Linear Quadratic Control Using Model-Free Reinforcement Learning

Farnaz Adib Yaghmaie, Fredrik Gustafsson and Lennart Ljung

Adib Yaghmaie, F., Gustafsson, F., Ljung, L., (2023), Linear Quadratic Control Using Model-Free Reinforcement Learning, *IEEE Transactions on Automatic Control*, 68(2), 737-752.
https://doi.org/10.1109/TAC.2022.3145632

Linear Quadratic Control using Model-free Reinforcement Learning

Farnaz Adib Yaghmaie, Fredrik Gustafsson, *Fellow, IEEE*, and Lennart Ljung, *Fellow, IEEE*

***Abstract*— In this paper, we consider Linear Quadratic (LQ) control problem with process and measurement noises. We analyze the LQ problem in terms of the average cost and the structure of the value function. We assume that the dynamics of the linear system is unknown and only noisy measurements of the state variable are available. Using noisy measurements of the state variable, we propose two model-free iterative algorithms to solve the LQ problem. The proposed algorithms are variants of policy iteration routine where the policy is greedy with respect to the average of all previous iterations. We rigorously analyze the properties of the proposed algorithms, including stability of the generated controllers and convergence. We analyze the effect of measurement noise on the performance of the proposed algorithms, the classical off-policy, and the classical Q-learning routines. We also propose a model-building approach where a model of the dynamical system is estimated and the optimal control problem is solved assuming that the estimated model is the true model. We use a benchmark to evaluate and compare our proposed algorithms with the classical off-policy, the classical Q-learning, and the policy gradient. We show that our model-building approach performs nearly identical to the analytical solution and our proposed policy iteration-based algorithms outperform the classical off-policy and the classical Q-learning algorithms on this benchmark but do not outperform the model-building approach.**

***Index Terms*— Linear Quadratic Control, Reinforcement Learning**

I. Introduction

Machine Learning (ML) has surpassed human performance in many challenging tasks like pattern recognition [1] and playing video games [2]. By recent progress in ML, specifically using deep networks, there is a renewed interest in applying ML techniques to control dynamical systems interacting with a physical environment [3], [4] to do more demanding tasks like autonomous driving, agile robotics [5], solving decision-making problems [6], etc.

Adaptive control studies data-driven approaches for control of unknown dynamical systems [7], [8]. If we combine adaptive control with optimal control theory, it is possible to control unknown dynamical systems adaptively and optimally.

Farnaz Adib Yaghmaie is supported by the Vinnova Competence Center LINK-SIC and by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP).

Farnaz Adib Yaghmaie, Fredrik Gustafsson and Lennart Ljung are with Department of Electrical Engineering, Linköping University, Linköping, Sweden (email: email: farnaz.adib.yaghmaie@liu.se, fredrik.gustafsson@liu.se, lennart.ljung@liu.se).

Reinforcement Learning (RL) refers to a class of such routines and its history dates back decades [9], [10]. By recent progress in ML, specifically using deep networks, the RL field is also reinvented. RL algorithms have recently shown impressive performances in many challenging problems including playing Atari games [2], agile robotics [5], control of continuous-time systems [3], [11]–[17], and distributed control of multi-agent systems [18], [19].

In a typical RL setting, the model of the dynamical system is unknown and the optimal controller is learned through interaction with the dynamical system. One possible approach to solve an RL problem is the class of *Dynamic Programming (DP)*-based solutions, also called *Temporal Difference (TD) learning*, where the idea is to learn the value function by minimizing the Bellman error [20]. The celebrated Q-learning algorithm belongs to this class [21]. *Least Squares Temporal Difference learning (LSTD)* [21], [22] refers to the case where the least-squares is used to minimize the TD error. Another class contains *policy search* algorithms where the policy is learned by directly optimizing the performance index. Examples are policy-gradient methods that use sampled returns from the dynamical system [23] and actor-critic methods that use estimated value functions in learning the policy [24]. In the control community, adaptive control is used for optimal control of the unknown system by first estimating (possibly recursively) a model [25], [26] and then, solving the optimal control problem using the estimated model [8]. This category of solutions is called *model-based RL* in the RL community but we coin the term *model-building RL* to emphasize that no known/given model is used or assumed.

As the scope of RL expands to more demanding tasks, it is crucial for the RL algorithms to provide guaranteed stability and performance. Still, we are far away from analyzing the RL algorithms because of the inherent complexity of the tasks and deep networks. This motivates considering a simplified case study where analysis is possible. *Linear Quadratic (LQ)* problem is a classical control problem where the dynamical system obeys linear dynamics and the cost function to be minimized is quadratic. The LQ problem has a celebrated closed-form solution and is an ideal benchmark for studying and comparing the RL algorithms because firstly, it is theoretically tractable and secondly, it is practical in various engineering domains.

Because of the aforementioned reasons, the LQ problem has attracted much attention from the RL community [14], [27]–[29], see [30] for a complete survey on RL approaches and properties for the LQ problems. For example, [15], [16]

TABLE I: Notation and abbreviations.

The subscript k	The time step
The superscript i	The iteration index of algorithm
The superscript $*$	The optimal value of the superscripted quantity
$x_k,\ u_k,\ y_k$	The state, control and output at k
(A, B)	System dynamics
$r_k = y_k^T R_y y_k + u_k^T R_u u_k$	The running cost associated with $y_k,\ u_k$
$w_k,\ v_k,\ \eta_k$	The process, measurement and probing noises at k
$W_w,\ W_v,\ W_\eta$	The covariances of the process, measurement and probing noises
$K^i,\ L^i = A + BK^i,\ \lambda^i$	The policy gain at iteration i, the closed-loop dynamics and the average cost using K^i
$V^i,\ P^i$	The value function and its quadratic kernel associated with K^i
$Q^i,\ G^i$	The quality function and its quadratic kernel associated with K^i
$\hat{V}^i,\ \hat{Q}^i,\ \hat{P}^i,\ \hat{G}^i,\ \bar{\lambda}^i$	The estimations of $V^i,\ Q^i,\ P^i,\ G^i$ and the empirical estimation of λ^i
$I,\ \mathbf{0}$	The identity and zero matrices with appropriate dimensions
$P > 0 (P \geq 0)$	$P \in \mathbb{R}^{n\times n}$ is (semi) positive definite
$\mu(P),\ \underline{\mu}(P),\ \rho(P),\ \mathrm{Tr}(P)$	An eigenvalue, the minimum eigenvalue, the spectral radius and the trace of $P \in \mathbb{R}^{n\times n}$
$\mathrm{vec}(A) = [a_1^T, a_2^T, ..., a_m^T]^T$	The vectorization of matrix $A = [a_1, ..., a_m],\ a_i \in \mathbb{R}^n,\ i = 1, ..., m$
$\mathrm{vecs}(P) = [p_{11}, ..., p_{1n}, p_{22}\ , ..., p_{2n}, ..., p_{nn}]^T$	The vectorization of the upper-triangular part of a symmetric matrix $P \in \mathbb{R}^{n\times n}$
$\mathrm{vecv}(v) = [v_1^2, 2v_1v_2, ..., 2v_1v_n,\ v_2^2, ...,\ 2v_2v_n,\ ..., v_n^2]^T$	The quadratic vector of the vector $v \in \mathbb{R}^n$
$\otimes$	The Kronecker product

study the *Linear Quadratic Regulator (LQR)* control problems in completely deterministic setups via LSTD. Global convergence of policy gradient methods for deterministic LQR problem is shown in [31]. Considering a stochastic setup is more demanding [14], [27]–[29]. In [14], a stochastic LQ problem with state- and control-dependent process noise is considered. The authors assume that the noise is measurable and propose an LSTD approach to solve the optimal control problem. In [27], the authors give LSTD algorithms for the LQ problem with process noise and analyze the regret bound, i.e. the difference between the occurred cost and the optimal cost. The regret bound is also studied in [32] for a model-building routine. The sample complexity of the LSTD for the LQ problem is studied in [29] and the gap between a model-building approach and a model-free routine is analyzed in [33]. In [28], a model-building approach, called coarse ID control, is developed to solve the LQ problem with process noise and sample complexity bounds are derived. The idea of coarse-ID control is to estimate the linear model and the uncertainty, and then to solve the optimal control problem using the information of the model and uncertainty. A convex procedure to design a robust controller for the LQ problem with unknown dynamics is suggested in [34]. The total cost to be optimized in the LQ problem or in general in RL can be considered as the average cost [33] or discounted cost [16], [29] and it is a design parameter. If the cost is equally important at all times, for example, the cost is energy consumption, one may consider an average cost setting. If the immediate costs are more important, one may consider a discounted cost. In a discounted setting, one needs to select the discounting factor carefully to prevent loss of stability while this can be avoided by considering the average cost.

In all the aforementioned results, only process noise is considered in the problem formulation. In practice, both process and measurement noises appear in the dynamical systems and one should carefully consider the effect of measurement noise. This is particularly important in feedback controller design, where only a noisy measurement of the output is available for the control purpose. When a model-free approach is used to control the system, it is not possible to use a filter, e.g. a Kalman filter, to estimate the state variable because the dynamics of the system is unknown. Reference [35] considers both process and measurement noises in the problem formulation but only stability of the generated policy is established and the effect of measurement noise is not studied. A closely related topic is RL for Partial Observable Markov Decision Processes (POMDP), where the state is only partially observable [36]–[38]. The references [36], [37] consider LQ systems with both process and observation noises and propose algorithms to estimate the model of the dynamical system and to learn a near optimal controller via gradient descent. [38] considers noise-free LQ problem and proposes to use a history of input-output data in the controller design.

In this paper, we focus on Dynamic Programming-based RL routines to solve LQ problem. There is much research on RL algorithms for LQ problem assuming that the state variable is exactly measurable [15], [16], [27], [29]. We take one step further and replace the state with a noisy measurement of the state; i.e. the output. There are two contributions to this paper. Our first contribution is to study the LQ problem in terms of the average cost and the structure of the value function. Our second contribution is to propose two model-free Dynamic Programming-based RL algorithms and analyze their properties. We study the effect of measurement noise in the proposed algorithms and the classical off-policy and the classical Q-learning routines. Such a discussion is absent in [35]. Moreover, we provide the convergence proof for the proposed algorithms while such a discussion in a simpler setup in [27] is missing. We also propose a model-building RL which can be easily extended to cover the partially observable dynamical systems. We leave the analysis of such model-building approaches for our future research. We would like to stress that the solutions we discuss (including the model-building one) are all in the model-free family of classical RL. The only information about the data generation that is used is that the outputs come from a linear system of known order; no known model is assumed to generate the data.

The rest of the paper is organized as follows. In Section

II, we define the stochastic linear quadratic problem and discuss the analytical model-based solution. In Section III, we define the value and quality functions for the LQ problem. In Section IV, we present two model-free algorithms to solve the LQ problem. In Section V, we analyze the properties of the proposed algorithms like stability of the generated controllers, convergence, and the bias in the estimations. Anyone not interested in the theoretical details can skip this section. In Section VI, we give the simulation result and compare our proposed algorithms with the classical model-free RL approaches and the model-building approach. In Section VII, we conclude the paper. To be self-contained, we bring the classical off-policy learning, the classical Q-learning and the model-building approach in Appendix I. The proofs of all theorems and lemmas are given in Appendix II. We have summarized the notations and abbreviations in Table I. The reader may refer to this table to find the frequently used notations.

II. Linear Quadratic Problem

A. Linear Gaussian dynamical systems

Consider a linear Gaussian dynamical system

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{1}$$
$$y_k = x_k + v_k, \tag{2}$$

where $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ are the state and the control input vectors respectively. The vector $w_k \in \mathbb{R}^n$ denotes the process noise drawn i.i.d. from a Gaussian distribution $\mathcal{N}(\mathbf{0}, W_w)$. The state variable x_k is not measurable and the output $y_k \in \mathbb{R}^n$ in (2) denotes a noisy measurement of the state x_k where $v_k \in \mathbb{R}^n$ is the measurement noise drawn i.i.d. from a Gaussian distribution $\mathcal{N}(\mathbf{0}, W_v)$ where W_v is diagonal.

The model (A, B) is unknown but stabilizable. The model order n is assumed to be known. The measurements y_k, u_k can be used to form the next output in order to achieve a specific goal of the control. When the control input u_k is selected as a function of the output y_k, it is called a policy. In this paper, we design stationary policies of the form $u_k = Ky_k$ to control the system in (1)-(2). Let $L := A + BK$. The policy gain K is stabilizing if $\rho(L) < 1$.

B. The Linear quadratic optimal control problem

We define the *quadratic running cost* as

$$r(y_k, u_k) = r_k = y_k^T R_y y_k + u_k^T R_u u_k \tag{3}$$

where $R_y \geq 0$ and $R_u > 0$ are the output and the control weighting matrices respectively. The *average cost associated with the policy* $u_k = Ky_k$ is defined by

$$\lambda(K) = \lim_{\tau \to \infty} \frac{1}{\tau} \mathbf{E}[\sum_{t=1}^{\tau} r(y_t, Ky_t)] \tag{4}$$

which does not depend on the initial state of the system [9]. For the dynamical system in (1)-(2), we define the *value function* [9] (or bias function [39]) associated with a given policy $u_k = Ky_k$

$$V(y_k, K) = \mathbf{E}[\sum_{t=k}^{+\infty} (r(y_t, Ky_t) - \lambda(K)) | y_k] \tag{5}$$

where the expectation is taken with respect to w_k, v_k.

