2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Model-Free Optimal Control Based on Reinforcement Learning for Rotary Inverted Pendulum

Eduardo Yudho
Departamento de Control Automático
CINVESTAV-IPN
Mexico City, 0736, Mexico

Xiaoou Li
Departamento de Computación
CINVESTAV-IPN
Mexico City, 0736, Mexico

Brisbane Ovilla-Martínez
Departamento de Computación
CINVESTAV-IPN
Mexico City, 0736, Mexico

Wen Yu
Departamento de Control Automático
CINVESTAV-IPN
Mexico City, Mexico

***Abstract*—In this article, an algorithm based on reinforcement learning was used to design an optimal adaptive controller for the rotary inverted pendulum, without explicitly solving the algebraic Riccati equation and without knowing the dynamics of the system. The rotary inverted pendulum is an underactuated mechanical system with two degrees of freedom. Parametric variation was used to evaluate the capacity of the algorithm. The results were satisfactory.**

***Index Terms*—reinforcement learning, rotary inverted pendulum, LQR**

I. Introduction

The rotary inverted pendulum, also known as the Furuta pendulum [1]–[3], constitutes a mechanism composed of a horizontally rotating arm and a vertically rotating pendulum [4]. The system is equipped with a sole actuator, which imparts torque to the arm and indirectly influences the pendulum's position. Consequently, it embodies a underactuated mechanical system with two degrees of freedom. Because the rotary inverted pendulum is a highly non-linear, non-minimal phase, simple-structured, multivariable, and unstable system, it is frequently employed to validate the effectiveness and performance of various control algorithms [5].

Feedback control theory constitutes the systematic exploration of methodologies aimed at formulating control algorithms for human-engineered systems, ensuring the compliance of predetermined performance and safety criteria [6]. Feedback control systems operate on the fundamental principle of observing a system's output, juxtaposing it against the desired trajectory, and subsequently computing the requisite control signal to effectuate adjustments to the system's performance [7]. However, the application of conventional control strategies typically requires a degree of familiarity with the dynamics of the underlying system.

When the model of a dynamic system is known, it's possible to linearize it around an operating point and design linear controllers based on the model. A process of linearization about an operational point affords the opportunity to devise linear control strategies. Among these strategies, the linear quadratic regulator (LQR) is a notable example. The LQR controller is architected through an offline procedure involving the solution of the algebraic Riccati equation, thereby facilitating the minimization of a predetermined cost function. Nevertheless, it is noteworthy that alterations in system parameters can potentially undermine the optimality of the LQR controller, thereby engendering the need for periodic recalibration.

According to Lewis [6]–[8], adaptive control and optimal control represent two different philosophies of feedback controller design. Typically, computing optimal controllers for linear systems involves an offline design by solving the Hamilton-Jacobi-Bellman (HJB) equations, such as the Riccati equation, using complete knowledge of the dynamic system.

On the other hand, adaptive controllers learn online to control unknown systems based on measured information in real-time along the system trajectories. Usually, adaptive controllers are not designed to be optimal in the sense of minimizing user-specified performance functions, but they must satisfy certain conditions of inverse optimality.

Reinforcement learning, on the other hand, involves a cause-and-effect relationship between actions and rewards or penalties [9]. For this, reinforcement learning algorithms are built on the idea that effective control decisions should be reinforced through signals, making it more likely to reuse them. Additionally, reinforcement learning is based on evaluated environmental information in real time and can be seen as action-based learning [7]. Thus, from a theoretical point of view, reinforcement learning is related to adaptive and optimal control methods. This makes it possible to use reinforcement learning to solve optimal control problems.

Therefore, in this paper we present the use of an algorithm based on reinforcement learning to achieve online learning of the solution for a linear quadratic regulator applied to the rotary inverted pendulum. Subsequently, the algorithm's

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

development is presented, commencing with an analysis rooted in the Linear Quadratic Regulator (LQR) framework and the algebraic Riccati equation. Following this, the obtained results are presented through simulation, whereby a comparison between optimal solutions and solutions derived from reinforcement learning estimations is delineated.

