EE899 Project Report:
Cost-Constrained LQGs via Convex Reformulation

Kihyun Yu

1 Introduction

While optimal control algorithms have demonstrated their effectiveness in many real-world applications, several tasks require additional constraints to guarantee safety. For instance, in drone control, the available battery power is limited, while in robotics, safety constraints must ensure that robots do not harm humans. To address such requirements, constrained optimal control has gained significant attention, including formulations such as constrained Markov Decision Processes (MDPs) and constrained Linear Quadratic Gaussians (LQGs).

In this report, we focus on discrete-time average-loss LQGs with cost constraints, whose formal definition is provided in Section 2. The key challenges in solving cost-constrained LQGs are as follows: (i) the failure of Bellman's optimality principle and (ii) the presence of infinitely many states and actions. Due to (i), we cannot use the Discrete Algebraic Riccati Equation (DARE) as in standard LQGs. Due to (ii), we cannot apply useful techniques commonly used for constrained MDPs, which rely on the finiteness of the state and action spaces. Both of these challenges will be discussed in Section 3 with detailed examples.

Contributions Despite these challenges, we solve the cost-constrained LQG problem via a convex reformulation. Our technical contributions and results are summarized as follows.

- We apply a known semi-definite program (SDP) reformulation for unconstrained LQGs to the cost-constrained setting. Specifically, the SDP reformulation proposed by [1] was originally designed for unconstrained LQGs. We extend this to LQGs with cost constraints and prove, both theoretically and empirically, that it yields an optimal policy. [1]
- We further extend our approach to an online setting to handle scenarios with unknown transition dynamics, and empirically demonstrate the effectiveness of the resulting algorithm. In this case, the agent must not only minimize the loss but also learn the underlying dynamics through interaction with the environment. To resolve this, we incorporate a transition-estimation step (e.g., [2]) into our algorithm. As performance metrics, we consider regret and constraint violation, which are standard in online settings. The source code is available at: `https://github.com/kihyun-yu/cost-constrained-lqg`.

[1] It is worth noting how we construct a precomputed optimal policy for comparison with our solution, as there is no widely accepted method for solving constrained LQGs. To this end, we first solve an unconstrained LQG to obtain an optimal policy π^*, and then set the cost budget according to the cost value function induced by π^*. For further details, we refer the reader to Section 6.2.

Organization The rest of the report is organized as follows. In Section 1.1, we discuss related work, focusing on algorithms for unconstrained LQGs with unknown transition functions and algorithms for LQGs with various constraints. In Section 2, we specify the discrete-time average-loss cost-constrained LQG setting and the standard performance metrics for online algorithms, namely, regret and constraint violation. In Section 4, we introduce a convex reformulation. In Section 5, we present an online algorithm along with its empirical performance. Finally, we provide detailed proofs and parameter settings in Section A and Section B, respectively.

1.1 Related Work

LQG with Unknown Transition Function To handle an unknown transition function, previous work has primarily focused on online algorithms that simultaneously learn the transition function while minimizing cumulative loss. In this setting, regret serves as the standard performance metric, measuring the suboptimality of the cumulative loss relative. Specifically, [3] proposed the first algorithm to solve LQGs with unknown dynamics achieving a regret bound of $\widetilde{\mathcal{O}}(\sqrt{T})$, where T is the number of time steps. While [4] improved the regret upper bound, both algorithms are computationally inefficient due to non-convex optimization steps. Subsequently, [5] proposed a computationally efficient algorithm with a $\widetilde{\mathcal{O}}(T^{2/3})$ regret bound. Recently, [2, 6] developed computationally efficient algorithms with $\widetilde{\mathcal{O}}(\sqrt{T})$ regret bounds. Notably, [7] achieved the first nearly minimax-optimal algorithm, in the sense that its regret upper bound matches the lower bound. Among them, [1, 2] utilized convex reformulations of LQGs, although their approaches are limited to the unconstrained setting.

LQG with Constraints LQGs with constraints have been studied with various types of constraints: (i) affine constraints: $Q^g\mathbf{x}_t \le b_{\mathbf{x}},\ Q^f\mathbf{u}_t \le b_{\mathbf{u}}$ [8]; (ii) risk constraints: $\sum_{t=1}^{T}\mathbb{E}[\mathrm{Var}(r_t \mid \mathcal{F}_{t-1})] \le \epsilon$ where $r_t = \mathbf{x}_t^\top Q\mathbf{x}_t + \mathbf{u}_{t-1}^\top R\mathbf{u}_{t-1}$ and $\mathcal{F}_{t-1}$ is the information up to step $t-1$ [9]; (iii) chance constraints: $\mathbb{P}[(q^g)^\top\mathbf{x}_t \le b] \ge 1-\epsilon$ [10]; and (iv) quadratic constraints: $\mathbf{x}_t^\top Q^g\mathbf{x}_t + \mathbf{u}_t^\top R^g\mathbf{u}_t \le b$ [11]. Among them, the most relevant work with ours is [11], which considers quadratic constraints. However, their algorithm requires gradient computations that assume knowledge of the transition function. Thus, our setting is more challenging and realistic, since the transition function is unknown.

