Coordinating Planning and Tracking in Layered Control Policies via Actor-Critic Learning

Fengjun Yang and Nikolai Matni*

April 1, 2024

Abstract

We propose a reinforcement learning (RL)-based algorithm to jointly train (1) a trajectory planner and (2) a tracking controller in a layered control architecture. Our algorithm arises naturally from a rewrite of the underlying optimal control problem that lends itself to an actor-critic learning approach. By explicitly learning a *dual* network to coordinate the interaction between the planning and tracking layers, we demonstrate the ability to achieve an effective consensus between the two components, leading to an interpretable policy. We theoretically prove that our algorithm converges to the optimal dual network in the Linear Quadratic Regulator (LQR) setting and empirically validate its applicability to nonlinear systems through simulation experiments on a unicycle model.

1 Introduction

Layered control architectures (Matni et al., 2024; Chiang et al., 2007) are ubiquitous in complex cyber-physical systems, such as power networks, communication networks, and autonomous robots. For example, a typical autonomous robot has an autonomy stack consisting of decision-making, trajectory optimization, and low-level control. However, despite the widespread presence of such layered control architectures, there has been a lack of a principled framework for their design, especially in the data-driven regime.

In this work, we propose an algorithm for jointly learning a trajectory planner and a tracking controller. We start from an optimal control problem and show that a suitable relaxation of the problem naturally decomposes into reference generation and trajectory tracking layers. We then propose an algorithm to train a layered policy parameterized in a way that parallels this decomposition using actor-critic methods. Different from previous methods, we show how a *dual network* can be trained to coordinate the trajectory optimizer and the tracking controller. Our theoretical analysis and numerical experiments demonstrate that the proposed algorithm can achieve good performance in various settings while enjoying inherent interpretability and modularity.

1.1 Related Work

1.1.1 Layered control architectures

The idea of layering has been studied extensively in the multi-rate control literature (Rosolia et al., 2022; Csomay-Shanklin et al., 2022), through the lens of optimization decomposition (Chiang et al., 2007; Matni and Doyle, 2016), and for specific application domains (Samad et al., 2007; Samad and Annaswamy, 2017; Jiang, 2018). Recently, Matni et al. (Matni et al., 2024) proposed a quantitative framework for the design and analysis of layered control architectures, which has since been instantiated to various control and robotics applications (Srikanthan et al., 2023a,b; Zhang et al., 2024). Within this framework, our work is most related to Srikanthan et al. (2023b); Zhang et al. (2024), which seek to design trajectory planners based on past data of a tracking controller. However, we consider the case where the low-level tracking controller is not given and has to be learned with the trajectory planner. We also provide a more principled approach to coordinating planning and tracking that leverages a dual network.

1.1.2 Hierarchical reinforcement learning

Recently, reinforcement learning-based methods have demonstrated impressive performance on highly complex dynamical systems (Kumar et al., 2021; Kaufmann et al., 2023). Within the RL literature, our approach is most closely related to the idea of *goal-conditioned* reinforcement learning (Dayan and Hinton, 1992; Kulkarni et al., 2016; Levy et al., 2017; Nachum et al., 2018a; Vezhnevets et al., 2017; Nachum et al., 2018b). In this framework, an upper-level agent periodically specifies a goal for the lower-level agent to execute. However, the "intrinsic" reward used to train the lower-level agent is usually heuristically chosen. Nachum et al. (Nachum et al., 2018b) derived a principled objective for the lower-level agent based on a suboptimality bound introduced by the hierarchical structure, but they focus on the case where the goal is specified as a learned low-dimensional representation. We focus on the case where the dynamics are deterministic and derive a simple quadratic objective for the lower-level agent (tracking layer). We also structure our upper-level agent (planning layer) to generate full trajectories instead of single waypoints.

1.1.3 Actor-critic methods

The actor-critic method (Silver et al., 2014; Lillicrap et al., 2015; Fujimoto et al., 2018) describes a class of reinforcement learning algorithms that simultaneously learn a policy and its associated value function. These algorithms have achieved great success with continuous control tasks and have found various applications in the controls and robotics community (Wang and Fazlyab, 2024; Grandesso et al., 2023). In this paper, we use actor-critic methods to learn a tracking controller and its value function, where the latter is used to help the trajectory planner determine how difficult a generated trajectory is for the tracking controller to follow.

1.2 Statement of Contributions

Our contribution is three-fold. First, we propose a novel way of parameterizing layered policies based on principled derivation. In this parameterization, we introduce a *dual network* to coordinate the trajectory planner and the tracking controller. We show how this dual network can be trained jointly with other components in the layered policy in an RL fashion. Secondly, we

show theoretically and empirically that our algorithm for updating the dual network can recover the optimal dual network parameters for unconstrained linear quadratic regulator (LQR) problems. Finally, we evaluate our method empirically on constrained LQR problems and the unicycle environment to demonstrate its potential to be applied to more complex systems.

2 Problem Formulation

We consider a discrete-time finite-horizon optimal control problem with state $x_t \in \mathbb{R}^{d_x}$ and control input $u_t \in \mathbb{R}^{d_u}$:

$$
\begin{aligned}
\underset{x_{0:T}, u_{0:T-1}}{\text{minimize}} \quad & \mathbb{E}_{\xi \sim D_\xi} \left[\mathcal{C}(x_{0:T}) + \mathcal{D}(u_{0:T-1}) \right] \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t = 0, 1, ..., T-1, \\
& x_{0:T} \in \mathcal{X}, \quad u_{0:T-1} \in \mathcal{U} \\
& x_0 = \xi.
\end{aligned}
\tag{1}
$$

Here, $T \in \mathbb{Z}^+$ is a fixed time horizon, $x_{0:T} = [x_0^\top, ..., x_T^\top]^\top$ and $u_{0:T-1} = [u_0^\top, ..., u_{T-1}^\top]^\top$ respectively denote the state and control trajectory, and $\mathcal{C}(x_{0:T})$ and $\mathcal{D}(u_{0:T-1})$ are the state and control costs, respectively. We assume that the state and input constraints decouple across time, and denote the state constraints by $\mathcal{X} = \prod_{t=0}^{T} \mathcal{X}_t$, and the control input constraints $\mathcal{U} = \prod_{t=0}^{T} \mathcal{U}_t$. The initial condition ξ is sampled i.i.d. from a possibly unknown distribution D_ξ.

As per the reinforcement learning convention, we assume that we only have access to the dynamics via a simulator, i.e., that we do not know $f(x_t, u_t)$ explicitly, but can simulate the dynamics for any x_t and u_t. However, we do assume that we have access to the cost functions $\mathcal{C}, \mathcal{D}$, as they are usually designed by the users, instead of being an inherent hidden part of the system. We also assume that we know the constraints $\mathcal{X}$ and $\mathcal{U}$ for the same reason.

Our goal is to learned a layered policy $\pi = (\pi^{\text{plan}}, \pi^{\text{track}})$ that consists of 1) a trajectory planner

$$
\pi^{\text{plan}} : \mathbb{R}^{d_x} \to \mathbb{R}^{T d_x}
$$

that takes in an initial condition $\xi \in \mathbb{R}^{d_x}$ and outputs a reference trajectory $r_{0:T} \in \mathcal{X}$, and 2) a tracking controller

$$
\pi^{\text{track}} : \mathbb{R}^{d_x} \times \mathbb{R}^{T d_x} \to \mathbb{R}^{d_u}
$$

that takes in the current state and a reference trajectory to output a control action to best track the given trajectory. We now decompose problem (1) such that it may inform a suitable parameterization for the planning and tracking policies, π^{plan} and π^{track}.

3 Layered Approach to Optimal Control

We first consider a variation of problem (1) with a fixed initial condition ξ, and rewrite it into a form that has a natural layered control architecture interpretation. For ease of notation, we use unsubscripted letters x, u, r to denote the respective trajectories stacked as a column vector

$$
x := x_{0:T}, \quad u := u_{0:T-1}, \quad r := r_{0:T}.
$$

We begin the rewrite of problem (1) by introducing a redundant variable $r = x$ to get an equivalent optimization problem

$$\begin{aligned}
\underset{r,x,u}{\text{minimize}} \quad & \mathcal{C}(r) + \mathcal{D}(u) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t = 0, 1, ..., T-1, \\
& r \in \mathcal{X}, \quad u \in \mathcal{U}, \\
& x_0 = \xi, \quad r = x,
\end{aligned} \tag{2}$$

where we use the fact that $r = x$ to move the state cost and constraint from x onto r. Defining the indicator functions

$$\mathbb{I}_{dyn}(x,u) = \begin{cases} 0, & \begin{aligned} & x_0 = \xi, \\ & x_{t+1} = f(x_t, u_t), \forall t, \\ & u \in \mathcal{U}, \end{aligned} \\ \infty, & \text{otherwise} \end{cases}, \qquad \mathbb{I}_{state}(r) = \begin{cases} 0, & r \in \mathcal{X} \\ \infty, & \text{otherwise} \end{cases},$$

we write the partial augmented Lagrangian of problem (2) in terms of the (scaled) dual variable ν

$$\mathcal{L}_\rho(r, x, u, \nu) = \mathcal{C}(r) + \mathcal{D}(u) + \mathbb{I}_{dyn}(x, u) + \mathbb{I}_s(r) + \frac{\rho}{2}\|r + \nu - x\|_2^2 - \frac{\rho}{2}\|\nu\|_2^2. \tag{3}$$

Applying dual ascent to this augmented Lagrangian, we obtain the following method-of-multiplier updates

$$\begin{aligned}
(r^+, x^+, u^+) = \underset{x,u,r}{\text{argmin}} \ & \mathcal{C}(r) + \mathcal{D}(u) + \frac{\rho}{2}\|r + \nu - x\|_2^2 \\
\text{s.t.} \ & x_{t+1} = f(x_t, u_t), \quad \forall t, \\
& r \in \mathcal{X}, u \in \mathcal{U}, \\
& x_0 = \xi
\end{aligned} \tag{4}$$

$$\nu^+ = \nu + (r^+ - x^+), \tag{5}$$

which will converge to locally optimal primal and dual variables r^*, x^*, u^*, ν^* given mild assumptions on the smoothness and convexity of $\mathcal{C}, \mathcal{D}$ and the constraints in the neighborhood of the optimal point (See Bertsekas (2014, §2)).