II. Rotary inverted pendulum

The rotary inverted pendulum is a two-degree-of-freedom underactuated mechanical system with a horizontally rotating arm and a vertically rotating pendulum [4].

The geometric analysis of the rotational joints was conducted, as depicted in Fig. 1. The left side corresponds to the horizontal plane, encompassing the analysis of the arm and the projection of the pendulum onto this plane. The right side displays the pendulum on the vertical plane.

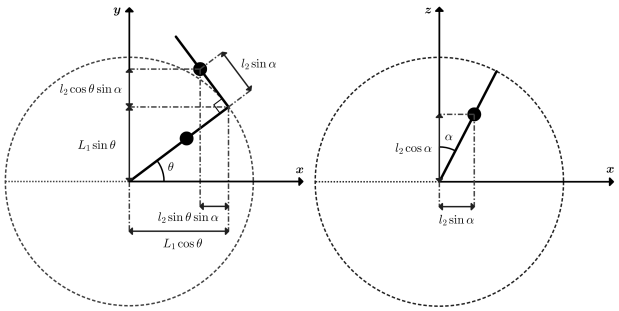

Fig. 1. The horizontal and vertical planes.

The arm has a length L_1, mass m_1, distance to the center of mass l_1, and a moment of inertia J_1. Similarly, the pendulum has a length L_2, mass m_2, distance to the center of mass l_2, and a moment of inertia J_2.

The motion equations for this kind of manipulator systems are the characteristic the manipulator equation:

$$M\left(q\right)\ddot{q} + C\left(q,\dot{q}\right)\dot{q} + G\left(q\right) = \tau \tag{1}$$

where q is a generalized coordinate vector, $M(q) \in \mathbb{R}^{n\times n}$ is the inertial matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ captures Coriolis forces, $G \in \mathbb{R}^n$ captures the gravitational torques and $\tau \in \mathbb{R}^n$ represents the generalized forces. For the rotary inverted pendulum, we have $q = [\theta, \alpha]^T$, where θ and α are the rotational angles for the arm and pendulum, respectively. and:

$$M(q) = \begin{bmatrix} \theta_1 + \theta_2 \sin^2 q_2 & \theta_3 \cos q_2 \\ \theta_3 \cos q_2 & \theta_4 \end{bmatrix}$$

$$C\left(q,\dot{q}\right) = \begin{bmatrix} \frac{1}{2}\theta_2 \sin 2q_2\dot{q}_2 & \frac{1}{2}\theta_2 \sin 2q_2\dot{q}_1 - \theta_3 \sin q_2\dot{q}_2 \\ -\frac{1}{2}\theta_2 \sin 2q_2\dot{q}_1 & 0 \end{bmatrix}$$

$$G\left(q\right) = [0, -\theta_5 \sin q_2]^T,\ \tau = [\tau_1, 0]^T$$

with $\theta_1 = J_1 + m_1 l_1^2 + m_2 L_1^2$, $\theta_2 = m_2 l_2^2$, $\theta_3 = m_2 L_1 l_2$, $\theta_4 = I_2 + m_2 l_2^2$ and $\theta_5 = g m_2 l_2$.

Taking into consideration viscous friction in the rotational joints, b_1 and b_2, and the electromechanical model of the actuator with the constant β_m, the linear state-space representation of the system is sought in the form $\dot{x} = Ax + Bu$. For the linearization of the system, the equilibrium point $\mathbf{x}^* = [0,\ 0,\ 0,\ 0]^T$ was considered, resulting in:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix},\ \mathbf{B} = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix} \tag{2}$$

with:

$$\begin{aligned}
a_{22} &= -\frac{\left(J_2 + m_2 l_2^2\right)\left(b_1 + \beta_m K_g K_b\right)}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
a_{23} &= -\frac{g m_2^2 L_1 l_2^2}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
a_{24} &= \frac{m_2 L_1 l_2 b_2}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
a_{42} &= \frac{m_2 L_1 l_2 \left(b_1 + \beta_m K_g K_b\right)}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
a_{43} &= \frac{g m_2 l_2 \left(J_1 + m_1 l_1^2 + m_2 L_1^2\right)}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
a_{44} &= -\frac{b_2 \left(J_1 + m_1 l_1^2 + m_2 L_1^2\right)}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
b_2 &= \frac{\beta_m \left(J_2 + m_2 l_2^2\right)}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2} \\
b_4 &= -\frac{\beta_m m_2 L_1 l_2}{\left(J_1 + m_1 l_1^2\right)\left(J_2 + m_2 l_2^2\right) + J_2 m_2 L_1^2}
\end{aligned}$$

with $\beta_m = (\eta_g K_g \eta_m K_m)/R_m$.

III. Linear Quadratic Regulator

A wide range of discrete-time systems can be characterized through the linear time-invariant state-space representation:

$$x_{k+1} = Ax_k + Bu_k \tag{3}$$

with the discrete time index k, the state vector $x_k \in \mathbb{R}^n$, the control input vector $u_k\mathbb{R}^m$, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. Furthermore, the state transition equation corresponds to a deterministic Markov Decision Process (MDP) [2]. The policies of interest in this type of systems are state feedback policies of the form:

$$u_k = h(x_k) = -K^T x_k \tag{4}$$

with the control policy $K \in \mathbb{R}^n$ a constant feedback gain vector. The gain vector K should be chosen in such a way that the closed-loop system matrix, $A_{cl} = A - BK^T$, has all its eigenvalues strictly within the unit circle.

Likewise, it is possible to assign a per-step cost or utility, which represents a measure of the control cost at each step. A standard form is the quadratic function:

$$r\left(x_k, u_k\right) = x_k^T Q x_k + u_k^T R u_k \tag{5}$$

where $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$, with $Q = Q^T \geq 0$ and $R = R^T > 0$ to ensure a well-defined cost function. It is assumed that the pair (A, B) is stabilizable, meaning that there exists a feedback gain K that makes the closed-loop system asymptotically stable.

On the other hand, the total cost of a state x_k under the control policy $h(x_k)$, denoted as $V_h(x_k)$, is defined as the sum of utilities at each step that the policy $h(x_k)$ has incurred, starting from step k, where:

$$V_h(x_k) = \frac{1}{2}\sum_{i=k}^{\infty}\left(x_i^T Q x_i + u_i^T R u_i\right) \tag{6}$$

This definition implies the following recursive relationship:

$$\begin{aligned} V_h(x_k) &= \frac{1}{2}r(x_k,u_k) + \frac{1}{2}\sum_{i=k+1}^{\infty}\left(x_i^T Q x_i + u_i^T R u_i\right) \\ &= \frac{1}{2}r(x_k,u_k) + V_h(x_{k+1}) \end{aligned} \tag{7}$$

By assuming that the optimal cost function, V^*, is quadratic in the state, it can be expressed as:

$$V^*(x_k) = \frac{1}{2}x_k^T P x_k \tag{8}$$

where $P \in \mathbb{R}^{n\times n}$ is the cost matrix for the policy K. Thus, the Bellman equation is expressed as:

$$x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \tag{9}$$

Substituting for the system dynamics (3) and the feedback gain (4), assuming a constant state feedback policy, $h(x_k)$, that is stationary for some stabilizing gain K. Since this must hold for all states x_k, it is considered:

$$\left(A-BK^T\right)^T P\left(A-BK^T\right) - P + Q + KRK^T = 0 \tag{10}$$

This matrix equation is linear in P and is known as the Lyapunov equation when K is fixed [6]. Thus, it is evident that the discrete-time LQR Bellman equation is equivalent to a Lyapunov equation [7]. Solving this equation with a predetermined gain K yields $P = P^T > 0$.