2 Problem Statement

Let $\mathbb{R}^{d_{\mathbf{x}}}$ and $\mathbb{R}^{d_{\mathbf{u}}}$ be the state and action spaces, where $d_{\mathbf{x}}$ and $d_{\mathbf{u}}$ are the dimensions of state and action, respectively. Given $A \in \mathbb{R}^{d_{\mathbf{x}}\times d_{\mathbf{x}}}$, $B \in \mathbb{R}^{d_{\mathbf{x}}\times d_{\mathbf{u}}}$, and $\sigma > 0$, we define the transition function as follows

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w} \quad \text{where} \quad \mathbf{w} \sim \mathcal{N}(0, \sigma^2 I).$$

Given $Q^f, Q^g \in \mathbb{R}^{d_{\mathbf{x}}\times d_{\mathbf{x}}}$, $R^f, R^g \in \mathbb{R}^{d_{\mathbf{u}}\times d_{\mathbf{u}}}$,[2], and $\mathbf{x}_1 = \mathbf{0}$, we define the cost functions with respect to a policy π as

$$J^f(\pi) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{x}_t^\top Q^f\mathbf{x}_t + \mathbf{u}_t^\top R^f\mathbf{u}_t\right].$$

[2]It is common to further assume that the function parameters satisfy $\alpha_1 \preceq Q^f, Q^g, R^f, R^g \preceq \alpha_2$ for some α_1, α_2 in order to derive regret upper bounds. In this project, however, we do not impose such assumptions, as deriving theoretical regret upper bounds is beyond our scope.

We define J^g in the same way. Finally, we define the discrete-time average-loss cost-constrained LQG:

$$\begin{aligned} \min_{\pi \in \Pi} \quad & J^f(\pi) \\ \text{s.t.} \quad & J^g(\pi) \le b \\ & \mathbf{u}_t = \pi(\mathbf{x}_t) \\ & \mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t \end{aligned} \tag{CLQG}$$

where b is the cost budget, $\pi : \mathbb{R}^{d_{\mathbf{x}}} \to \mathbb{R}^{d_{\mathbf{u}}}$ is a policy, along with Π is the set of all possible policies. Furthermore, we note that the constraint $\lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{x}_t^\top Q^g \mathbf{x}_t + \mathbf{u}_t^\top R^g \mathbf{u}_t\right] \le b$ ensures that the expected cumulative cost does not exceed the budget b over the process.

We assume that the loss and cost parameters Q^f, Q^g, R^f, R^g are known, while the dynamics parameters A, B are possibly unknown. Moreover, let π^* denote an optimal policy to (CLQG). Next, we define performance metrics for online algorithms.

Definition 1 (Regret and Constraint Violation)**.** The regret, denoted by Regret(T), measures the suboptimality of cumulative loss compared to that of an optimal policy π^*:

$$\text{Regret}(T) \triangleq \sum_{t=1}^{T} (\mathbf{x}_t^\top Q^f \mathbf{x}_t + \mathbf{u}_t^\top R^f \mathbf{u}_t) - TJ^f(\pi^*).$$

The constraint violation, denoted by Violation(T), quantifies how much the agent exceeds the cost budget during the learning process:

$$\text{Violation}(T) \triangleq \left[\sum_{t=1}^{T} (\mathbf{x}_t^\top Q^g \mathbf{x}_t + \mathbf{u}_t R^g \mathbf{u}_t) - Tb\right]_+$$

3 Challenges in Constrained LQGs

3.1 Challenge 1: Failure of Bellman Principle

Bellman's optimality principle fails in our setting because the optimal action may depend on the previous trajectory. To illustrate, we provide the following example:

Example 1 (Shortest Path with Costs)**.** Suppose that the agent's goal is to find the shortest path from $\mathbf{x}_1$ to $\mathbf{x}_5$, while ensuring the total cost does not exceed some budget (e.g., highway tolls). The environment is illustrated in Figure 1.

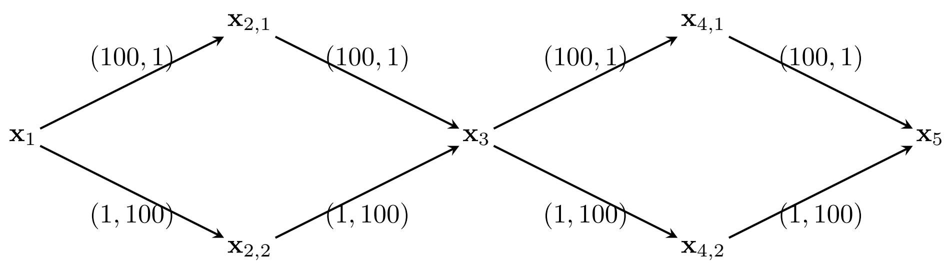


Figure 1: Shortest path with costs. Each pair indicates (distance, cost) of the corresponding path.

When the budget is $+\infty$ (i.e., the unconstrained case), it is easy to see that the optimal path is $\mathbf{x}_1 \to \mathbf{x}_{2,2} \to \mathbf{x}_3 \to \mathbf{x}_{4,2} \to \mathbf{x}_5$, which has a total distance of 4 and a total cost of 400.

Suppose that the budget is 300. Although the path $\mathbf{x}_1 \to \mathbf{x}_{2,2} \to \mathbf{x}_3 \to \mathbf{x}_{4,2} \to \mathbf{x}_5$ is the shortest, it is not optimal because its cost exceeds the budget. In this case, the optimal paths are $\mathbf{x}_1 \to \mathbf{x}_{2,1} \to \mathbf{x}_3 \to \mathbf{x}_{4,2} \to \mathbf{x}_5$ and $\mathbf{x}_1 \to \mathbf{x}_{2,2} \to \mathbf{x}_3 \to \mathbf{x}_{4,1} \to \mathbf{x}_5$, whose total distance and total cost are 202. Here, we can observe that the optimal action in $\mathbf{x}_3$ depends on the previous trajectory. For instance, if the agent passes through $\mathbf{x}_{2,1}$, then the optimal direction is $\mathbf{x}_{4,2}$. Therefore, when the agent is in $\mathbf{x}_3$, it is impossible to determine the optimal direction without knowing which path the agent has previously taken.