Problem 1: Consider the dynamical system in (1)-(2). Design the gain K^* such that the policy $K^* y_k$ minimizes (5).

In this paper, we are interested in optimizing (5). The value function (5) quantifies the quality of transient response using $u_k = Ky_k$. Because of the presence of stationary process and measurement noises in (1)-(2), the controller $u_k = Ky_k$ results in a permanent average cost which is represented by $\lambda(K)$. In the next section, we will show that $\lambda(K)$ is a function of W_w, W_v and has a nonzero value if the process noise or the measurement noise, or both are present. By subtracting $\lambda(K)$ in (5), we end up with the transient cost or the controller cost. We will also prove in Lemma 1 that minimizing $V(y_k, K)$ is equivalent to an LQR problem and can be solved through an *Algebraic Riccati Equation (ARE)*.

Another relevant problem is the classical *Linear Quadratic Gaussian (LQG)* problem which optimizes the average cost (4):

Problem 2 (LQG): Consider the dynamical system in (1)-(2). Design the controller $u_k(y_1, ..., y_k)$ to minimize (4).

The optimal controller for Problem 2 is $u_k = K_{\text{LQR}} \hat{x}_{k|k}$ where $\hat{x}_{k|k}$ is a posteriori state estimation at time k given observations $(y_1, ..., y_k)$ [8]. The cost to be minimized in the LQG problem contains two parts: 1- The Linear Quadratic Regulator (LQR) cost and 2- the mean-square state estimation error. The LQR cost measures the quality of transient response. The K_{LQR} minimizes the LQR cost and it is obtained from ARE using (A, B, R_y, R_u). The mean-square error of the state estimation $\hat{x}_{k|k}$ is minimized by the means of a Kalman Filter where the Kalman gain depends on (A, B, W_w, W_v). We will show in Lemma 1 that solving Problem 1 is equivalent to solving the LQR problem with $K^* \equiv K_{\text{LQR}}$.

In the next result, we show that the solution to Problem 1 can be found from a standard ARE machinery if a model of the true system is known. The proof of Lemma 1 is given in Appendix II-A.

Lemma 1: Consider Problem 1 and (1)-(5). The optimal policy gain K^* is

$$K^* = -(R_u + B^T P^* B)^{-1} B^T P^* A \tag{6}$$

where $P^* > 0$ is the solution to the ARE

$$A^T P^* A - P^* - A^T P^* B (B^T P^* B + R_u)^{-1} B^T P^* A + R_y = \mathbf{0}. \tag{7}$$

The average cost associated with K^* is given by

$$\lambda(K^*) = \text{Tr}(K^{*T} B^T P^* B K^* W_v) + \text{Tr}(P^*(W_w + W_v)) - \text{Tr}((A + BK^*)^T P^* (A + BK^*) W_v). \tag{8}$$

Remark 1: Lemma 1 states that Problem 1 is equivalent to the classical LQR problem and the solution can be found from

Algorithm 1 Model-based LQ [40]

1: **Initialize:** Select a stabilizing policy gain K^1, set $i=1$.
2: **for** $i=1,..,\mathbf{N}$ **do**
3: Find P^i from *the model-based Bellman equation*

$$(A+BK^i)^TP^i(A+BK^i)-P^i+R_y + K^{iT}R_uK^i=\mathbf{0}. \quad (10)$$

4: Update the policy gain

$$K^{i+1}=-(R_u+B^TP^iB)^{-1}B^TP^iA. \quad (11)$$

a standard ARE machinery. We explain this point intuitively: The process and measurement noises result in a permanent average cost. We take away the effect of the process and measurement noises by subtracting the average cost in (5). This leaves us with the transient cost or the controller cost which can be optimized through the ARE in (7). Also, note that we can cast solving Problem 1 to minimizing the regret function. The regret framework R^{u_k} evaluates the quality of a policy u_k by comparing its running cost to a baseline b [30]

$$R^{u_k}=\sum_{t=1}^{+\infty} r_k - b. \quad (9)$$

Let $b:=\lambda(K)$ in (4) represent the baseline. Then the regret framework in (9) is indeed equivalent to the value function in (5), and minimizing V with respect to K (Problem 1) is cast to minimizing the regret function.

C. Iterative model-based approach for the LQ problem

According to Lemma 1, the solution to Problem 1 can be found from the ARE in (7). If the dynamics is known, one can use the Hewer's method [40] in Algorithm 1. The algorithm is initiated with a stabilizing gain. In each iteration, the gain is evaluated by solving the model-based Bellman equation in (10). Then, using the solution to the Bellman equation, the gain will be improved. It has been proved in [40] that K^i converges to K_{LQR}.

III. Value and Quality functions

In this paper, we will present two iterative model-free algorithms to solve Problem 1. Our algorithms rely on the evaluation of a policy gain K which is done by finding its associated value function V (defined in Subsection II-B) or quality function Q (to be defined in this section).

A. Value function V

The value function is defined in (5). Using (5), *the value-based Bellman equation* reads

$$V(y_k,K)=r(y_k,Ky_k)-\lambda(K)+\mathbf{E}[V(y_{k+1},K)|y_k]. \quad (12)$$

It is straight forward to show that the value function associated with a linear policy is quadratic and it is equivalent to the model-based Bellman equation (10):

Lemma 2 ([35]): Consider (1)-(5). Assume that the policy gain K is stabilizing. Then, $V(y_k,K)=y_k^TPy_k$ where $P>0$ satisfies

$$(A+BK)^TP(A+BK)-P+R_y+K^TR_uK=\mathbf{0} \quad (13)$$

and the average cost is given by

$$\lambda(K)=\text{Tr}(K^TB^TPBKW_v)+\text{Tr}(P(W_w+W_v)) - \text{Tr}((A+BK)^TP(A+BK)W_v). \quad (14)$$

Proof: See the proof of Lemma 1 in [35]. ■

B. Quality Q function

Along with the value function (5), it is also handy to define the *Quality (Q) function*. Let $z_k=[y_k^T,u_k^T]^T$. The Q function is equal to the expected return cost for taking an action u_k at time k and then following the existing policy Ky_k

$$Q(y_k,u_k,K)=r(y_k,u_k)-\lambda(K)+\mathbf{E}[V(y_{k+1},K)|z_k]. \quad (15)$$

Note that (15) is indeed *the quality-based Bellman equation*. The value function $V(y_k,K)$ can be obtained from $Q(y_k,u_k,K)$ by selecting $u_k=Ky_k$

$$V(y_k,K)=Q(y_k,Ky_k,K). \quad (16)$$

Lemma 3: Consider (1)-(5) and (15). Assume that the policy gain K is stabilizing. Then, $Q(z_k,K)=z_k^TGz_k$ where $G\geq 0$ is a solution to

$$\begin{bmatrix}A^T\\B^T\end{bmatrix}\begin{bmatrix}I & K^T\end{bmatrix}G\begin{bmatrix}I\\K\end{bmatrix}\begin{bmatrix}A & B\end{bmatrix}-G+\begin{bmatrix}R_y & 0\\0 & R_u\end{bmatrix}=\mathbf{0}, \quad (17)$$

and the average cost is given by

$$\begin{aligned}\lambda(K)=&\text{Tr}(K^TB^T\begin{bmatrix}I & K^T\end{bmatrix}G\begin{bmatrix}I\\K\end{bmatrix}BKW_v)\\&+\text{Tr}(\begin{bmatrix}I & K^T\end{bmatrix}G\begin{bmatrix}I\\K\end{bmatrix}(W_w+W_v))\\&-\text{Tr}(L^T\begin{bmatrix}I & K^T\end{bmatrix}G\begin{bmatrix}I\\K\end{bmatrix}LW_v).\end{aligned} \quad (18)$$

Lemma 4: Consider (1)-(5) and (15). Assume that the policy gain K is stabilizing. Then, the quality-based Bellman equation (15) is equivalent to the model-based Bellman equation (10); that is the solutions are the same.
The proofs of Lemmas 3-4 are given in Appendices II-B-II-C.

IV. Proposed model-free algorithms

In this section, we present two model-free iterative algorithms to solve Problem 1. At a high level, each of these algorithms is a variant of policy iteration routine. Each algorithm iterates $\mathbf{N}$ times over two steps: 1- policy evaluation and 2- policy improvement. The policy is evaluated by finding the associated value function or Q function, and the average cost. The improved policy is then selected to be greedy with respect to the average of all previously estimated value or Q functions. The number of iterations $\mathbf{N}$ is a design variable, often a small number.

Algorithm 2 Average Off-policy Learning

1: **Initialize:** Select a stabilizing policy gain K^1, set $i=1$.
2: **for** $i=1,..,\mathbf{N}$ **do**
3: Execute $K^i y_k$ for τ rounds and estimate $\hat{P}^i$, $\bar{\lambda}^i$ from (21)-(22).
4: $\mathcal{Z}=$`CollectData`$(K^i,\tau',\tau'',W_\eta)$.
5: Estimate $\hat{\xi}^i$ by (27) using $\mathcal{Z}$.
6: Update the policy gain K^{i+1} by (28).

Algorithm 3 `CollectData`

Input: K, τ', τ'', W_η.
Output: $\mathcal{Z}$.
$\mathcal{Z}=\{\}$.
for $t=1,..,\tau'$ **do**
Execute Ky for τ'' rounds and observe y.
Sample $\eta\sim\mathcal{N}(\mathbf{0},W_\eta)$ and set $u=Ky+\eta$.
Take u and observe y_+.
Add (y,u,y_+) to $\mathcal{Z}$.

A. Average off-policy learning

Our first routine is called *average off-policy learning* in Algorithm 2. This routine is given in [35] but its properties are not discussed. We analyze the properties in Subsection V-A.

The algorithm is initiated with a stabilizing policy gain K^1 in **Line 1**. Then, the algorithm iterates **N** times in **Line 2** over the policy evaluation and the policy improvement steps. The policy evaluation step is given in **Line 3** and discussed in Subsubsection IV-A.1. The policy gain K^i is evaluated by estimating its associated value function and average cost. The policy improvement step is given in **Lines 4-6** and discussed in Subsubsection IV-A.2.

1) Policy evaluation:

Line 3: Considering $V^i(y_k,K^i)=y_k^TP^iy_k$, the value-based Bellman equation (12) reads

$$r_k-\lambda^i=(\Phi_k-\Phi_{k+1})^T\text{vecs}(P^i)+\mathfrak{n}_1 \tag{19}$$

where

$$\begin{aligned}&\Phi_k=\text{vecv}(y_k),\quad r_k=r(y_k,K^iy_k),\\&\mathfrak{n}_1=y_{k+1}^TP^iy_{k+1}-\mathbf{E}[y_{k+1}^TP^iy_{k+1}|y_k].\end{aligned} \tag{20}$$

To estimate P^i and λ^i, we execute K^iy_k and collect τ samples of the output y_k. We use the empirical average cost from τ samples, as an estimate of λ^i

$$\bar{\lambda}^i=\frac{1}{\tau}\sum_{t=1}^{\tau}r_t. \tag{21}$$

The average-cost LSTD estimator of P^i is given by [39]

$$\text{vecs}(\hat{P}^i)=(\sum_{t=1}^{\tau}\Phi_t(\Phi_t-\Phi_{t+1})^T)^{-1}(\sum_{t=1}^{\tau}\Phi_t(r_t-\bar{\lambda}^i)). \tag{22}$$

2) Policy improvement: We find the improved policy gain by the concept of off-policy learning. The idea is to apply a behavioral policy u_k to the system while learning the target (or optimal) policy K^iy_k. Note that the behavioral policy should be different from the target policy. Define the behavioral policy as $u_k=K^iy_k+\eta_k$ where we sample η_k from $\mathcal{N}(\mathbf{0},W_\eta)$ at each time k and W_η is the covariance of probing. Using the behavioral policy, the closed-loop system of (1) reads

$$\begin{aligned}y_{k+1}=&Ax_k+Bu_k+w_k+v_{k+1}\\=&L^ix_k+B(u_k-K^iy_k)+BK^iv_k+w_k+v_{k+1}.\end{aligned} \tag{23}$$

Using the above equation, we can obtain *the off-policy Bellman equation*

$$\begin{aligned}&y_k^T(R_y+K^{iT}R_uK^i-P^i)y_k+\mathbf{E}[y_{k+1}^TP^iy_{k+1}|y_k]-\lambda^i\\&=2(u_k-K^iy_k)^TB^TP^iAy_k\\&+(u_k+K^iy_k)^TB^TP^iB(u_k-K^iy_k).\end{aligned} \tag{24}$$

We will show in Theorem 1 that (24) is equivalent to the model-based Bellman equation (10). We can write (24) as a linear regression

$$c_k=\Upsilon_k^T\xi^i+\mathfrak{n}_2 \tag{25}$$

where

$$\begin{aligned}&c_k=y_k^T(R_y+K^{iT}R_uK^i-\hat{P}^i)y_k+y_{k+1}^T\hat{P}^iy_{k+1}-\bar{\lambda}^i,\\&\Upsilon_k=\begin{bmatrix}2y_k\otimes(u_k-K^iy_k)\\\text{vecv}(u_k)-\text{vecv}(K^iy_k)\end{bmatrix},\\&\xi^i=\begin{bmatrix}\text{vec}(B^TP^iA)\\\text{vecs}(B^TP^iB)\end{bmatrix},\\&\mathfrak{n}_2=y_{k+1}^T\hat{P}^iy_{k+1}-\mathbf{E}[y_{k+1}^TP^iy_{k+1}|y_k]+y_k^TP^iy_k-y_k^T\hat{P}^iy_k.\end{aligned} \tag{26}$$

In (25), ξ^i is the parameter vector to be estimated and c_k and Υ_k are known using the collected data.