For a layered interpretation, we note that the primal update (4) can be written as a nested optimization problem

$$\begin{aligned}
r^+ = \underset{r}{\text{minimize}} \quad & \mathcal{C}(r) + p^*(r + \nu; \xi) \\
\text{s.t.} \quad & r \in \mathcal{X}
\end{aligned} \tag{6}$$

where $p^*(r + \nu; \xi)$ is the locally optimal value of the (x, u)-minimization step

$$\begin{aligned}
p^*(r + \nu; \xi) = \underset{x,u}{\min} \quad & \mathcal{D}(u) + \frac{\rho}{2}\|r + \nu - x\|_2^2 \\
\text{s.t.} \quad & x_{t+1} = f(x_t, u_t), \ u \in \mathcal{U}, \ \forall t, \\
& x_0 = \xi.
\end{aligned} \tag{7}$$

We immediately recognize that optimal control problem (7) is finding the control action u to minimize a quadratic tracking cost for the reference trajectory

$$\tilde{r} := r + \nu.$$

Thus, this nested rewrite can be seen as breaking the primal minimization problem (4) into a trajectory optimization problem (6) that seeks to find the best reference r and a tracking problem (7) that seeks to best track the *perturbed* trajectory $\tilde{r}$. An important subtlety here is that the planned trajectory, r, and the trajectory sent to the tracking controller, $\tilde{r}$, are different. To understand this discrepancy, let us first consider a similar, but perhaps more intuitive, reference optimization problem:

$$\begin{aligned} \underset{r}{\text{minimize}} \quad & \mathcal{C}(r) + p^*(r; \xi) \\ \text{s.t.} \quad & r \in \mathcal{X}. \end{aligned} \tag{8}$$

This heuristics-based approach, employed in previous works such as Srikanthan et al. (2023b); Zhang et al. (2024), seeks to find a reference that balances minimizing the nominal cost $\mathcal{C}(r)$ and not incurring high tracking cost $p^*(r; \xi)$. In these works, the solution r is then sent to the tracking controller unperturbed. A problem with this approach is that unless the tracking controller can execute the given reference perfectly, the executed trajectory x will differ from the planned reference r. One can mitigate this deviation by multiplying the tracking cost with a large weight, but this can quickly become numerically ill-conditioned, or bias the planned trajectory towards overly conservative and easy-to-track behaviors.

Returning to the method-of-multiplier updates (4) and (5), we note that, under suitable technical conditions, solving the planning layer problem (6) using the locally optimal dual variable ν^* leads to the feasible solution satisfying $r^* = x^*$. In particular, the perturbed reference trajectory $\tilde{r} = r + \nu^*$ is sent to the tracking controller defined by problem (7), and this results in the executed state trajecotry x^* matching the reference $x^* = r^*$. This discussion highlights the role of the locally optimal dual variable as coordinating the planning and tracking layers, and motivates our approach of explicitly modeling this dual variable in our learning framework.

Following this intuition, in the next section, we show how to parameterize π^{plan} and π^{track} to approximately solve (6) and (7), respectively. In practice, finding ν^* with the iterative update in (5) can be prohibitively expensive. To circumvent this issue, we recognize that any locally optimal dual variable ν^* can be written as a function of the initial condition ξ. We thus seek to learn an approximate map to predict this locally optimal dual variable ν^* from the initial condition ξ.[1]

We close this section by noting that the above derivation assumes that the reference trajectory is of the same dimension as the state, i.e., that $r_t = x_t$. However, if the state cost $\mathcal{C}$ and constraints $\mathcal{X}$ only require a subset of the states, i.e., if they are defined in terms of $z_t = g(x_t) \in \mathbb{R}^{d_z}$, with $d_z < d_x$, then one can modify the discussion above by replacing the redundant constraint $x = r$ with $z = r$, so that the reference only needs to be specified on the lower dimensional output z. We refer the readers to Appendix D for the details.

4 Actor-Critic Learning in the Layered Control Architecture

4.1 Parameterization of the Layered Policy

We parameterize our layered policy $\pi = (\pi^{\text{plan}}, \pi^{\text{track}})$ so that its structure parallels the dual ascent updates (6) and (7). The tracking controller $\pi_\phi^{\text{track}} : \mathbb{R}^{d_x} \to \mathbb{R}^{T d_x}$, specified by learnable parameters

---

[1]We have been somewhat cavalier in our assumption that such a locally optimal dual variable ν^* exists. We note that notions of local duality theory, see for example (Luenberger et al., 1984, Ch 14.2), guarantee the existence of such a locally optimal dual variable under mild assumptions of local convexity.

ϕ, seeks to approximate a feedback controller that solves the tracking problem (7).[2] The trajectory generator π^{plan} seeks to approximately solve the planning problem (6) and is parameterized as the solution to the optimization problem

$$\pi_{\theta,\psi}^{\mathrm{plan}}(\xi) = \underset{r}{\mathrm{minimize}} \quad \mathcal{C}(r) + p_{\psi}^{\pi^{\mathrm{track}}}(r + v_{\theta}(\xi); \xi) \tag{9}$$
$$\text{s.t.} \quad r \in \mathcal{X}.$$

This is to say that π^{plan} generates a reference trajectory from initial condition ξ by solving an optimization problem. The objective of this optimization problem, however, contains two learned components, v_{θ} and $p_{\psi}^{\pi^{\mathrm{track}}}$, specified by parameters θ and ψ, respectively. First, $v_{\theta} : \mathbb{R}^{d_x} \to \mathbb{R}^{T d_x}$ is a dual network that seeks to predict the locally optimal dual variable ν^* from initial condition ξ. Then, the tracking value function $p_{\psi}^{\pi^{\mathrm{track}}} : \mathbb{R}^{T d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$ takes in an initial state ξ and a reference trajectory r and learns to predict the quadratic tracking cost (7) that the policy π^{track} will incur on this reference trajectory. Summarizing, our layered policy consists of three learned components: the dual network v_{θ}, the low-layer tracking policy $\pi_{\phi}^{\mathrm{track}}$, and its associated value function $p_{\psi}^{\pi^{\mathrm{track}}}$. In what follows, we explain how we learn the tracking value function $p_{\psi}^{\pi^{\mathrm{track}}}$ and policy $\pi_{\phi}^{\mathrm{track}}$ jointly via the actor-critic method, and how to update the dual network v_{θ} in a way similar to dual ascent.

4.2 Learning the Tracking Controller via Actor-Critic Method

We use the actor-critic method to jointly learn the tracking value function $p_{\psi}^{\pi^{\mathrm{track}}}$ and policy $\pi_{\phi}^{\mathrm{track}}$. We are learning a deterministic policy and its value function, a setting which has been extensively explored and for which many off-the-shelf algorithms exist (Silver et al., 2014; Lillicrap et al., 2015; Fujimoto et al., 2018). In what follows, we specify the RL problem for learning the tracking controller and treat the actor-critic algorithm as a black-box solver for finding our desired parameters ϕ and ψ.

We define an augmented system with the state $x_t^{aug} = (x_t, \mathbf{r}_t)^{\top} \in \mathbb{R}^{(H+1)d_x}$, which concatenates x_t with a H-step reference trajectory $\mathbf{r}_t = (r_t^{\top} \ r_{t+1}^{\top} \ \cdots \ r_{t+H-1}^{\top})^{\top}$, where $H \in \mathbb{Z}^+$ specifies the tracking controller's horizon of look-ahead. The augmented state transitions are then given by

$$x_{t+1}^{aug} = \begin{bmatrix} f(x_t, u_t) \\ \mathcal{Z}\mathbf{r}_t \end{bmatrix}, t = 1, \dots T, \tag{10}$$

where $\mathcal{Z}$ is a block-upshift operator that shifts the reference trajectory forward by one timestep. The cost of the augmented system c^{aug} is chosen to match the tracking optimization problem (7), i.e., we set

$$c^{aug}(x_t^{aug}, u_t) = \frac{\rho}{2} \|x_{t+1} - r_{t+1}\|_2^2 + \mathcal{D}_t(u_t). \tag{11}$$

The initial condition x_0^{aug} is found by first sampling $\xi \sim D_{\xi}$, and then setting $\mathbf{r}_0$ to the first H steps of the reference generated by $\pi^{\mathrm{plan}}(\xi)$. We then run the actor-critic algorithm on this augmented system to jointly learn $p_{\psi}^{\pi^{\mathrm{track}}}$ and $\pi_{\phi}^{\mathrm{track}}$.

---

[2]The finite-horizon nature of (7) calls for a time-varying controller. Thus, the correct π^{track} and associated value function p^{π} need to be conditioned on the time step t. In our experiments, we show that approximating this with a time-invariant controller works well for the time horizons we consider.

4.3 Learning the Dual Network

We design our dual network update as an iterative procedure that mirrors the dual ascent update step (5), which moves the dual variable in the direction of the mismatch between reference r^+ and execution x^+. At each iteration, we sample a batch of initial conditions $\{\xi_i\}_{i=1}^B$, and for each ξ_i, we solve the planning problem (9) with current parameters ϕ and θ to obtain reference trajectories $r_i = \pi_{\theta^{(k)}, \psi}^{\text{plan}}(\xi)$. We then send the perturbed trajectories $\tilde{r}_i = r_i + v_\theta(\xi_i)$ to the tracking controller to obtain the executed trajectories

$$x_{i,t+1} = f(x_{i,t}, \pi_\psi^{\text{track}}(x_{i,t}, \tilde{\mathbf{r}}_{i,t})), \quad t = 0, ..., T.$$

Similar to the dual ascent step, we then perform a gradient ascent step in θ to move $v_{\theta^{(k)}}(\xi_i)$ in the direction of $r_i - x_i$:

$$
\begin{aligned}
\theta^+ &\leftarrow \theta + \eta \left(\nabla_\theta \sum_{i=1}^B \frac{1}{B} (r_i - x_i)^\top v_\theta(\xi_i) \right) \\
&= \theta + \eta \sum_{i=1}^B \frac{1}{B} (r_i - x_i)^\top J_{v,\theta}(\xi_i; \theta),
\end{aligned}
\tag{12}
$$

where $J_{v,\theta}$ denotes the Jacobian of v w.r.t. θ. Note that even though r_i and x_i implicitly depend on θ, similar to the dual ascent step (5), we do not differentiate through these two terms when computing this gradient. In the next section, we show that for the case of linear quadratic regulators, this update for the dual network parameter θ converges to the vicinity of the optimal parameter θ^* if the tracking problem is solved to sufficient accuracy.