To summarize, Bellman's optimality principle implies that any sub-trajectory of an optimal trajectory is also optimal. However, as shown in Example 1, the presence of constraints causes the optimality of a sub-trajectory to depend on previous trajectories. Consequently, we cannot break the control problem into a sequence of a sub-problem, requiring new approaches to solve optimal control with constraints.

3.2 Challenge 2: Infinite States and Actions

The second challenge of constrained LQGs is that the number of states and actions is infinite, which hinders the use of well-known reformulation techniques for constrained MDPs. For constrained MDPs, there exist simple and efficient algorithms via linear programming (LP) reformulation. Before introducing the reformulation, we specify the constrained MDP. Given cost functions f, g, budget b, and transition function P, let $V_f^\pi(x) \triangleq (1-\gamma)\mathbb{E}\left[\sum_{t=1}^\infty \gamma^t f(x_t, u_t) \mid P, \pi, x_1 = x\right]$ for some discount factor $\gamma \in (0,1)$, and let us consider the following constrained MDP:

$$\begin{aligned} \max_{\pi \in \Pi} \quad & V_f^\pi(x_1) \\ \text{s.t.} \quad & V_g^\pi(x_1) \le b. \end{aligned} \tag{CMDP}$$

As shown in [12], (CMDP) can be reformulated into an LP using the notion of occupancy measure. The occupancy measure $q^{\pi,P} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{A}|}$ is defined as follows:

$$q^{\pi,P}(x,u) \triangleq (1-\gamma)\sum_{t=1}^{\infty} \gamma^t \Pr[x_t = x, u_t = u | P, \pi, x_1]. \tag{1}$$

Here, the value functions are linear in $q^{\pi,P}$, i.e., $V_f^\pi(x_1) = \sum_{x\in\mathcal{X}, u\in\mathcal{U}} f(x,u) q^{\pi,P}(x,u) \triangleq \langle f, q^{\pi,P}\rangle$. It leads to the following LP reformulation:

$$\begin{aligned} \max_{q \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{A}|}} \quad & \langle f, q \rangle \\ \text{s.t.} \quad & \langle g, q \rangle \le b \\ & \sum_u q(x,u) = (1-\gamma)\mathbb{1}_{\{x = x_1\}} + \gamma \sum_{x',u'} P(x|x',u') q(x',u') \quad \forall x \in \mathcal{X} \\ & q(x,a) \ge 0 \quad \forall x, a \in \mathcal{X}\times\mathcal{A}. \end{aligned} \tag{LP-CMDP}$$

Consequently, we reformulate (CMDP) into (LP-CMDP), which becomes an LP with respect to q. Furthermore, letting q^* denote an optimal solution to this LP, a policy defined by $\pi^*(u|x) = q^*(x,u)/\sum_{u'} q^*(x,u')$ is proven to be optimal for (CMDP). For more details, we refer the reader to Chapter 3 of [12].

However, such reformulation cannot be applied to constrained LQGs. The main technical difficulty arises from the infinite number of states and actions. In the LQG setting, since $\mathcal{X}$ and $\mathcal{A}$ could be continuous, the decision variable $q \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{A}|}$ becomes a infinite-dimensional vector. Consequently, the reformulation is no longer LP, and a fundamentally different approach must be considered.

4 SDP Reformulation

To address the challenges mentioned previously, we introduce the semi-definite programming (SDP) reformulation for LQGs. This reformulation was proposed by [1] for unconstrained LQGs, and the only difference in our reformulation is the addition of the cost constraint.

Before introducing it, we first define a matrix notation. Let $\Sigma \in \mathbb{R}^{(d_{\mathbf{x}}+d_{\mathbf{u}})\times(d_{\mathbf{x}}+d_{\mathbf{u}})}$ be a symmetric matrix. Then let $\Sigma_{xx} \in \mathbb{R}^{d_{\mathbf{x}}\times d_{\mathbf{x}}}, \Sigma_{xu} \in \mathbb{R}^{d_{\mathbf{x}}\times d_{\mathbf{u}}}, \Sigma_{ux} \in \mathbb{R}^{d_{\mathbf{u}}\times d_{\mathbf{x}}}, \Sigma_{uu} \in \mathbb{R}^{d_{\mathbf{u}}\times d_{\mathbf{u}}}$ denote the corresponding blocks, i.e.,

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xu} \\ \Sigma_{ux} & \Sigma_{uu} \end{bmatrix}.$$

Next, we provide the SDP reformulation as follows. For simplicity, let $d \triangleq d_{\mathbf{x}}+d_{\mathbf{u}}$, and let $P \bullet Q \triangleq \operatorname{tr}(P^\top Q)$ for any matrices P, Q.

$$\begin{aligned} \min_{\Sigma\in\mathbb{R}^{d\times d}} \quad & \begin{bmatrix} Q^f & 0 \\ 0 & R^f \end{bmatrix} \bullet \Sigma \\ \text{s.t.} \quad & \begin{bmatrix} Q^g & 0 \\ 0 & R^g \end{bmatrix} \bullet \Sigma \leq b \\ & \Sigma_{xx} = \begin{bmatrix} A & B \end{bmatrix} \Sigma \begin{bmatrix} A^\top \\ B^\top \end{bmatrix} + \sigma^2 I \\ & \Sigma \succeq 0. \end{aligned} \tag{SDP}$$

We note that (SDP) is a convex program in Σ. Consequently, well-established algorithms and optimization libraries for solving constrained convex programs can be applied, such as projected gradient descent [13] or `CVXPY` [14].