Line 4: To estimate ξ^i, we run the `CollectData` routine in Algorithm 3 to collect τ' sample points. The procedure is as follows: First, the policy K^iy_k is applied for τ'' rounds to ensure that $\mathbf{E}[x_k]=\mathbf{0}$ and the output y_k is sampled. Then, we sample a random action η_k from $\eta_k\sim\mathcal{N}(\mathbf{0},W_\eta)$. We apply the action $u_k=K^iy_k+\eta_k$ and sample the next output y_{k+1}. This builds one data point (y_k,u_k,y_{k+1}). We repeat this procedure and collect τ' data points.

Line 5: The estimation of ξ^i is given by

$$\hat{\xi}^i=(\sum_{t=1}^{\tau'}\Upsilon_t\Upsilon_t^T)^{-1}(\sum_{t=1}^{\tau'}\Upsilon_tc_t). \tag{27}$$

Line 6: Using $\hat{\xi}^i$, we now obtain the improved policy. Let $H^i:=B^TP^iA$ and $N^i:=B^TP^iB$, and let $\hat{H}^i$ and $\hat{N}^i$ be their estimates using (27) (see (26)-(27)). The improved policy is selected to be greedy with respect to the average of all previously estimated value functions

$$K^{i+1}=-(\sum_{j=1}^{i}(\hat{N}^j+R_u))^{-1}(\sum_{j=1}^{i}\hat{H}^j). \tag{28}$$

B. Average Q-learning

Our second routine is called *average Q-learning* in Algorithm 4. Similar to the average off-policy learning in Subsection IV-A, the average Q-learning is a policy iteration routine. Different from the average off-policy learning, the policy is evaluated by finding the Q function. The algorithm is an extended version of the Model-Free Linear Quadratic control (MFLQ) in [27] to accommodate the effect of observation noise. In [27], only process noise is present and the average cost is parameterized based on the covariance of the process noise. Such an approach is not possible here because the average cost depends on the process and observation noises, and the dynamics of the system. Note that there is no proof of convergence for the MFLQ algorithm in [27]. We give a proof of convergence for Algorithm 4 in Subsection V-B which is also applicable to the MFLQ algorithm in [27].

The algorithm is initiated with a stabilizing policy gain K^1 in **Line 1**. Then, the algorithm iterates $\mathbf{N}$ times in **Line 2** over the policy evaluation and the policy improvement steps. The policy evaluation step is given in **Lines 3-5** and discussed in Subsubsection IV-B.1. The policy gain K^i is evaluated by estimating its associated Q function and average cost. The policy improvement step is given in **Line 6** and discussed in Subsubsection IV-B.2.

1) Policy evaluation: We evaluate the policy gain K^i by finding the quadratic kernel G^i of the Q function and the average cost λ^i.

Line 3: We execute $K^i y_k$ and collect τ samples of the output y_k and then we estimated P^i and λ^i from (21)-(22).

Line 4: We run `CollectData` routine in Algorithm 3 to collect τ' sample points.

Line 5: Considering $Q^i(z_k, K^i) = z_k^T G^i z_k$, the quality-based Bellman equation (15) can be written as

$$c'_k = \Psi_k^T \text{vecs}(G^i) + \mathfrak{n}_3 \tag{29}$$

where

$$\begin{aligned} c'_k &= r(y_k, u_k) - \bar{\lambda}^i + y_{k+1}^T \hat{P}^i y_{k+1}, \\ \Psi_k &= \text{vecv}(z_k), \quad z_k = [y_k^T, u_k^T]^T, \\ \mathfrak{n}_3 &= y_{k+1}^T \hat{P}^i y_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | z_k]. \end{aligned} \tag{30}$$

We estimate G^i using τ' sample points collected in **Line 5**

$$\text{vecs}(\hat{G}^i) = (\sum_{t=1}^{\tau'} \Psi_t \Psi_t^T)^{-1} (\sum_{t=1}^{\tau'} \Psi_t c'_t). \tag{31}$$

2) Policy improvement:

Line 6: Let partition matrix $\hat{G}^i$ as

$$\hat{G}^i = \begin{bmatrix} \hat{G}^i_{11} & \hat{G}^i_{12} \\ \hat{G}^{iT}_{12} & \hat{G}^i_{22} \end{bmatrix} \tag{32}$$

such that $\hat{G}^i_{11} \in \mathbb{R}^{n\times n}$, $\hat{G}^i_{12} \in \mathbb{R}^{n\times m}$, $\hat{G}^i_{22} \in \mathbb{R}^{m\times m}$. The improved policy is selected to be greedy with respect to the average of all previously estimated Q functions

$$K^{i+1} = \arg\min_a \frac{1}{i} \sum_{j=1}^{i} \hat{Q}^j(y_k, a) = \sum_{j=1}^{i} -(\hat{G}^j_{22})^{-1} \hat{G}^{jT}_{12}. \tag{33}$$

Algorithm 4 Average Q-learning

1: **Initialize:** Select a stabilizing policy gain K^1, set $i = 1$.
2: **for** $i = 1, .., \mathbf{N}$ **do**
3: Execute $K^i y_k$ for τ rounds and estimate $\hat{P}^i$, $\bar{\lambda}^i$ from (21)-(22).
4: $\mathcal{Z}$ =`CollectData`$(K^i, \tau', \tau'', W_\eta)$.
5: Estimate $\hat{G}^i$ from (31) using $\mathcal{Z}$.
6: Update the policy gain K^{i+1} by (33).

Remark 2: In Appendices I-A-I-B, we bring two classical model-free algorithms, namely the classical off-policy learning (Algorithm 5) and Q-leaning (Algorithm 6). We also give a model-building approach (Algorithm 8) using the same information as the RL schemes in Appendix I-C.

V. Analysis of the Proposed Algorithms

The aim of this section is to analyze properties of the proposed algorithms and compare them with the classical off-policy and the classical Q-learning routines. The reader may refer to Appendix I for overviews of the classical off-policy, the classical Q-learning and the model-building approaches.

There are two main differences between our proposed routines in Algorithms 2 and 4, and the classical off-policy and the classical Q-learning routines in Algorithms 5 and 6: 1) We use Algorithm 3 to collect data for estimation. The main feature of this algorithm is that it runs the dynamics for τ'' steps before collecting a sample data point. It helps us to have independent sample points and $\mathbf{E}[x_k] = \mathbf{0}$. 2) The policy is greedy with respect to the average of all previously estimated value functions rather than the last value function. This causes the gain to adapt slowly towards its optimal value. However if the estimation of the value function is poor (so it is away from its correct value), it helps us to avoid getting an unstable controller gain. This feature has a positive effect when the trajectory length to estimate the value or Q function is short.

A. Properties of Average off-policy learning

Theorem 1 ([35]): The off-policy Bellman equation (24) is equivalent to the model-based Bellman equation (10).

Theorem 2 ([35]): Assume that the estimation errors in (22) and (27) are small enough. Then, Algorithm 2 produces stabilizing policy gains $K^{i+1}, i = 2..., \mathbf{N}$.

Theorem 3: Assume that the estimation errors in (22) and (27) are small enough. Then, the sequence of P^i, $i = 1, ..., \mathbf{N}$ associated with the controller gain K^i, $i = 1, ..., \mathbf{N}$ generated in Algorithm 2 is converging $P^* \leq P^{i+1} \leq P^i \leq P^1$.

The off-policy Bellman equation is given in (24). In both the average off-policy learning and the classical off-policy learning, the solution to this equation is found by estimating P^i, $B^T P^i A$ and $B^T P^i B$. In the classical off-policy learning P^i, $B^T P^i A$ and $B^T P^i B$ are estimated at the same time, see (35)-(36). In the average off-policy learning, we solve (24) by first estimating the quadratic kernel of the value function; i.e. P^i and then, the matrices $B^T P^i A$ and $B^T P^i B$ are estimated by running the `CollectData` routine in Algorithm 3. In the sequel, we show that the estimation of P^i in (22) is biased and

as a result, the average off-policy learning routine in Algorithm 2 is biased. But if this bias in the estimation of P^i is near zero, the average off-policy learning returns an unbiased estimate.

Lemma 5: Consider Problem 1 and the estimation problem in (19)-(22). The LSTD estimate of $\hat{P}^i$ in (22) is biased by an amount that is related to the correlation between the noise $\mathfrak{n}_1$ in (20) and the square of y_k.

Theorem 4: Consider Problem 1 and the estimation problem in (25)-(27). Assume that we run Algorithm 3 to collect data to find the estimated solution $\hat{\xi}^i$ in (27). The estimate $\hat{\xi}^i$ in (27) in the average off-policy learning is biased and the bias in $\hat{\xi}^i$ is proportional to the bias in $\hat{P}^i$ in (22). If the bias in $\hat{P}^i$ is negligible, the estimate $\hat{\xi}^i$ is unbiased.

Lemma 6: Consider Problem 1 and the estimation problem in (34)-(36). The estimated solution ϖ^i in (36) in the classical off-policy learning is biased by an amount that is related to the correlation between $\mathfrak{n}_1$ and y_k, and the correlation between $\mathfrak{n}_1$ and the square of y_k where $\mathfrak{n}_1$ is the noise in (35).
The proofs of Theorems 3-4 and Lemmas 5-6 are given in Appendices II-D-II-G.

B. Properties of Average Q-learning

Based on Lemma 4 and Theorems 1-3, the following corollaries are concluded immediately.

Corollary 1: Assume that the estimation errors in (22) and (31) are small enough. Then, Algorithm 4 produces stabilizing policy gains $K^{i+1}, i = 2..., \mathbf{N}$.

Corollary 2: Assume that the estimation errors in (22) and (31) are small enough. Then, the sequence of P^i, $i = 1, ..., \mathbf{N}$ associated with the controller gain K^i, $i = 1, ..., \mathbf{N}$ generated in Algorithm 4 is converging $P^* \leq P^{i+1} \leq P^i \leq P^1$.

The quality-based Bellman equation is given in (15). In the average Q-learning, we solve this equation by first estimating P^i and then, estimating G^i by collecting data according to the `CollectData` routine in Algorithm 3. In comparison, the classical Q-learning algorithm estimates the kernel G^i directly, without going through the estimation of P^i. Both algorithms return biased estimates.

Lemma 7: Consider Problem 1 and the estimation problem in (29)-(31). Assume that we run Algorithm 3 to collect data for the estimation of G^i in (31). The estimate (31) in the average Q-learning is biased by an amount that is related to the average of noise $\mathfrak{n}_3$ in (30), the correlation between $\mathfrak{n}_3$ and y_k, and the correlation between $\mathfrak{n}_3$ and the square of y_k.

Lemma 8: Consider Problem 1 and the estimation problem in (38)-(40). The estimated solution G^i in (40) in the classical Q-learning is biased by an amount that is related to the correlation between $\mathfrak{n}_4$ and y_k, and the correlation between $\mathfrak{n}_4$ and the square of y_k where $\mathfrak{n}_4$ is the noise in (39).

Remark 3: It is not possible to compare the bias terms in the average Q-learning and the classical Q learning, as they contain different terms that are not comparable.

Remark 4: Setting $v_k \equiv \mathbf{0}$ and $W_v \equiv \mathbf{0}$ in the proofs of Lemmas 5-8 and Theorem 4, it is easy to verify that in the absence of measurement noise, all estimates in Algorithms 2, 4, 5, 6 are unbiased.
The proofs of Lemmas 7-8 are given in Appendices II-H-II-I.

C. Computational complexity

TABLE II: Computational complexity in each iteration of Algorithms 2, 4-6, and 8. d : the number of parameters to be estimated, n, m : the dimensions of the state and the input, τ : the rollout length, and τ' : the exploration length in `CollectData` in Algorithm 3 (only for the average off-policy and the average Q-learning routines).† The (n^2) parameters in the model-building approach are only estimated if the open-loop system is unstable.

Algorithm 2- The average off-policy learning

d :	$(n+m)(n+m+1)/2$
Complexity:	$\mathcal{O}(n^4\tau) + \mathcal{O}((m^4 + 4m^2n^2 + 4m^3n)\tau')$

Algorithm 4- The average Q-learning

d :	$(n+m)(n+m+1)/2 + n(n+1)/2$
Complexity:	$\mathcal{O}(n^4\tau) + \mathcal{O}((n+m)^4\tau')$

Algorithm 5- The classical off-policy learning in App. I-A

d :	$(n+m)(n+m+1)/2$
Complexity:	$\mathcal{O}((n+m)^4\tau)$

Algorithm 6- The classical Q-learning in App. I-B

d :	$(n+m)(n+m+1)/2$
Complexity:	$\mathcal{O}((n+m)^4\tau)$

Algorithm 8- The model-building approach in App. I-C

d :	$n^2 + nm + (n^2)^\dagger$

Let d denote the number of parameters to be estimated. In the average off-policy, the classical off-policy and the classical Q-learning, we estimate the same number of parameters $d = (n+m)(n+m+1)/2$ but we need to estimate $n(n+1)/2$ more parameters in the average Q-learning, so $d = (n+m)(n+m+1)/2 + n(n+1)/2$. In the model-building approach, we need to estimate $n^2 + nm$ parameters in (A, B). If the open-loop system is unstable, it is also required to estimate n^2 parameters of the so-called disturbance model which are denoted by the superscript $\dagger$ in the corresponding cell in Table II.

