2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

An Efficient Off-Policy Reinforcement Learning Algorithm for the Continuous-Time LQR Problem

Victor G. Lopez and Matthias A. Müller

Abstract— **In this paper, an off-policy reinforcement learning algorithm is designed to solve the continuous-time linear quadratic regulator (LQR) problem using only input-state data measured from the system. Different from other algorithms in the literature, we propose the use of a specific persistently exciting input as the exploration signal during the data collection step. We then show that, using this persistently excited data, the solution of the matrix equation in our algorithm is guaranteed to exist and to be unique at every iteration. Convergence of the algorithm to the optimal control input is also proven. Moreover, we formulate the policy evaluation step as the solution of a Sylvester-transpose equation, which increases the efficiency of its solution. A method to determine an initial stabilizing policy using only measured data is proposed. Finally, the advantages of the proposed method are tested via simulation.**

I. INTRODUCTION

Reinforcement learning (RL) is a set of iterative algorithms that allow a system to learn its optimal behavior as it interacts with its environment [1], [2]. In the context of linear optimal control, RL has been used in the last few decades to solve the linear quadratic regulator (LQR) problem in continuous-time [3], [4], [5], [6], [7], [8] and in discrete time [9], [10], [11], [12]. For applications of RL procedures to nonlinear systems and other extensions, the reader is referred to the surveys [13], [14], [15] and the references therein.

In the continuous-time linear time-invariant (CT-LTI) case, several RL algorithms with attractive properties have been designed. Although the first proposed algorithms required at least partial knowledge of the system model (e.g., [3]), completely data-based methods are now well known [4], [5], [6], [7]. These data-based algorithms replace the need for model knowledge by measuring persistently excited data directly from the system. Most of these data-based methods are *on-policy* algorithms, meaning that they require the application (or simulation) of an exciting input to the system at every iteration, such that a new set of data can be collected. In contrast, the authors in [8] proposed a data-based *off-policy* RL algorithm. This method has the advantage of requiring to collect data from the system only once, and then every iteration of the algorithm is performed using the same batch of measurements.

The method in [8], as well as most on-policy methods, is formulated as the problem of determining the values of certain unknown matrices from a set of equations derived from the Bellman equation. Taking advantage of the properties of the Kronecker product, this problem is then expressed as a set of linear equations that can be easily solved. However, the Kronecker product formulation generates matrices of large size, and this procedure presents a high computational burden that increases rapidly with the system dimension.

Another important issue in the existing learning-based control literature is the selection of a proper persistently exciting (PE) input. In most of the above literature, heuristic approaches for persistence of excitation are employed, often designing exciting inputs by adding sinusoidal, exponential and/or random signals [14]. A different approach for persistence of excitation was studied in [16], where conditions for the design of a discrete-time PE input are formally established. It is shown in [16] that their definition of persistence of excitation provides data measurements that are so rich in information that every possible trajectory of a controllable discrete-time linear system can be expressed in terms of such data. This result is now known as Willems' lemma, and has been successfully used in recent years in data-based analysis, estimation and control of discrete-time systems (see, e.g., the survey [17] and the references therein). In [6], it was proposed to use a PE signal as defined in [16] to excite a continuous-time system during a Q-learning procedure, which guarantees solvability of their policy evaluation step. However, the method in [6] is an *on-policy* algorithm and the authors require persistence of excitation of a signal composed of both the input and the state of the system. This contrasts with our objective of considering a PE signal in terms of the input only. Moreover, in [6] a high order of persistence of excitation is needed.

The contributions of this paper are as follows. We propose a novel data-based *off-policy* RL algorithm to solve the LQR problem for continuous-time systems. As in [8], we perform the policy evaluation and policy improvement steps simultaneously. Different from the existing algorithms, we formulate a Sylvester-transpose equation that can be efficiently solved using known methods [18], [19], [20]. This avoids the use of the Kronecker product and the ensuing large matrices in our computations. Moreover, we use the results in [21], where a continuous-time version of Willems' lemma was proposed. This allows us to design a PE input that guarantees the solvability of the Sylvester-transpose equation in a data-based fashion. In our formulation, persistence of excitation depends only on the input of the system, and we require the use of a PE input of lower order compared to [6]. Finally, we propose a method to determine the required initial stabilizing policy for the proposed algorithm using only measured data.

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 948679).

V. G. Lopez and M. A. Müller are with the Leibniz University Hannover, Institute of Automatic Control, 30167 Hannover, Germany {lopez,mueller}@irt.uni-hannover.de

Different from [7], this method does not require the solution of linear matrix inequalities (LMIs).

In the following, Section II introduces the preliminary results that are used throughout the paper. The development of the proposed efficient RL algorithm and its theoretical analysis are shown in Section III. Section IV analyses the computational efficiency of the proposed algorithm and presents a procedure to compute the initial stabilizing gain. In Section V, we illustrate the theoretical results with numerical examples, and Section VI concludes the paper.

II. PRELIMINARIES

In this section, we present existing results from the literature that are relevant for the remainder of this paper.