4.4 Summary of the Algorithm

We summarize our algorithm in Algorithm 1. The outer loop of the algorithm (Line 1-9) corresponds to the dual update procedure described in Section 4.3. Within each iteration of the outer loop, we also run the actor-critic algorithm to update the tracking policy $\pi_{\phi^{(k)}}^{\text{track}}$ and its value function p_ψ^π (Line 5-8). Note that we do not wait for the tracking controller to converge before starting the dual update. In Section 6, we empirically validate that dual learning can start to make progress even when the tracking controller is still suboptimal. After the components are learned for the specified iterations, we directly apply the learned policy $\pi^{\text{plan}}, \pi^{\text{track}}$ for any new initial condition ξ.

5 Analysis for Linear Quadratic Regulator

In this section, we consider the unconstrained linear quadratic regulator (LQR) problem and show that our method learns to predict the optimal dual variable if we solve the tracking problem well enough. We focus on the dual update because the tracking problem (7) reduces to standard LQR, to which existing results (Bradtke et al., 1994; Tu and Recht, 2018) are readily applicable. In what follows, we define the problem we analyze, and first show that dual network updates of the form (12) converge to the optimal dual map if one perfectly solves the planning (6) and tracking problem (7). We then present a robustness result which shows that the algorithm will converge to the vicinity of the optimal dual variable if we solve the tracking problem with a small error.

We consider the instantiation of (2) with the dynamics

$$x_{t+1} = f(x_t, u_t) = Ax_t + Bu_t \tag{13}$$

Algorithm 1: Layered Actor-Critic

Result: Policy parameters ϕ, ψ, θ

1 **for** $k = 1, ..., K$ **do**

2 $\quad$ Sample a batch of initial conditions $\{\xi_i^{(k)}\}_{i=1}^B$;

3 $\quad$ Predict the optimal dual variables $\hat{\nu}_i^{(k)} = v_{\theta^{(k)}}(\xi_i^{(k)})$;

4 $\quad$ Solve (9) to find reference trajectories $\{r_i^{(k)}\}_{i=1}^B$;

5 $\quad$ Construct augmented state $x_{i,0}^{aug} = [\xi_i, r_i^{(k)}]^\top$;

6 $\quad$ **for** $t = 0, ..., T-1$ **do**

7 $\quad\quad$ Roll augmented dynamics forward with $\pi_{\phi^{(k)}}^{\text{track}}$ to get $\{x_{i,t+1}^{(k)}\}_{i=1}^B$;

8 $\quad\quad$ Update π_ϕ^{track} and p_ψ^π with observed transition using actor-critic algorithm;

9 $\quad$ Update the dual network parameter per (12);
---

and cost functions

$$\mathcal{C}(r) = \sum_{t=0}^T r_t^\top Q r_t =: r^\top \mathcal{Q} r,$$

$$\mathcal{D}(u) = \sum_{t=0}^{T-1} u_t^\top R u_t =: u^\top \mathcal{R} u, \tag{14}$$

where $Q \succeq 0, R \succ 0, \mathcal{Q} = I_T \otimes Q$ and $\mathcal{R} = I_{T-1} \otimes R$. States and control inputs are unconstrained, i.e., $\mathcal{X} = \mathbb{R}^{d_x}, \mathcal{U} = \mathbb{R}^{d_u}$. The initial condition ξ is sampled i.i.d. from the standard normal distribution $\mathcal{N}(0, I)$.

In this case, strong duality holds, and the optimal dual variable[3] ν^* is a linear function of the initial condition ξ. (See Lemma 2 in Appendix B.) We thus parameterize the dual network as a linear map

$$v_\theta(\xi) = \Theta \xi. \tag{15}$$

5.1 With Optimal Tracking

We first consider the following update rule, wherein we assume that the planning (6) and tracking problems (7) are solved optimally. At each iteration, we first sample a minibatch of initial conditions $\{\xi_i^{(k)}\}_{i=1}^B, \xi_i^{(k)} \overset{i.i.d.}{\sim} \mathcal{N}(0, I)$, and use the current $\Theta^{(k)}$ to predict the optimal dual variable

$$\hat{v}_i^{(k)} = v_{\theta^{(k)}}(\xi_i) = \Theta^{(k)} \xi_i.$$

We assume we perfectly solve the trajectory optimization problem

$$r_i^{(k)} = \underset{r}{\arg\min} \, r^\top \mathcal{Q} r + p^*(r + \hat{v}_i^{(k)}; \xi_i), \tag{16}$$

---

[3]If not further specified, when we refer to ν or the dual variable, we mean the dual variable associated with the constraint $r = x$ in problem (2)

where $p^*(\cdot)$ is the optimal value of the tracking problem

$$
\begin{aligned}
x_i^{(k)}, u_i^{(k)} = \underset{x,u}{\operatorname{argmin}} \ & u^\top \mathcal{R} u + \frac{\rho}{2} \| r_i^{(k)} + \hat{v}_i^{(k)} - x \|_2^2 \\
\text{s.t.} \quad & x_{t+1} = A x_t + B u_t, \\
& x_0 = \xi.
\end{aligned}
\tag{17}
$$

This is a standard LQR optimal control problem, and closed-form expressions for the optimizers and the value function are readily expressed in terms of the solution to a discrete algebraic Riccati equation.

After solving (17), we update the dual map Θ as

$$
\begin{aligned}
\Theta^{(k+1)} &= \Theta^{(k)} + \eta \nabla_\Theta \left(\sum_{i=1}^B \frac{1}{B} \left(r_i^{(k)} - x_i^{(k)} \right)^\top v_{\theta^{(k)}} (\xi_i) \right) \\
&= \Theta^{(k)} + \eta \sum_{i=1}^B \frac{1}{B} \left(r_i^{(k)} - x_i^{(k)} \right) \xi_i^\top
\end{aligned}
\tag{18}
$$

A feature of this update rule is that the difference between the reference r_i and the executed trajectory x_i can be written out in closed form as follows.

Lemma 1. *Given the update rules* (16), (17), *the difference between the updates $r_i^{(k)}$ and $x_i^{(k)}$ can be written as a linear map of the initial condition ξ as*

$$
r_i^{(k)} - x_i^{(k)} = H \Theta^{(k)} \xi_i + G \xi_i,
$$

where H and G are matrices of appropriate dimensions that depend on A, B, Q, R, and H is symmetric negative definite. See Lemma 3 in Appendix B for definitions of H and G.

We leverage Lemma 1, and that the matrix H is negative definite, to show that the updates (15)-(18) make progress in expectation.

Theorem 1. *Consider the cost functions* (14) *and dynamics* (13), *and fix an initial $\Theta^{(0)}$. Fix a step size $\eta = \frac{2}{\sigma_{max}(H) - \sigma_{min}(H)}$ and mini-batch size $B > \frac{2 d_x \sigma_{max}^2(H)}{\sigma_{min}^2(H)}$. The the iterates generated by the updates* (15)-(18) *satisfy*

$$
\mathbb{E} \left\| \Theta^{(k)} - \Theta^* \right\|_2 \leq \gamma^k \mathbb{E} \left\| \Theta^{(0)} - \Theta^* \right\|_2,
$$

where $\gamma \in (0,1)$ is a function of η, H, B, and d_x.

Proof. See Appendix B. $\qquad\square$

5.2 With Suboptimal Tracking

We consider the case where we only have approximate solutions to the updates (17) and (16). We leverage the structural properties of the LQR problem, and parameterize the optimal tracking controller as a linear map, and its value function as a quadratic function of the augmented state. We thus consider perturbations in the optimal value function p^* as

$$
\hat{p}(\tilde{r}, \xi) = p^*(\tilde{r}; \xi) + (\tilde{r} - F\xi) \Delta_P (\tilde{r} - F\xi),
\tag{19}
$$

and perturbations in the control action as

$$\hat{u}(\tilde{r}, \xi) = u^*(\tilde{r}, \xi) + \Delta_{u,r}\tilde{r} + \Delta_{u,\xi}\xi \tag{20}$$

where u^* denotes the u solution of (17). We note that the perturbations $\Delta_P, \Delta_{u,r}, \Delta_{u,x_0}$ represent the difference between learned and optimal policies, and have been shown to decay with the number of transitions used for training (Bradtke et al., 1994; Tu and Recht, 2018). Perturbation analysis on Theorem 1 shows that if the learned controller is close to optimal, the dual map Θ will converge to a small ball around Θ^*, where the radius of the ball depends on the error of the learned tracking controller. Due to space constraints, we present an informal version of this result here, and relegate a precise statement and proof to Appendix C.

Theorem 2. *(informal) Consider the dynamics* (13) *and cost* (14). *Consider the update rules* (15)-(18) *with the perturbations* (19) *and* (20). *Denote the size of the perturbations as* $\epsilon_P = \|\Delta_P\|, \epsilon_{u,r} = \|\Delta_{u,r}\|, \epsilon_{u,\xi} = \|\Delta_{u,\xi}\|$. *Given any* $\Theta^{(0)}$, *if the perturbations* $\epsilon_P, \epsilon_{u,r}$ *are small enough, there exist step size* η *and batch size* B *such that*

$$\mathbb{E}\left\|\Theta^{(k)} - \Theta^*\right\| \leq \gamma^k \mathbb{E}\left\|\Theta^{(0)} - \Theta^*\right\| + \frac{1 - \rho^k}{1 - \rho}e(\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi}),$$

where $0 < \gamma < 1$, $e(\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi})$ *is an error term depending polynomially on its arguments.*

6   Experiments

We now proceed to evaluate our algorithm numerically on LQR and unicycle systems. For all the experiments, we use the CleanRL (Huang et al., 2022) implementation of Twin-Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018) as our actor-critic algorithm. All code needed to reproduce the examples found in this section will be made available at the following repository: https://github.com/unstable-zeros/layered-ac.