After obtaining an optimal solution Σ^* from (SDP), we construct a policy using the following rule:

$$\pi(\mathbf{x}) = \mathcal{K}(\Sigma^*)\mathbf{x} \quad \text{where} \quad \mathcal{K}(\Sigma^*) \triangleq (\Sigma^*_{xu})^\top (\Sigma^*_{xx})^{-1}. \tag{2}$$

In Section 6, we present theoretical guarantees and empirical evidence showing that (SDP) is indeed a reformulation. That is, an optimal policy obtained from (SDP) is also optimal to the original problem (CLQG).

Intuition The key intuition behind the reformulation is to convert the decision variable from policies into matrices, because optimizing over matrices Σ is relatively easier than optimizing over policies π. To achieve this, we may consider the following steps: (i) constructing a mapping from policies to matrices, and then (ii) appropriately reformulating the objective function and constraints. These steps result in (SDP).

Since we change the decision variable space, an important task is to construct mappings between the policy space and matrix space. To this end, $\mathcal{E}$ defines a map from policies to matrices, while $\mathcal{K}$ defines a map from matrices to policies as follows: given the steady-state covariance $\mathbb{E}[\mathbf{x}\mathbf{x}^\top]$ and $\mathbf{u} = \pi(\mathbf{x})$,

$$\mathcal{E}(\pi) = \begin{bmatrix} \mathbb{E}[\mathbf{x}\mathbf{x}^\top] & \mathbb{E}[\mathbf{x}\mathbf{u}^\top] \\ \mathbb{E}[\mathbf{u}\mathbf{x}^\top] & \mathbb{E}[\mathbf{u}\mathbf{u}^\top] \end{bmatrix}, \quad \mathcal{K}(\Sigma) = \Sigma_{xu}^\top \Sigma_{xx}^{-1}.$$

5 Online Algorithm

Based on [2]—an online algorithm for unconstrained LQGs with unknown A, B, we design Algorithm 1, an online algorithm for cost-constrained LQGs with unknown A, B. Basically, the algorithm operates as follows. In lines 2 - 15, the algorithm prepares the policy π_t for each step t. In the warm-up phase ($t \leq T_{\text{warm}}$), it takes a known stable policy induced by K_{warm}. After the warm-up phase, it prepares the policy by estimating the transition function and then solving an SDP. In line 17, the algorithm takes the action $\mathbf{u}_t = \pi_t(\mathbf{x}_t)$, and observe the next state $\mathbf{x}_{t+1}$. Then the design matrix V_{t+1} is updated.

Algorithm 1 Online algorithm for cost-constrained LQGs with unknown A, B

Input: Function parameters Q^f, R^f, Q^g, R^g; Noise parameter σ^2; Hyperparameters β, λ, μ; Warm-up phase T_{warm} and stable policy K_{warm}

Initialization: $A_0, B_0 \leftarrow 0$; $V_1 \leftarrow \lambda I$; $\mathbf{x}_1 \leftarrow 0$

```
 1: for $t = 1, \dots, T$ do
 2:   if $t \leq T_{\text{warm}}$ then
 3:     $\pi_t(\mathbf{x}) \leftarrow K_{\text{warm}}\mathbf{x}$
 4:   end if
 5:   if $t \geq T_{\text{warm}} \wedge (\det(V_t) > 2\det(V_\tau) \vee t = T_{\text{warm}} + 1)$ then
 6:     $\tau \leftarrow t$
 7:     Estimate $A_t, B_t$:
```

$$A_t, B_t \in \arg\min_{A,B} \frac{1}{\beta} \sum_{s=1}^{t-1} \| \begin{bmatrix} A & B \end{bmatrix} z_s - x_{s+1} \|_2^2 + \lambda \| \begin{bmatrix} A & B \end{bmatrix} - \begin{bmatrix} A_0 & B_0 \end{bmatrix} \|_F^2$$

```
 8:     Solve SDP:
```