A. Matrix definitions for continuous-time data

Consider the integer $N \in \mathbb{N}$ and the positive scalar $T \in \mathbb{R}_+$. Let $\xi : [0, NT] \to \mathbb{R}^\sigma$, with $[0, NT] \subset \mathbb{R}$, denote a continuous-time signal of length NT. Using the trajectory ξ, we define the following matrix

$$\mathcal{H}_T(\xi(t)) := \begin{bmatrix} \xi(t) & \xi(t+T) & \cdots & \xi(t+(N-1)T) \end{bmatrix} \tag{1}$$

for $0 \leq t \leq T$. Notice that (1) is a time-varying matrix defined on the interval $t \in [0, T]$.

Now, consider the following CT-LTI system

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{2}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state and input vectors of the system, respectively. The pair (A, B) is assumed to be controllable throughout the paper.

Suppose that the input signal $u : [0, NT] \to \mathbb{R}^m$ is applied to (2), and the resulting state trajectory $x : [0, NT] \to \mathbb{R}^n$ is collected. From (2) and the definition in (1), we can write

$$\mathcal{H}_T(\dot{x}(t)) = A\mathcal{H}_T(x(t)) + B\mathcal{H}_T(u(t)).$$

Since it is unusual to have the state derivative $\dot{x}$ available as a measurement, integrate the expression above to obtain

$$\begin{aligned} &\mathcal{H}_T(x(T)) - \mathcal{H}_T(x(0)) \\ &\qquad = A\int_0^T \mathcal{H}_T(x(\tau))d\tau + B\int_0^T \mathcal{H}_T(u(\tau))d\tau. \end{aligned}$$

For convenience of notation, define the matrices

$$\tilde{X} = \mathcal{H}_T(x(T)) - \mathcal{H}_T(x(0)), \tag{3}$$

$$X = \int_0^T \mathcal{H}_T(x(\tau))d\tau, \quad U = \int_0^T \mathcal{H}_T(u(\tau))d\tau.$$

Notice that the matrix X (and similarly U) only requires the computation of integrals of the form $\int_0^T x(\tau + jT)d\tau$, $j = 0, \ldots, N-1$. This corresponds to integrating vectors of smaller dimension when compared to the existing methods in the RL literature [6], [7], [8].

By definition, the following expression holds

$$\tilde{X} = AX + BU. \tag{4}$$

B. Persistence of excitation for discrete-time systems

Define the integer constants $L, N \in \mathbb{N}$. The Hankel matrix of depth L of a discrete-time sequence $\{\mu_k\}_{k=0}^{N-1} = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, $\mu_k \in \mathbb{R}^m$, is defined as

$$H_L(\mu) := \begin{bmatrix} \mu_0 & \mu_1 & \cdots & \mu_{N-L} \\ \mu_1 & \mu_2 & \cdots & \mu_{N-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{L-1} & \mu_L & \cdots & \mu_{N-1} \end{bmatrix}.$$

In [16], the following definition of a PE input for discrete-time systems is made.

Definition 1: The discrete sequence $\{\mu_k\}_{k=0}^{N-1}$, $\mu_k \in \mathbb{R}^m$, is said to be persistently exciting of order L if its Hankel matrix of depth L has full row rank, i.e.,

$$\text{rank}(H_L(\mu)) = mL. \tag{5}$$

It is important to highlight the fact that Definition 1 provides a condition that enables a straightforward design of a PE input and that is easy to verify for any discrete sequence.

Remark 1: A necessary condition for (5) to hold is that $N \geq (m+1)L - 1$. This provides a minimum length for a PE input sequence.

C. Persistence of excitation for continuous-time systems

It is shown in [21] that a piecewise constant input designed by exploiting Definition 1 is persistently exciting for the continuous-time system (2). This class of inputs is formally described in the following definition.

Definition 2 (Piecewise constant PE input): Consider a time interval $T > 0$ such that

$$T \neq \frac{2\pi k}{|\mathcal{I}_m(\lambda_i - \lambda_j)|}, \qquad \forall k \in \mathbb{Z}. \tag{6}$$

where λ_i and λ_j are any two eigenvalues of matrix A in (2), and $\mathcal{I}_m(\cdot)$ is the imaginary part of a complex number. A piecewise constant persistently exciting (PCPE) input of order L for continuous-time systems is defined as $u(t + iT) = \mu_i$ for all $0 \leq t < T$, $i = 0, \ldots, N-1$, where $\{\mu_i\}_{i=0}^{N-1}$ is a sequence of constant vectors $\mu_i \in \mathbb{R}^m$ that is persistently exciting of order L in the sense of Definition 1.

Remark 2: Notice that the condition (6) is not restrictive, even with no knowledge of the system model (2). This is because the values of T that make this condition fail form a set of measure zero and are unlikely to be encountered in practice.

When a PCPE input is applied to system (2), the obtained input-state data set satisfies an important rank condition, as shown below.

Lemma 1 ([21]): Consider system (2), let the pair (A, B) be controllable, and let u be a PCPE input of order $n+1$ as defined in Definition 2. Then, the rank condition

$$\text{rank}\left(\begin{bmatrix} \mathcal{H}_T(x(t)) \\ \mathcal{H}_T(u(t)) \end{bmatrix}\right) = \text{rank}\left(\begin{bmatrix} \mathcal{H}_T(x(t)) \\ H_1(\mu) \end{bmatrix}\right) = m + n \tag{7}$$

holds for all $0 \leq t \leq T$.