In each iteration of model-free algorithms, we solve one or more linear regression problems with solutions like (22). Considering d as the number of parameters and τ as the number of samples, the complexity of (22) for $\tau >> d$ is $\mathcal{O}(d^2\tau)$ [41]. Table II summarizes the computational complexities of the estimations in each iteration of the routines where we have simplified the complexity terms and kept the dominant terms only.

For the sake of comparison, assume $\tau' = \tau$. According to Table II, the average off-policy learning has a lower complexity in compared to the classical off-policy and Q-learning algorithms. To see this point, note that the complexity $\mathcal{O}(d^2\tau)$ is quadratic in the number of parameters d. In the average off-policy learning, these $d = (n+m)(n+m+1)/2$ unknown parameters are estimated in two parts: first $n(n+1)/2$ parameters are estimated in (22) and then, $nm + m(m+1)/2$ parameters are estimated in (27). So, the overall complexity of the average off-policy learning is $\mathcal{O}((n(n+1)/2)^2\tau) + \mathcal{O}((mn + m(m+1)/2)^2\tau')$. In the classical off-policy and Q-learning algorithms all $d = (n+m)(n+m+1)/2$ parameters

are estimated at the same time which has the higher complexity $\mathcal{O}(((n+m)(n+m+1)/2)^2\tau)$.

The complexity of the model-building approach can be difficult to establish. For example, the command SSEST in MATLAB initializes the parameter estimation using a nonit-erative subspace approach where the numerical calculations consist of a QR factorization and an SVD (Singular Value Decomposition). It then refines the parameter values iteratively using the prediction error minimization approach (see section 10.7 of [25]). One can also use total least squares [42] to estimates the dynamics [35]. For this, one needs to find Singular Value Decomposition of a matrix of order $\tau \times (n^2 + nm + 1)$. For $\tau >> d$, the complexity of the computation is $\mathcal{O}(2(n^2+mn+1)^2\tau + 11(n^2+mn+1)^3)$ [42].

VI. Simulation Results

We consider an idealized instance of data center cooling with three sources coupled to their own cooling devices with the following dynamics [23], [27], [28]

$$x_{k+1} = \begin{bmatrix} 1.01 & 0.01 & 0 \\ 0.01 & 1.01 & 0.01 \\ 0 & 0.01 & 1.01 \end{bmatrix} x_k + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u_k + w_k,$$
$$y_k = x_k + v_k,$$

and the quadratic running cost

$$r(y_k, u_k) = 0.001 y_k^T y_k + u_k^T u_k.$$

Let $W_w = I$, $W_v = I$. We select the initial state of the system as zero. We set the initial stabilizing policy gain K^1 for all algorithms by solving a modified $ARE(A, B, 200R_y, R_u)$. We set the covariance of the probing noise as $W_\eta = I$. If an algorithm produces an unstable controller gain, we assign an infinite cost to that iteration and algorithm, and restart the system. Let τ and $\mathbf{N}$ denote the rollout length and the number of iterations in Algorithms 2, 4, 5, 6.

Average off-policy learning: In Algorithm 2, we set $\tau' = \tau$ and select $\tau'' = 10$. We select the behavioral policy as $u_k = K^i y_k + \eta_k$ where we sample η_k from $\mathcal{N}(\mathbf{0}, W_\eta)$.

Average Q-learning: In Algorithm 4, we set $\tau' = \tau$ and select $\tau'' = 10$. We select the behavioral policy as $u_k = K^i y_k + \eta_k$ where we sample η_k from $\mathcal{N}(\mathbf{0}, W_\eta)$.

Classical off-policy learning: We run the classical off-policy learning in Algorithm 5. We select the behavioral policy as $u_k = K^i y_k + \eta_k$ where we sample η_k from $\mathcal{N}(\mathbf{0}, W_\eta)$.

Classical Q-learning: We run the classical Q-learning in Algorithm 6. We set $u_k = K^i y_k + \eta_k$ where we sample η_k from $\mathcal{N}(\mathbf{0}, W_\eta)$.

Model-building approach: We run Algorithm 8 for the model-building approach. We collect the input-output data by applying the control signal $u_k = K^i y_k + 10\eta_k$, where we sample η_k from $\mathcal{N}(\mathbf{0}, W_\eta)$. We estimate the matrices $(\hat{A}, \hat{B})$ from the experimental data as explained in Appendix I-C.

Policy gradient: While this paper studies Dynamic Programming-based RL routines, we also evaluate a policy gradient algorithm [23] in our simulation. To be self-contained, we have summarized this routine in Algorithm 9 in Subsection I-D. We consider a multivariate Gaussian distribution for the probability density function of the probabilistic policy with the covariance $0.01I$. In each iteration, we collect 10 mini-batches of trajectories of length $T = 10$, so in each iteration, 10×10 data is used. We update the controller gain using an ADAM optimizer with the learning rate as 0.01 and the default values for other hyper parameters. To match the sample budget of Algorithms 2, 4, 5, 6, we set the number of iterations in the policy gradient as $\tau \times \mathbf{N}/100$.

Analytical solution: Our baseline for comparison is the analytical solution assuming that the matrices (A, B) are exactly known. Using the full information of the system dynamics (A, B), we solve $ARE(A, B, R_y, R_u)$ and obtain P^* and K^*. We compute the analytical average cost $\lambda(K^*)$ from (8) using A, B, P^*, K^*, W_v and W_w.

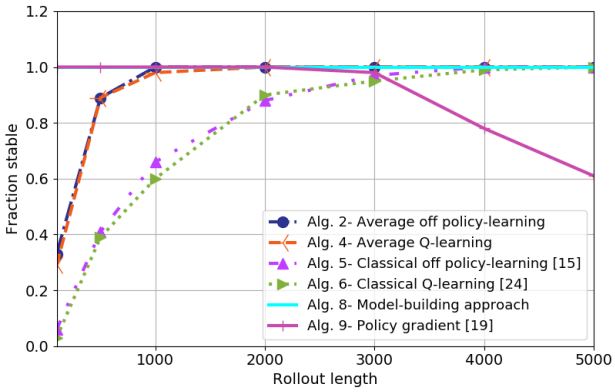

Fig. 1: The fraction of stable policy gains generated by each algorithm in all iterations.

TABLE III: Percentage of stability in all iterations in 100 simulations.

T=	100	500	1000	2000	3000	4000	5000
Alg. 2	33	89	100	100	100	100	100
Alg. 4	29	89	98	100	100	100	100
Alg. 5	6	41	66	88	97	100	100
Alg. 6	3	39	60	90	95	99	100
Alg. 8	100	100	100	100	100	100	100
Alg. 9	100	100	100	100	98	78	61

A. Stability analysis

For Algorithms 2, 4, 5, 6, we set the number of iterations as $\mathbf{N} = 3$ and we change the rollout length $\tau = [100, 500, 1000, 2000, 3000, 4000, 5000]$. In the policy gradient algorithm, we change the number of iterations as $\tau \times 3/100 = [3, 15, 30, 60, 90, 120, 150]$ to match the sample budget of Algorithms 2, 4, 5, 6. We run each algorithm 100 times and in Fig. 1, we show the fraction of times each algorithm produces stable policy gains in all iterations. In Table III, we list the percentage of stability numerically. From this figure, we can see that the model-building approach always produces stable policies. We can also see that the average off-policy learning and the average Q-learning algorithms produce more stable policies among DP-based model-free approaches and they are the most stable model-free routines. This is

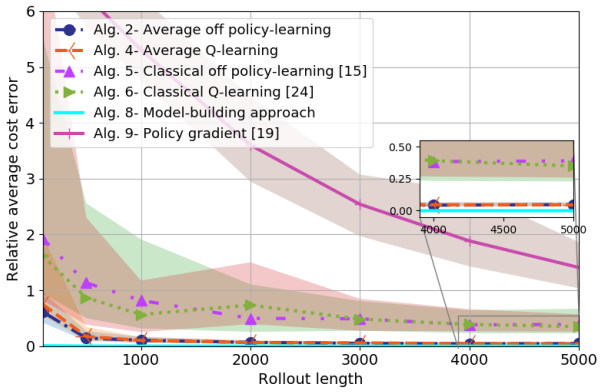

Fig. 2: Median of the relative error $\frac{|\lambda(K)-\lambda(K^*)|}{\lambda(K^*)}$ for 100 stable policies. Shaded regions display quartiles. The numerical values are given in Table IV.

TABLE IV: The median of the relative error $\frac{|\lambda(K)-\lambda(K^*)|}{\lambda(K^*)}$.

T=	100	500	1000	2000	3000	4000	5000
Alg. 2	0.616	0.145	0.106	0.067	0.051	0.043	0.047
Alg. 4	0.751	0.173	0.104	0.064	0.058	0.046	0.045
Alg. 5	1.930	1.134	0.820	0.498	0.491	0.383	0.389
Alg. 6	1.619	0.868	0.566	0.736	0.484	0.390	0.349
Alg. 8	0.006	0.001	0.001	0.000	0.000	0.000	0.000
Alg. 9	7.056	6.280	5.309	3.602	2.541	1.885	1.409

because in the average off-policy and the average Q-learning routines, the policy is greedy with respect to the average of all previously estimated value functions rather than the last value function. This causes the gain to adapt slowly towards its optimal value and helps us to get less unstable gains when the trajectory length to estimate the value or Q function is short. By increasing the rollout length, $\tau \geq 2000$, the average off-policy learning and the average Q-learning routines produce stable policy always while the classical Q-learning and the classical off-policy learning need at least $4000-5000$ samples to produce stable policies always. For the policy gradient algorithm, as more iteration is done, the algorithm returns less stable policies.

B. Performance analysis

Our metric of interest for analyzing the performance is the relative error $\frac{|\lambda(K)-\lambda(K^*)|}{\lambda(K^*)}$ where K is the policy gain obtained from a model-free algorithm or the model-building approach and K^* is the optimal policy gain obtained analytically using full information of the system. We run each algorithm until we obtain 100 stable policies. We set the number of iterations as $\mathbf{N} = 3$ and we change the rollout length $\tau = [100,\ 500,\ 1000,\ 2000,\ 3000,\ 4000,\ 5000]$. In the policy gradient algorithm, we change the number of iterations as $\tau \times 3/100 = [3,\ 15,\ 30,\ 60,\ 90,\ 120,\ 150]$.

In Fig. 2, we plot the median of the relative errors for the aforementioned approaches and in Table IV, we list the median of the relative errors numerically. From this figure, we can see that the model-building approach has the lowest relative error and it is almost identical to the analytical solution. The average off-policy and the average Q-learning routines suffer from $4-5\%$ relative error. The reason is that the estimations of P^i in the average off-policy and G^i in the average Q-learning are biased and they will not improve further by increasing the rollout length. However, the relative errors for the average off-policy and the average Q-learning are much less than those of the classical off-policy and Q-learning algorithms ($\sim 35-40\%$). When τ is small, the average off-policy and the average Q-learning reach their best performance. This is because Algorithm 3 is used to collect data for estimation where the dynamics is run for τ'' steps before collecting a sample data point. It helps us to have independent sample points and $\mathbf{E}[x] = \mathbf{0}$. The policy gradient algorithm performance is worse than other algorithms. It is because the policy gradient algorithms try to directly optimize the performance index and as such, they are very sensitive to noise.

C. Discussion

In summary, one may consider different factors when choosing a model-building or a model-free approach. Our empirical evaluation shows that the model-building solution outperforms the model-free approaches and is the closest one to the analytical solution. This point has also been shown in [27], [28] for the LQ problem with the process noise only (no measurement noise), where it is possible to estimate (A, B) using the ordinary least-squares. Though when the measurement noise is present, this point needs to be proved.

The main difference between the model-building and the model-free approaches comes from the way the data is used. In the model-building approach, the data is used to identify the dynamics of the system while in the model-free approach, to learn the value function and the optimal policy. The main advantage of a model-free approach is to eliminate the system identification part and to directly solve the optimal control problem. Such an approach can be extremely useful for nonlinear systems where it is difficult to identify the dynamics and solve the optimal control problem.

VII. Conclusion and Future Works

In this paper, we have considered the LQ control problem with process and measurement noises. We have assumed that the dynamical model is unknown and developed two model-free DP-based routines (Algorithms 2 and 4) to solve Problem 1. We have proved stability of the generated controllers and convergence of the algorithms. We have shown that the measurement noise can deteriorate the performance of DP-based RL routines. We have also presented a model-building approach in Algorithm 8 as a natural way of solving Problem 1 in the adaptive control context, using the same information from the system as Algorithms 2 and 4. We have used a popular RL benchmark for the LQ problem to evaluate our proposed algorithms. Our empirical evaluation shows that our DP-based algorithms produce more stable policies in comparison with the classical off-policy and the classical Q-learning routines and they are the closest ones to the analytical solution. It has

Algorithm 5 Classical Off-policy Learning

1: **Initialize:** Select a stabilizing policy gain K^1, set $i=1$.
2: **for** $i=1,..,\mathbf{N}$ **do**
3: $\mathcal{Z}=\{\}$.
4: **for** $t=1,...,\tau$ **do**
Sample $\eta_t\sim\mathcal{N}(\mathbf{0},W_\eta)$ and let $u_t=K^iy_t+\eta_t$.
Take u_t and observe y_{t+1}.
Add (y_t,u_t,y_{t+1}) to $\mathcal{Z}$.
5: Estimate $\bar{\lambda}^i$ from (21), and P^i, B^TP^iA, B^TP^iB from (36).
6: Update the policy gain K^{i+1} by (37).