$$\begin{aligned} \Sigma_t^* = \min_{\Sigma \in \mathbb{R}^{d \times d}} \quad & \begin{bmatrix} Q^f & 0 \\ 0 & R^f \end{bmatrix} \bullet \Sigma \\ \text{s.t.} \quad & \begin{bmatrix} Q^g & 0 \\ 0 & R^g \end{bmatrix} \bullet \Sigma \leq b \\ & \Sigma_{xx} \succeq \begin{bmatrix} A_t & B_t \end{bmatrix} \Sigma \begin{bmatrix} A_t^\top \\ B_t^\top \end{bmatrix} + \sigma^2 I - \mu \cdot \Sigma \bullet V_t^{-1} \cdot I \\ & \Sigma \succeq 0. \end{aligned} \tag{Optimistic SDP}$$

```
 9:     if (Optimistic SDP) is feasible then
10:       $\pi_t(\mathbf{x}) = (\Sigma_t^*)_{ux}(\Sigma_t^*)_{xx}^{-1}\mathbf{x}$
11:     else
12:       $\pi_t(\mathbf{x}) \leftarrow K_{\text{warm}}\mathbf{x}$
13:     end if
14:   else
15:     $\pi_t \leftarrow \pi_{t-1}, A_t \leftarrow A_{t-1}, B_t \leftarrow B_{t-1}$
16:   end if
17:   Take action $\mathbf{u}_t = \pi_t(\mathbf{x}_t)$ and observe $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t$ where $\mathbf{w}_t \sim \mathcal{N}(0, \sigma^2 I)$
18:   Update $z_t \leftarrow \begin{bmatrix} x_t \\ u_t \end{bmatrix}$ and $V_{t+1} \leftarrow V_t + \frac{1}{\beta} z_t z_t^\top$
19: end for
```

To handle the uncertain transition function, we borrow two key techniques from [2], originally proposed for handling unknown A, B in the unconstrained LQG setting. More detailed descriptions are as follows:

(i) Estimating $\boldsymbol{A}, \boldsymbol{B}$ (lines 5-7) To estimate A, B, the algorithm solves a regularized least-squares regression based on the data gathered up to the current step t. The obtained estimates A_t, B_t are then used to construct an SDP.

(ii) Solving optimistic SDP (line 8) Using the estimates A_t, B_t, the algorithm constructs (Optimistic SDP). The key difference from (SDP) is that the constraint for the transition kernel is relaxed; specifically, the right-hand side of the constraint is additionally subtracted by $\mu \cdot \Sigma \bullet V_t^{-1} \cdot I$, and $=$ is replaced with $\succeq$. This technique is called *the optimism in the face of uncertainty principle*—choosing the best decision among all plausible models. In general, the primary purpose of this technique is to balance exploitation and exploration in order to handle uncertainty in the environment—in our case, the transition function. This principle is widely used in the bandit and reinforcement learning literature (e.g., see the seminal work by [3]).

6 Results

6.1 Theoretical Result

We prove the following theorem by extending Theorem 4.2 of [1] to our setting, in which the cost constraint is considered. This theorem states that the SDP reformulation remains valid under the cost constraint.

Theorem 1. Suppose that the optimal policy for (CLQG) is stable. Let Σ^* be an optimal solution to (SDP), and let $\pi(\mathbf{x}) = \mathcal{K}(\Sigma^*)\mathbf{x}$. Then π is optimal for the original problem (CLQG).

6.2 Empirical Results

Parameter Setting We generate an instance of cost-constrained LQGs as follows. We set the state and action dimensions as $d_{\mathbf{x}} = 4, d_{\mathbf{u}} = 2$, and the covariance parameter as $\sigma^2 = 0.1$ for the noise term in the transition function. The parameters for the functions Q_f, R_f, Q_g, and R_g are randomly generated while satisfying positive definiteness. On the other hand, the transition function parameters A and B are randomly generated and may not be positive definite. The choice of the cost budget will be described in the next paragraph.

An optimal policy for a cost-constrained LQG instance is computed as follows. Since there is no widely used algorithm for computing an optimal solution under constraints, we proceed in two steps. First, using the DARE, we compute an optimal solution without constraints, i.e., ignoring $J^g(\pi) \le b$. Based on this unconstrained optimal solution π, we compute the expected cost $J^g(\pi)$, and set the cost budget as $b = J^g(\pi) + 0.01$. The detailed parameter settings can be found in Section B.

Result when $\boldsymbol{A}, \boldsymbol{B}$ are known In Table 1, we present convergence results when A, B are known. The results are obtained by solving (SDP). We observe that $J^f(\pi^*) = J^f(\pi_{\text{SDP}})$ and $J^g(\pi_{\text{SDP}}) \le b$. Thus, we conclude that (SDP) successfully solves (CLQG) when A, B are known.

Table 1: Comparison of π^* and π_{SDP}. π^* denotes a precomputed optimal policy, and π_{SDP} denotes a solution obtained by solving (SDP). In the second row, 0.9385 and 0.9285 correspond to b and $J^g(\pi_{\text{SDP}})$, respectively.

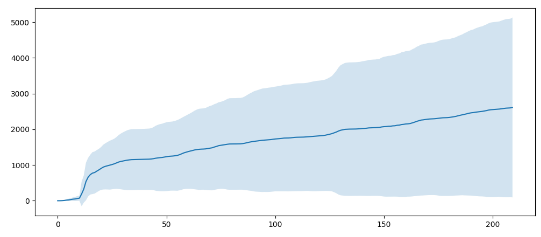
	π^*	π_{SDP}
$J^f(\cdot)$	0.9152	0.9152
b and $J^g(\cdot)$	0.9385	0.9285

Result when A, B are unknown In Table 2, we present convergence results when A, B are unknown. The results are obtained by running Algorithm 1. Unlike the known A, B case, we observe that π_{OSDP} does not converge to an optimal solution; although the constraint holds, $\pi_{\text{OSDP}} < b$, the objective is suboptimal, i.e., $J^f(\pi^*) < J^f(\pi_{\text{OSDP}})$. The main reason for this gap comes from the estimation error in A and B.