Remark 3: In [21], the result in Lemma 1 was presented considering persistence of excitation of any order L. For simplicity of notation, we presented Lemma 1 directly for PE inputs of order $n+1$. This is the only order of persistence of excitation used throughout the paper.

D. The LQR problem and Kleinman's algorithm

For a CT-LTI system (2), the infinite-horizon LQR problem concerns determining the control input u that minimizes a cost function of the form

$$J(x(0),u) := \int_0^\infty \left(x^\top(t)Qx(t) + u^\top(t)Ru(t)\right) dt, \quad (8)$$

where $Q \succeq 0$ and $R \succ 0$. Throughout the paper, we assume that the pair $(A, Q^{1/2})$ is observable. This, together with the assumed controllability of (A, B), implies that the optimal control input is given by $u^*(x) = -K^*x$, where

$$K^* = R^{-1}B^\top P^*$$

and the matrix $P^* \succ 0$ solves the algebraic Riccati equation

$$Q + P^*A + A^\top P^* - P^*BR^{-1}B^\top P^* = 0.$$

In [22], Kleinman proposed a model-based iterative algorithm to solve the LQR problem. This algorithm starts by selecting an initial stabilizing matrix K_0, i.e., a matrix such that $A - BK_0$ is Hurwitz stable. At every iteration i, the Lyapunov equation

$$P_i(A - BK_i) + (A - BK_i)^\top P_i + Q + K_i^\top RK_i = 0 \quad (9)$$

is solved for P_i. Then, a new feedback matrix is defined as

$$K_{i+1} = R^{-1}B^\top P_i. \quad (10)$$

The algorithm iterates the equations (9) and (10) until convergence. With the main drawback of being a model-based method, Kleinman's algorithm otherwise possesses highly attractive features. Namely, at each iteration the matrix K_{i+1} is stabilizing, the algorithm converges such that

$$\lim_{i\to\infty} K_{i+1} = K^*,$$

and convergence occurs at a quadratic rate [22].

The following section presents the main developments of this paper.

III. AN EFFICIENT DATA-BASED ALGORITHM FOR THE CT LQR PROBLEM

In this section, we present an efficient data-based off-policy RL algorithm to determine the optimal controller that minimizes (8). We show that the proposed procedure is equivalent to Kleinman's algorithm (9)-(10), and therefore preserves all of its theoretical properties. For the clarity of exposition, we introduce first a model-based algorithm that is then used as the basis of our data-based method.

A. A model-based algorithm

Combining (9) and (10), we readily obtain the following expressions

$$P_iA - K_{i+1}^\top RK_i + A^\top P_i - K_i^\top RK_{i+1} + Q + K_i^\top RK_i = 0$$

and $B^\top P_i - RK_{i+1} = 0$. Therefore, the matrix equation

$$\begin{bmatrix} A & B \\ -RK_i & -R \end{bmatrix}^\top \begin{bmatrix} P_i \\ K_{i+1} \end{bmatrix} \begin{bmatrix} I_n & 0 \end{bmatrix} + \begin{bmatrix} I_n \\ 0 \end{bmatrix} \begin{bmatrix} P_i \\ K_{i+1} \end{bmatrix}^\top \begin{bmatrix} A & B \\ -RK_i & -R \end{bmatrix} + \begin{bmatrix} Q + K_i^\top RK_i & 0 \\ 0 & 0 \end{bmatrix} = 0 \quad (11)$$

holds, where I_n is an $n \times n$ identity matrix and 0 represents a matrix of zeros with appropriate dimensions.

Denoting the fixed matrices as

$$\Phi_i := \begin{bmatrix} A & B \\ -RK_i & -R \end{bmatrix}, \quad E := \begin{bmatrix} I_n & 0 \end{bmatrix}, \quad \bar{Q}_i := \begin{bmatrix} Q + K_i^\top RK_i & 0 \\ 0 & 0 \end{bmatrix} \quad (12)$$

and the unknown matrix as

$$\Theta_{i+1} := \begin{bmatrix} P_i \\ K_{i+1} \end{bmatrix}, \quad (13)$$

we can write (11) in the compact form

$$\Phi_i^\top \Theta_{i+1} E + E^\top \Theta_{i+1}^\top \Phi_i + \bar{Q}_i = 0. \quad (14)$$

The matrix $\Theta_{i+1} \in \mathbb{R}^{(n+m)\times n}$ consists of the unknown matrices in Kleinman's algorithm, P_i and K_{i+1}. It is of our interest to design a method in which solving a matrix equation as in (14) for Θ_{i+1} corresponds to solving both (9) and (10) simultaneously. However, it can be noted that (14), as it is formulated, in general does not have a unique solution Θ_{i+1}. To address this issue, first express the unknown submatrices of Θ_{i+1} as

$$\Theta_{i+1} = \begin{bmatrix} \Theta_{i+1}^1 \\ \Theta_{i+1}^2 \end{bmatrix}, \quad (15)$$

with $\Theta_{i+1}^1 \in \mathbb{R}^{n\times n}$ and $\Theta_{i+1}^2 \in \mathbb{R}^{m\times n}$. In the following lemma, we show that there exists only one matrix Θ_{i+1} that solves (14) such that the submatrix Θ_{i+1}^1 is symmetric.

