x))\big) + W\big(F(x,\pi(x))\big) - W(x,\pi(x)) + \ell(x,\pi(x)). \tag{16}$$

In our stability analysis we will use the following composite function as a candidate CLF for (1):

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) = W(x) + \gamma \tilde{V}_\gamma^\pi(x). \tag{17}$$

Defining for each $\pi \in \hat{\Pi}$ and $x \in \mathcal{X}$ the new optimality gap:

$$\tilde{\epsilon}_\gamma^\pi(x) = \tilde{V}_\gamma^*(x) - \tilde{V}_\gamma^\pi(x), \tag{18}$$

and following steps analogous to those taken in (8)-(10), we can obtain the following:

$$\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi\big(F(x,\pi(x))\big) - \tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi(x) = -\ell(x,\pi(x)) + (1-\gamma)\tilde{V}_\gamma^\pi(x) \tag{19}$$
$$= -\ell(x,\pi(x)) + (1-\gamma)\big[\tilde{V}_\gamma^*(x) + \tilde{\epsilon}_\gamma^\pi(x)\big] \tag{20}$$
$$\leq -Q(x) + (1-\gamma)\big[\tilde{V}_\gamma^*(x) + \tilde{\epsilon}_\gamma^\pi(x)\big]. \tag{21}$$

Similar to the analysis in the previous section, we will aim to understand when the right hand side of (21) is negative, as this will characterize when π stabilizes the system. One key difference between the inequality (10) and (21) is that, while the original value function V_γ^* is necessarily positive definite, $\tilde{V}_\gamma^*$ can actually take on negative values since the addition of the CLF term allows the new running cost in (12) to be negative. As we shall see, this forms the basis for the stability and robustness properties our cost formulation enjoys when W is designed properly.

Remark 5. *(Learning Corrections to W) Even though $\tilde{V}_\gamma^\pi$ can take on negative values, in Lemma 2 in the appendix we demonstrate that $\tilde{\boldsymbol{\mathcal{V}}}_\gamma^\pi = W + \tilde{V}_\gamma^\pi$ is a positive definite function. Thus, when the right hand side of (21) is negative for each $x \in \mathcal{X} \setminus \{0\}$, inequality (21) demonstrates that $\boldsymbol{\mathcal{V}}_\gamma^\pi$ is in fact a CLF for (1) and that π stabilizes the system. We can think of W as an 'initial guess' for a CLF for the system, while γV_γ^π is a 'correction' to W that is implicitly made by a learned policy π. Roughly speaking, the larger the discount factor, the larger this correction. Thus the user can trade-off how much the learned policy is able to correct the candidate CLF W against the additional complexity of solving a problem with a higher discount factor, depending on how good of a CLF candidate W is believed to be for the real robot.*

We first state a general stability result for suboptimal policies associated to the new cost, and then discuss how the choice of W affects the stability of suboptimal control policies:

Assumption 3. *There exists $\tilde{C} \in \mathbb{R}$ such that $V_\gamma^*(x) \leq \tilde{C}Q(x)$ for each $\gamma \in [0,1]$ and $x \in \mathcal{X}$.*

Assumption 4. *There exists $\tilde{\delta} > 0$ such that the optimality gap for the policy $\pi \in \hat{\Pi}$ satisfies the bound $\tilde{\epsilon}_\gamma^\pi(x) \leq \tilde{\delta}Q(x)$ for each $x \in \mathcal{X}$ and a desired discount factor $\gamma \in (0,1]$.*

These Assumptions are entirely analogous to Assumptions 1 and 2, except in Assumption 3 the upper bound on the value function is allowed to take on negative values.