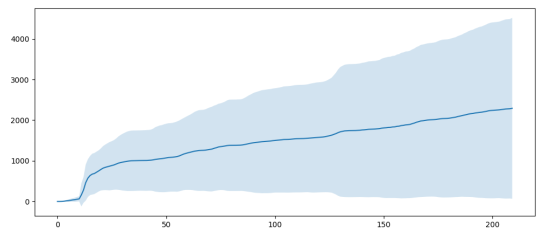
Table 2: Comparison of π^* and π_{OSDP}. By running Algorithm 1 for $T = 200$ steps, we obtain π_{OSDP}. In particular, π_{OSDP} denotes the policy obtained by solving (Optimistic SDP) with the estimates A_{200} and B_{200}.

	π^*	π_{OSDP}
$J^f(\cdot)$	0.9152	0.9205
b and $J^g(\cdot)$	0.9385	0.9323

Additionally, in Figure 2, we show the performance of Algorithm 1 using other performance metrics, i.e., regret and constraint violation given in Definition 1. During the warm-up phase $t = 1, \ldots, 10$, we observe that the regret and violation grow slowly, while they increase rapidly during $t = 10, \ldots, 20$. After this phase, they grow slowly again, exhibiting sublinear growth in both regret and constraint violation. These observations indicate that A and B are learned by the algorithm. On the other hand, the standard deviation in both cases are relatively large, suggesting that the algorithm requires improvement for empirical stabilization.



(a) Regret



(b) Constraint Violation

Figure 2: Performance of Algorithm 1 in terms of regret and constraint violation. We conducted 10 trials with different random seeds. The plots show the means of regret and constraint violation, respectively, and the shaded regions denote ±`std`. We set the hyperparameters as follows: $T = 210$, $T_{\text{warm}} = 10$, $\beta = 1$, $\mu = 0.02$, $\lambda = 1.0$.

7 Conclusion

In this project, we study discrete-time average-loss cost-constrained LQGs. Inspired by the SDP reformulation for unconstrained LQGs [1], we extend this reformulation to the setting in which a cost constraint is considered. Both theoretically and empirically, we show that the extended reformulation remains valid under the cost constraint. Moreover, by adopting the techniques from [2], we design an online algorithm to handle unknown A and B, and empirically demonstrate its effectiveness by reporting regret and constraint violation. Finally, we conclude the project by pointing out limitations and future directions as follows:

- **Deriving theoretical upper bounds on regret and constraint violation;** In the RL theory literature, deriving regret bounds—e.g., $\text{Regret}(T) = O(\sqrt{T})$—is regarded as an important step for demonstrating the theoretical effectiveness of proposed online algorithms. While we provide empirical results for Algorithm 1, this is not sufficient.

- **Discounted-loss setting;** The current SDP reformulation works only for the average-loss setting. This is because the relation $J(\pi) = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \bullet \mathcal{E}(\pi)$, which is the key of the reformulation, holds when $J(\pi)$ is defined as the expected average loss. Extending this to the discounted setting is therefore an important problem, yet non-trivial problem.

References

[1] A. Cohen, A. Hasidim, T. Koren, N. Lazic, Y. Mansour, and K. Talwar, "Online linear quadratic control," in *International Conference on Machine Learning.* PMLR, 2018, pp. 1029–1038.

[2] A. Cohen, T. Koren, and Y. Mansour, "Learning linear-quadratic regulators efficiently with only $\sqrt{T}$ regret," in *International Conference on Machine Learning.* PMLR, 2019, pp. 1300–1309.

[3] Y. Abbasi-Yadkori and C. Szepesvári, "Regret bounds for the adaptive control of linear quadratic systems," in *Proceedings of the 24th Annual Conference on Learning Theory.* JMLR Workshop and Conference Proceedings, 2011, pp. 1–26.

[4] M. Ibrahimi, A. Javanmard, and B. Roy, "Efficient reinforcement learning for high dimensional linear quadratic systems," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[5] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "Regret bounds for robust adaptive control of the linear quadratic regulator," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[6] H. Mania, S. Tu, and B. Recht, "Certainty equivalence is efficient for linear quadratic control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] M. Simchowitz and D. Foster, "Naive exploration is optimal for online lqr," in *International Conference on Machine Learning.* PMLR, 2020, pp. 8937–8948.

[8] Y. Li, S. Das, and N. Li, "Online optimal control with affine constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 8527–8537.

[9] A. Tsiamis, D. S. Kalogerias, A. Ribeiro, and G. J. Pappas, "Linear quadratic control with risk constraints," *arXiv preprint arXiv:2112.07564*, 2021.

[10] G. Schildbach, P. Goulart, and M. Morari, "The linear quadratic regulator with chance constraints," in *2013 European Control Conference (ECC).* IEEE, 2013, pp. 2746–2751.

[11] F. Zhao and K. You, "Policy gradient methods for the cost-constrained lqr: Strong duality and global convergence," *IEEE Transactions on Automatic Control*, 2025.

[12] E. Altman, *Constrained Markov decision processes.* Routledge, 2021.

[13] A. Beck, *First-order methods in optimization.* SIAM, 2017.

[14] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[15] J. Gallier, "Notes on the schur complement," 2010.

A Missing Proofs

Lemma 1. Let π be a stable policy. If π is feasible to (CLQG), then $\mathcal{E}(\pi)$ is feasible to (SDP).