Lemma 2: Consider the equation (14) with the matrices Φ_i, E and $\bar{Q}_i$ defined as in (12). Moreover, let the matrix K_i be stabilizing. Then, there exists a unique solution (15) to this equation for which $\Theta_{i+1}^1 = (\Theta_{i+1}^1)^\top$.

Proof: Considering the partition in (15), notice that (14) holds for any matrix Θ_{i+1} such that

$$A^\top\Theta_{i+1}^1 - K_i^\top R\Theta_{i+1}^2 + (\Theta_{i+1}^1)^\top A - (\Theta_{i+1}^2)^\top RK_i + Q + K_i^\top RK_i = 0,$$

and

$$B^\top\Theta_{i+1}^1 - R\Theta_{i+1}^2 = 0.$$

From the second equation it is clear that $\Theta_{i+1}^2 = R^{-1}B^\top\Theta_{i+1}^1$. Substituting this and the fact that $\Theta_{i+1}^1 = (\Theta_{i+1}^1)^\top$ in the first equation, we get

$$(A - BK_i)^\top\Theta_{i+1}^1 + \Theta_{i+1}^1(A - BK_i) + Q + K_i^\top RK_i = 0. \quad (16)$$

Since K_i is stabilizing, we use Lyapunov arguments to

conclude that Θ^1_{i+1} (and therefore also Θ^2_{i+1}) is unique. ■

Lemma 2 implies that constraining the solution of (14) to include a symmetric submatrix Θ^1_{i+1} leads to the desired solution (13). The following lemma shows that we achieve this by properly modifying Φ_i in (12).

Lemma 3: Consider the matrix equation

$$(\Phi_i^-)^\top \Theta_{i+1} E + E^\top \Theta_{i+1}^\top \Phi_i^+ + \bar{Q}_i = 0, \tag{17}$$

where

$$\Phi_i^+ := \begin{bmatrix} A+I & B \\ -RK_i & -R \end{bmatrix}, \quad \Phi_i^- := \begin{bmatrix} A-I & B \\ -RK_i & -R \end{bmatrix}, \tag{18}$$

and the matrices E and $\bar{Q}_i$ are defined as in (12). Moreover, let the matrix K_i be stabilizing. Then, the solution (15) of (17) is unique, and $\Theta^1_{i+1} = (\Theta^1_{i+1})^\top$. Moreover, the solution of (17) is also a solution of (14).

Proof: First, define the matrix

$$S = \begin{bmatrix} \Theta^1_{i+1} - (\Theta^1_{i+1})^\top & 0 \\ 0 & 0 \end{bmatrix}.$$

Using this definition, it is straightforward to express (17) in terms of the matrix Φ_i in (12) as

$$\Phi_i^\top \Theta_{i+1} E + E^\top \Theta_{i+1}^\top \Phi_i + \bar{Q}_i = S.$$

Notice that the left-hand side of this expression is symmetric, and therefore so must be S. Now, S is symmetric if and only if $\Theta^1_{i+1} = (\Theta^1_{i+1})^\top$, that is, $S = 0$. This implies both that the solution of (17) also solves (14) and, by Lemma 2, that this solution is unique. ■

Remark 4: Equation (17) is a case of the generalized Sylvester-transpose equation, and algorithms to solve it efficiently are well known [18], [19], [20].

Using this result, we formulate Algorithm 1 below. As in any policy iteration procedure, Algorithm 1 is initialized with a stabilizing matrix K_0. Using this matrix (as well as model knowledge), (17) is solved for Θ_{i+1}. Then, partitioning Θ_{i+1} as in (15), a new feedback matrix is obtained as $K_{i+1} = \Theta^2_{i+1}$.

Algorithm 1: Model-based RL algorithm

1: **procedure**
2: Let $i = 0$ and initialize a stabilizing feedback matrix K_0.
3: Using the definitions in (12) and (18), solve for Θ_{i+1} from the equation

$$(\Phi_i^-)^\top \Theta_{i+1} E + E^\top \Theta_{i+1}^\top \Phi_i^+ + \bar{Q}_i = 0.$$

4: Partitioning Θ_{i+1} as in (15), define

$$K_{i+1} = \Theta^2_{i+1}.$$

5: If $\|K_{i+1} - K_i\| > \varepsilon$ for some $\varepsilon > 0$, let $i = i + 1$ and go to Step 3. Otherwise, stop.
6: **end procedure**

Using the results obtained so far, we conclude that Algorithm 1 is equivalent to Kleinman's algorithm in the sense that, starting from the same initial matrix K_0, they provide the same updated policies K_{i+1} at every iteration. This implies that Algorithm 1 preserves all the properties of Kleinman's algorithm. In the following, we use this result to design a data-based algorithm.

B. The data-based algorithm

To avoid the need for model knowledge in Algorithm 1, we collect persistently excited data from the system (2) as described in Section II-C. Using this data, we define the constant matrices X, U and $\tilde{X}$ as in (3).