6.1   Unconstrained LQR

Experiment Setup   We begin by validating our algorithm on unconstrained LQR problems and show that our algorithm achieves near-optimal performance and near-perfect reference tracking. We consider linear systems (13) with dimensions $d_x = d_u = 2, 4, 6, 8$ and horizon $T = 20$. For each system size, we randomly sample 15 pairs of dynamics matrices (A, B)[4] and normalize A so that the system is marginally stable ($\rho(A) = 1$). For all setups, we consider a quadratic cost (14) with $Q = I_{d_x}, R = 0.01I_{d_u}$. We have $\mathcal{X} = \mathbb{R}^{d_x}, \mathcal{U} = \mathbb{R}^{d_u}$, and the initial state $\xi \sim \mathcal{N}(0, I_{d_x})$. We leverage the linearity of the dynamics to parameterize the tracking controller π^{track} to be linear, and the value function p^π to be quadratic in the augmented state (10). Since p^π is quadratic, the optimization problem for the trajectory planner (9) is a QP, which we solve with CVXPY (Diamond and Boyd, 2016). We parameterize the dual network to be a linear map as in (15). We train the tracking policy and the dual network jointly for $100,000$ transitions ($5,000$ episodes) with dual batch size $B = 5$, before freezing the tracking policy and just updating the dual network for another $5,000$ transitions (250 episodes). We specify the detailed training parameters in Table 6 in the Appendix. During training, we periodically evaluate the learned policy by applying it

---

[4]Each entry is sampled i.i.d from the standard normal distribution.

d_x, d_u	Relative Cost ($\downarrow$)	Mean Tracking Deviation ($\downarrow$)
2	1.004	0.002
4	1.009	0.003
6	1.020	0.008
8	1.031	0.009

Table 1: LQR Results on Varying System Sizes.

d_x, d_u	Relative Cost ($\downarrow$)	Mean Tracking Deviation ($\downarrow$)
2	1.012 ($+0.7\%$)	0.046 ($+2,300\%$)
4	1.028 ($+1.8\%$)	0.045 ($+1,500\%$)
6	1.036 ($+1.5\%$)	0.061 ($+763\%$)
8	1.052 ($+2.0\%$)	0.062 ($+689\%$)

Table 2: LQR Results without Dual Learning. Numbers in parentheses denote the percentage difference from the approach with dual learning.

on 50 initial conditions. We then record the cost it achieved and the average tracking deviation $\frac{1}{T}\sum_{t=1}^{T}|r_t - x_t|$. To make the cost interpretable, we report the relative costs normalized by the optimal cost of solving (2) directly with the corresponding dynamics and cost. Thus, a relative cost of 1 is optimal. The results are summarized below.

Varying System Sizes In Table 1, we summarize the cost and mean tracking deviations evaluated at the end of training.[5] We first note that the learned policy achieves near-optimal cost and near-perfect tracking for all the system sizes considered. To get an intuitive idea of the mean tracking deviation, we refer the reader to Figure 2 for a sample trajectory that has a mean tracking deviation of 0.005. This shows that our parameterization and learning algorithm are able to find good policies with only black-box access to the underlying dynamics. We note that the performance degrades slightly as the size of the system grows. This is likely because learning the tracking controller becomes more difficult as the size of the state space increases. However, even for the largest system we considered ($N = 8$), the cost of the learned controller is still only 3% above optimal.

Visualization of Dual Learning We visualize the algorithm's progress for learning the dual map in Figure 1. Recall that our theory suggests that in the unconstrained LQR case, the dual map weight Θ will converge to the neighborhood of the optimal dual map Θ^*, where the radius of the neighborhood depends on the quality of the learned controller. This is indeed the case shown in Figure 1, where the norm of the difference $\Theta - \Theta^*$ first decays exponentially before reaching a plateau. We note that this plot also validates our choice to start learning the dual network before the tracking controller training has converged, as progress is made starting at the very beginning of the training.

Comparison to heuristic approach We now compare our approach to the heuristic approach of generating trajectories without using the learned dual variable (Srikanthan et al., 2023b; Zhang et al., 2024), summarized in equation (8). We use the same parameters to train a tracking controller

---

[5]The reported numbers are their respective medians taken over their 15 LQR instances.
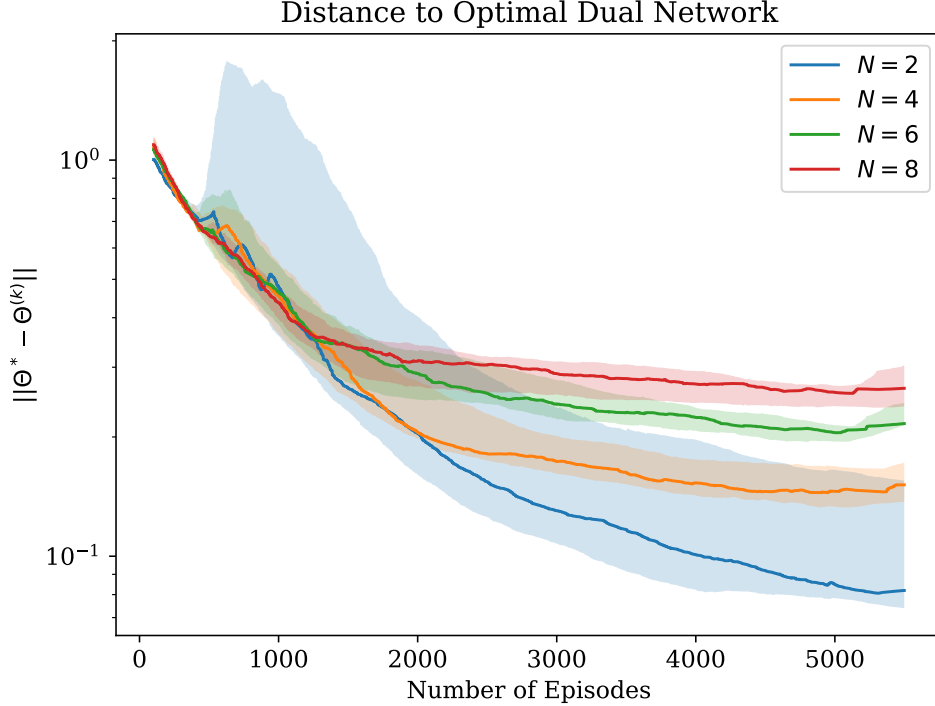
Figure 1: Training progress for the dual map parameter Θ. Here, the solid lines are the median over 15 LQR instances, and the shaded regions represent the 25th to 75th percentile.

ρ	0.5	1	2	4	8
Relative Cost ($\downarrow$)	2.04	1.24	1.11	1.10	1.19
Mean Deviation ($\downarrow$)	0.039	0.01	0.005	0.003	0.003

Table 3: LQR Results on Varying Hyperparameter ρ

and a value function, with the only difference being that π^{plan} solves (8) instead of (9). We show the results in Table 2. First, despite the heuristic approach managing to learn a good policy, it is outperformed by our approach both in terms of cost and tracking deviation across all the different system sizes, showing the value of learning to predict the dual variable. We note that the difference is especially pronounced for tracking deviation. Since the dual network learned to preemptively perturb the reference to minimize tracking error, it achieves near-perfect tracking and an order of magnitude lower tracking error. This suggests that learning the dual network is especially important in achieving good coordination between the trajectory planner and the tracking controller.

The role of ρ Finally, we note that the penalty parameter ρ is a hyperparameter which needs to be tuned when implementing Algorithm 1. Since ρ directly affects the objective of the tracking problem, it begs the question of whether the choice of ρ significantly affects the performance of our algorithm. We test this hypothesis on 15 randomly sampled *underactuated* systems where $d_x = 4$ and $d_u = 2$. We use the same set of hyperparameters as above except for ρ. We report the results in Table 3. From Table 3, we see that algorithm behavior is robust to the choice of ρ, so long as it is large enough; indeed, only the case of $\rho = 0.5$ leads to significant performance degradation.

Method	Relative Cost ($\downarrow$)	Mean Constraint Violation ($\downarrow$)
Ours	1.011	0.0002
No Dual (8)	1.014	0.002

Table 4: Constrained LQR Results

6.2 LQR with State Constraints

In the unconstrained case, the map from the initial condition ξ to the optimal dual variable ν^* is linear. In this section, we consider the case where inequality constraints are introduced and this map is no longer linear. We show that by parameterizing the dual map $v_\theta(\xi)$ as a neural network, we can learn well-performing policies that respect the constraints. Similar to the experiments above, we randomly sample 10 LQR systems where $d_x = d_u = 2$. Here we consider stable systems with $\rho(A) = 0.995$. The time horizon is fixed to $T = 20$ and cost matrices are $Q = I, R = 0.01I$. We add the constraint that

$$\mathcal{X} = \{x_{0:T} \mid x_{t,i} \geq -0.05, \quad 1 \leq t \leq 20, i = 1, 2\},$$

i.e., that we restrict all states except for the initial state to be above -0.05. Since adding the state constraint does not affect the tracking problem, we still parametrize the actor and critic as linear and quadratic, respectively. Since the optimal dual map is no longer linear, we parameterize the dual network as a neural network with a single hidden layer with ReLU activation. However, note that the optimization problem for trajectory planning (9) is still a QP as it does not depend on the form of the dual network. To account for the nonlinearity of the dual network, we increase the dual batch size to 40 trajectories, and train the policy and dual network for $150,000$ transitions, before freezing the tracking controller and training the dual network for another $600,000$ transitions ($30,000$ episodes). We specify the detailed training parameters in Table 7 in the Appendix. We report the relative cost and mean constraint violation[6] in Table 4 and show a representative sample trajectory in Figure 2.