Theorem 1. *Let Assumption 3 hold. Let $\pi \in \hat{\Pi}$ be a policy which satisfies Assumption 4 for discount factor $\gamma \in (0,1]$. Then π asymptotically stabilizes (1) if $\tilde{C} + \tilde{\delta} < \frac{1}{1-\gamma}$.*

The proof is conceptually similar to the proof of Proposition 1, and follows from the inequality (21) and Assumptions 3 and 4; we delegate the proof to Appendix C for brevity. Indeed, note that the conditions for stability under the new cost are essentially identical to those for the previous cost in Proposition 1. As alluded to in Remark 4, we will primarily focus on comparing how large the constants $C \geq 1$ and $\tilde{C} \in \mathbb{R}$ are for the two problems, as they control the discount factor required to learn a stabilizing policy and also the 'robustness' of the cost. The following result provides a condition on W which ensures that $\tilde{C} \leq 0$. This condition is from the model-predictive control literature [45, 46], which use CLFs as terminal costs. Proof can be found in Appendix C.

Lemma 1. *Suppose there exists $M \geq 0$ such that for each $x \in \mathcal{X}$ the following condition holds:*

$$\inf_{u \in U} W(F(x,u)) - W(x) + \ell(x,u) \leq -MQ(x). \tag{22}$$

Then Assumption 3 is satisfied with constant $\tilde{C} = -M$.

Remark 6. *(Robustness of reshaped cost) The condition in Lemma 1 implies that W is a CLF for the system which 'dominates' the original running cost ℓ, in the sense that at each time step W can be decreased faster than the running cost ℓ accumulates. In this case, we will have $\tilde{C} < C$ implying the new cost enjoys the desirable properties discussed above. Note that the constant $M > 0$ in Lemma 1 can be increased by putting more weight on W in the cost (12). In cases where W is only an 'approximate' CLF for the system, meaning that W cannot be decreased at each time step, but can instead be decreased over a window of a small number of time steps, V_γ^* may still take on negative values and contribute to the right hand side of (21) being negative. However, in this case moderate values of the discount factor may be needed to see these benefits, as policy optimization algorithms must implicitly plan further into the future to see the benefits of 'going down' W. The inverted pendulum example in Section 4 illustrates this point.*

4 Examples and Practical Implementations

We highlight the main results for each of our examples, but leave certain details to Appendix D.

Fine-tuning a Learned Policy for Cartpole Swing-Up: We first fine-tune a swing-up controller for the `Quanser` cartpole system [1] using real world data and an initial policy which was pre-trained

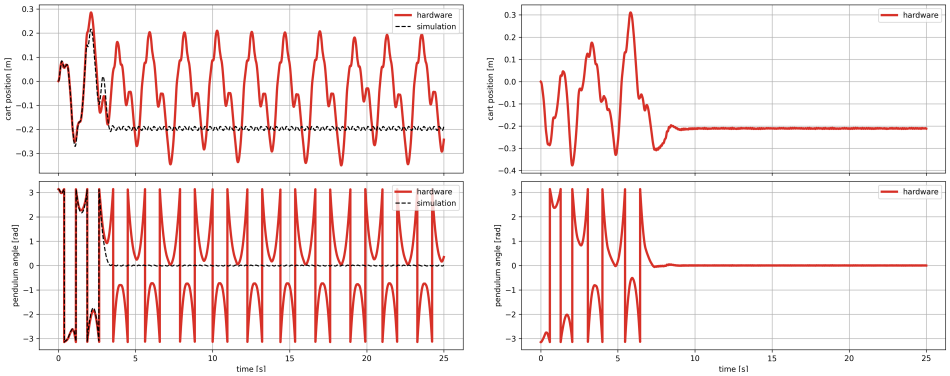

Figure 2: Plots of the cart position and pendulum angle of the cartpole system. (left) the policy trained only in simulation fails to bring the real cartpole system to the upright position; (right) by fine-tuning the learned policy with $20s$ of real-world data using our CLF-based reward function, we obtain a successful policy. Videos of the experiments can be found in the supplementary material.