Lemma 1 showed that the collected data set satisfies the rank condition (7). In the following lemma, we extend this result to the matrices X and U.

Lemma 4: Consider system (2), let the pair (A, B) be controllable, and let u be a PCPE input of order $n + 1$ as defined in Definition 2. Using the resulting input-state data, define the matrices X and U as in (3). Then,

$$\operatorname{rank}\left(\begin{bmatrix} X \\ U \end{bmatrix}\right) = n + m. \tag{19}$$

Proof: Notice that, since the applied input is piecewise constant, an expression for the resulting state of (2) is

$$x(t + iT) = e^{At}x(iT) + \int_0^t e^{A\tau} d\tau B\mu_i,$$

for $i = 0, \ldots, N-1$ and $0 \le t \le T$. Thus, we can write

$$\begin{aligned} \begin{bmatrix} X \\ U \end{bmatrix} &= \int_0^T \begin{bmatrix} \mathcal{H}_T(x(\tau)) \\ H_1(\mu) \end{bmatrix} d\tau \\ &= \underbrace{\int_0^T \begin{bmatrix} e^{A\tau} & \int_0^\tau e^{As} ds B \\ 0 & I \end{bmatrix} d\tau}_{W} \begin{bmatrix} \mathcal{H}_T(x(0)) \\ H_1(\mu) \end{bmatrix}. \end{aligned}$$

Notice that W is nonsingular since the condition (6) holds (the fact that $\int_0^T e^{A\tau} d\tau$ is nonsingular follows from the fact that T corresponds to a non-pathological sampling time [23]). Moreover, by Lemma 1 the second matrix on the right-hand side has full row rank, completing the proof. ■

Define $Z = [X^\top \quad U^\top]^\top$. Since Z has full row rank by Lemma 4, we can select $n+m$ linearly independent columns from it. Let z_k represent the kth column of Z, and let $\eta = \{k_1, \ldots, k_{n+m}\}$ be a set of indices such that

$$Z_\eta := \begin{bmatrix} z_{k_1} & \cdots & z_{k_{n+m}} \end{bmatrix} \tag{20}$$

is a nonsingular matrix. Then, Θ_{i+1} is a solution of (17) if and only if it is a solution of

$$Z_\eta^\top (\Phi_i^-)^\top \Theta_{i+1} E Z_\eta + Z_\eta^\top E^\top \Theta_{i+1}^\top \Phi_i^+ Z_\eta + Z_\eta^\top \bar{Q}_i Z_\eta = 0. \tag{21}$$

From the definitions in (12) and (18), and using the expression (4), we have the following

$$\Phi_i^+ Z_\eta = \begin{bmatrix} AX_\eta + X_\eta + BU_\eta \\ -RK_iX_\eta - RU_\eta \end{bmatrix} = \begin{bmatrix} \tilde{X}_\eta + X_\eta \\ -RK_iX_\eta - RU_\eta \end{bmatrix},$$

$$\Phi_i^- Z_\eta = \begin{bmatrix} AX_\eta - X_\eta + BU_\eta \\ -RK_iX_\eta - RU_\eta \end{bmatrix} = \begin{bmatrix} \tilde{X}_\eta - X_\eta \\ -RK_iX_\eta - RU_\eta \end{bmatrix},$$

$Z_\eta^\top \bar{Q}_i Z_\eta = X_\eta^\top (Q + K_i^\top R K_i) X_\eta$ and $EZ_\eta = X_\eta$, where the subindex η represents a matrix constructed using the columns specified by the set η from the corresponding

original matrix as in (20), e.g.,

$$X_\eta := \begin{bmatrix} x_{k_1} & \cdots & x_{k_{n+m}} \end{bmatrix}.$$

Substituting in (21), we obtain

$$(Y_i^-)^\top \Theta_{i+1} X_\eta + X_\eta^\top \Theta_{i+1}^\top Y_i^+ + X_\eta^\top (Q + K_i^\top R K_i) X_\eta = 0. \tag{22}$$

where

$$\begin{aligned} Y_i^- &:= \begin{bmatrix} \tilde{X}_\eta - X_\eta \\ -RK_iX_\eta - RU_\eta \end{bmatrix}, \\ Y_i^+ &:= \begin{bmatrix} \tilde{X}_\eta + X_\eta \\ -RK_iX_\eta - RU_\eta \end{bmatrix}. \end{aligned} \tag{23}$$

Now, (22) is a data-based equation that does not require any knowledge about the system model. Algorithm 2 uses this expression to solve the LQR problem. For convenience, for Algorithm 2 we define

$$Q_i := X_\eta^\top (Q + K_i^\top R K_i) X_\eta. \tag{24}$$

Algorithm 2: Data-based RL algorithm

1: **procedure**
2: Select $N \geq (n+1)m + n$ and $T > 0$, apply a PCPE input of order $n+1$ to (2) and collect an NT-long input-state trajectory.
3: Compute the matrices X, U, and $\tilde{X}$ as in (3).
4: Select a set of indices $\eta = \{k_1, \ldots, k_{n+m}\}$ such that $[X_\eta^\top \quad U_\eta^\top]^\top$ is nonsingular.
5: Let $i = 0$ and initialize a stabilizing feedback matrix K_0.
6: Define the matrices Y_i^+, Y_i^- and Q_i as in (23)-(24), and solve for Θ_{i+1} from the equation

$$(Y_i^-)^\top \Theta_{i+1} X_\eta + X_\eta^\top \Theta_{i+1}^\top Y_i^+ + Q_i = 0. \tag{25}$$

7: Partitioning Θ_{i+1} as in (15), define

$$K_{i+1} = \Theta_{i+1}^2.$$

8: If $\|K_{i+1} - K_i\| > \varepsilon$ for some $\varepsilon > 0$, let $i = i+1$ and go to Step 6. Otherwise, stop.
9: **end procedure**

The following theorem states the main properties of this algorithm.