As seen in Table 4 and the sample trajectories Figure 2, we are able to learn a policy that correctly plans reference trajectories within the constraint of the problem. The planned trajectory is well-adapted to the learned tracking controller so that the executed trajectory avoids constraint violation. This shows that empirically, our algorithm can also effectively learn to predict the dual variable even when the map from the initial condition to the dual variable is nonlinear. We again compare the results with solving for the reference without learning a dual network (8), which shows again that learning the dual network results in better coordination between the trajectory planner and the tracking controller. As a result, the approach with dual learning achieves better constraint satisfaction. We conclude this set of experiments by noting that in practice, one can tighten the constraints $x \in \mathcal{X}$ to further avoid constraint violation. How to leverage the learned dual network to inform constraint tightening could constitute an interesting direction of future work.

---

[6]We measure the constraint violation as $max(0.05 - x_t^{(i)}, 0), t = 17, ..., 20$. Reported values are the medians over the 10 systems.
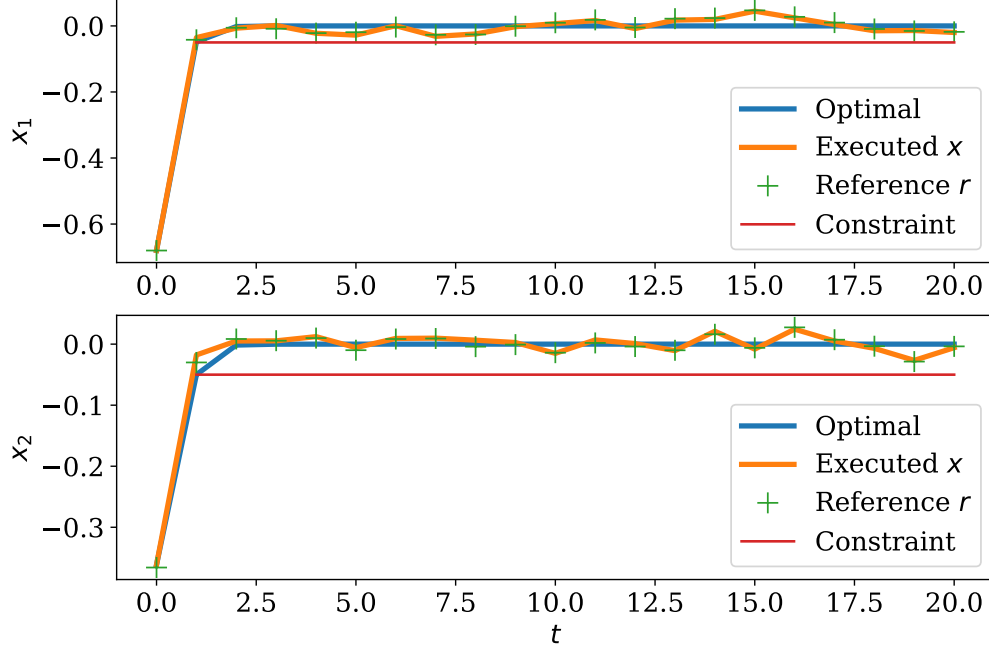
Figure 2: A Representative Sample Trajectory for Constrained LQR.

6.3 Unicycle

Finally, we apply our algorithm to controlling a nonlinear unicycle system with state and control input

$$x_t = \begin{bmatrix} p_{x,t} \\ p_{y,t} \\ \theta_t \\ v_t \end{bmatrix} \in \mathbb{R}^4, \quad u_t = \begin{bmatrix} a_t \\ \omega_t \end{bmatrix} \in \mathbb{R}^2,$$

where p_x, p_y are the x and y positions of the unicycle, respectively, θ is the heading angle of the vehicle, and v is the vehicle velocity. The two control inputs are the acceleration a and the angular velocity (steering) ω. We consider the discrete-time nonlinear dynamics given by

$$x_{t+1} = f(x_t, u_t) = \begin{bmatrix} p_{x,t} \\ p_{y,t} \\ \theta_t \\ v_t \end{bmatrix} + 0.1 \begin{bmatrix} \cos(\theta_t)v_t \\ \sin(\theta_t)v_t \\ \omega_t \\ a_t \end{bmatrix}.$$

We consider the problem of steering the vehicle to the origin, which is specified by a quadratic objective (14) with $Q = \mathrm{diag}([1,1,0,0])$, and $R = 0.01I_2$. The initial condition ξ is sampled uniformly on the unit circle. We take $T = 20$. We learn a trajectory planner π^{plan} to generate reference trajectories for the positions (p_x, p_y).

The nonlinearity of the dynamics presents several challenges. First, we can no longer assume the form of the optimal tracking controller and its value function and have to parameterize both as neural networks. As a result of this non-convex parameterization of p^π, the reference generation problem (9) now becomes nonconvex. We thus use gradient descent to find reference trajectories

Method	Relative Cost (↓)	Mean Tracking Deviation (↓)
iLQR	1	-
Ours	1.04	0.02
No Dual	1.04	0.05

Table 5: Unicycle Results

that are locally optimal solutions to the trajectory planning problem. Secondly, the nonlinear nature of the dynamics makes the learning of a tracking controller more difficult. To address this, we warmstart the tracking controller by training on simple line trajectories before running Algorithm 1 in full where we generate the reference with π^{plan} by solving (9).[7] This curriculum approach overcomes the difficulty that (9) tends to generate bad trajectories when p^π is randomly initialized. We train the tracking controller on simple references for $100,000$ transitions ($5,000$ episodes) as a warmstart, and then run Algorithm 1 for $500,000$ transitions ($25,000$ episodes). We run the experiment both with and without training the dual network and report our results in Table 5. To make the result interpretable, we normalize the cost against iLQR as a baseline.[8]

First, we see that our learned policy achieves comparable performance as iLQR. However, unlike iLQR, our policy is trained without explicit knowledge of the dynamics of the system. We note that the costs achieved by the policy learned with and without a dual network are similar. This could be a consequence of the fact that we cannot solve the trajectory generation problem (9) exactly. However, learning with a dual network again leads to significantly better tracking performance, highlighting the importance of dual networks in coordinating the planning and tracking layers.

7 Discussion

In this paper, we proposed a principled way for parameterizing and learning a layered control policy consisting of a trajectory planner and a tracking controller. We derived our parameterization from an underlying optimal control problem and showed that a dual network emerges naturally as a way to coordinate the two components. We showed theoretically that our algorithm can learn to predict the optimal dual variable for unconstrained LQR problems and validated this theory via simulation experiments. Further simulation experiments demonstrated the potential of applying this method to more nonlinear problems as well. Future work will explore learning the trajectory planner parameterized as a neural network and using the dual network to inform constraint tightening.

References

Nikolai Matni, Aaron D Ames, and John C Doyle. Towards a theory of control architecture: A quantitative framework for layered multi-rate control. *arXiv preprint arXiv:2401.15185*, 2024.

Mung Chiang, Steven H Low, A Robert Calderbank, and John C Doyle. Layering as optimization

---

[7]In this case, the simple reference trajectories are equally spaced line segments that connect initial states to the origin.

[8]For each initial condition, we run iLQR with two random dynamically feasible initial trajectories. We take the lesser cost as iLQR's cost.

decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95 (1):255–312, 2007.

Ugo Rosolia, Andrew Singletary, and Aaron D Ames. Unified multirate control: From low-level actuation to high-level planning. *IEEE Transactions on Automatic Control*, 67(12):6627–6640, 2022.

Noel Csomay-Shanklin, Andrew J Taylor, Ugo Rosolia, and Aaron D Ames. Multi-rate planning and control of uncertain nonlinear systems: Model predictive control and control lyapunov functions. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3732–3739. IEEE, 2022.

Nikolai Matni and John C Doyle. A theory of dynamics, control and optimization in layered architectures. In *2016 American Control Conference (ACC)*, pages 2886–2893. IEEE, 2016.

Tariq Samad, Paul McLaughlin, and Joseph Lu. System architecture for process automation: Review and trends. *Journal of Process Control*, 17(3):191–201, 2007.

Tariq Samad and Anuradha M Annaswamy. Controls for smart grids: Architectures and applications. *Proceedings of the IEEE*, 105(11):2244–2261, 2017.

Jehn-Ruey Jiang. An improved cyber-physical systems architecture for industry 4.0 smart factories. *Advances in Mechanical Engineering*, 10(6):1687814018784192, 2018.

Anusha Srikanthan, Vijay Kumar, and Nikolai Matni. Augmented lagrangian methods as layered control architectures. *arXiv preprint arXiv:2311.06404*, 2023a.

Anusha Srikanthan, Fengjun Yang, Igor Spasojevic, Dinesh Thakur, Vijay Kumar, and Nikolai Matni. A data-driven approach to synthesizing dynamics-aware trajectories for underactuated robotic systems. *arXiv preprint arXiv:2307.13782*, 2023b.

Hanli Zhang, Anusha Srikanthan, Spencer Folk, Vijay Kumar, and Nikolai Matni. Why change your controller when you can change your planner: Drag-aware trajectory generation for quadrotor systems. *arXiv preprint arXiv:2401.04960*, 2024.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018a.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018b.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Jiarui Wang and Mahyar Fazlyab. Actor-critic physics-informed neural lyapunov control. *arXiv preprint arXiv:2403.08448*, 2024.

Gianluigi Grandesso, Elisa Alboni, Gastone P Rosati Papini, Patrick M Wensing, and Andrea Del Prete. Cacto: Continuous actor-critic with trajectory optimization—towards global optimality. *IEEE Robotics and Automation Letters*, 2023.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

Steven J Bradtke, B Erik Ydstie, and Andrew G Barto. Adaptive linear quadratic control using policy iteration. In *Proceedings of 1994 American Control Conference-ACC'94*, volume 3, pages 3475–3479. IEEE, 1994.

Stephen Tu and Benjamin Recht. Least-squares temporal difference learning for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 5005–5014. PMLR, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

A   Experiment Setup

A.1   Hyperparameters for the Unconstrained LQR Experiments

See Table 6.

