AN OPTIMAL ADAPTIVE CONTROLLER BASED ON ONLINE ACTOR-CRITIC LEARNING FOR A ROBOT MANIPULATOR

UM CONTROLADOR ADAPTATIVO ÓTIMO BASEADO EM APRENDIZADO ONLINE ATOR-CRÍTICO PARA UM MANIPULADOR ROBÓTICO

UN CONTROLADOR ADAPTATIVO ÓPTIMO BASADO EN EL APRENDIZAJE ACTOR-CRÍTICO EN LÍNEA PARA UN MANIPULADOR ROBÓTICO

https://doi.org/10.56238/sevened2026.008-036

Patrícia Helena Moraes Rêgo[1], Joelson Miller Bezerra de Sousa[2]

ABSTRACT

The uncertainties in the parameters of a robot manipulator can significantly affect the robot performance, causing steady-state and trajectory following errors. Adaptive controllers are a good alternative for these systems, since their main feature is the capability to learn online using real-time parameter estimation. Nevertheless, adaptive controllers are not usually designed to be optimal to a prescribed performance index, and thus, are not suitable to applications in which optimal use of resources is highly desirable, for instance humanoid and service robots. This paper presents the design and performance study of a controller that combine features of adaptive control and optimal control applied to a robot manipulator. Specifically, the proposed control scheme is implemented as an actor-critic structure, which is in the reinforcement learning context, characterizing this design as a model-free approach. In contrast to others actor-critic systems in which two independent neural networks are used, one for approximating the value function and another for learning the control actions, in this scheme, a single neural network is defined, reducing the number of parameters to be estimated. The simulation results validate the desired performance of the proposed controller applied in a two-link robot manipulator with revolute joints.

Keywords: Robot Manipulator. Adaptive Control. Optimal Control. Reinforcement Learning. Actor-Critic Scheme.

RESUMO

As incertezas nos parâmetros de um manipulador robótico podem afetar, de forma significativa, o desempenho do manipulador, ocasionando erros de regime e de seguimento de trajetória. Controladores adaptativos apresentam-se como uma boa alternativa para esses sistemas, pois possuem como principal característica a capacidade de aprenderem online usando estimação de parâmetros em tempo real. No entanto, controladores adaptativos não são geralmente projetados com a qualidade de serem ótimos com respeito

[1] Dr. in Electrical Engineering. Universidade Estadual do Maranhão (UEMA). E-mail: phmrego@yahoo.com.br Orcid: https://orcid.org/0000-0001-5899-0623 Lattes: http://lattes.cnpq.br/6535271381344851
[2] Master's degree of Science in Computer Engineering. Universidade Estadual do Maranhão (UEMA). E-mail: joelsonmiller@hotmail.com Lattes: http://lattes.cnpq.br/6867462270921352

aos critérios de desempenho especificados e, desta forma, não são viáveis para aplicações onde o uso ótimo de recursos é altamente desejável, como por exemplo em robôs humanoides e robôs de serviços. Este artigo apresenta o projeto e investigação de desempenho de um controlador que combina características de controle adaptativo e controle ótimo para um manipulador robótico. Especificamente, o esquema de controle proposto é implementado como uma estrutura ator-crítico, a qual está inserida no contexto de aprendizado por reforço, caracterizando este projeto como uma abordagem independente do modelo da planta. Em contraste a outros sistemas ator-críticos em que são usadas duas redes neurais independentes, uma para aproximar a função valor, e a outra para aprender ações de controle, neste esquema, se define uma única rede neural, o que reduz o número de parâmetros a serem estimados. Os resultados de simulação demonstram o desempenho desejado do controlador proposto que atua em um manipulador de juntas rotativas com dois graus de liberdade.

Palavras-chave: Manipulador Robótico. Controle Adaptativo. Controle Ótimo. Aprendizado por Reforço. Esquema Ator-Crítico.