Similarly, the Hamiltonian function for a discrete-time LQR is given by:

$$H(x_k,u_k) = r(x_k,u_k) + x_{k+1}^T P x_{k+1} - x_k^T P x_k \tag{11}$$

A necessary condition for optimality is the stationary value condition, given by $\partial H(x_k,u_k)/\partial u_k = 0$. Such that we obtain:

$$Ru_k + B^T P(Ax_k + Bu_k) = 0 \tag{12}$$

From which the control signal is derived as:

$$u_k = -Kx_k = -\left(R + B^T P B\right)^{-1} B^T P A x_k \tag{13}$$

Therefore, the policy will be:

$$K = \left(R + B^T P B\right)^{-1} B^T P A \tag{14}$$

Substituting for the control policy into the Bellman equation (9) and simplifying the result, we obtain the discrete-time Hamilton-Jacobi-Bellman equation:

$$A^T P A - P + Q - A^T P B\left(R + B^T P B\right)^{-1} B^T P A = 0 \tag{15}$$

This equation is quadratic in P and is known as the algebraic Riccati equation, equivalent to the Bellman optimality equation for an LQR [7]. According to Bradtke [10], this is a straightforward yet computationally expensive way to obtain K^*, if precise system models and cost functions are known.

IV. Reinforcement learning for optimal control

It is possible to implement an online algorithm based on the Q-function without knowing the system dynamics, only measuring information along the system trajectories. This yields adaptable optimal control algorithms that converge online to optimal control solutions.

The Q-function for a discrete-time LQR, following the policy K, is derived from the value function as [6]:

$$Q(x_k,u_k) = \frac{1}{2}r(x_k,u_k) + V^*(x_{k+1}) \tag{16}$$

Where the control signal u_k is arbitrary, and the policy $u_k = h(x_k)$ is followed for $k+1$ and subsequent steps. Expressing the Q-function using the solution of the algebraic Riccati equation, P, the Q-function for discrete-time LQR is as follows:

$$Q(x_k,u_k) = \frac{1}{2}z_k^T\begin{bmatrix} A^TPA+Q & A^TPB \\ B^TPA & B^TPB+R \end{bmatrix} z_k \tag{17}$$

with $z_k = [x_k, u_k]^T$. Additionally, the following is proposed:

$$Q(x_k,u_k) = \frac{1}{2}z_k^T S z_k = \frac{1}{2}z_k^T\begin{bmatrix} S_{xx} & S_{xu} \\ S_{ux} & S_{uu} \end{bmatrix} z_k \tag{18}$$

and the matrix $S = S^T > 0$ with $S \in \mathbb{R}^{l\times l}$, $S_{xx} \in \mathbb{R}^{n\times n}$, $S_{xu} = S_{ux}^T \in \mathbb{R}^{n\times m}$, $S_{uu} \in \mathbb{R}^{m\times m}$, where:

$$\begin{bmatrix} A^TPA+Q & A^TPB \\ B^TPA & B^TPB+R \end{bmatrix} = \begin{bmatrix} S_{xx} & S_{xu} \\ S_{ux} & S_{uu} \end{bmatrix}$$

On the other hand, for the policy update, $\partial Q(x_k,u_k)/\partial u = 0$ is considered. From the representation (17), we obtain:

$$0 = B^TPAx_k + \left(B^TPB+R\right)u_k \tag{19}$$

Therefore, it's possible to express the optimal control signal as a function dependent on the state vector x_k:

$$u_k = -(B^TPB+R)^{-1}B^TPAx_k \tag{20}$$

However, upon considering the partial derivative of expression (18) with respect to the control signal, we have:

$$u_k = -S_{uu}^{-1}S_{ux}x_k \tag{21}$$

The expression (20) requires knowledge of the system dynamics (A,B) to perform control policy improvement, whereas (21) only requires knowledge about the matrix S from the Q-function [10].