Parameter	Value
TD3 Policy Noise	5e-4
TD3 Noise Clip	1e-3
TD3 Exploration Noise	0
actor learning rate	3e-3
actor batch size	256
critic learning rate	3e-3
critic batch size	256
dual learning rate	0.1
dual batch size	5

Table 6: Hyperparameters for the Unconstrained LQR Experiments

A.2   Hyperparameters for the Constrained LQR Experiments

See Table 7. The dual network is chosen to be a MLP with one hidden layer of 128 neurons. The

Parameter	Value
TD3 Policy Noise	5e-4
TD3 Noise Clip	1e-3
TD3 Exploration Noise	0
actor learning rate	3e-3
actor batch size	256
critic learning rate	3e-3
critic batch size	256
dual learning rate	3e-4
dual batch size	40

Table 7: Hyperparameters for the Constrained LQR Experiments

activation is chosen to be ReLU.

A.3   Hyperparameters for the Unicycle Experiments

See Table 8. The dual network is chosen to be a MLP with one hidden layer of 128 neurons. The actor and critic are both MLPs with a single hidden layer of 256 neurons. All activation functions are ReLU.

Parameter	Value
TD3 Policy Noise	1e-3
TD3 Noise Clip	1e-2
TD3 Exploration Noise	6e-2
actor learning rate	1e-3
actor batch size	256
critic learning rate	1e-3
critic batch size	256
dual learning rate	5e-3
dual batch size	60

Table 8: Hyperparameters for the Unicycle Experiments

B Proofs for Theorem 1

The LQR problem considered in Section 5 with costs (14) and dynamics (13) admits closed-form solutions to the r-update (16) and x-update (17). In this section, we begin by showing that for this problem, the dual variable can indeed be written as a linear map of the initial condition ξ. We then derive the closed-form solutions to the updates (16) and (17). Finally, we use a contraction argument to show our desired result in Theorem 1. In the process, we make clear the conditions on step size η and batch size B to guarantee the contraction.

For easing notation, for the rest of this section, we again define $\tilde{r} := r + \nu$. We also define the matrices

$$
E := \begin{bmatrix}
0 & 0 & \cdots & 0 \\
B & 0 & \cdots & 0 \\
AB & B & \cdots & 0 \\
\vdots & \vdots & & \vdots \\
A^{T-1}B & A^{T-2}B & \cdots & B
\end{bmatrix}, \qquad
F := \begin{bmatrix}
I \\
A \\
A^2 \\
\vdots \\
A^T
\end{bmatrix}.
$$

Lemma 2. *For the problem considered in section 2, given the initial condition ξ, the optimal dual variable can be expressed as a unique linear map from ξ as*

$$
\nu^* = -\frac{2}{\rho}Q\left(-E(E^\top QE + R)^{-1}E^\top QF + F\right)\xi.
$$

Proof. From the KKT condition for the optimization problem (2), we have that

$$
\nabla_r \left(r^\top Qr + \frac{\rho}{2}\|r + \nu^* - x^*\|_2^2\right)\Big|_{r=r^*} = 0.
$$

Solving for r^*, we get that

$$
2(Q + \frac{\rho}{2}I)r^* = -\rho\nu^* + \rho x^*.
$$

Also from the KKT condition, we have that $r^* = x^*$. Subbing this into the above expression and rearranging terms, we get that

$$
\nu^* = -\frac{2}{\rho}Qr^* = -\frac{2}{\rho}Qx^*.
$$

Finally, from the equivalence of the original problem (1) and the redundant problem (2), we see that x^* can be expressed in closed form as

$$x^* = \left(-E(E^\top QE + R)^{-1}E^\top QF + F \right)\xi.$$

$\square$

We now derive the closed-form update rules and show the following.

Lemma 3. *In the LQR setting, we have that the difference between the updates r^+ and x^+ can be written as a linear map of the initial condition ξ as*

$$r^+ - x^+ = H\Theta^{(k)}\xi + G\xi,$$

where

$$H := -\frac{2}{\rho}P(Q + P)^{-1}P - \frac{\rho}{2}E\bar{R}^{-1}E^\top,$$
$$G := HF + F.$$

H is symmetric negative definite.

Proof. We start by deriving the closed-form expressions for solving both the updates (16) and (17). We begin by writing out the update rule more explicitly. First, we note that we can write all x, u satisfying the dynamics constraint as

$$x = Eu + F\xi, \quad u_t \in \mathbb{R}^{d_u}.$$

Define

$$\begin{aligned}
p(\tilde{r}, \xi, u) &= u^\top Ru + \frac{\rho}{2}\|\tilde{r} - x\|_2^2 \\
&= u^\top Ru + \frac{\rho}{2}\|\tilde{r} - (Eu + F\xi))\|_2^2.
\end{aligned}$$

We can solve for the optimal control action in closed-form as

$$\begin{aligned}
u^* &= \operatorname{argmin}_u p(\tilde{r}, \xi, u) \\
&= -\frac{\rho}{2}\bar{R}^{-1}E^\top(F\xi - \tilde{r}),
\end{aligned}$$

where we defined

$$\bar{R} = R + \frac{\rho}{2}E^\top E.$$

Subbing this back, we get that

$$\begin{aligned}
p^*(\tilde{r}, \xi) &= p(\tilde{r}, \xi, u^*) \\
&= (F\xi - \tilde{r})^\top(\frac{\rho}{2}I - \frac{\rho^2}{4}E\bar{R}^{-1}E^\top)(F\xi - \tilde{r}) \\
&=: (F\xi - \tilde{r})^\top P(F\xi - \tilde{r}),
\end{aligned}$$

where we defined

$$P := \frac{\rho}{2}I - \frac{\rho^2}{4}E\bar{R}^{-1}E^\top.$$

Note that we arrived at p^* from a partial minimization on u, which preserves the convexity of the problem on r. Thus, we have that
$$P \succ 0.$$

With the knowledge of p^*, we can now solve for r in closed-form. Since $Q + P \succ 0$, we have that
$$\begin{aligned}
r^+ &= \mathrm{argmin}_r r^\top Q r + p^*(r + \nu, \xi) \\
&= \mathrm{argmin}_r r^\top Q r + (F\xi - r - \nu)^\top P (F\xi - r - \nu) \\
&= (Q + P)^{-1} P (F\xi - \nu).
\end{aligned}$$

Thus, overall, we update rules are given as
$$\begin{aligned}
r^+ &= (Q + P)^{-1} P (F\xi - \nu), \\
u^+ &= -\frac{\rho}{2} \bar{R}^{-1} E^\top (F\xi - r^{(k+1)} - \hat{\nu}^{(k)}), \\
x^+ &= E u^{(k+1)} + F\xi.
\end{aligned}$$

From the closed-form update rules specified above, we have that
$$\begin{aligned}
&r^+ - x^+ \\
=& r^+ - (E(-\frac{\rho}{2}\bar{R}^{-1} E^\top (F\xi - r^{(k+1)} - \hat{\nu}^{(k)})) + F\xi) \\
=& (I - \frac{\rho}{2} E\bar{R}^{-1} E^\top) r^+ + \frac{\rho}{2} E\bar{R}^{-1} E^\top (F\xi - \nu) - F\xi \\
=& -(\frac{2}{\rho} P(Q+P)^{-1} P + \frac{\rho}{2} E\bar{R}^{-1} E^\top)\nu - (HF + F)\xi
\end{aligned}$$

Denote
$$\begin{aligned}
H &:= -\frac{2}{\rho} P(Q+P)^{-1} P - \frac{\rho}{2} E\bar{R}^{-1} E^\top \\
G &:= HF + F,
\end{aligned}$$

we get the expression that we desire. From the fact that $P \succ 0$, $Q \succeq 0$, and $\bar{R} \succ 0$, it follows that $H \prec 0$. $\qquad\qquad\square$

Now, recall that for any $\xi \in \mathbb{R}^{d_x}$, the optimal dual map Θ^* satisfies that $\nu^* = \Theta^*\xi$, where ν^* is the optimal dual variable for the given ξ. From the KKT condition of (2), we know that ν^* induces a fixed point to the update rules (16) and (17). Thus,
$$r^+ - x^+ = r^* - x^* = H\Theta^*\xi + G\xi = 0.$$

Since this holds for all $\xi \in \mathbb{R}^{d_x}$, we have that
$$H\Theta^* + G = 0.$$

Before starting with the main proof, we present the following lemma.

Lemma 4. *For a set of i.i.d normal vectors $\xi_i \sim \mathcal{N}(0, I)$, we have that*
$$\mathbb{E}\left\| \sum_{i=1}^B \frac{1}{B} \xi_i \xi_i^\top - I \right\|_F \le \sqrt{\frac{2d_x}{B}}.$$

Proof. We begin by rewriting the above expression with a change of variables, where we define

$$X := \sum_{i=1}^{B} \frac{1}{B} \xi_i \xi_i^\top.$$

Since X is a sum of outer products of independently distributed normal random vectors, X follows the Wishart distribution. Specifically, we have that

$$X \sim W_{d_x}(I, B).$$

The above expression can then be bounded as

$$
\begin{aligned}
\mathbb{E} \left\| \sum_{i=1}^{B} \frac{1}{B} \xi_i \xi_i^\top - I \right\|_F &= \frac{1}{B} \mathbb{E} \left\| X - BI \right\|_F \\
&\leq \frac{1}{B} \sqrt{\mathbb{E} \left\| X - BI \right\|_F^2} \\
&= \frac{1}{B} \sqrt{\operatorname{Tr} \mathbb{E}(X - BI)(X - BI)^\top} \\
&= \frac{1}{B} \sqrt{\operatorname{Tr} \operatorname{Var}(X)} \\
&= \frac{1}{B} \sqrt{2 d_x B} \\
&= \sqrt{\frac{2 d_x}{B}},
\end{aligned}
$$

where we first used Jensen's inequality and then, in the following equalities, used the properties of Wishart random variables. □

We can now start analyzing the progress of the dual update. First, combing Lemma 3 with the dual update rule (18), we have that

$$
\begin{aligned}
\Theta^{(k+1)} &= \Theta^{(k)} + \eta \sum_{i=0}^{B} \frac{1}{B} (H\Theta^{(k)} \xi_i + G\xi_i) \xi_i^\top \\
&= \Theta^{(k)} + \eta (H\Theta^{(k)} + G) \sum_{i=0}^{B} \frac{1}{B} \xi_i \xi_i^\top.
\end{aligned}
\tag{21}
$$