RESUMEN

Las incertidumbres en los parámetros de un manipulador robótico pueden afectar significativamente al rendimiento del manipulador, provocando errores de régimen y de seguimiento de la trayectoria. Los controladores adaptativos se presentan como una buena alternativa para estos sistemas, ya que su principal característica es la capacidad de aprender en línea utilizando la estimación de parámetros en tiempo real. Sin embargo, los controladores adaptativos no suelen diseñarse con la calidad de ser óptimos con respecto a los criterios de rendimiento especificados y, por lo tanto, no son viables para aplicaciones en las que es muy deseable el uso óptimo de los recursos, como por ejemplo en robots humanoides y robots de servicio. Este artículo presenta el diseño y la investigación del rendimiento de un controlador que combina características de control adaptativo y control óptimo para un manipulador robótico. En concreto, el esquema de control propuesto se implementa como una estructura actor-crítico, que se inserta en el contexto del aprendizaje por refuerzo, lo que caracteriza a este diseño como un enfoque independiente del modelo de la planta. A diferencia de otros sistemas actor-crítico en los que se utilizan dos redes neuronales independientes, una para aproximar la función de valor y otra para aprender acciones de control, en este esquema se define una única red neuronal, lo que reduce el número de parámetros que deben estimarse. Los resultados de la simulación demuestran el rendimiento deseado del controlador propuesto, que actúa en un manipulador de juntas rotativas con dos grados de libertad.

Palabras clave: Manipulador Robótico. Control Adaptativo. Control Óptimo. Aprendizaje por Refuerzo. Esquema Actor-Crítico.

1 INTRODUCTION

The development of control strategies for robotic manipulators presents difficulties due to the characteristics of the system itself, i.e., an articulating robot is a multivariable dynamic system, with strong nonlinearities due to the couplings of its joints and movements, in addition to presenting uncertain or time-varying parameters, such as the mass and inertia of the links, friction or backlash in the joint gears, payload variations, location of the center of mass (which can change when the robot is under load), among others (Fateh; Fateh, 2019). These parametric inaccuracies result in losses of accuracy and speed in manipulator movements, which in certain applications is highly undesirable. Nonlinear dynamics, on the other hand, can lead the system to instability at certain operating points (Craig, 2021).

Conventional feedback controllers, such as PID (Proportional-Integral-Derivative), are widely used in industry because they are simple, easy to implement, and perform well in various applications (Borase *et al*., 2021). However, this control scheme, as it is a type of control with fixed gains, becomes insufficient when applied to systems with nonlinearities and/or uncertainties (imprecise parameters, non-modeled high-frequency dynamics, and disturbances), i.e., systems that have variable operating points (Konstantopoulos; Baldivieso-Monasterios, 2020).

Among the classical controllers applied to manipulators, there are those based on model (kinematic and/or dynamic for controlling position, velocity and force). However, these approaches require complete knowledge of the equations that describe the behavior of the system, which are quite complex and with parameters that are often uncertain. The complexity of the model also increases with the increase in joints and links of the manipulator, increasing the computational cost to solve these equations (Moosavi; Zafar; Sanfilippo, 2022).

Adaptive control theory provides a means to develop solutions for dynamic systems that require more complex controllers. This approach allows for online compensation for *parametric variations and uncertainties in the system, ensuring that the desired performance criteria are met* (Sun *et al*., 2020). Traditionally, adaptive control methods can be divided into two approaches: indirect control and direct control (Qi; Tao; Jiang, 2019). In indirect control, the estimation of the system parameters precedes the generation of an control input. In direct control, the controller parameters are directly adjusted without the need for the equations that govern the behavior of the system.

In the Adaptive Control literature, there are several studies and methods applied to the trajectory control of robotic manipulators. Dubowsky and Desforges (1979) are the pioneers in employing adaptive control techniques in articulated robots. The approach used by these

researchers was the *Model Reference Adaptive System* (MRAS). Practical results also showed the benefits of approaches based on *self-tuning* and *backstepping techniques* in relation to conventional control with fixed gains (Clegg; Dunnigan; Lane, 2001) (Sasaki *et al*., 2009) (Hu; Xu; Zhang, 2012). Hybrid approaches have also been explored (Maliotis, 1991) (Al-Olimat; Ghandakly, 2002) (Chen, 2005) (Alqaudi *et al*., 2016) (Zhang; Wei, 2017). In (Wu; Yan; Cai, 2019) (Fateh; Fateh, 2019) (Yilmaz *et al*., 2022) (Freire; Rossomando; Soria, 2018) adaptive control designs based on artificial intelligence techniques, such as neural networks and *fuzzy logic*, are proposed, which are able to compensate for the uncertainties of the model of a manipulator robot.

Although adaptive control techniques have been successful in many applications, one aspect that should be noted is that the controller designs resulting from these methods, in general, have been structured without considering the optimization of the control action and, therefore, are not feasible for applications where the use of optimal control strategies is required. such as humanoid robots/service robots (Khan *et al*., 2012). In this case, a joint approach of adaptive control and optimal control techniques is desired. Optimal control basically consists of determining a control law in order to minimize a desired performance criterion. In the context of robotics, performance criteria may involve the energy or force for the execution of the motion, while the physical constraints of the system, such as actuator or joint limits, must be satisfied.