According to [7], the matrix S from the Q-function (18) can be estimated online, without knowing the system dynamics, using identification techniques. To accomplish this, the Q-function (18) is expressed in a parametric form:

$$Q(x,u) = Q(z) = W^T(z\otimes z) = W^T\phi(z) \tag{22}$$

with W being the vector formed by the elements of S and $\otimes$ denoting the Kronecker product. The function $\phi(z) = z\otimes z$ represents the quadratic polynomial basis in terms of the elements of z. By eliminating redundant entries of $\phi(z)$, it is ensured that W will solely contain the $(n+m)(n+m+1)/2$ elements from the upper half of S, taking the form:

$$W = [s_{11}, 2s_{12}, \cdots, s_{22}, 2s_{23}, \cdots, s_{33}, 2s_{34}, , \cdots, s_{ll}]^T$$

Meanwhile, the vector of quadratic functions takes the form:

$$\phi(z) = \left[z_1^2, z_1 z_2, \cdots, z_1 z_l, z_2^2, z_2 z_3, \cdots, z_2 z_l, \cdots, z_l^2\right]^T$$

The temporal difference error is defined as:

$$e_k = r(x_k, u_k) + Q(x_{k+1}, h(x_{k+1})) - Q(x_k, u_k) \quad (23)$$

Substituting the approximation of the function Q into the temporal difference error:

$$e_k = r(x_k, u_k) + W^T \phi(z_{k+1}) - W^T \phi(z_k) \quad (24)$$

The step of evaluating the function Q in a policy iteration algorithm is given as:

$$W_{j+1}^T (\phi(z_k) - \phi(z_{k+1})) = \frac{1}{2}\left(x_k^T Q x_k + u_k^T R u_k\right) \quad (25)$$

with the policy improvement step:

$$h_{j+1}(x_k) = \arg\min_u \left(W_{j+1}^T \phi(x_k, u)\right), \ \forall x \in X \quad (26)$$

Since (25) involves $(n+m)(n+m+1)/2$ unknowns, which form the vector W, it perfectly corresponds to the type of equations used in system identification. Therefore, for its online implementation, Recursive Least Squares (RLS) was utilized for updating the parameter vector W_{j+1} with the regression vector [10]:

$$\Phi_k = \phi(z_k) - \phi(z_{k+1}) \quad (27)$$

The data to be measured at each step include $(x_k, u_k, r(x_k, u_k), x_{k+1}, u_{k+1})$, where u_{k+1} is computed as $h_j(x_{k+1})$, with $h_j(\cdot)$ representing the current policy. It's necessary to add test noise to the control signal u_k to ensure persistent excitation. The recursive relations of the RLS algorithm are provided by [10]:

$$\begin{aligned} W_{j+1} &= W_j + \frac{P_{c_j}\Phi_k\left(r(x_k, u_k) - W_j^T \Phi_k\right)}{1 + \Phi_k^T P_{c_j} \Phi_k} \\ P_{c_{j+1}} &= P_{c_j} - \frac{P_{c_j}\Phi_k \Phi_k^T P_{c_j}}{1 + \Phi_k^T P_{c_j} \Phi_k} \end{aligned} \quad (28)$$

This algorithm requires the initial feedback gain to be stabilizing. This gain can be easily determined through initial testing on the plant [10].

The update of the parameter vector W is performed until the RLS algorithm converges, utilizing the Euclidean norm of the difference between each update, $\|W_{j+1} - W_j\| \leq \varepsilon_W$, where ε_W is a small constant, as a convergence parameter. Similarly, the Euclidean norm of the variation in estimation $\|K_{j+1} - K_j\| \leq \varepsilon_K$ is employed, where ε_K is a small constant. This way, the convergence of the algorithm is determined, and the estimation process is halted.

V. Simulations

Simulations were conducted using a Matlab Simulink model to implement the reinforcement learning algorithm. The nonlinear model of the rotary inverted pendulum was employed during the simulations to assess the algorithm's capability to approximate the optimal solution of the linear system.

For the simulations of the rotary inverted pendulum, the parameters were utilized as $m_1 = 0.2570$ *kg*, $L_1 = 0.2159$ *m*, $l_1 = 0.0619$ *m*, $b_1 = 2.4\times10^{-3}$ *N*, $J_1 = 9.9829\times10^{-4}$ *kg·m²*, $R_m = 2.6\ \Omega$, $K_m = 7.68\times10^{-3}$ *V·s/rad*, $K_b = 7.68\times10^{-3}$ *N·m/A*, $K_g = 70$, $\eta_m = 0.69$ and $\eta_g = 0.9$.