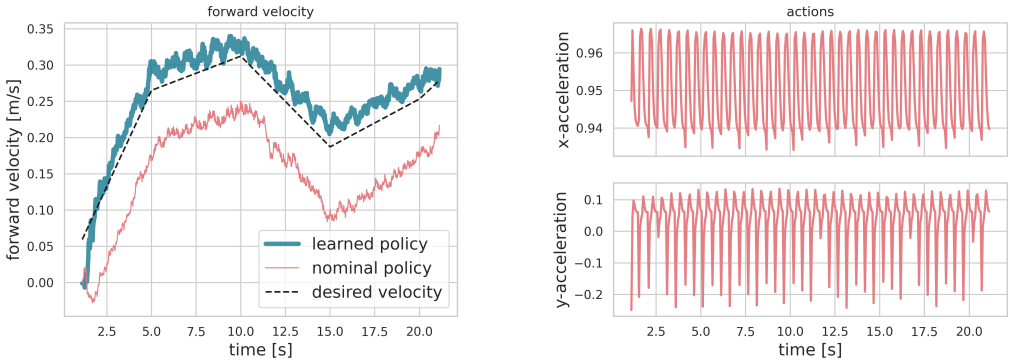

Figure 3: Plots illustrating our proposed method to fine-tune a locomotion controller on the `Unitree A1` robot hardware. (left) we illustrate improved velocity tracking of the learned policy (in dark green) compared to the nominal locomotion controller (in pink) to track a desired velocity profile (in dashed black line); (right) we show the normalized actions of the learned policy, which display a highly periodic profile which correlates strongly with individual walking steps.

in simulation but that does not translate well to the real system. Due to the underactuated nature of the system, synthesizing a CLF, even when input constraints are ignored, is challenging. Thus, as alluded to previously, we use a 'typical' cost function of the form (3) and a discount factor of $\gamma = 0.999$ to learn a stabilizing neural network policy π_ϕ for a simulation model of the system using the SAC algorithm [47]. Given the discussion in Section 2.2, the value function V_θ associated with the learned policy will be a CLF for the simulation environment, and we use this as the candidate CLF ($W = V_\theta$) in our reward reshaping formulation (12) for fine-tuning on hardware. In particular, when improving the simulation-based policy π_ϕ with real world data, we keep the parameters of this network fixed and learn an additional smaller policy π_ψ (so that the overall control action is produced by $\pi_\phi + \pi_\psi$) using our proposed CLF-based cost formulation. We solve the reshaped problem with a discount factor $\gamma = 0$ and collect rollouts of $10s$ on hardware. Our CLF-based fine-tuning approach is able to successfully complete the swing-up task after collecting data from just one rollout, however the behavior of the policy near the top is not smooth and sometimes fails. After collecting data from an additional rollout, the controller is much more reliable and robust to recover from several pushes, as demonstrated in the supplementary video. A successful swing-up motion is shown in Figure 2. Finally, in Appendix D, we compare our method with a typical fine-tuning approach which uses the same cost in both phases. We show that our method outperforms the baseline under moderate perturbations to the dynamics model.

Improving Velocity Tracking for a Quadruped Locomotion Controller: Next, we fine-tune a model-based trotting controller [19] for a quadruped robot walking at different speeds using a CLF which is designed around the desired gait. The tracking performance of this controller depends heavily on the chosen feedback gains. As illustrated by the pink curve in Fig. 3 (left), large steady-

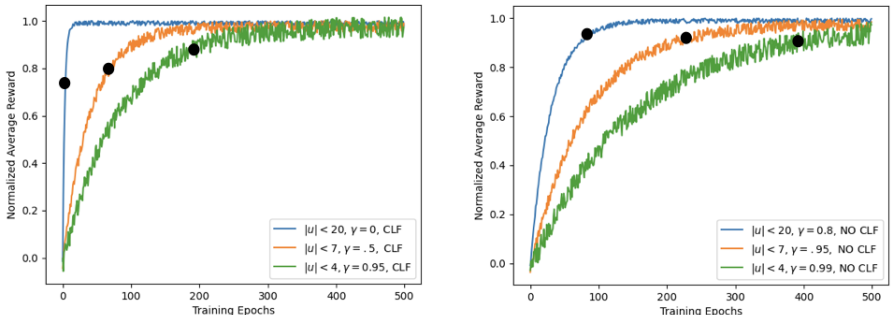

Figure 4: Learning curves for an inverted pendulum system under different input constraints. The curves plotted correspond to the smallest discount factors that led to stabilizing policies. On the left, the obtained learning curves use a CLF in the reward. On the right, the reward does not include the CLF term. The black dots denote the first stabilizing policy for each training.