Algorithm 6 Classical Q-policy Learning

1: **Initialize:** Select a stabilizing policy gain K^1, set $i=1$.
2: **for** $i=1,..,\mathbf{N}$ **do**
3: $\mathcal{Z}=\{\}$.
4: **for** $t=1,...,\tau$ **do**
Sample $\eta_t\sim\mathcal{N}(\mathbf{0},W_\eta)$ and let $u_t=K^iy_t+\eta_t$.
Take u_t and observe y_{t+1}.
Add (y_t,u_t,y_{t+1}) to $\mathcal{Z}$.
5: Estimate $\bar{\lambda}^i$ from (21) and $\hat{G}^i$ from (40).
6: Update the policy gain K^{i+1} by (41).

also turned out that our model-building approach outperforms the DP-based solutions which is consistent with the previous results on linear systems with process noise only [23], [28].

Possible future research avenues are to study sample complexity of the proposed methods and extension to nonlinear systems. We are also interested in extending the current results for partially observable LQ problems.

APPENDIX I
CLASSICAL MODEL-FREE ALGORITHMS AND THE MODEL-BUILDING APPROACH

A. Classical off-policy learning

The classical off-policy learning algorithm for deterministic systems is given in [15]. Here, we bring a modified version in Algorithm 5 to accommodate the effect of process and observation noises. The algorithm is initiated with a stabilizing policy gain K^1 in **Line 1**. Then, the algorithm iterates $\mathbf{N}$ times in **Line 2** over the policy evaluation and the policy improvement steps. The policy evaluation step is given in **Lines 3-5** and discussed in Appendix I-A.1. The policy improvement step is given in **Line 6** and discussed in Appendix I-A.2.

1) Policy evaluation:

Lines 3-4: To evaluate the policy gain K^i, we collect τ samples of (y_k,u_k,y_{k+1}) in the following way: we observe y_k and then, we sample η_k from $\mathcal{N}(\mathbf{0},W_\eta)$. We apply the policy $u_k=K^iy_k+\eta_k$ and observe y_{k+1}.

Line 5: We set the average cost λ^i to the empirical average cost from τ samples (21). The off-policy Bellman equation (24) reads

$$r_k-\lambda^i=\Omega_k^T\varpi^i+\mathfrak{n}_1 \tag{34}$$

where

$$\varpi^i=\begin{bmatrix}\text{vecs}(P^i)\\ \text{vec}(B^TP^iA)\\ \text{vecs}(B^TP^iB)\end{bmatrix},\ \Omega_k=\begin{bmatrix}\text{vecv}(y_k)-\text{vecv}(y_{k+1})\\ 2y_k\otimes(u_k-K^iy_k)\\ \text{vecv}(u_k)-\text{vecv}(K^iy_k)\end{bmatrix}$$
$$\Theta_k=\begin{bmatrix}\text{vecv}(y_k)\\ 2y_k\otimes(u_k-K^iy_k)\\ \text{vecv}(u_k)-\text{vecv}(K^iy_k)\end{bmatrix}, \tag{35}$$
$$\mathfrak{n}_1=y_{k+1}^TP^iy_{k+1}-\mathbf{E}[y_{k+1}^TP^iy_{k+1}|y_k].$$

The estimated solution to (24) in the classical off-policy learning using instrumental variable method is given by

$$\hat{\varpi}^i=(\sum_{t=1}^{\tau}\Theta_t\Omega_t^T)^{-1}(\sum_{t=1}^{\tau}\Theta_t(r_t-\bar{\lambda}^i)). \tag{36}$$

2) Policy improvement:

Line 6: Let $H^i:=B^TP^iA$ and $N^i:=B^TP^iB$, and let $\hat{H}^i$ and $\hat{N}^i$ be their estimates using (36). The improved policy is given by

$$K^{i+1}=-(\hat{N}^i+R_u)^{-1}\hat{H}^i. \tag{37}$$

B. Classical Q-learning

We bring the classical Q-learning routine from [29] in Algorithm 6 and modify it to accommodate the effect of process and measurement noises. The algorithm is initiated with a stabilizing policy gain K^1 in **Line 1**. Then, the algorithm iterates $\mathbf{N}$ times in **Line 2** over the policy evaluation and the policy improvement steps. The policy evaluation step is given in **Lines 3-5** and discussed in Appendix I-B.1. The policy improvement step is given in **Line 6** and discussed in Appendix I-B.2.

1) Policy evaluation: The policy gain K^i is evaluated by estimating the quadratic kernel G^i of the Q function.

Lines 3-4: We collect τ samples of (y_k,u_k,y_{k+1}) in the following way: we observe y_k and then, we sample η_k from $\mathcal{N}(\mathbf{0},W_\eta)$. We apply the policy $u_k=K^iy_k+\eta_k$ and observe y_{k+1}.

Line 5: We set λ^i to the empirical average cost from τ samples (21). The quality-based Bellman equation (15) reads

$$r_k-\lambda^i=(\Psi_k-\Psi_{k+1})\text{vecs}(G^i)+\mathfrak{n}_4 \tag{38}$$

where

$$\Psi_k=\text{vecv}(z_k),$$
$$\mathfrak{n}_4=z_{k+1}^TG^iz_{k+1}-\mathbf{E}[z_{k+1}^TG^iz_{k+1}|z_k]. \tag{39}$$

Let $z_k=[y_k^T,u_k^T]$ and $z_{k+1}=[y_{k+1}^T,(K^iy_{k+1}^T)]$. In the classical Q-learning, the LSTD estimator of G^i approximates the solution to the quality-based Bellman equation (15) by

$$\text{vecs}(\hat{G}^i)=(\sum_{t=1}^{\tau}\Psi_t(\Psi_t-\Psi_{t+1})^T)^{-1}(\sum_{t=1}^{\tau}\Psi_t(r_t-\bar{\lambda}^i)). \tag{40}$$

Algorithm 7 `EstimateDynamics`

Input: U, Y, $m0$.
Output: `(A_hat,B_hat)`

```
data_exp = iddata(U,Y);
me=ssest(data_exp,m0,ssestOptions,'
DisturbanceModel', 'estimate');
A_hat=me.a; B_hat=me.b;
```

Algorithm 8 Model-building approach

1: **Initialize:** Select a stabilizing policy gain K^1, set $i = 1$, $U = \{\}, Y = \{\}$.
`m0=idss(zeros(n,n),zeros(n,m),eye(n), zeros(n,m),'ts',1); m0.Structure.C.Free =zeros(n,n);`
2: **for** $i = 1, .., \mathbf{N}$ **do**
3: **for** $t = 1, ..., \tau$ **do**
Sample $\eta_t \sim \mathcal{N}(\mathbf{0}, W_\eta)$ and let $u_t = K^i y_t + \eta_t$.
Take u_t and $U \leftarrow u_t, Y \leftarrow y_t$.
4: $(\hat{A}, \hat{B})$ =`EstimateDynamics`$(U, Y, m0)$
5: `m0.A=`$\hat{A}$; `m0.B=`$\hat{B}$;
6: Find the optimal policy gain K^* from (6)-(7) and set $K^{i+1} = K^*$.

2) Policy improvement:

Line 6: Let partition matrix $\hat{G}^i$ as (32). The improved policy gain is given by

$$K^{i+1} = \arg\min_a \hat{Q}^i(y_k, a) = -(\hat{G}^i_{22})^{-1}\hat{G}^{iT}_{12}. \quad (41)$$

C. Model-building approach

A possible variant to model-free solutions to Problem 1 where the dynamics is not known is to estimate (A, B) as a separate system identification problem and use these estimates for the solution of Problem 1. We can call this a model-building approach to the problem. The estimated $(\hat{A}, \hat{B})$ can be used in the ARE (7) to solve for optimal policy gain in the corresponding LQ control (see Lemma 1).

The system identification problem to find (A, B) in (1) is actually a simple problem since the model order n is known. Let $[U, Y]$ denote the input-output data and $m0$ denote the initial state-space model (which can be set to zero matrices when no initial knowledge about the dynamics is known). To solve this structured problem in a general system identification code, like the system identification toolbox in MATLAB [43], the code in Algorithm 7 can be used. One can also use total least squares [42] to estimate the dynamics [35].

Since in this paper we examine the model-building approach along with the model-free algorithms, we bring the model-building approach in a recursive way such that the controller gain in the model-building approach is updated at the same pace as the model-free approaches. Algorithm 8 summarizes the model-building approach. In **Line 1**, we initialize the algorithm by selecting a stabilizing controller gain. Note that if the uncontrolled system is unstable, using only noise for system identification results in numerical instability. Otherwise, we can set $K^1 = \mathbf{0}$. We set empty matrices for the input-output data U, Y, assign zero matrices to the initial state-space model, and fix $C = I$ ($y_k = Ix_k + v_k$). In **Line 2**, the algorithm iterates $\mathbf{N}$ times. In **Line 3**, we collect τ input-output samples and append to U, Y. In **Line 4**, we estimate $(\hat{A}, \hat{B})$ using U, Y and $m0$. In **Line 5**, we update $m0$ by $(\hat{A}, \hat{B})$. In **Line 6**, we solve the ARE (7) using $(\hat{A}, \hat{B})$.

Our proposed model-building routine in Algorithm 7 can be easily extended (by further estimating C) to cover the partially observable dynamical systems similar to [36], [37].

Algorithm 9 Policy gradient

1: **Initialize:** Policy gain K^1, $i = 1$, probability density function $p(a; K^i y_k)$.
2: **while** ending condition is not satisfied **do**
3: $\mathcal{Z} = \{\}$.
4: **for** $t = 1, ..., T$ **do**
Observe y_t, sample $u_t \sim p(a; K^i y_k)$, take u_t.
Observe r_t. Add (y_t, u_t, r_t) to $\mathcal{Z}$.
5: Let $R(T) = \sum_{t=1}^T r_t$.
6: Set $\delta K^i = \sum_{t=1}^T (R(T) - \mathfrak{b}) \nabla_{K^i} \log p(u_t, K^i y_t)$.
7: Update K^{i+1} by gradient descent using δK^i.

D. Policy gradient

A Policy gradient routine is given in Algorithm 9 [23]. In **Line 1**, we initialize the algorithm by a controller gain and selecting a probability density function (pdf) for the probabilistic policy (usually a multivariate Gaussian distribution). In **Line 2**, the algorithm is iterated until the ending condition is satisfied. In **Lines 3-4**, we collect T samples of (y_k, u_k, r_k) by sampling the actions from the pdf $u_k \sim p(a; K^i y_k)$ and storing in $\mathcal{Z}$. In **Line 5**, we calculate the total cost of T steps. In **Line 6**, we give the gradient of the total cost with respect to K^i. Note that $\mathfrak{b}$ in **Line 6** is a baseline to reduce variance [30]. Among many options, one can select the baseline as the mean cost of previous iterates. In **Line 7**, we update the policy gain by gradient descent using δK^i.

APPENDIX II
THE PROOFS

In this section, we bring the proofs of theorems and lemmas in the body of the paper. To facilitate the derivations, we use the following identity frequently $\text{vecv}(v)^T \text{vecs}(P) = (v \otimes v)\text{vec}(P) = v^T P v$, where P is a symmetric matrix and v is a vector with appropriate dimension. The following lemma is useful through the proofs.

Lemma 9: Consider (1)-(5). For $\Xi \in \mathbb{R}^{n\times n}$, we have

$$\mathbf{E}[x_k^T \Xi v_k | y_k] = \mathbf{E}[x_k^T \Xi w_k | y_k] = \mathbf{E}[x_k^T \Xi v_{k+1} | y_k] = 0, \quad (42)$$

$$\mathbf{E}[x_k^T \Xi x_k | y_k] = y_k^T \Xi y_k - \text{Tr}(\Xi W_v). \quad (43)$$

Proof: Based on (1), w_k, v_k, v_{k+1} do not affect x_k and (42) follows. To see (43), note that

$$\begin{aligned} y_k^T \Xi y_k &= \mathbf{E}[y_k^T \Xi y_k | y_k] = \mathbf{E}[(x_k + v_k)^T \Xi (x_k + v_k) | y_k] \\ &= \mathbf{E}[x_k^T \Xi x_k | y_k] + 2\mathbf{E}[x_k^T \Xi v_k | y_k] + \mathbf{E}[v_k^T \Xi v_k | y_k] \\ &= \mathbf{E}[x_k^T \Xi x_k | y_k] + \text{Tr}(\Xi W_v). \end{aligned}$$

■

A. Proof of Lemma 1

Let $V^* = y_k^T P^* y_k$ be the optimal value function. Define the Bellman equation for the optimal value function (5)

$$\begin{aligned} y_k^T P^* y_k =& y_k^T R_y y_k + y_k^T K^{*T} R_u K^* y_k - \lambda(K^*) \\ &+ \mathbf{E}[y_{k+1}^T P^* y_{k+1}|y_k]. \end{aligned} \tag{44}$$

The term $\mathbf{E}[y_{k+1}^T P^* y_{k+1}|y_k]$ reads

$$\begin{aligned} &\mathbf{E}[y_{k+1}^T P^* y_{k+1}|y_k] \\ =&\mathbf{E}[x_k^T (A+BK^*)^T P^* (A+BK^*) x_k|y_k] \\ &+ 2\mathbf{E}[x_k^T (A+BK^*)^T P^* B K^* v_k|y_k] \\ &+ 2\mathbf{E}[x_k^T (A+BK^*)^T P^* (w_k + v_{k+1})|y_k] \\ &+ \mathbf{E}[v_k^T K^{*T} B^T P^* B K^* v_k|y_k] \\ &+ 2\mathbf{E}[v_k^T K^{*T} B^T P^* (w_k + v_{k+1})|y_k] \\ &+ \mathbf{E}[(w_k + v_{k+1})^T P^* (w_k + v_{k+1})|y_k]. \end{aligned}$$

Using (42)-(43) in the above equation

$$\begin{aligned} \mathbf{E}[y_{k+1}^T P^* y_{k+1}|y_k] &= y_k^T (A+BK^*)^T P^* (A+BK^*) y_k \\ &+ \text{Tr}(K^{*T} B^T P^* B K^* W_v) + \text{Tr}(P^* W_w) \\ &- \text{Tr}((A+BK^*)^T P^* (A+BK^*) W_v) + \text{Tr}(P^* W_v). \end{aligned}$$

Substituting the above result in (44) and matching terms, we have the optimal average cost in (8) and

$$(A+BK^*)^T P^* (A+BK^*) - P^* + R_y + K^{*T} R_u K^* = \mathbf{0}.$$

Optimizing the above equation with respect to K^*, results the optimal policy gain in (6).