Many efforts in systems control theory are currently concentrated in an area of machine learning based on animal behavior studies and cognitive psychology, called *Reinforcement Learning* (RL), which aims to incorporate features of biological systems into the treatment of systems with uncertainties, introducing several terms, such as adaptation, learning, pattern recognition, and self-organization (Guo; Yan; Cui, 2020) (Yaghmaie; Gustafsson; Ljung, 2023) (Chen; Hence; Dong, 2024a) (Chen; Dong; Dai, 2024b) (Zhao *et al*., 2025) (Su *et al*., 2025) (Wang *et al*., 2025). The central theme in RL research is the design of algorithms that learn optimal control policies through knowledge only of transition samples of states or trajectories, which are collected in advance or by real-time interaction with the system.

Actor-Critic methods constitute a class of reinforcement learning techniques that essentially consist of two independent parametric structures (e.g., neural networks), one to represent the control policy, called the Actor, and the other network structure is to represent the value function, called the critic (Sutton; Barto, 2018). The actor is an agent who interacts with the environment, i.e., the actor is the controller who establishes control actions, while

the critic evaluates the effect of control actions and provides guidelines on how to improve the control law.

Reinforcement learning can be seen in (Kiumarsi *et al*, 2018) from the perspective of a promising field of research for the design of a class of adaptive controllers with actor-Critic structure that learn *optimal* control solutions online without making use of the system dynamics model (plant). This approach solves the optimization equation (Hamilton-Jacobi-Bellman equation - HJB) in a "forward in time" manner using methods of temporal differences, approximation of functions, and policy improvements. Such controllers are inspired by biological neural structures that provide capabilities to effectively deal with the degree of complexity of nonlinear, uncertain, and partially observable systems. In (Kiumarsi *et al*, 2018), the main ideas and algorithms of reinforcement learning are presented, as well as their applications in optimal control of dynamical systems.

1.1 OBJECTIVES

The present article aims to evaluate the potential of a reinforcement learning algorithm to solve problems of online optimal control of the trajectory of a robotic manipulator with continuous state space (joint space). In contrast to most of the actor-Critic algorithms reported in the literature (see Section 2), in which two neural networks are used, one to approximate the value function, and the other to learn control actions, the algorithm proposed in this work employs an actor-Critic architecture where a single neural network is used to approximate the solution of the equation HJB, which significantly reduces the number of parameters to be estimated. Specifically, in this scheme, control actions are calculated accurately using a greedy policy scheme with respect to the value function, rather than using a parametric approximator to represent the control policy. Experiments performed on a UR10 robotic arm of the V-REP simulator show that such an algorithm successfully learns the optimal control law for the regulation and tracking tasks for different reference signals.

2 RELATED WORKS

Important previous contributions to the design of RL-based control include the work of Peters and Schaal (2008a) (2008b), who investigated various methods of reinforcement learning for humanoid robots. These methods were classified into three categories: greedy policy, vanilla policy gradient, and natural policy gradient. The natural Actor-Critic approach, which explores the formulation of the natural policy gradient, was highlighted by the authors for presenting better convergence properties. An extension of this study is shown in (Bhatnagar *et al*., 2009).

Already (Shah; Gopal, 2009) presented a Q-learning-based control approach to manipulator robots in uncertain environments and provided a comparative study of different methods of function approximation, such as fuzzy, neural networks, decision tree, and support vector machine.

In (Khan *et al*., 2011, 2012), the authors emphasized applications of RL controllers in robotic systems and proposed an optimal adaptive control scheme based on *Q-Learning* and Approximate Dynamic Programming. The strategy was implemented on the arm of a humanoid robot (Bristol Elumotion-Robotic-Torso II) considering one case without restrictions and another with movement restrictions.

In (Pane; Nageshrao; Babuška, 2016), the authors provided experimental validation of an Actor-Critic learning-based compensator to improve the performance of a manipulator robot. The proposed method eliminates the need to learn the system model and can be used in any feedback controller (PID, LQR, etc.). The validation of the method was demonstrated through experiments on an industrial manipulator robot with six degrees of freedom for different types of reference trajectories. An extension of this work is presented in (Pane *et al*., 2019).

The application of RL controllers in robotic manipulators is also shown in (Hu; Si, 2018). In this work, an Actor-Critic Learning strategy with state observer via neural network was implemented to control a robotic arm with unknown parameters and subject to unknown dead zones.