Theorem 1: Consider the CT-LTI system (2), and the partitioning (15) of Θ_{i+1}. Every iteration of Algorithm 2 has the following properties: (i) the solution Θ_{i+1} of (25) exists and is unique; (ii) the gain K_{i+1} is stabilizing; and (iii) $\Theta_i^1 \succeq \Theta_{i+1}^1 \succeq P^*$. Moreover,

$$\lim_{i \to \infty} K_i = K^*$$

and the rate of convergence of the algorithm is quadratic.

Proof: The proof is obtained by showing that Algorithm 2 is equivalent to Kleinman's algorithm at every iteration. First, notice that by Lemma 4, the matrix $[X^\top \quad U^\top]^\top$ has full row rank and, therefore, a nonsingular matrix $[X_\eta^\top \quad U_\eta^\top]^\top$ can always be constructed. This means that (25) is equivalent to (17). Now, noting that K_0 is stabilizing, use an induction argument to assume that K_i is stabilizing. Lemma 3 shows the existence and uniqueness of Θ_{i+1} from (17). Moreover, the expression (16) in the proof of Lemma 2 shows that $\Theta_{i+1}^1 = P_i$, where P_i is the solution of the Lyapunov equation (9). Also in the proof of Lemma 2 it was shown that $\Theta_{i+1}^2 = R^{-1}B^\top \Theta_{i+1}^1$, which now corresponds to Kleinman's updated gain (10). Therefore, Algorithm 2 is equivalent to Kleinman's algorithm and shares all of its properties [22]. ■

Algorithm 2 is a purely data-based, off-policy method to solve the continuous-time LQR problem. Using Definition 2, we are able to guarantee the existence of a solution Θ_{i+1} of (25) at every iteration for data trajectories of fixed length. This contrasts with the methods in the literature that must keep collecting data until a matrix gets full rank, such as, e.g., [7], [8]. Moreover, we avoid the use of the Kronecker product and its resulting large matrices in Algorithm 2. As stated in Remark 4, methods to efficiently solve a Sylvester-transpose equation as in (25) are well known.

Remark 5: Step 4 of Algorithm 2 instructs to select $n+m$ linearly independent columns of $Z = [X^\top \quad U^\top]^\top$. This step is only performed in benefit of efficiency, as it decreases the size of the matrices in (25). However, since Z has full row rank, skipping this step in Algorithm 2 and using the complete data matrices instead does not affect the result at each iteration.

IV. PRACTICAL CONSIDERATIONS

A. Efficiency analysis of Algorithm 2

In this subsection, we analyze the theoretical computational complexity of Algorithm 2. Moreover, we compare this complexity with that of the algorithm proposed in [8]. This is because [8] is also an off-policy data-based method that shares many of the characteristics of Algorithm 2.

The most expensive steps in Algorithm 2 are obtaining the solution of (25) and selecting $n+m$ linearly independent vectors from $[X^\top \quad U^\top]^\top$. Methods to solve the Sylvester-transpose equation (25) with a complexity of $\mathcal{O}((n+m)^3)$ are known [19]. The selection of linearly independent vectors can be performed using a simple procedure like Gaussian elimination to transform the matrix of interest into row echelon form. This method has a complexity of $\mathcal{O}((n+m)^2 N)$ operations [24]. This step, however, only needs to be performed once in Algorithm 2 (in Step 4). Thus, we conclude that Algorithm 2 requires once $\mathcal{O}((n+m)^2 N)$ and then in each iteration $\mathcal{O}((n+m)^3)$ floating point operations.

The algorithm in [8] was also shown to be equivalent to Kleinman's algorithm at every iteration. However, their method uses a Kronecker product formulation that yields matrices of large dimensions. Let $N_\otimes$ be the amount of data samples used in [8]. Then, the most expensive step at each iteration of their algorithm is the product of a matrix with dimensions $(\frac{1}{2}n(n+1) + mn) \times N_\otimes$ times its transpose. This product, and hence each iteration of the algorithm, requires $\mathcal{O}((\frac{1}{2}n(n+1) + mn)^2 N_\otimes)$ floating point operations [25]. Clearly, as the dimension of the system increases, the difference in performance of both algorithms becomes more significant. Moreover, we notice from [8] that the amount of collected data must satisfy $N_\otimes \geq \frac{1}{2}n(n+1) + mn$ for the algorithm to yield a unique solution at every iteration.

Compare this with the bound $N \geq (n+1)m+n$ in Algorithm 2. In Section V, we test this theoretical comparison using numerical examples.