It was proposed to evaluate the algorithm's ability to find the online solution of a discrete-time LQR in the presence of parametric uncertainty. The experiments were initiated with the medium pendulum, having a length of $L_2 = 0.3365$ *m*, with a center of mass $l_2 = 0.1556$ *m*, and a mass of $m_2 = 0.127$ *kg*. Later, the short pendulum was replaced with a length of $L_2 = 0.20$ *m*, with a center of mass $l_2 = 0.1635$ *m*, and a mass of $m_2 = 0.0970$ *kg*.

The linear Furuta pendulum model was discretized using Matlab with the zero-order hold (zoh) method for a sampling time of 2 ms. Subsequently, an LQR was designed for each system, considering the quadratic cost function. The matrices Q and R were chosen as $Q = 20I$ and $R = 1$. Thus, the gain vectors were $K_m = [-3.6148, -5.4437, -69.5539, -10.3393]^T$ and $K_c = [-3.2097, -4.8446, -58.4409, -8.0511]^T$.

In the simulation of the reinforcement learning-based algorithm, $Q = 20I$ and $R = 1$ were kept for the utility function. The initial value of the covariance matrix was set as $P_{c_0} = 90I$. Concerning the parameter vector W, better performance was observed when $W_0 = \mathbf{0}$.

To determine an initial gain vector, tests were conducted with the closed-loop system and the vector $K_0 = [-1, -1, -10, -1]^T$ was chosen. Additionally, to establish algorithm convergence, ε_W and ε_K were set to $\varepsilon = 1\times10^{-9}$. As a result, the K vector was updated when $\|Wj + 1 - Wj\| \leq \varepsilon$ or when the discrete time steps k since the last update exceeded a maximum of $150,000$. The algorithm also halts when $\|K_{j+1} - K_j\| \leq \varepsilon$.

Fig. 2 displays the tuning of the gain vector K for the LQR of the Furuta pendulum with both the medium and short pendulums. The top portion illustrates the 3 components with the smallest magnitude for easier visualization.

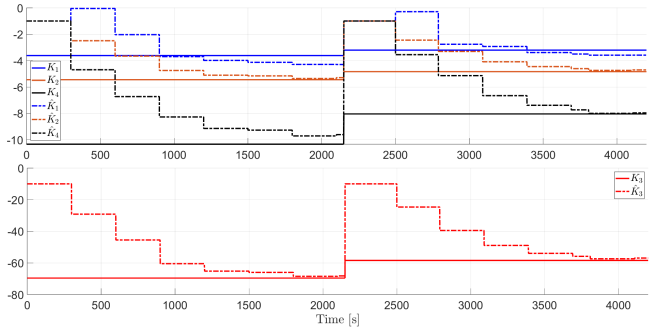

Fig. 2. Tuning of the LQR.

The vector obtained for the medium pendulum was $\hat{K}_m = [-4.2965, -5.2935, -68.2348, -9.6118]^T$ and for the short pendulum $\hat{K}_c = [-3.5859, -4.7077, -56.9293, -7.9692]^T$.

Subsequently, an experiment was proposed to compare the response of the system with the vectors K. The initial condi-

tion vector $x_0 = [\theta_0, 0, \alpha_0, 0]^T$ was chosen, with $\theta_0 = -15$ and $\alpha_0 = 10$. Simulations were conducted for both short and medium pendulums, comparing Km and Kc against the reinforcement learning obtained vectors $\hat{K}m$ and $\hat{K}c$, respectively.

Fig. 3 and Fig. 4 presents the comparison of the four controllers, where labels LQR_i indicate the use of analytically calculated vectors, and labels RL_i indicate the use of vectors obtained by reinforcement learning.

Upon analyzing the behavior of the arm, it was observed that the controllers obtained through RL exhibited very similar behaviors. Furthermore, they yielded a quicker response, which might indicate a slightly higher control signal compared to the optimal controllers.