state errors persist when the controller is not well tuned. Furthermore, when the space of control parameters is large, tuning them by hand may not be straightforward. To address these limitations, we augment a policy π_ψ, to our nominal controller. To learn the parameters ψ, we first design a candidate CLF (W) based on a nominal linearized model of the robot, which is used to define our reward function. We then collect rollouts of $10s$ on the robot hardware with randomly chosen desired velocity profiles, and a discount factor $\gamma = 0$. Our approach is able to learn a policy which significantly improves the tracking performance of the nominal controller within 5 minutes (30 episodes) of hardware data. More details of this experiment are provided in Appendix D.

Inverted Pendulum with Input Constraints: Our final example demonstrates the utility of our method even when W is a crude guess for a CLF for the system, through the use of moderate discount factors. We illustrate this for a simple inverted pendulum by varying the magnitude of the input constraints for the system. We use the procedure from [39] to design a candidate CLF for the system. Like many CLF design techniques, this approach assumes there are no input constraints, and guides the pendulum to swing directly to the top. However, when the input constraints are too strict, this motion is infeasible and multiple pumps of the arm are required to swing upright. Thus, W is only a *local* CLF for the system, meaning it satisfies the CLF condition (2) only near the upright position. In the following simulations, we consider inputs bounds of the form $|u| \leq M$ for $M \in \{20, 7, 4\}$. For $M = 20$ the system has the control authority to swing directly to the top, while for $M = \{7, 4\}$ multiple pumping motions are required to gain enough momentum. Using the candidate CLF, we first train a stabilizing controller using a 'typical' cost function of the form (3), and then train a controller using the reshaped cost (12). For both forms of cost function, we sweep across different values of discount factor to approximately determine the smallest discount factor that can learn a stabilizing controller for each cost. In Figure 4, we display the training curves for the smallest discount factor which lead to a stabilizing controller (see Appendix D for details on how we determined if a controller is stabilizing). On each of the training curves the black dot denotes the first training epoch at which a stabilizing controller was obtained. As the plots clearly demonstrate, the addition of the CLF enables our method to more rapidly learn a stabilizing controller, but the effect is more pronounced when W is a better candidate CLF, namely when the input bound is larger. Moreover, in line with the presented theory, these results show that our method is able to more efficiently learn stabilizing policies with smaller discount factors.

5 Discussion and Limitations

As we have mentioned previously, our approach has several limitations. The cost shaping technique we introduce in Section 3 only provides benefits when W is in-fact a reasonable guess for a CLF for the true system. This requires that the user have a dynamics model which captures the primary features of the environments which affect the structure of CLFs for the system. While the cartpole simulations we provide in the Appendix D provide some intuition for when this will be the case, further research is needed to better understand in what scenarios we can see significant benefits from our method. Nonetheless, our two hardware experiments provide encouraging initial results which indicate that our method can rapidly learn stabilizing controllers using CLFs which are constructed using a nominal dynamics model. More broadly, there are many exciting avenues for further incorporating Lyapunov design techniques with RL, especially offline learning [11].

References

[1] Quanser. Linear servo base unit with inverted pendulum, Apr 2021. URL https://www.quanser.com/products/linear-servo-base-unit-inverted-pendulum/.

[2] U. Robotics. A1. URL https://www.unitree.com/products/a1/.

[3] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[4] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[5] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[6] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020.

[7] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[8] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[9] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. *arXiv preprint arXiv:2103.14295*, 2021.

[10] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robotics and Automation Letters*, 6(2):1471–1478, 2021.

[11] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[12] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. *arXiv preprint arXiv:2110.05457*, 2021.

[13] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv preprint arXiv:2004.10190*, 2020.

[14] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman. Efficient adaptation for end-to-end vision-based robotic manipulation. 2020.

[15] Z. Mandi, P. Abbeel, and S. James. On the effectiveness of fine-tuning versus meta-reinforcement learning. *arXiv preprint arXiv:2206.03271*, 2022.