Khan *et al*. (2019) proposed an optimal adaptive compliance control for a robotic locomotion aid device. The suggested control scheme is based on Q-Learning and approximate dynamic programming. This scheme is completely independent of dynamic model and employs feedback of joint position and speed, as well as the detected joint torque (applied by the user during walking) for compliance control. The efficiency of the controller is tested in simulation on a model of a robotic locomotion aid device.

Kamboj *et al*. (2020) presented an optimal kinematic control strategy in discrete time for a manipulator using the Actor-Critic structure. The methodology exposed was applied to a 3D model of a manipulator with six degrees of freedom in experiments carried out in a simulation software. Then, the strategy was implemented in a real robot of the same model as the simulated one.

In (He *et al*., 2021), the authors discussed the design of control and validation of experiments of a flexible two-link manipulator system. A reinforcement learning control strategy is developed based on the actor-Critic framework to attenuate vibrations while maintaining trajectory tracking.

An Actor-Critic Learning-based tracking controller for a handler has also been studied by (Cao *et al*., 2023). In this work, the sliding modes technique is used so that the action obtained by the Actor-Critic scheme ensures the convergence of the tracking error in a fixed time. In addition, an *antiwindup compensator* has been designed to handle the effects of saturation of the joint actuator.

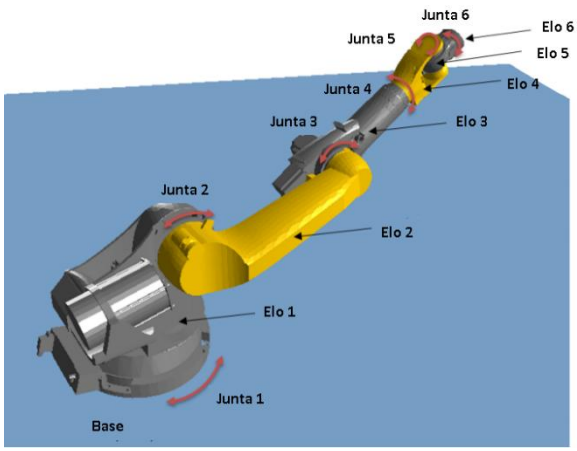
In the above literature, most actor-Critic RL algorithms are implemented using two neural networks, one to approximate the value function, and the other to learn control actions. To reduce the computational complexity associated with actor-Critic methods, we propose in this paper an architecture where a single neural network is used to approximate the optimal control solution, which significantly reduces the number of parameters to be estimated. Specifically, control actions are calculated accurately through a greedy policy schema with respect to the value function, rather than using a parametric approximation to represent the control policy.

3 DESCRIPTION OF THE ROBOTIC MANIPULATOR SYSTEM

A robotic manipulator, or articulated robot, is formed by a set of individual bodies connected to each other forming a kinematic chain capable of performing tasks through interaction with the environment (Craig, 2021). The two fundamental parts that make up an articulated robot are the links, or joints, and the joints. Links are the physical structures (rigid or flexible) that make up the robot. Joints, on the other hand, are responsible for promoting relative movement between joints through actuators and are commonly classified according to the mobility they enable. The most common types found in industry are rotational joints and prismatic joints.

Figure 1

Links and Joints of an articulated robot



Source: Abbas, 2018.

Figure 1 illustrates a sequence of links and joints of a robotic arm. The ends of the articulating robot are called the base and effector. The base attaches to the first link and secures the mechanism somewhere in the task space. The effector is a tool connected to the last link of the articulator and it is at this point that there is interaction with the environment. The type of actuator installed will depend on the task to be performed.

3.1 DYNAMIC EQUATIONS OF A ROBOTIC MANIPULATOR

The dynamics of manipulators studies the relationship between the forces applied to the actuators of the joints and the movement of the mechanism. The Lagrange formulation allows modeling the dynamic behavior of a body in terms of kinetic and potential energies rather than considering the moments and forces applied individually to each joint. The Lagrange equation is expressed by

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \quad (1)$$

$$L(\dot{q}, q) = K(\dot{q}, q) - U(q)\,, \quad (2)$$

where $K(\cdot)$ is the kinetic energy and $U(\cdot)$ is the potential energy stored in the mechanism. This equation is written in terms of the generalized coordinates q of the articulator and its derivative $\dot{q}$ in time. The term τ, in turn, represents the generalized vector of forces, including the forces and torques applied to the system.