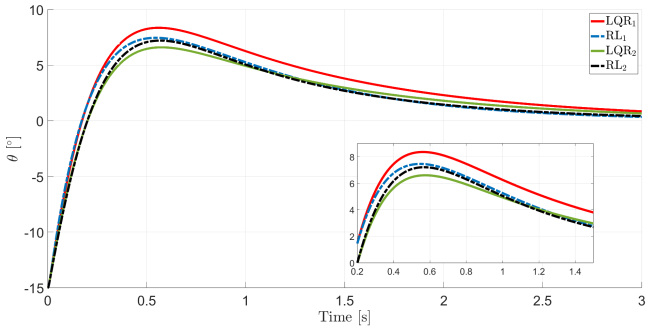

Fig. 3. LQR responses comparison

On the other hand, the pendulum's response exhibited a similar behavior across all four controllers. It was observed that the optimal controllers maintained a very close overshoot. However, the controllers tuned through reinforcement learning displayed a greater difference in the magnitudes of the maximum overshoot values.

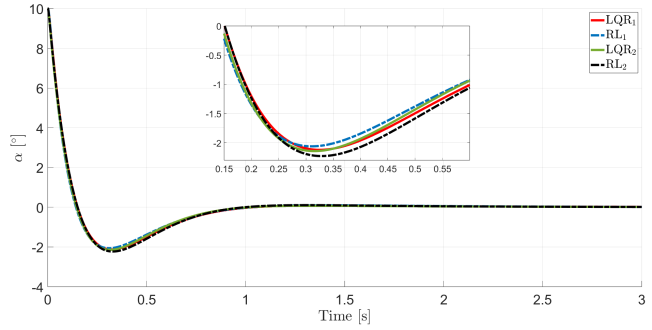

Fig. 4. LQR responses comparison

In order to compare the performance of each controller, the LQR cost function with matrices $Q = 20I$ and $R = 1$ is used.

$$J(x_k, u_k) = \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \tag{29}$$

The obtained results are presented in Table I.

TABLE I
LQR COST FUNCTION RESULTS

Controller	Cost
LQR_1	$8,114.9$
RL_1	$8,151.2$
LQR_2	$7,121.1$
RL_2	$7,134.8$

VI. CONCLUSIONS

Simulations were conducted in Matlab Simulink to implement a reinforcement learning-based algorithm that enables online solution of a discrete-time LQR to control the Furuta pendulum around its unstable equilibrium. Two pendulum scenarios were considered with different lengths and masses, referred to as the medium pendulum and the small pendulum. The tuning results demonstrated that by implementing the reinforcement learning-based algorithm, it's possible to find solutions close to the calculated values. The controllers obtained through reinforcement learning provided a performance very close to that achieved with the optimal controllers. In both cases, the increase observed in the cost function was minimal, with an increase of 0.45% for the medium pendulum and 0.19% for the short pendulum. Thus, the reinforcement learning-based algorithm presents an alternative that doesn't necessitate knowledge of the system's dynamics to find solutions close to optimal controllers.

REFERENCES

[1] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing up control of inverted pendulum," in *Proceedings IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation*, vol. 3, pp. 2193–2198, 1991.

[2] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 206, no. 4, pp. 263–269, 1992.

[3] M. Yamakita, K. Furuta, K. Konohara, J. Hamada, and H. Kusano, "Vss adaptive control based on nonlinear model for titech pendulum," in *Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, vol. 3, pp. 1488–1493, 1992.

[4] I. Fantoni and R. Lozano, *Non-linear Control for Underactuated Mechanical Systems*. London: Springer, 2002.

[5] J. Moreno-Valenzuela and C. Aguilar-Avelar, *Motion Control of Underactuated Mechanical Systems*. Cham, Switzerland: Springer, 2018.

[6] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[7] D. Vrabie, K. G. Vamvoudakis, and F. L. Lewis, *Optimal adaptive control and differential games by reinforcement learning principles*. Control, Robotics and Sensors, Institution of Engineering and Technology, 2012.

[8] 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 Magazine*, vol. 32, no. 6, pp. 76–105, 2012.

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

[10] S. J. Bradtke, *Incremental Dynamic Programming for On-Line Adaptive Optimal Control*. PhD thesis, University of Massachusetts, USA, 1995. UMI Order No. GAX95-10446.