B. An initial stabilizing policy

In [26, Remark 2], a procedure to design a stabilizing controller for continuous-time systems using only measured data was described. This method is based on the solution of a linear matrix inequality (LMI). The authors in [7] proposed to use a similar LMI-based procedure to determine the initial stabilizing gain for a Q-learning algorithm. Since one of the goals in this paper is computational efficiency, we would like to avoid the computationally expensive step of solving an LMI. In this subsection, we present an alternative method to determine the initial stabilizing matrix K_0 for Algorithm 2. The following development follows closely a procedure proposed in [27, Section IV] for discrete-time systems.

Let F be the Moore-Penrose pseudoinverse of the matrix X in (3). Since X has full row rank (see Lemma 4), F is a right inverse of X. Furthermore, let G be a basis for the null space of X, such that $X(F - G\bar{K}) = I$ for any matrix $\bar{K}$ of appropriate dimensions. Using the matrices F, G and U from (3), we propose to compute the initial stabilizing gain K_0 for Algorithm 2 as

$$K_0 = -U(F - G\bar{K}) \tag{26}$$

where $\bar{K}$ is a matrix to be determined.

From (4) and (26), notice that

$$\begin{aligned}\tilde{X}(F - G\bar{K}) &= [A \quad B]\begin{bmatrix} X \\ U \end{bmatrix}(F - G\bar{K}) \\ &= [A \quad B]\begin{bmatrix} I \\ -K_0 \end{bmatrix}\end{aligned}$$

Therefore, by designing the poles of the matrix $\tilde{X}(F - G\bar{K})$, we also set the poles of $A - BK_0$. Since (A, B) is controllable and hence the poles of $A-BK_0$ can be assigned arbitrarily, also the poles of $\tilde{X}(F - G\bar{K})$ can be placed arbitrarily by a suitable choice of $\bar{K}$. Moreover, since $\tilde{X}$, F and G are matrices obtained from data, we can operate with them without any need of model knowledge. This procedure is summarized in the following theorem. The proof of this theorem is straightforward considering the procedure described in this subsection and is hence omitted.

Theorem 2: Let the matrices $\tilde{X}$, X and U be defined as in (3) using data collected from (2) during the application of a PCPE input of order $n+1$. Define F as the Moore-Penrose pseudoinverse of X and G as a basis for the null space of X. Moreover, define the *virtual* system matrices $\bar{A} = \tilde{X}F$ and $\bar{B} = \tilde{X}G$. Using pole-placement methods, determine a matrix $\bar{K}$ such that $\bar{A} - \bar{B}\bar{K}$ is Hurwitz. Then, the matrix K_0 defined by (26) is stabilizing for system (2).

Remark 6: Notice that the matrices $\bar{A} = \tilde{X}F$ and $\bar{B} = \tilde{X}G$ in Theorem 2 do not correspond to the actual system matrices A and B. In fact, B and $\bar{B}$ in general do not have the same dimensions. No model identification is performed in the proposed procedure.

V. NUMERICAL EXPERIMENTS

In this section, we compare in simulation the efficiency of the proposed Algorithm 2 with that of the algorithm presented in [8]. As described above, these algorithms have the same characteristics: they are data-based off-policy methods that are equivalent to Kleinman's algorithm at every iteration.

To compare the efficiency of both algorithms, several simulations are performed for different, randomly generated linear systems (2). In particular, 100 different linear systems are generated using the command *rss* in Matlab, and both algorithms are applied to each of them. The system dimensions considered for each set of 100 experiments are $n = 2, 3, 5$ and 7. In every case, we consider single input systems ($m = 1$), and we define the cost function (8) with $Q = I$ and $R = 2$.

Each implementation of Algorithm 2 had the following characteristics. A PCPE input as in Definition 2 was used to collect data from the system. A sample of data was collected every 10^{-4} time units. We considered a time interval of $T = 0.2$, and we collected data for a total of NT time units, with $N = (n+1)m+n$. The method described in [18] was used to solve the Sylvester-transpose equation (25) at every iteration.

For the implementation of the Kronecker product-based method in [8], we followed the same simulation characteristics described in the simulation section of that paper. The only exception is in the amount of data collected, which was reduced for small system dimensions in order to make a fairer comparison.

Finally, notice that the command *rss* in Matlab yields stable systems. Thus, an initial stabilizing matrix of $K_0 = 0$ was used for all experiments and both algorithms. The simulations were performed using Matlab R2020b on an Intel i7-10875H (2.30 GHz) with 16 GB of memory.

The results of our simulations are displayed in Table I. In this table, we refer to Algorithm 2, which is based on the solution of a Sylvester-transpose equation, as 'SYL'. The algorithm in [8] that is based on the use of the Kronecker product is denoted as 'KRO'. To compare the computational efficiency of the methods, we present the average time that it takes the algorithms to complete 10 iterations. Due to their quadratic rate of convergence, 10 iterations yield a very accurate result of the optimal control gain for both algorithms. In the table we can observe a confirmation of our theoretical analysis regarding the improved performance of Algorithm 2.