B. Proof of Lemma 3

Similar to the proof of Lemma 2, we show that the given quadratic form satisfies the quality-based Bellman equation (15). Let $z_k = [y_k^T, u_k^T]^T$, $z_{k+1} = [y_{k+1}^T, (Ky_{k+1})^T]^T$ and $M = \begin{bmatrix} I & K^T \end{bmatrix} G \begin{bmatrix} I \\ K \end{bmatrix}$. Let u_k as $u_k = Ky_k + \eta_k$. Using the control u_k at time k, the next output y_{k+1} reads

$$y_{k+1} = Lx_k + B\eta_k + BKv_k + w_k + v_{k+1}. \tag{45}$$

By (16), $\mathbf{E}[V(y_{k+1}, K)|z_k] = \mathbf{E}[Q(y_{k+1}, Ky_{k+1}, K)|z_k]$. Now, we use (45) to compute $\mathbf{E}[Q(y_{k+1}, Ky_{k+1}, K)|z_k]$

$$\begin{aligned} &\mathbf{E}[Q(y_{k+1}, Ky_{k+1}, K)|z_k] = \mathbf{E}[y_{k+1}^T M y_{k+1}|z_k] \\ &= \mathbf{E}[(Lx_k + B\eta_k + BKv_k + w_k + v_{k+1})^T M \\ &\quad (Lx_k + B\eta_k + BKv_k + w_k + v_{k+1})|z_k] \\ &= \mathbf{E}[x_k^T L^T M L x_k|z_k] + 2\mathbf{E}[x_k^T L^T M B \eta_k|z_k] \\ &+ \mathbf{E}[\eta_k^T B^T M B \eta_k|z_k] + \mathbf{E}[v_k^T K^T B^T M B K v_k|z_k] \\ &+ \mathbf{E}[w_k^T M w_k + v_{k+1}^T M v_{k+1}|z_k]. \end{aligned}$$

Using (43), the above equation reads

$$\begin{aligned} &\mathbf{E}[y_{k+1}^T M y_{k+1}|z_k] \\ =& y_k^T L^T M L y_k + 2y_k^T L^T M B \eta_k + \eta_k^T B^T M B \eta_k \\ &- \text{Tr}(L^T M L W_v) + \text{Tr}(K^T B^T M B K W_v) \\ &+ \text{Tr}(M W_w) + \text{Tr}(M W_v) \\ =& \begin{bmatrix} y_k^T & u_k^T \end{bmatrix} \begin{bmatrix} A^T M A & A^T M B \\ B^T M A & B^T M B \end{bmatrix} \begin{bmatrix} y_k \\ u_k \end{bmatrix} \\ &- \text{Tr}(L^T M L W_v) + \text{Tr}(K^T B^T M B K W_v) \\ &+ \text{Tr}(M W_w) + \text{Tr}(M W_v) \\ =& z_k^T \begin{bmatrix} A^T M A & A^T M B \\ B^T M A & B^T M B \end{bmatrix} z_k - \text{Tr}(L^T M L W_v) \\ &+ \text{Tr}(K^T B^T M B K W_v) + \text{Tr}(M W_w) + Tr(M W_v). \end{aligned}$$

Replacing the above result in the quality-based Bellman equation (15)

$$\begin{aligned} z_k^T G z_k = z_k^T &\begin{bmatrix} A^T M A + R_y & A^T M B \\ B^T M A & B^T M B + R_u \end{bmatrix} z_k \\ &- \text{Tr}(L^T M L W_v) + \text{Tr}(K^T B^T M B K W_v) \\ &+ \text{Tr}(M W_w) + Tr(M W_v) - \lambda(K). \end{aligned}$$

By matching terms (17)-(18) are concluded.

C. Proof of Lemma 4

According to (16),

$$P = \begin{bmatrix} I & K^T \end{bmatrix} G \begin{bmatrix} I \\ K \end{bmatrix}. \tag{46}$$

By Lemma 3, the quadratic kernel of the Q function, satisfies (17). Pre-multiplying (17) by $\begin{bmatrix} I & K^T \end{bmatrix}$ and post-multiplying (17) by $\begin{bmatrix} I \\ K \end{bmatrix}$, we have

$$\begin{bmatrix} I & K^T \end{bmatrix} \left(\begin{bmatrix} A^T \\ B^T \end{bmatrix} P \begin{bmatrix} A & B \end{bmatrix} + \begin{bmatrix} R_y & 0 \\ 0 & R_u \end{bmatrix} \right) \begin{bmatrix} I \\ K \end{bmatrix} - P = \mathbf{0},$$

where we have used (46) in the above equation. By expanding the above equation, (10) is concluded.

D. Proof of Theorem 3

In Algorithm 5, first, a solution to the off-policy Bellman equation (24) is estimated and then the controller is selected to be greedy with respect to the average of all previously estimated value functions, see (28). The proof of convergence of the algorithm contains two steps. The first step is to show that solving the off-policy Bellman equation (24) is equivalent to the model-based Bellman equation (13). This is guaranteed by Theorem 1 and assuming that the estimation errors in (22) and (27) are small enough $\hat{P}^i \approx P^i$, $B^T \hat{P}^i A \approx B^T P^i A$, $B^T \hat{P}^i B \approx B^T P^i B$. The second step is to show that by improving the policy to be greedy with respect to the average of all previous value functions (28), $P^{i+1} \leq P^i$ for $i = 1, ..., \mathbf{N}$. By Theorem 2, the algorithm produces stabilizing controller gain K^i. Let $P^i > 0$ denote the solution to the model-based Bellman equation (13)

$$P^i = (A+BK^i)^T P^i (A+BK^i) + K^{iT} R_u K^i + R_y. \tag{47}$$

Since $A+BK^i$ is stable, the unique positive definite solution of (47) may be written as [40]

$$P^i=\sum_{k=0}^{+\infty}((A+BK^i)^T)^k(K^{iT}R_uK^i+R_y)(A+BK^i)^k. \tag{48}$$

Consider two iteration indices i and j. Using (47) P^i-P^j reads

$$\begin{aligned}P^i-P^j=&(A+BK^i)^TP^i(A+BK^i)+K^{iT}R_uK^i\\&-(A+BK^j)^TP^j(A+BK^j)-K^{jT}R_uK^j\\=&(A+BK^j)^TP^i(A+BK^j)+K^{iT}R_uK^i\\&-K^{jT}B^TP^i(A+BK^j)-A^TP^iK^j\\&+K^{iT}B^TP^i(A+BK^i)+A^TP^iK^i\\&-(A+BK^j)^TP^j(A+BK^j)-K^{jT}R_uK^j\\=&(A+BK^j)^T(P^i-P^j)(A+BK^j)\\&+(K^i-K^j)^T(R_u+B^TP^iB)(K^i-K^j)\\&+(K^i-K^j)^T[(R_u+B^TP^iB)K^j+B^TP^iA]\\&+[K^{jT}(R_u+B^TP^iB)+A^TP^iB](K^i-K^j)\\=&\sum_{k=0}^{+\infty}((A+BK^j)^T)^k\\&((K^i-K^j)^T(R_u+B^TP^iB)(K^i-K^j)\\&+(K^i-K^j)^T[(R_u+B^TP^iB)K^j+B^TP^iA]\\&[K^{jT}(R_u+B^TP^iB)+A^TP^iB](K^i-K^j))\\&(A+BK^j)^k,\end{aligned} \tag{49}$$

where we have used (48) to obtain the last equation. By (28)

$$\begin{aligned}K^{i+1}&=-(\sum_{j=1}^{i}(B^TP^jB+R_u))^{-1}(\sum_{j=1}^{i}B^TP^jA),\\K^{i}&=-(\sum_{j=1}^{i-1}(B^TP^jB+R_u))^{-1}(\sum_{j=1}^{i-1}B^TP^jA).\end{aligned} \tag{50}$$

Rearranging K^{i+1} in (50) and let $M:=\sum_{j=1}^{i-1}(B^TP^jB+R_u)>0$

$$M(K^i-K^{i+1})=[(R_u+B^TP^iB)K^{i+1}+B^TP^iA]. \tag{51}$$

By setting $j=i+1$ in (49) and using (51), P^i-P^{i+1} reads

$$\begin{aligned}P^i-P^{i+1}=&\sum_{k=0}^{+\infty}((A+BK^j)^T)^k\\&[(K^i-K^j)^T(R_u+B^TP^iB)(K^i-K^j)\\&+2(K^i-K^j)^TM(K^i-K^j)](A+BK^j)^k\geq\mathbf{0},\end{aligned}$$

which shows that $P^{i+1}\leq P^i$ and the sequence is converging. By repeating this procedure for $i=1,...,$ we can see that $P^*\leq P^{i+1}\leq P^i\leq P^1$.

E. Proof of Lemma 5

We repeat the estimation problem in (19)-(22)

$$\begin{aligned}r_k-\lambda^i&=(y_k\otimes y_k-y_{k+1}\otimes y_{k+1})^T\mathrm{vec}(P^i)+\mathfrak{n}_1,\\\mathfrak{n}_1&=y_{k+1}^TP^iy_{k+1}-\mathbf{E}[y_{k+1}^TP^iy_{k+1}|y_k].\end{aligned} \tag{52}$$

The noise in the above equation appears in the regressor (also called error-in-variables [22])) and can be written as $\mathfrak{n}_1=\nu^T\mathrm{vec}(P^i)$ where $\mathrm{vec}(P^i)$ is our parameter vector and

$$\begin{aligned}\nu^T=&y_{k+1}^T\otimes y_{k+1}^T-\mathbf{E}[y_{k+1}^T\otimes y_{k+1}^T|y_k]\\=&(2(w_k+v_{k+1})^T\otimes x_k^T)(I\otimes L^{iT})\\&+(2(w_k+v_{k+1})^T\otimes v_k^T)(I\otimes(K^{iT}B^T))\\&+(w_k+v_{k+1})^T\otimes(w_k+v_{k+1})^T\\&-(v_k^T\otimes v_k^T)(A^T\otimes A^T)\\&-(2v_k^T\otimes v_k^T)((K^{iT}B^T)\otimes A^T)\\&-(2v_k^T\otimes x_k^T)(A^T\otimes L^{iT})+(\mathrm{vec}(W_v)^T)(L^{iT}\otimes L^{iT})\\&-\mathrm{vec}(W_w)^T-\mathrm{vec}(W_v)^T\\&-(\mathrm{vec}(W_v)^T)((K^{iT}B^T)\otimes(K^{iT}B^T)).\end{aligned} \tag{53}$$

The instrumental variable in the estimation (22) is $y_k\otimes y_k$. To have an unbiased estimate, $\mathbf{E}[(y_k\otimes y_k)\nu^T]=\mathbf{0}$. In the sequel, we analyze the correlation matrix

$$\begin{aligned}\mathbf{E}[(y_k\otimes y_k)\nu^T]=&\mathbf{E}[(x_k\otimes x_k)\nu^T]+2\mathbf{E}[(x_k\otimes v_k)\nu^T]\\&+\mathbf{E}[(v_k\otimes v_k)\nu^T].\end{aligned}$$

Remembering the facts that x_k, v_k, v_{k+1}, w_k are mutually independent, W_v is diagonal and using (14), we get the following results. The term $\mathbf{E}[(x_k\otimes x_k)\nu^T]$ reads

$$\begin{aligned}&\mathbf{E}[(x_k\otimes x_k)\nu^T]\\=&\mathbf{E}[x_k\otimes x_k](\mathrm{vec}(W_w)^T+\mathrm{vec}(W_v)^T\\&-\mathrm{vec}(W_v)^T(A^T\otimes A^T+2(K^{iT}B^T)\otimes A^T)\\&+(\mathrm{vec}(W_v)^T)(L^{iT}\otimes L^{iT})-\mathrm{vec}(W_w)^T-\mathrm{vec}(W_v)^T\\&-(\mathrm{vec}(W_v)^T)((K^{iT}B^T)\otimes(K^{iT}B^T)))=\mathbf{0}.\end{aligned}$$

The term $2\mathbf{E}[(x_k\otimes v_k)\nu^T]$ reads

$$\begin{aligned}&2\mathbf{E}[(x_k\otimes v_k)\nu^T]\\&=-4\mathbf{E}[(x_k\otimes v_k)(v_k^T\otimes x_k^T)](A^T\otimes L^{iT}).\end{aligned} \tag{54}$$

The term $\mathbf{E}[(v_k\otimes v_k)\nu^T]$ reads

$$\begin{aligned}&\mathbf{E}[(v_k\otimes v_k)\nu^T]\\&=-\mathbf{E}[(v_k\otimes v_k)(v_k^T\otimes v_k^T)](A^T\otimes A^T+2(K^{iT}B^T)\otimes A^T)\\&+\mathrm{vec}(W_v)(\mathrm{vec}(W_v)^T)(L^{iT}\otimes L^{iT})\\&-\mathrm{vec}(W_v)(\mathrm{vec}(W_v)^T)((K^{iT}B^T)\otimes(K^{iT}B^T)).\end{aligned} \tag{55}$$

Finally by summing (54)-(55), we have the bias term in the estimation (22)

$$\begin{aligned}&\mathbf{E}[(y_k\otimes y_k)\nu^T]\\&=-4\mathbf{E}[(x_k\otimes v_k)(v_k^T\otimes x_k^T)](A^T\otimes L^{iT})\\&-\mathbf{E}[(v_k\otimes v_k)(v_k^T\otimes v_k^T)](A^T\otimes A^T+2(K^{iT}B^T)\otimes A^T)\\&+\mathrm{vec}(W_v)(\mathrm{vec}(W_v)^T)(L^{iT}\otimes L^{iT})\\&-\mathrm{vec}(W_v)(\mathrm{vec}(W_v)^T)((K^{iT}B^T)\otimes(K^{iT}B^T))\end{aligned} \tag{56}$$

which is nonzero. Meaning that the estimate is biased.