Thus, for the expected norm of interest, we have

$$\mathbb{E}\left\|\Theta^{(k+1)} - \Theta^*\right\|$$

$$=\mathbb{E}\left\|\Theta^{(k)} + \eta(H\Theta^{(k)} + G)\sum_{i=0}^{B}\frac{1}{B}\xi_i\xi_i^\top - \Theta^*\right\|$$

$$\overset{(a)}{=}\mathbb{E}\left\|\Theta^{(k)} - \Theta^* + \eta(H\Theta^{(k)} + G)\sum_{i=0}^{B}\frac{1}{B}\xi_i\xi_i^\top - \eta(H\Theta^* + G)\sum_{i=0}^{B}\frac{1}{B}\xi_i\xi_i^\top\right\|$$

$$=\mathbb{E}\left\|\Theta^{(k)} - \Theta^* + \eta H(\Theta^{(k)} - \Theta^*)\sum_{i=0}^{B}\frac{1}{B}\xi_i\xi_i^\top\right\|$$

$$=\mathbb{E}\left\|(I + \eta H)(\Theta^{(k)} - \Theta^*) + \eta H(\Theta^{(k)} - \Theta^*)(\sum_{i=0}^{B}\frac{1}{B}\xi_i\xi_i^\top - I)\right\|$$

$$\leq\left(\|I + \eta H\|_2 + \eta\|H\|_2\,\mathbb{E}\left\|\sum_{i=0}^{B}\frac{1}{B}\xi_i\xi_i^\top - I\right\|\right)\left\|\Theta^{(k)} - \Theta^*\right\|$$

$$\leq\left(\|I + \eta H\|_2 + \eta\|H\|_2\sqrt{\frac{2d_x}{B}}\right)\left\|\Theta^{(k)} - \Theta^*\right\|,$$

where in step (a) we used the above fact that $H\Theta^* + G = 0$, and in the final step, we used Lemma 4. We have the desired contraction if

$$\gamma(\eta, B) := \|I + \eta H\|_2 + \eta\|H\|_2\sqrt{\frac{2d_x}{B}} < 1.$$

Note that choosing

$$\eta^* = \frac{2}{\sigma_{\max}(H) + \sigma_{\min}(H)}$$

minimizes the norm $\|I + \eta H\|$. Solving for B with this choice of η, we have that B needs to satisfy that

$$B > \frac{2d_x\sigma_{\max}^2(H)}{\sigma_{\min}(H)^2}.$$

Following these appropriate choices of η and B, we have

$$\mathbb{E}\left\|\Theta^{(k)} - \Theta^*\right\| \leq \gamma^k\mathbb{E}\left\|\Theta^{(0)} - \Theta^*\right\|.$$

C   Proof of Theorem 2

We start by showing that when H and G (See Lemma 3 above) are perturbed by small additive perturbations, the algorithm can still converge to the vicinity of the optimal Θ^* if the perturbations are small enough.

Lemma 5. *Consider the perturbations in H and G as*

$$H' = H + \Delta_H, \quad G' = G + \Delta_G.$$

If the perturbation in H satisfies that

$$\|\Delta_H\| < \frac{1}{1 + \sqrt{\frac{2d_x}{B_0}}} \frac{\sigma_{max}(H) - \sigma_{min}(H)}{\sigma_{max}(H) + \sigma_{min}(H)},$$

for any $B_0 \geq 1$, then we can pick step size η and batch size B such that the update (21) converges to the vicinity of the optimal dual variable, i.e., that

$$\mathbb{E}\left\|\Theta^{(k)} - \Theta^*\right\| \leq \gamma^k \mathbb{E}\left\|\Theta^{(0)} - \Theta^*\right\| + \frac{1 - \gamma^k}{1 - \gamma} e(\|\Delta_H\|, \|\Delta_G\|),$$

where $0 < \gamma < 1$, and

$$e(\|\Delta_H\|, \|\Delta_G\|) := (1 + \sqrt{\frac{2d_x}{B}})(\|\Delta_G\| + \|\Delta_H\|\|\Theta^*\|) \tag{22}$$

Proof. We follow similar steps as when we showed the similar result for the unperturbed case.

$$\mathbb{E}\left\|\Theta^{(k+1)} - \Theta^*\right\|$$

$$= \mathbb{E}\left\|\Theta^{(k)} - \Theta^* + \eta((H + \Delta_H)\Theta^{(k)} + (G + \Delta_G)) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top\right\|$$

$$= \mathbb{E}\left\|\Theta^{(k)} - \Theta^* + \eta(H\Theta^{(k)} + G + (\Delta_H\Theta^{(k)} + \Delta_G)) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top - \eta(H\Theta^* + G) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top\right\|$$

$$\leq \mathbb{E}\left\|\Theta^{(k)} - \Theta^* + \eta H(\Theta^{(k)} - \Theta^*) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top\right\| + \mathbb{E}\left\|(\Delta_H\Theta^{(k)} + \Delta_G)) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top\right\|$$

$$\leq \left(\|I + \eta H\|_2 + \eta\|H\|_2\sqrt{\frac{2d_x}{B}}\right)\left\|\Theta^{(k)} - \Theta^*\right\|_F + \mathbb{E}\left\|(\Delta_H\Theta^{(k)} + \Delta_G)) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top\right\|,$$

where the last step followed the same steps in the proof of Theorem 1. We now proceed to bound the second term.

$$\mathbb{E}\left\|(\Delta_H\Theta^{(k)} + \Delta_G) \sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i^\top\right\|$$

$$\leq \left\|(\Delta_H(\Theta^{(k)} - \Theta^*) + (\Delta_G + \Delta_H\Theta^*))\right\| \mathbb{E}\left\|\sum_{i=0}^{B} \frac{1}{B}\xi_i\xi_i\right\|$$

$$\leq (1 + \sqrt{\frac{2d_x}{B}})\left(\|\Delta_H\|\left\|\Theta^{(k)} - \Theta^*\right\| + \|\Delta_G\| + \|\Delta_H\|\|\Theta^*\|\right)$$

Combining this with the unperturbed result, we get that

$$\mathbb{E}\left\|\Theta^{(k+1)} - \Theta^*\right\|_F$$

$$\leq \left(\|I + \eta H\|_2 + \eta\|H\|_2\sqrt{\frac{2d_x}{B}} + (1 + \sqrt{\frac{2d_x}{B}})\|\Delta_H\|\right)\left\|\Theta^{(0)} - \Theta^*\right\| + e(\|\Delta_H\|, \|\Delta_G\|),$$

where

$$e(\|\Delta_H\|, \|\Delta_G\|) := (1 + \sqrt{\frac{2d_x}{B}})(\|\Delta_G\| + \|\Delta_H\| \|\Theta^*\|)$$

For the iterations to be contractive, we would need that

$$\left(\|I + \eta H\|_2 + \eta \|H\|_2 \sqrt{\frac{2d_x}{B}} + (1 + \sqrt{\frac{2d_x}{B}}) \|\Delta_H\|\right) < 1.$$

Again, note that choosing

$$\eta^* = -\frac{2}{\lambda_{\max}(H) + \lambda_{\min}(H)}$$

minimizes the norm $\|I + \eta H\|$. Thus, for the inequality to hold, $\|\Delta_H\|$ needs to satisfy

$$\left(1 + \sqrt{\frac{2d_x}{B_0}}\right) \|\Delta_H\| < \|1 + \eta^* H\| = \frac{\lambda_{\max}(H) - \lambda_{\min}(H)}{\lambda_{\max}(H) + \lambda_{\min}(H)},$$

for some $B_0 \geq 1$ or equivalently

$$\|\Delta_H\| < \frac{1}{1 + \sqrt{\frac{2d_x}{B_0}}} \frac{\lambda_{\max}(H) - \lambda_{\min}(H)}{\lambda_{\max}(H) + \lambda_{\min}(H)}.$$

We can then pick

$$B > \max\left(\frac{2d_x \eta \|H\|_2^2}{(1 - \|I + \eta H\|)^2 - (1 + \sqrt{\frac{2d_x}{B_0}}) \|\Delta_H\|}, B_0\right),$$

so that

$$\gamma := \|I + \eta H\|_2 + \eta \|H\|_2 \sqrt{\frac{2d_x}{B}} + (1 + \sqrt{\frac{2d_x}{B}}) \|\Delta_H\| < 1.$$

The result then follows from telescoping the sum. $\qquad\square$

We now consider the perturbations we described in Section 5.2 and bound the terms $\|\Delta_H\|$ and $\|\Delta_G\|$ in terms of the perturbations $\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi}$.