Proof. We closely follow the proof of Lemma 4.1 of [1], where the main difference comes from the cost constraint. Let $\mathbf{x}$ be the stable state, and let $\mathbf{u} = \pi(\mathbf{x})$. It follows that

$$\begin{aligned} J^g(\pi) &= \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{x}_t^\top Q^g \mathbf{x}_t + \mathbf{u}_t^\top R^g \mathbf{u}_t\right] \\ &= \mathbf{x}^\top Q^g \mathbf{x} + \mathbf{u}^\top R^g \mathbf{u} \\ &= \operatorname{tr}(Q^g \mathbf{x}\mathbf{x}^\top) + \operatorname{tr}(R^g \mathbf{u}\mathbf{u}^\top) \\ &= \operatorname{tr}\left(\begin{bmatrix} Q^g & 0 \\ 0 & R^g \end{bmatrix}^\top \mathcal{E}(\pi)\right) \\ &= \begin{bmatrix} Q^g & 0 \\ 0 & R^g \end{bmatrix} \bullet \mathcal{E}(\pi). \end{aligned}$$

This implies that

$$J^g(\pi) \le b \quad \Leftrightarrow \quad \begin{bmatrix} Q^g & 0 \\ 0 & R^g \end{bmatrix} \bullet \mathcal{E}(\pi) \le b.$$

Furthermore, since $\mathbf{x}$ is a stable state, $\mathcal{E}(\pi)$ satisfies that

$$\begin{aligned} \mathcal{E}(\pi)_{xx} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] &= \mathbb{E}\left[\left(\begin{bmatrix} A & B \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} + \mathbf{w}\right)\left(\begin{bmatrix} A & B \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} + \mathbf{w}\right)^\top\right] \\ &= \begin{bmatrix} A & B \end{bmatrix}\begin{bmatrix} \mathbf{x}\mathbf{x}^\top & \mathbf{x}\mathbf{u}^\top \\ \mathbf{u}\mathbf{x}^\top & \mathbf{u}\mathbf{u}^\top \end{bmatrix}\begin{bmatrix} A^\top \\ B^\top \end{bmatrix} + \sigma^2 I \\ &= \begin{bmatrix} A & B \end{bmatrix}\mathcal{E}(\pi)\begin{bmatrix} A^\top \\ B^\top \end{bmatrix} + \sigma^2 I. \end{aligned}$$

Finally, we note that $\operatorname{tr}(\mathcal{E}(\pi)) = \operatorname{tr}(\mathbf{x}\mathbf{x}^\top) + \operatorname{tr}(\mathbf{u}\mathbf{u}^\top) = \operatorname{tr}(\mathbf{x}^\top\mathbf{x}) + \operatorname{tr}(\mathbf{u}^\top\mathbf{u}) = \|\mathbf{x}\|_2^2 + \|\mathbf{u}\|_2^2 \ge 0$. This implies that every eigenvalues of $\mathcal{E}(\pi)$ is nonnegative, i.e., $\mathcal{E}(\pi) \succeq 0$. $\square$

Proof of Theorem 1 The proof closely follows the proof of Theorem 4.2 of [1], and the only distinction is the cost constraint. For clarity, recall the key notations. Let Σ denote a feasible solution to (SDP), and it is written as

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xu} \\ \Sigma_{xu}^\top & \Sigma_{uu} \end{bmatrix}.$$

Given Σ, let

$$K \triangleq \mathcal{K}(\Sigma) = (\Sigma_{xu})^\top(\Sigma_{xx})^{-1}.$$

Step 1: π is stable To show the stability of π, it is sufficient to show that $\rho(A + BK) < 1$, where ρ denotes the spectral radius, i.e., the maximum of absolute eigenvalues. For this, we additionally define

$$\Sigma' \triangleq \begin{bmatrix} \Sigma_{xx} & \Sigma_{xx}K^\top \\ K\Sigma_{xx} & K\Sigma_{xx}K^\top \end{bmatrix}.$$

By the definition of K, we know that $\Sigma_{xu} = \Sigma_{xx} K^\top$. It follows that

$$\Sigma - \Sigma' = \begin{bmatrix} 0 & 0 \\ 0 & \Sigma_{uu} - K\Sigma_{xx}K^\top \end{bmatrix}.$$

Note that $\Sigma_{uu} - K\Sigma_{xx}K^\top$ is the Schur complement of Σ, and thus $\Sigma_{uu} - K\Sigma_{xx}K^\top \succ 0$ ([15]). Therefore, this implies that

$$\Sigma \succ \Sigma'. \tag{3}$$

By left multiplying by $\begin{bmatrix} A & B \end{bmatrix}$ and right multiplying by $\begin{bmatrix} A^\top \\ B^\top \end{bmatrix}$ on both sides, we have

$$\begin{bmatrix} A & B \end{bmatrix} \Sigma \begin{bmatrix} A^\top \\ B^\top \end{bmatrix} \succeq \begin{bmatrix} A & B \end{bmatrix} \Sigma' \begin{bmatrix} A^\top \\ B^\top \end{bmatrix}.$$

The left-hand side can be rewritten as $\Sigma_{xx} - \sigma^2 I = \begin{bmatrix} A & B \end{bmatrix} \Sigma \begin{bmatrix} A^\top \\ B^\top \end{bmatrix}$, since σ is a feasible solution of (SDP). Moreover, the right-hand side can be written as, by the definition of Σ',

$$\begin{bmatrix} A & B \end{bmatrix} \Sigma' \begin{bmatrix} A^\top \\ B^\top \end{bmatrix} = (A + BK)\Sigma_{xx}(A + BK)^\top.$$

Consequently, since $\sigma^2 I \succ 0$, we have

$$\Sigma_{xx} \succ (A + BK)\Sigma_{xx}(A + BK)^\top.$$

Let λ, v be any eigen pair of $A + BK$, and let $v^\dagger$ denote the conjugate transpose of v. Then it follows that

$$v^\dagger \Sigma_{xx} v > |\lambda|^2 v^\dagger \Sigma_{xx} v.$$

Note that $\Sigma_{xx} \succ 0$. Thus, $|\lambda|^2 < 1$, which means that $\rho(A + BK) < 1$.