F. Proof of Theorem 4

We repeat the estimation problem in (25)-(27)

$$
\begin{aligned}
&c_k = \Upsilon_k^T \xi^i + \mathfrak{n}_2, \qquad (57)\\
&c_k = y_k^T (R_y + K^{iT} R_u K^i - \hat{P}^i) y_k + y_{k+1}^T \hat{P}^i y_{k+1} - \bar{\lambda}^i,\\
&\Upsilon_k = \begin{bmatrix} 2y_k \otimes (u_k - K^i y_k) \\ (u_k - K^i y_k) \otimes (u_k + K^i y_k) \end{bmatrix}, \ \xi^i = \begin{bmatrix} \mathrm{vec}(B^T P^i A) \\ \mathrm{vec}(B^T P^i B) \end{bmatrix},\\
&\mathfrak{n}_2 = y_{k+1}^T \hat{P}^i y_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | y_k] + y_k^T P^i y_k - y_k^T \hat{P}^i y_k.
\end{aligned}
$$

The noise term $\mathfrak{n}_2$ reads

$$
\begin{aligned}
\mathfrak{n}_2 =& y_{k+1}^T \hat{P}^i y_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | y_k] + y_k^T (P^i - \hat{P}^i) y_k\\
=& x_k^T L^{iT} (\hat{P}^i - P^i) L^i x_k + 2 x_k^T L^{iT} (\hat{P}^i - P^i) B K^i v_k\\
&+ 2 x_k^T L^{iT} \hat{P}^i (w_k + v_{k+1}) + v_k^T K^{iT} B^T (\hat{P}^i - P^i) B K^i v_k\\
&+ 2 v_k^T K^{iT} B^T \hat{P}^i (w_k + v_{k+1}) + x_k^T (P^i - \hat{P}^i) x_k\\
&+ v_k^T (P^i - \hat{P}^i) v_k + 2 x_k^T (P^i - \hat{P}^i) v_k\\
&+ (w_k + v_{k+1})^T \hat{P}^i (w_k + v_{k+1}) - \lambda^i \qquad (58)\\
&- v_k^T A^T P^i A v_k - 2 v_k^T A^T P^i B K^i v_k - 2 x_k^T L^{iT} P^i A v_k.
\end{aligned}
$$

To see if the estimate is biased, we examine $\mathbf{E}[\Upsilon_k \mathfrak{n}_2]$

$$
\mathbf{E}[\Upsilon_k \mathfrak{n}_2] = \begin{bmatrix} 0 \\ \eta_k \otimes \eta_k \end{bmatrix} \mathbf{E}[\mathfrak{n}_2] + \begin{bmatrix} 2\mathbf{E}[y_k \mathfrak{n}_2] \otimes \eta_k \\ 2\eta_k \otimes K^i \mathbf{E}[y_k \mathfrak{n}_2] \end{bmatrix}.
$$

So, it is enough to analyze $\mathbf{E}[\mathfrak{n}_2]$ and $\mathbf{E}[y_k \mathfrak{n}_2]$. Remembering the facts that x_k, v_k, v_{k+1}, w_k are mutually independent, W_v is diagonal, $\mathbf{E}[x_k] = \mathbf{0}$ (because of running Algorithm 3 to collect data) and using (14), we have

$$
\begin{aligned}
\mathbf{E}[\mathfrak{n}_2] =& \mathbf{E}[x_k^T L^{iT} (\hat{P}^i - P^i) L^i x_k] + \mathbf{E}[x_k^T (P^i - \hat{P}^i) x_k]\\
&+ \mathbf{E}[v_k^T K^{iT} B^T (\hat{P}^i - P^i) B K^i v_k]\\
&+ \mathrm{Tr}((\hat{P}^i - P)(W_w + W_v)), \qquad (59)
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{E}[y_k \mathfrak{n}_2] = \mathbf{E}[(x_k + v_k)\mathfrak{n}_2]\\
&= \underbrace{\mathbf{E}[(x_k + v_k) x_k^T L^{iT} (\hat{P}^i - P^i) L^i x_k]}_{=0}\\
&+ 2\underbrace{\mathbf{E}[(x_k + v_k) x_k^T L^{iT} (\hat{P}^i - P^i) B K^i v_k]}_{=0}\\
&\underbrace{\mathbf{E}[2(x_k + v_k) x_k^T L^{iT} \hat{P}^i (w_k + v_{k+1})]}_{=0}\\
&+ \underbrace{\mathbf{E}[(x_k + v_k) v_k^T K^{iT} B^T (\hat{P}^i - P^i) B K^i v_k]}_{=0}\\
&+ \underbrace{\mathbf{E}[2(x_k + v_k) v_k^T K^{iT} B^T \hat{P}^i (w_k + v_{k+1})]}_{=0}\\
&+ \underbrace{\mathbf{E}[(x_k + v_k)(x_k^T (P^i - \hat{P}^i) x_k + v_k^T (P^i - \hat{P}^i) v_k)]}_{=0}\\
&+ 2\underbrace{\mathbf{E}[(x_k + v_k) x_k^T (P^i - \hat{P}^i) v_k]}_{=0}\\
&+ \underbrace{\mathbf{E}[(x_k + v_k)(w_k + v_{k+1})^T \hat{P}^i (w_k + v_{k+1}) - (x_k + v_k)\lambda^i]}_{=0}\\
&+ \underbrace{\mathbf{E}[(x_k + v_k)(-v_k^T A^T P^i A v_k - 2 v_k^T A^T P^i B K^i v_k)]}_{=0}\\
&- 2\underbrace{\mathbf{E}[(x_k + v_k)(x_k^T L^{iT} P^i A v_k)]}_{=0} = \mathbf{0}. \qquad (60)
\end{aligned}
$$

Based (60), $\mathbf{E}[y_k \mathfrak{n}_2] = \mathbf{0}$ and only $\mathbf{E}[\mathfrak{n}_2]$ contributes to bias. If the estimation error of P^i is negligible $\hat{P}^i - P^i \approx \mathbf{0}$, then by (59), $\mathbf{E}[\mathfrak{n}_2] = \mathbf{0}$ and the estimate is unbiased.

G. Proof of Lemma 6

We repeat the estimation problem in (34)-(36)

$$
\begin{aligned}
&r_k - \lambda^i = \Omega_k^T \varpi^i + \mathfrak{n}_1,\\
&\mathfrak{n}_1 = y_{k+1}^T P^i y_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | y_k],\\
&\varpi^i = \begin{bmatrix} \mathrm{vec}(P^i) \\ \mathrm{vec}(B^T P^i A) \\ \mathrm{vec}(B^T P^i B) \end{bmatrix}, \Omega_k = \begin{bmatrix} y_k \otimes y_k - y_{k+1} \otimes y_{k+1} \\ 2y_k \otimes (u_k - K^i y_k) \\ (u_k - K^i y_k) \otimes (u_k + K^i y_k) \end{bmatrix},\\
&\Theta_k = \begin{bmatrix} y_k \otimes y_k \\ 2y_k \otimes \eta_k \\ \eta_k \otimes (\eta_k + 2K^i y_k) \end{bmatrix}. \qquad (61)
\end{aligned}
$$

We have shown in the proof of Lemma 5 that the noise term can be written as $\mathfrak{n}_1 = \nu^T \mathrm{vec}(P^i)$ where ν is given in (53). In the sequel, we study $\mathbf{E}[\Theta_k \nu^T]$. Since ν is zero mean and η_k is known, we only need to analyze $\mathbf{E}[y_k \otimes y_k \nu^T]$ and $\mathbf{E}[y_k \nu^T]$. The term $\mathbf{E}[y_k \otimes y_k \nu^T]$ is given in (56). Remembering the facts that x_k, v_k, v_{k+1}, w_k are mutually independent and W_v is diagonal, we compute $\mathbf{E}[y_k \nu^T]$ where ν is given in (53)

$$
\begin{aligned}
&\mathbf{E}[y_k \nu^T] = \mathbf{E}[(x_k + v_k)\nu^T]\\
&= -\mathbf{E}[(x_k)(v_k^T \otimes v_k^T)](A^T \otimes A^T + 2(K^{iT} B^T) \otimes A^T)\\
&\quad - 2\mathbf{E}[(v_k)(v_k^T \otimes x_k^T)](A^T \otimes L^{iT})\\
&\quad + \mathbf{E}[x_k](\mathrm{vec}(W_v)^T)(L^{iT} \otimes L^{iT})\\
&\quad - \mathbf{E}[x_k](\mathrm{vec}(W_v)^T)((K^{iT} B^T) \otimes (K^{iT} B^T)). \qquad (62)
\end{aligned}
$$

Hence, the bias in the classical off-policy is a function of (56) and (62).

H. Proof of Lemma 7

We repeat the estimation problem in (29)-(31)

$$
\begin{aligned}
&c_k' = \Psi_k^T \mathrm{vec}(G^i) + \mathfrak{n}_3, \qquad (63)\\
&c_k' = r(y_k, u_k) - \lambda^i + y_{k+1}^T \hat{P}^i y_{k+1},\\
&\Psi_k = \begin{bmatrix} y_k^T \otimes y_k^T & y_k^T \otimes u_k^T & u_k^T \otimes y_k^T & u_k^T \otimes u_k^T \end{bmatrix}^T,\\
&\mathfrak{n}_3 = y_{k+1}^T \hat{P}^i y_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | z_k].
\end{aligned}
$$

The noise term $\mathfrak{n}_3$ reads

$$
\begin{aligned}
\mathfrak{n}_3 =& y_{k+1}^T \hat{P}^i y_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | z_k]\\
=& x_k^T A^T (\hat{P}^i - P^i) A x_k + 2 x_k^T A^T (\hat{P}^i - P^i) B u_k\\
&+ u_k^T B^T (\hat{P}^i - P^i) B u_k - 2 v_k^T A^T P^i (A x_k + B u_k)\\
&+ 2(x_k^T A^T + u_k^T B^T) \hat{P}^i (w_k + v_{k+1})\\
&+ (w_k + v_{k+1})^T \hat{P}^i (w_k + v_{k+1}) - v_k^T A^T P^i A v_k\\
&+ \mathrm{Tr}(A^T P^i A W_v) - \mathrm{Tr}(P^i W_w) - \mathrm{Tr}(P^i W_v). \qquad (64)
\end{aligned}
$$

To see if the estimate is biased, we need to analyze $\mathbf{E}[\Psi_k \mathfrak{n}_3]$. Since u_k is deterministic (we know u_k), we examine $i := u_k \otimes u_k \mathbf{E}[\mathfrak{n}_3]$, $ii := u_k \otimes \mathbf{E}[y_k \mathfrak{n}_3]$ and $iii := \mathbf{E}[y_k \otimes y_k \mathfrak{n}_3]$. Remembering the facts that x_k, v_k, v_{k+1}, w_k are mutually independent, W_v is diagonal, $\mathbf{E}[x_k] = \mathbf{0}$ (because of running Algorithm 3 to collect data), we get the following results

$$\begin{aligned} i :=& (u_k \otimes u_k)(\mathbf{E}[x_k^T A^T (\hat{P}^i - P^i) A x_k] \\ &+ u_k^T B^T (\hat{P}^i - P^i) B u_k + \mathrm{Tr}((\hat{P}^i - P^i)(W_w + W_v)). \end{aligned} \tag{65}$$

$$\begin{aligned} ii :=& u_k \otimes \mathbf{E}[(x_k + v_k)\mathfrak{n}_3] \\ =& 2u_k \otimes \mathbf{E}[x_k x_k^T A^T (\hat{P}^i - P^i) B u_k] \\ &- 2u_k \otimes \mathbf{E}[v_k v_k^T A^T P^i B u_k]. \end{aligned} \tag{66}$$

Next, we analyze $iii := \mathbf{E}[y_k \otimes y_k \mathfrak{n}_3] = \mathbf{E}[x_k \otimes x_k \mathfrak{n}_3] + 2\mathbf{E}[x_k \otimes v_k \mathfrak{n}_3] + \mathbf{E}[v_k \otimes v_k \mathfrak{n}_3]$. First, we obtain $\mathbf{E}[x_k \otimes x_k \mathfrak{n}_3]$

$$\begin{aligned} &\mathbf{E}[x_k \otimes x_k \mathfrak{n}_3] \\ =& \mathbf{E}[x_k \otimes x_k x_k^T A^T (\hat{P}^i - P^i) A x_k] \\ &+ \mathbf{E}[x_k \otimes x_k] u_k^T B^T (\hat{P}^i - P^i) B u_k \\ &+ \mathbf{E}[x_k \otimes x_k] \mathrm{Tr}((\hat{P}^i - P^i)(W_w + W_v). \end{aligned} \tag{67}$$

Second, we obtain $2\mathbf{E}[x_k \otimes v_k \mathfrak{n}_3]$

$$2\mathbf{E}[x_k \otimes v_k n_3] = -4\mathbf{E}[x_k \otimes v_k v_k^T A^T P^i A x_k]. \tag{68}$$

Third, we obtain $\mathbf{E}[v_k \otimes v_k \mathfrak{n}_3]$

$$\begin{aligned} &\mathbf{E}[v_k \otimes v_k \mathfrak{n}_3] \\ =& \mathbf{E}[v_k \otimes v_k x_k^T A^T (\hat{P}^i - P^i) A x_k] \\ &+ \mathbf{E}[v_k \otimes v_k u_k^T B^T (\hat{P}^i - P^i) B u_k] \\ &+ \mathbf{E}[v_k \otimes v_k] \mathrm{Tr}((\hat{P}^i - P^i)(W_v + W_w)) \\ &+ \mathbf{E}[v_k \otimes v_k] \mathrm{Tr}(A^T P^i A W_v) - \mathbf{E}[v_k \otimes v_k v_k^T A^T P^i A v_k]. \end{aligned} \tag{69}$$

As a result, iii is given by the summation of the terms in (67)-(69). Because the terms in (65)-(69) are nonzero, the estimate in (31) is biased.