Lemma 6. *Consider the perturbations specified in* (19) *and* (20). *Denote the norms of the perturbations as*

$$\epsilon_P = \|\Delta_P\|, \epsilon_{u,r} = \|\Delta_{u,r}\|, \epsilon_{u,\xi} = \|\Delta_{u,\xi}\|.$$

If $\epsilon_P < \frac{\lambda_{\min}(Q+P)}{2}$, *we have that* $\|\Delta_H\|$ *and* $\|\Delta_G\|$ *as*

$$\|\Delta_H\| < e_H(\epsilon_P, \epsilon_{u,r}),$$
$$:= \frac{2\epsilon \|P\|}{\lambda_{\min}(Q + P)} + \frac{\epsilon^2}{\lambda_{\min}(Q + P)} + \frac{2\epsilon(\|P\| + \epsilon)^2}{\lambda_{\min}(Q + P)^2}, \tag{23}$$

where $\epsilon = max(\epsilon_P, \frac{\rho}{2} \|E\| \epsilon_{u,r})$.

$$\|\Delta_G\| < e_G(\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi})$$
$$:= \|F\| e_H(\epsilon_P, \epsilon_{u,r}) + \epsilon_{u,\xi}. \tag{24}$$

Proof. We first note that the perturbed update rule gives the updates

$$r^+ = (Q + P + \Delta_P)^{-1}(P + \Delta_P)(r + \nu - F\xi),$$
$$\hat{u}(\tilde{r}, \xi) = u^*(\tilde{r}, \xi) + \Delta_{u,r}\tilde{r} + \Delta_{u,\xi}\xi.$$

The difference between r^+ and x^+ can be summarized as

$$
\begin{aligned}
&r^+ - x^+ \\
=&(I - \frac{\rho}{2}E\bar{R}^{-1}E^\top - E\Delta_{u,r})r^+ + \frac{\rho}{2}E\bar{R}^{-1}E^\top(F\xi - \nu) - (F + \Delta_{u,\xi})\xi \\
=&\frac{2}{\rho}(P - \frac{\rho}{2}E\Delta_{u,r})r^+ + \frac{\rho}{2}E\bar{R}^{-1}E^\top(F\xi - \nu) - (F + \Delta_{u,\xi})\xi \\
=&(H + \Delta_H)\nu - (G + \Delta_G)\xi
\end{aligned}
$$

where

$$\Delta_H := \frac{2}{\rho}(P - \frac{\rho}{2}E\Delta_{u,r})(Q + P + \Delta_P)^{-1}(P + \Delta_P) - \frac{2}{\rho}P(Q + P)^{-1}P$$
$$\Delta_G := \Delta_H F + E\Delta_{u,\xi}.$$

We now proceed to bound $\|\Delta_H\|$. Denote the reduced SVD of Δ_P as

$$\Delta_P = UCV^\top.$$

For the sake of simplicity, we denote $D := P + Q$. By Woodbury matrix identity, we have that

$$(A + UCV^\top)^{-1} = A^{-1} - A^{-1}U(C^{-1} + V^\top A^{-1}U)^{-1}V^\top A^{-1}.$$

Thus, we have that

$$
\begin{aligned}
&(P - \frac{\rho}{2}E\Delta_{u,r})(Q + P + \Delta_P)^{-1}(P + \Delta_P) \\
=&PD^{-1}P + \frac{\rho}{2}E\Delta_{u,r}D^{-1}P + PD^{-1}\Delta_P - \frac{\rho}{2}E\Delta_{u,r}D^{-1}\Delta_P + \\
&(P - \frac{\rho}{2}E\Delta_{u,r})D^{-1}U(C^{-1} + V^\top D^{-1}U)^{-1}V^\top D^{-1}(P + \Delta_P).
\end{aligned}
$$

The first term corresponds to the unperturbed H. We thus proceed to bound all the other terms left. Define

$$\epsilon = \max(\epsilon_P, \frac{\rho}{2}\|E\|\epsilon_{u,r}),$$

we have that

$$\left\|\frac{\rho}{2}E\Delta_{u,r}D^{-1}P\right\| \le \frac{\rho\|E\Delta_{u,r}\|\|P\|}{2\lambda_{\min}(D)} \le \frac{\epsilon\|P\|}{\lambda_{\min}(D)},$$
$$\left\|PD^{-1}\Delta_P\right\| \le \frac{\|\Delta_P\|\|P\|}{\lambda_{\min}(D)} \le \frac{\epsilon\|P\|}{\lambda_{\min}(D)},$$
$$\left\|\frac{\rho}{2}E\Delta_{u,r}D^{-1}\Delta_P\right\| \le \frac{\rho\|E\Delta_{u,r}\|\|\Delta_P\|}{2\lambda_{\min}(D)} \le \frac{\epsilon^2}{\lambda_{\min}(D)}.$$

To bound the last term, we use the fact that

$$\left\| (P - \frac{\rho}{2} E \Delta_{u,r}) D^{-1} U (C^{-1} + V^\top D^{-1} U)^{-1} V^\top D^{-1} (P + \Delta_P) \right\|$$

$$\leq (\|P\| + \epsilon)^2 \cdot \frac{1}{\lambda_{\min}(D)} \cdot \frac{1}{\sigma_{\min}(C^{-1} + V^\top D^{-1} U)}$$

We invoke the reverse triangle inequality to get that

$$\frac{1}{\sigma_{\min}(C^{-1} + V^\top D^{-1} U)} \leq \frac{1}{\sigma_{\min}(C^{-1}) - \|V^\top D^{-1} U\|}$$

$$= \frac{1}{\frac{1}{\|C\|} - \frac{1}{\lambda_{\min}(D)}}$$

$$\leq \epsilon_P \cdot \frac{\lambda_{\min}(D)}{\lambda_{\min}(D) - \epsilon_P}$$

From the assumption that $\epsilon_P < \frac{\lambda_{\min}(D)}{2}$, we have that

$$\frac{\lambda_{\min}(D)}{\lambda_{\min}(D) - \epsilon_P} \leq 2.$$

Thus, we have that

$$\left\| (P - \frac{\rho}{2} E \Delta_{u,r}) D^{-1} U (C^{-1} + V^\top D^{-1} U)^{-1} V^\top D^{-1} (P + \Delta_P) \right\|$$

$$\leq \frac{2\epsilon(\|P\| + \epsilon)^2}{\lambda_{\min}(D)^2}.$$

Thus, we overall have that

$$\|\Delta_H\| \leq \frac{2\epsilon \|P\|}{\lambda_{\min}(D)} + \frac{\epsilon^2}{\lambda_{\min}(D)} + \frac{2\epsilon(\|P\| + \epsilon)^2}{\lambda_{\min}(D)^2}$$

and that

$$\|\Delta_G\| \leq \|\Delta_H\| \|F\| + \epsilon_{u,\xi}.$$

$\square$

Combining the two Lemmas above, we can now state Theorem 2 formally.

Theorem 3. *(Formal statement of Theorem 2) Consider the cost functions* (14) *and dynamics* (13). *Consider the update rules* (15)-(18) *with the perturbations* (19) *and* (20). *Denote the size of the perturbations as*

$$\epsilon_P = \|\Delta_P\|, \epsilon_{u,r} = \|\Delta_{u,r}\|, \epsilon_{u,\xi} = \|\Delta_{u,\xi}\|.$$

Define e_H as in (23) *and e_G as in* (24). *Given any $\Theta^{(0)}$, if the perturbations $\epsilon_P, \epsilon_{u,r}$ satisfy that,*

$$e_H(\epsilon_P, \epsilon_{u,r}) < \frac{1}{1 + \sqrt{\frac{2d_x}{B}}} \frac{\sigma_{max}(H) - \sigma_{min}(H)}{\sigma_{max}(H) + \sigma_{min}(H)},$$

for any $B_0 \geq 1$, one can pick

$$\eta = \frac{2}{\sigma_{max}(H) + \sigma_{min}(H)}$$

and batch size

$$B > max \left(\frac{2d_x\eta \|H\|_2^2}{(1 - \|I + \eta H\|)^2 - (1 + \sqrt{\frac{2d_x}{B_0}})e_H(\epsilon_P, \epsilon_{u,r})}, B_0 \right),$$

such that

$$\mathbb{E} \left\| \Theta^{(k)} - \Theta^* \right\| \leq \gamma^k \mathbb{E} \left\| \Theta^{(0)} - \Theta^* \right\| + \frac{1 - \gamma^k}{1 - \gamma} e(\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi}),$$

where $0 < \gamma < 1$, and

$$e(\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi}) := (1 + \sqrt{\frac{2d_x}{B}}) \left(e_G(\epsilon_P, \epsilon_{u,r}, \epsilon_{u,\xi}) + e_H(\epsilon_P, \epsilon_{u,r}) \|\Theta^*\| \right).$$

We verify the predictions of the theorem qualitatively in the experiment section.

D Planning Only Subset of States

We consider the case where the state cost $\mathcal{C}$ and constraints $\mathcal{X}$ only require a subset of the states, i.e., if they are defined in terms of $z_t = g(x_t) \in \mathbb{R}^{d_z}$, with $d_z < d_x$. Specifically, we consider the problem

$$
\begin{aligned}
\underset{x,u,z}{\text{minimize}} \quad & \mathcal{C}(z) + \mathcal{D}(u) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t = 0, 1, ..., T - 1, \\
& z \in \mathcal{X}, \quad u \in \mathcal{U}, \\
& x_0 = \xi, \\
& z_t = g(x_t).
\end{aligned}
$$

In this case, one can modify the redundant constraint to be $r_t = z_t$ to arrive at the following redundant problem

$$
\begin{aligned}
\underset{r,x,u}{\text{minimize}} \quad & \mathcal{C}(r) + \mathcal{D}(u) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t = 0, 1, ..., T - 1, \\
& r \in \mathcal{X}, \quad u \in \mathcal{U}, \\
& x_0 = \xi, \quad r = g(x),
\end{aligned}
$$

where we wrote $g(x)$ to denote $[g(x_0), g(x_1), ..., g(x_T)]^\top$ with a slight abuse of notation. A similar derivation to that in Section 3 then arrives at the following iterative update

$$
\begin{aligned}
(r^+, x^+, u^+) = \underset{r,x,u}{\text{argmin}} \quad & \mathcal{C}(r) + \mathcal{D}(u) + \frac{\rho}{2} \|r + \nu - g(x)\|_2^2 \\
\text{s.t.} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t, \\
& r \in \mathcal{X}, u \in \mathcal{U}, \\
& x_0 = \xi
\end{aligned}
$$

$$\nu^+ = \nu + (r^+ - x^+)$$

and the nested optimization

$$r^+ = \underset{r}{\text{minimize}} \quad \mathcal{C}(r) + p^*(r + \nu; \xi)$$
$$\text{s.t.} \quad r \in \mathcal{X}$$

where $p^*(r + \nu; \xi)$ is the locally optimal value of the (x, u)-minimization step

$$p^*(r + \nu; \xi) = \underset{x, u}{\text{min}} \quad \mathcal{D}(u) + \frac{\rho}{2} \|r + \nu - g(x)\|_2^2$$
$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t), \ u \in \mathcal{U}, \ \forall t,$$
$$x_0 = \xi.$$

Note that the only difference is that the trajectory planner only generate reference trajectories on the states z required for the state cost and constraints, and that the tracking cost for the lower-level controller also only concerns tracking those states.