Step 2: $\boldsymbol{\mathcal{E}(K) \preceq \Sigma}$ Since we showed π is stable in the previous step, the steady-state covariance $X \triangleq \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$ exists under π. Given this, we can define $\mathcal{E}(K)$ as

$$\mathcal{E}(K) = \begin{bmatrix} X & XK^\top \\ KX & KXK^\top \end{bmatrix}.$$

To show this, it is sufficient to show that $\Delta \triangleq \Sigma_{xx} - X \succeq 0$. For this, we have

$$\begin{aligned} X + \Delta = \Sigma_{xx} &= \begin{bmatrix} A & B \end{bmatrix} \Sigma \begin{bmatrix} A^\top \\ B^\top \end{bmatrix} + \sigma^2 I \\ &\succeq (A + BK)\Sigma_{xx}(A + BK)^\top + \sigma^2 I \\ &= (A + BK)X(A + BK)^\top + (A + BK)\Delta(A + BK)^\top + \sigma^2 I \\ &= X + (A + BK)\Delta(A + BK)^\top \end{aligned}$$

where the inequality is due to (3), and the last equality follows from the fact that $X = (A + BK)X(A + BK)^\top + \sigma^2 I$. It follows that

$$\Delta \succeq (A + BK)\Delta(A + BK)^\top.$$

Note that $\rho(A + BK) < 1$, which is proven in the previous step. Then, recursively applying the above inequality yields

$$\Delta \succeq (A + BK)^n \Delta (A^\top + K^\top B^\top)^n \to 0 \quad \text{as } n \to \infty.$$

This implies that $\Delta \succeq 0$, i.e., $\Sigma_{xx} \succeq X$. Then we have

$$\mathcal{E}(K) \preceq \Sigma.$$

Step 3: $\boldsymbol{\pi}(\mathbf{x}) = K^*\mathbf{x}$ is optimal Let Σ^* be an optimal solution to (SDP) and $K^* = \mathcal{K}(\Sigma^*)$, and let $\pi(\mathbf{x}) = K^*\mathbf{x}$ be the policy induced by K^*. Furthermore, let π^* be an optimal solution to (CLQG).

Note that $J^g(\pi) = \begin{bmatrix} Q^f & 0 \\ 0 & R^f \end{bmatrix} \bullet \mathcal{E}(K^*) \leq b$, as Σ^* is feasible to (SDP). Therefore, π satisfies the cost constraint of (CLQG).

Now, it is sufficient to show its optimality, i.e., $J^f(\pi) \leq J^f(\pi^*)$. By Lemma 1, $\mathcal{E}(\pi^*)$ is an feasible solution to (SDP). Furthermore, by step 2, we have,

$$\mathcal{E}(K^*) \preceq \mathcal{E}(\pi^*).$$

It follows that

$$J^f(K^*) = \begin{bmatrix} Q^f & 0 \\ 0 & R^f \end{bmatrix} \bullet \mathcal{E}(K^*) \leq \begin{bmatrix} Q^f & 0 \\ 0 & R^f \end{bmatrix} \bullet \mathcal{E}(\pi^*) = J^f(\pi^*).$$

Finally, we conclude the proof. $\square$

B Detailed Parameters

$$A = \begin{bmatrix} 0.12203823 & 0.49517691 & 0.03438852 & 0.90932040 \\ 0.25877998 & 0.66252228 & 0.31171108 & 0.52006802 \\ 0.54671028 & 0.18485446 & 0.96958463 & 0.77513282 \\ 0.93949894 & 0.89482735 & 0.59789998 & 0.92187424 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.08849250 & 0.19598286 \\ 0.04522729 & 0.32533033 \\ 0.38867729 & 0.27134903 \\ 0.82873751 & 0.35675333 \end{bmatrix}$$

$$Q_f = \begin{bmatrix} 0.98431914 & 0.55336647 & 0.66655448 & 0.71555056 \\ 0.55336647 & 0.76577354 & 0.38307809 & 0.53925763 \\ 0.66655448 & 0.38307809 & 0.63036352 & 0.57586741 \\ 0.71555056 & 0.53925763 & 0.57586741 & 0.79318353 \end{bmatrix}$$

$$R_f = \begin{bmatrix} 0.58490153 & 0.47835073 \\ 0.47835073 & 0.57188842 \end{bmatrix}$$

$$Q_g = \begin{bmatrix} 0.89012261 & 0.29778192 & 0.44227961 & 0.21570672 \\ 0.29778192 & 1.06344568 & 0.12306210 & 0.73155999 \\ 0.44227961 & 0.12306210 & 0.88581457 & 0.56807808 \\ 0.21570672 & 0.73155999 & 0.56807808 & 1.08666707 \end{bmatrix}$$

$$R_g = \begin{bmatrix} 0.42901345 & 0.39095257 \\ 0.39095257 & 0.56455217 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

$$b = 0.9385$$