During the execution of these experiments, we noted some issues in the performance of both methods when applied to systems of large dimensions. First, Algorithm 2 requires the application of a solver from the literature to solve (25). We found that, if the data matrix Z_η in Algorithm 2 has a large condition number, the solvers considered often failed to provide the correct result. To address this problem, methods to construct a matrix Z_η with low condition number from a larger matrix Z could be considered. Regarding the algorithm in [8], determining a proper input in order to satisfy the required persistence of excitation condition for the collected

Dimension	Average time (sec)	
n	SYL	KRO
2	0.0099	0.0953
3	0.0144	0.1719
5	0.0287	0.4317
7	0.1046	1.8021

TABLE I

RUN-TIME COMPARISON BETWEEN ALGORITHM 2 (SYL) AND THE ALGORITHM IN [8] (KRO).

data (compare the discussion in the Introduction) becomes ever more difficult as the dimension of the system increases. In this case, it is uncertain how to solve this issue.

A Matlab code for Algorithm 2 has been made publicly available in https://codeocean.com/capsule/3042979/tree.

VI. CONCLUSIONS

In this paper, a computationally efficient algorithm was proposed to solve the continuous-time LQR problem. The proposed algorithm is equivalent to Kleinman's method, it does not require any knowledge from the system model and it requires collecting data from the system only once. We presented a persistently exciting input that guarantees that the matrix equation (25) in our algorithm has a unique solution at every iteration. Finally, we showed a method to determine an initial stabilizing feedback matrix using only measured data and that does not require to solve LMIs. Simulation results show that our algorithm significantly improves the performance of an algorithm with similar properties in the literature.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA: Athena Scientific, 2005, vol. I.

[3] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis, "Adaptive optimal control for continuous-time linear systems based on policy iteration," *Automatica*, vol. 45, pp. 477–484, 2009.

[4] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Systems & Control Letters*, vol. 100, pp. 14–20, 2017.

[5] H. Mohammadi, M. Soltanolkotabi, and M. R. Jovanovic, "Random search for learning the linear quadratic regulator," in *2020 American Control Conference (ACC)*, 2020, pp. 4798–4803.

[6] J. Y. Lee, J. B. Park, and Y. H. Choi, "Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems," *Automatica*, vol. 48, no. 11, pp. 2850–2859, 2012.

[7] C. Possieri and M. Sassano, "Q-learning for continuous-time linear systems: A data-driven implementation of the Kleinman algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 10, pp. 6487–6497, 2022.

[8] Y. Jiang and Z.-P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, pp. 2699–2704, 2012.

[9] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," in *Proceedings of 1994 American Control Conference*, vol. 3, 1994, pp. 3475–3479.

[10] M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 1–10.

[11] V. G. Lopez, M. Alsalti, and M. A. Müller, "Efficient off-policy Q-learning for data-based discrete-time LQR problems," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2922–2933, 2023.

[12] Y. Yang, B. Kiumarsi, H. Modares, and C. Xu, "Model-free λ-policy iteration for discrete-time linear quadratic regulation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 635–649, 2023.

[13] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2042–2062, 2018.

[14] Z.-P. Jiang, T. Bian, and W. Gao, "Learning-based control: A tutorial and some recent results," *Foundations and Trends in Systems and Control*, vol. 8, no. 3, pp. 176–284, 2020.

[15] D. Liu, S. Xue, B. Zhao, B. Luo, and Q. Wei, "Adaptive dynamic programming for control: A survey and recent advances," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 142–160, 2021.

[16] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[17] I. Markovsky and F. Dörfler, "Behavioral systems theory in data-driven analysis, signal processing, and control," *Annual Reviews in Control*, vol. 52, pp. 42–64, 2021.

[18] M. Hajarian, "Extending the CGLS algorithm for least squares solutions of the generalized Sylvester-transpose matrix equations," *Journal of the Franklin Institute*, vol. 353, no. 5, pp. 1168–1185, 2016, special Issue on Matrix Equations with Application to Control Theory.

[19] K. Tansri, S. Choomklang, and P. Chansangiam, "Conjugate gradient algorithm for consistent generalized Sylvester-transpose matrix equations," *AIMS Mathematics*, vol. 7, no. 4, pp. 5386–5407, 2022.

[20] C. Song, G. Chen, and L. Zhao, "Iterative solutions to coupled Sylvester-transpose matrix equations," *Applied Mathematical Modelling*, vol. 35, no. 10, pp. 4675–4683, 2011.

[21] V. G. Lopez and M. A. Müller, "On a continuous-time version of Willems' lemma," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 2759–2764.

[22] D. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 114–115, 1968.

[23] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. New York, USA: Oxford University Press, 1999.

[24] G. Strang, *Linear Algebra and its Applications*, 4th ed. Belmont, USA: Thomson Brooks/Cole, 2006.

[25] R. Hunger, "Floating point operations in matrix-vector calculus," Tech. Rep., 2007. [Online]. Available: https://mediatum.ub.tum.de/doc/625604/625604

[26] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2020.

[27] H. J. van Waarde, J. Eising, H. L. Trentelman, and M. K. Camlibel, "Data informativity: A new perspective on data-driven analysis and control," *IEEE Transactions on Automatic Control*, vol. 65, no. 11, pp. 4753–4768, 2020.