For a manipulator robot with n rigid links, kinetic energy can be written in the form

$$K(\dot{q}, q) = \sum_{i=0}^{n} k_i \quad (3)$$

$$k_i = \frac{1}{2} m_i v_{C_i}^T v_{C_i} + \frac{1}{2} \omega_i^{T\,{}^{C_i}} I_i \omega_i\,, \quad (4)$$

where k_i is the kinetic energy for the ith link. For each link, there are two components, one related to linear velocity v_{C_i}, and the other, angular velocity ω_i, relative to the center of mass of the respective joint, with m_i the mass of the link i, e ${}^{C_i}I_i$ and is the inertia matrix.

Potential energy can be expressed as

$$U(q) = \sum_{i=0}^{n} u_i \quad (5)$$

$$u_i = m_i g^T P_{C_i} \quad (6)$$

where u_i is the potential energy for the ith link, defined in terms of the mass m_i, the gravity vector g, and the location P_{C_i} of the center of mass relative to the base.

Applying Lagrangian $L(\cdot)$ to equation (1), one can reorder the terms of the resulting expression in order to obtain.

$$\tau = M(q)\ddot{q} + N(q, \dot{q}) + G(q), \quad (7)$$

where $M(q)$ is the mass matrix of the manipulator, $N(q, \dot{q})$ is a n-dimensional vector related to Coriolis and centripetal forces, and $G(q)$ is a vector $n \times 1$ with terms involving gravity.

In this way, the model of a manipulator can be written in the form of State Space by

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ -M^{-1}(N + G) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} \tau. \quad (8)$$

4 METHODOLOGY

In the context of optimal control and reinforcement learning, the notion of maximizing weighted future rewards is modified to minimize the cost of control. In this way, the objective is to determine a control law or control policy $h^*(x_k, d_k) = u_k^*$ that minimizes the performance index (value function)

$$V(x_k, d_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i, d_i), \quad (9)$$

where $x_k \in \mathbb{R}^n$ is the state vector, $u_k \in \mathbb{R}^m$ is the control input vector, d_k is the desired trajectory vector,$0 < \gamma \leq 1$ is the discount factor, and $r(\cdot)$ is the utility function that returns the control cost in a time step. A reasonably general utility function in energy minimization problems is given by:

$$r(x_i, u_i, d_i) = \tilde{r}(x_i, d_i) + u_i^T R u_i, \quad (10)$$

where R is a positive definite matrix. The vector d_i can be described as a design demand, making it $\tilde{r}(\cdot)$ represent the cost to perform the desired task, such as the tracking cost.

Using Bellman's principle of optimality (Vrabie; Vamvoudakis; Lewis, 2013), the optimal performance index can be written as

$$V^*(x_k, d_k) = \min_{u_k}\big(r(x_k, u_k, d_k) + \gamma V^*(x_{k+1}, d_{k+1})\big). \quad (11)$$

In reinforcement learning, a variant of the value function $V(\cdot)$, called a Q function (or action value function), is used. Such a function has an appropriate application in control desigs where the plant model is not available. The Q function associated with a control policy h is defined by

$$Q^h(x_k, u_k, d_k) = r(x_k, u_k, d_k) + \gamma V^h(x_{k+1}, d_{k+1}), \quad (12)$$

and the optimal Q function satisfies the following equation

$$Q^*(x_k, u_k, d_k) = r(x_k, u_k, d_k) + \gamma V^*(x_{k+1}, d_{k+1}). \quad (13)$$

Combining equations (11) and (13), Bellman's optimality equation in terms of the Q function is given by

$$V^*(x_k, d_k) = \min_{u_k}\big(Q^*(x_k, u_k, d_k)\big) \quad (14)$$

and the optimal control policy is obtained by

$$h^*(x_k, d_k) = \arg\min_{u_k} Q^*(x_k, u_k, d_k). \quad (15)$$

assuming Q^* sufficiently smooth (differentiable), the control signal can be obtained as a solution to the equation

$$\frac{\partial Q^*(x_k, u_k, d_k)}{\partial u_k} = 0. \quad (16)$$

4.1 ACTOR-CRITIC *ONLINE* LEARNING STRATEGY

The actor-Critic scheme described below considers a manipulator system with two degrees of freedom, which can be extended to manipulators with n degrees of freedom. The control law is synthesized in the optimal tracking problem of the manipulator's joint position. In particular, $x_k = [x_{k1} \quad x_{k2} \quad x_{k3} \quad x_{k4}]^T$ is defined as the state vector at time k, where $x_{k1} = q_1$ and $x_{k2} = q_2$ are the angular positions of joints 1 and 2, respectively, and $x_{k3} = \dot{q}_1$ e $x_{k4} = \dot{q}_2$ are, in due order, the angular velocities of joints 1 and 2. The control signal, of course, is a vector 2×1, where $u_k = \tau$ is the force applied on the joints. For the optimal tracking problem considered, the utility function reduces to