[16] V. François-Lavet, R. Fonteneau, and D. Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*, 2015.

[17] V. Gaitsgory, L. Grüne, and N. Thatcher. Stabilization with discounted optimal control. *Systems & Control Letters*, 82:91–98, 2015.

[18] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz. Stability analysis of discrete-time infinite-horizon optimal control with discounted cost. *IEEE Transactions on Automatic Control*, 62(6): 2736–2749, 2016.

[19] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg. Learning a contact-adaptive controller for robust, efficient legged locomotion. In *Conference on Robot Learning*, pages 883–894. PMLR, 2021.

[20] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[21] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] R. Munos and C. Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

[24] R. Postoyan, M. Granzotto, L. Buşoniu, B. Scherrer, D. Nešsić, and J. Daafouz. Stability guarantees for nonlinear discrete-time systems controlled by approximate value iteration. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 487–492. IEEE, 2019.

[25] N. Jiang, A. Kulesza, S. Singh, and R. Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189. Citeseer, 2015.

[26] M. Petrik and B. Scherrer. Biasing approximate dynamic programming with a lower discount factor. *Advances in neural information processing systems*, 21:1265–1272, 2008.

[27] C. Tessler and S. Mannor. Reward tweaking: Maximizing the total reward while planning for short horizons. *arXiv preprint arXiv:2002.03327*, 2020.

[28] C.-A. Cheng, A. Kolobov, and A. Swaminathan. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34:13550–13563, 2021.

[29] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[30] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames. Episodic learning with control lyapunov functions for uncertain robotic systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6878–6884. IEEE, 2019.

[31] A. J. Taylor, V. D. Dorobantu, M. Krishnamoorthy, H. M. Le, Y. Yue, and A. D. Ames. A control lyapunov perspective on episodic learning via projection to state stability. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1448–1455. IEEE, 2019.

[32] T. Westenbroek, F. Castañeda, A. Agrawal, S. S. Sastry, and K. Sreenath. Learning min-norm stabilizing control laws for systems with unknown dynamics. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 737–744. IEEE, 2020.

[33] T. Westenbroek, A. Agrawal, F. Castañeda, S. S. Sastry, and K. Sreenath. Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers. *IFAC-PapersOnLine*, 54(5):19–24, 2021.

[34] F. Castañeda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath. Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics. In *2021 American Control Conference (ACC)*, pages 3683–3690, 2021.

[35] J. Choi, F. Castañeda, C. Tomlin, and K. Sreenath. Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions. In *Robotics: Science and Systems*, Corvalis, OR, July 2020.

[36] Z. Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11):1163–1173, 1983.

[37] E. D. Sontag. A 'universal' construction of artstein's theorem on nonlinear stabilization. *Systems and Control Letters*, 13(2):117 – 123, 1989.

[38] A. D. Ames and M. Powell. Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs. *Lecture Notes in Control and Information Sciences*, 449:219–240, 2013.

[39] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, 2014.

[40] C. M. Kellett and A. R. Teel. Results on discrete-time control-lyapunov functions. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 6, pages 5961–5966. IEEE, 2003.

[41] R. Freeman and P. V. Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science and Business Media, 2008.

[42] Y. S. Ledyaev and E. D. Sontag. A lyapunov characterization of robust stabilization. *Nonlinear Analysis: Theory, Methods & Applications*, 37(7):813–840, 1999.

[43] B. Lincoln and A. Rantzer. Relaxing dynamic programming. *IEEE Transactions on Automatic Control*, 51(8):1249–1260, 2006.

[44] V. Gaitsgory, L. Grüne, C. M. Kellett, and S. R. Weller. Stabilization with discounted optimal control: the discrete time case. 2016.

[45] A. Jadbabaie, J. Yu, and J. Hauser. Receding horizon control of the caltech ducted fan: A control lyapunov function approach. In *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328)*, volume 1, pages 51–56. IEEE, 1999.

[46] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel. Model predictive control: for want of a local control lyapunov function, all is not lost. *IEEE Transactions on Automatic Control*, 50 (5):546–558, 2005.

[47] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.