I. Proof of Lemma 8

We repeat the estimation problem in (38)-(40)

$$\begin{aligned} r_k - \lambda^i &= (\Psi_k - \Psi_{k+1})\mathrm{vec}(G^i) + \mathfrak{n}_4, \\ \Psi_k^T &= [z_k^T \otimes z_k^T] = [y_k^T \otimes y_k^T, y_k^T \otimes u_k^T, u_k^T \otimes y_k^T, u_k^T \otimes u_k^T], \\ \mathfrak{n}_4 &= z_{k+1}^T G^i z_{k+1} - \mathbf{E}[z_{k+1}^T G^i z_{k+1} | z_k] \\ &= y_{k+1}^T P^i z_{k+1} - \mathbf{E}[y_{k+1}^T P^i y_{k+1} | z_k]. \end{aligned} \tag{70}$$

The noise term $\mathfrak{n}_4$ is obtained by $\hat{P}^i \equiv P^i$ in (64)

$$\begin{aligned} \mathfrak{n}_4 =& -2v_k^T A^T P^i (A x_k + B u_k) \\ &+ 2(x_k^T A^T + u_k^T B^T) P^i (w_k + v_{k+1}) \\ &+ (w_k + v_{k+1})^T P^i (w_k + v_{k+1}) - v_k^T A^T P^i A v_k \\ &+ \mathrm{Tr}(A^T P^i A W_v) - \mathrm{Tr}(P^i W_w) - \mathrm{Tr}(P^i W_v). \end{aligned} \tag{71}$$

We study $\mathbf{E}[\Psi_k \mathfrak{n}_4]$. Since u_k is deterministic (we know u_k), $\mathbf{E}[u_k \otimes u_k \mathfrak{n}_4] = 0$. We examine $u_k \otimes \mathbf{E}[y_k \mathfrak{n}_4]$, $\mathbf{E}[y_k \otimes y_k \mathfrak{n}_4]$.

$$\begin{aligned} u_k \otimes \mathbf{E}[y_k \mathfrak{n}_4] =& -2u_k \otimes (\mathbf{E}[v_k v_k^T A^T P^i (A x_k + B u_k)] \\ &+ \mathbf{E}[x_k]\mathbf{E}[(w_k + v_{k+1})^T P^i (w_k + v_{k+1})] \\ &- \mathbf{E}[x_k]\mathbf{E}[v_k^T A^T P^i A v_k] + \mathbf{E}[x_k]\mathrm{Tr}(A^T P^i A W_v) \\ &- \mathbf{E}[x_k](\mathrm{Tr}(P^i W_w) + \mathrm{Tr}(P^i W_v))) \\ =& -2u_k \otimes \mathbf{E}[v_k v_k^T A^T P^i (A x_k + B u_k)], \end{aligned} \tag{72}$$

$$\begin{aligned} \mathbf{E}[y_k \otimes y_k \mathfrak{n}_4] =& -4\mathbf{E}[x_k \otimes v_k v_k^T A^T P^i (A x_k + B u_k)] \\ &+ \mathbf{E}[v_k \otimes v_k]\mathrm{Tr}(A^T P^i A W_v) - \mathbf{E}[v_k \otimes v_k v_k^T A^T P^i A v_k]. \end{aligned} \tag{73}$$

Since (72)-(73) are nonzero, the estimate is biased.

REFERENCES

[1] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking Deep Reinforcement Learning for Continuous Control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338. [Online]. Available: http://arxiv.org/abs/1604.06778

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," in *Advances in Neural Information Processing Systems 19*, 2007, pp. 1–8.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and Others, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] M. Krstic, I. Kanellakopoulos, P. V. Kokotovic, and Others, *Nonlinear and adaptive control design*. Wiley New York, 1995, vol. 222.

[8] K. J. Åström and B. Wittenmark, *Adaptive control*, 2nd ed. Prentice Hall, 1994.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press Cambridge, 2018, vol. 1, no. 1.

[10] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[11] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *IEEE Control Systems*, vol. 32, no. 6, pp. 76–105, 2012.

[12] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, 2009.

[13] F. Adib Yaghmaie and D. J. Braun, "Reinforcement learning for a class of continuous-time input constrained optimal control problems," *Automatica*, vol. 99, pp. 221–227, 2019.

[14] T. Bian, Y. Jiang, and Z.-P. Jiang, "Adaptive dynamic programming for stochastic systems with state and control dependent noise," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 4170–4175, 2016.

[15] B. Kiumarsi, F. L. Lewis, and Z.-P. Jiang, "H_∞ control of linear discrete-time systems: Off-policy reinforcement learning," *Automatica*, vol. 78, pp. 144–152, 2017.

[16] H. Modares and F. L. Lewis, "Linear quadratic tracking control of partially-unknown continuous-time systems using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 59, no. 11, pp. 3051–3056, 2014.

[17] F. Adib Yaghmaie, S. Gunnarsson, and F. L. Lewis, "Output Regulation of Unknown Linear Systems using Average Cost Reinforcement Learning," *Automatica*, vol. 110, p. 108549, 2019.

[18] F. Adib Yaghmaie, F. L. Lewis, and R. Su, "Output regulation of heterogeneous linear multi-agent systems with differential graphical game," *International Journal of Robust and Nonlinear Control*, vol. 26, no. 10, pp. 2256–2278, 2016.

[19] F. Adib Yaghmaie, K. Hengster Movric, F. L. Lewis, and R. Su, "Differential graphical games for H_∞ control of linear heterogeneous multiagent systems," *International Journal of Robust and Nonlinear Control*, vol. 29, no. 10, pp. 2995–3013, 2019.
[20] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.
[21] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1107–1149, 2003.
[22] S. J. Bradtke and A. G. Barto, "Linear Least-Squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 1-3, pp. 33–57, 2004.
[23] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.
[24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, pp. 605–619, 2014.
[25] L. Ljung, *System Identification - Theory for the user*, 2nd ed. PTR Prentice Hall Inforamtion and System Sciences series, 1999.
[26] L. Ljung and T. Söderström, *Theory and practice of recursive identification*. MIT press, 1987.
[27] Y. Abbasi-Yadkori, N. Lazic, and C. Szepesvari, "Model-free linear quadratic control via reduction to expert prediction," *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 3108–3117, 2019.
[28] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the Sample Complexity of the Linear Quadratic Regulator," *Foundations of Computational Mathematics*, pp. 1–47, 2019.
[29] S. Tu and B. Recht, "Least-squares temporal difference learning for the linear quadratic regulator," *International Conference on Machine Learning*, pp. 5005–5014, 2018.
[30] N. Matni, A. Proutiere, A. Rantzer, and S. Tu, "From self-tuning regulators to reinforcement learning and back again," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 3724–3740.
[31] M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi, "Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator," in *International Conference on Machine Learning*, 2018, pp. 1467–1476.
[32] A. Cohen, A. Hassidim, T. Koren, N. Lazic, Y. Mansour, and K. Talwar, "Online linear quadratic control," *International Conference on Machine Learning*, pp. 1029–1038, 2018.
[33] S. Tu and B. Recht, "The Gap Between Model-Based and Model-Free Methods on the Linear Quadratic Regulator: An Asymptotic Viewpoint," in *Conference on Learning Theory*, 2019, pp. 3036–3083. [Online]. Available: http://arxiv.org/abs/1812.03565
[34] M. Ferizbegovic, J. Umenberger, H. Hjalmarsson, and T. B. Schon, "Learning robust LQ-controllers using application oriented exploration," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 19–24, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8732482/
[35] F. Adib Yaghmaie and F. Gustafsson, "Using Reinforcement Learning for Model-free Linear Quadratic Gaussian Control with Process and Measurement noises," in *IEEE Conference on Decision and Control*, 2019, pp. 6510–6517.
[36] M. Simchowitz, K. Singh, and E. Hazan, "Improper Learning for Non-Stochastic Control," in *CONFERENCE ON LEARNING THEORY (COLT)*, 2020. [Online]. Available: http://arxiv.org/abs/2001.09254
[37] S. Lale, K. Azizzadenesheli, B. Hassibi, and A. Anandkumar, "Logarithmic Regret Bound in Partially Observable Linear Dynamical Systems," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020, pp. 20 876—-20 888.
[38] F. L. Lewis and K. G. Vamvoudakis, "Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 1, pp. 14–25, 2011.
[39] H. Yu and D. P. Bertsekas, "Convergence results for some temporal difference methods based on least squares," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1515–1531, 2009.
[40] G. A. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *IEEE Transactions on Automatic Control*, vol. 16, no. 4, pp. 382–384, 1971.
[41] L. A. Prashanth, N. Korda, and R. Munos, "Fast LSTD using stochastic approximation: Finite time analysis and application to traffic control," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 66–81.
[42] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. The John Hopkins University Press, 2013.
[43] The MathWorks, "MATLAB system identification toolbox (R2018a)."

Farnaz Adib Yaghmaie received the B.E. and M.E. degrees from the school of Electrical Engineering department, K. N. Toosi University of Technology, Tehran, Iran in 2009 and 2011. She received her Ph.D degree from the school of Electrical and Electronic Engineering, Nanyang Technological University (EEE-NTU), Singapore in 2017. She is the recipient of the best thesis award from EEE-NTU among 160 Ph.D students. Currently, she is a postdoc research fellow at Linköping University, Sweden. Her current research interest is reinforcement learning and distributed control of multi-agent systems.

Fredrik Gustafsson is professor in Sensor Informatics at Department of Electrical Engineering, Linköping University, since 2005. He received the M.Sc. degree in electrical engineering 1988 and the Ph.D. degree in Automatic Control, 1992, both from Linköping University. His research interests are in stochastic signal processing, adaptive filtering and change detection, with applications to communication, vehicular, airborne, and audio systems. He is a co-founder of the companies NIRA Dynamics (automotive safety systems), Softube (audio effects) and Senion (indoor positioning systems). He was an associate editor for IEEE Transactions of Signal Processing 2000-2006, IEEE Transactions on Aerospace and Electronic Systems 2010-2012, and EURASIP Journal on Applied Signal Processing 2007-2012. He was awarded the Arnberg prize by the Royal Swedish Academy of Science (KVA) 2004, elected member of the Royal Academy of Engineering Sciences (IVA) 2007, and elevated to IEEE Fellow 2011. He was awarded the Harry Rowe Mimno Award 2011 for the tutorial "Particle Filter Theory and Practice with Positioning Applications", which was published in the AESS Magazine in July 2010, and was co-author of "Smoothed state estimates under abrupt changes using sum-of-norms regularization" that received the Automatica paper prize in 2014.

Lennart Ljung received his PhD in Automatic Control from Lund Institute of Technology in 1974. Since 1976 he is Professor of the chair of Automatic Control In Linkoping, Sweden. He has held visiting positions at Stanford and MIT and has written several books on System Identification and Estimation. He is an IEEE Fellow, an IFAC Fellow and an IFAC Advisor. He is as a member of the Royal Swedish Academy of Sciences (KVA), a member of the Royal Swedish Academy of Engineering Sciences (IVA), an Honorary Member of the Hungarian Academy of Engineering, an Honorary Professor of the Chinese Academy of Mathematics and Systems Science, and a Foreign Member of the US National Academy of Engineering (NAE). He has received honorary doctorates from the Baltic State Technical University in St Petersburg, from Uppsala University, Sweden, from the Technical University of Troyes, France, from the Catholic University of Leuven, Belgium and from Helsinki University of Technology, Finland. He has received both the Quazza Medal (2002) and the Nichols Medal (2017) from IFAC. In 2003 he received the Hendrik W. Bode Lecture Prize from the IEEE Control Systems Society, and he was the 2007 recipient of the IEEE Control Systems Award. In 2018 he received the Great Gold Medal from the Royal Swedish Academy of Engineering.