$$r(x_k, e_k, u_k) = e_k^T Q_c e_k + (u_{k+1} - u_k)^T S(u_{k+1} - u_k) + u_k^T R u_k \equiv r_k, \tag{17}$$

where $Q_c \in \mathbb{R}^{4\times4}$, $R \in \mathbb{R}^{2\times2}$, and $S \in \mathbb{R}^{2\times2}$ are diagonal and positive definite matrices.

In the present study, the parametric structure to approximate the Q function assumes the form given by

$$\hat{Q}_i(x_k, u_k, d_k, w_i) = w_i^T \phi(x_k, u_k, d_k), \tag{18}$$

where w_i is the ith estimate of the weight vector of the neural network and $\phi(\cdot)$ is the vector of activation functions or base functions. It is considered that the desired value for estimating the parameter w_i is given by

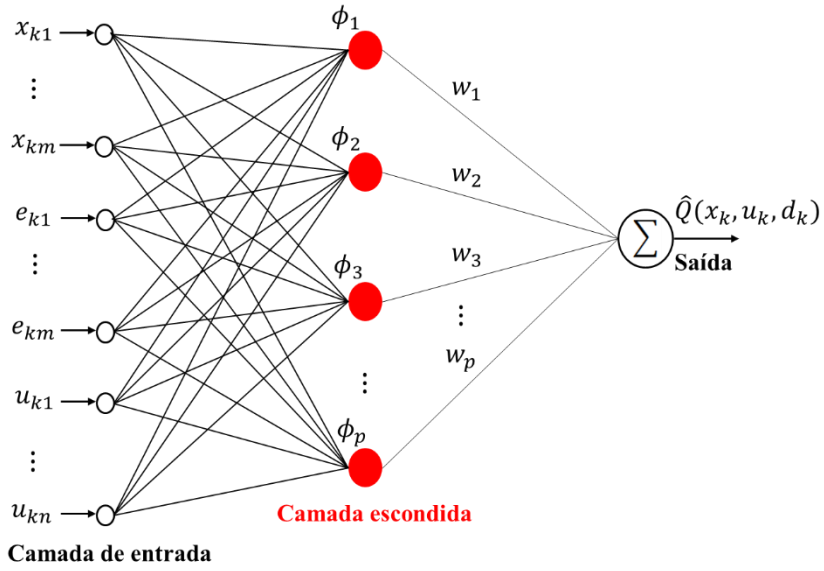
$$\Delta_{objetivo} = r(x_k, e_k, u_k) + \gamma \hat{Q}_i(x_{k+1}, u_{k+1}, d_{k+1}) \tag{19}$$

Figure 2 illustrates the neural network architecture used to estimate the Q function, in which, $m = 4$, $n = 2$ and $p = 105$. The functions ϕ_j, $j = 1, \ldots, p$, are the components of the vector of activation functions resulting from the Kronecker product given in equation (21).

The weight vector w_i is calculated by minimizing, in a least-squares sense, the temporal difference error, which is defined by

Figure 2

Architecture of the neural network used to estimate the function Q



Source: Prepared by the authors.

$$\delta_k = r_k + \gamma \hat{Q}_i(x_{k+1}, u_{k+1}, d_{k+1}, w_i) - \hat{Q}_i(x_k, u_k, d_k, w_i). \quad (20)$$

The vector of activation functions is constructed by higher-order polynomials. For simplicity, $\phi(\cdot)$ will be represented using Kronecker product $\otimes$ with the exclusion of redundant terms (Vrabie; Vamvoudakis; Lewis, 2013).

This exclusion is necessary so that the elements that make up the vector of base functions $\phi(\cdot)$ become linearly independent. The goal is to insert some quadratic elements and up to fourth-order terms into the tracking errors, states, and control signals, so that the neural network can learn the nonlinearities of the manipulator. Therefore,

$$\phi(z_k) = z_k \otimes z_k \,, \quad (21)$$

Where:

$$z_k = [u_k^T \quad e_k^T \quad e_{k1}^2 \quad e_{k2}^2 \quad e_{k3}^2 \quad e_{k4}^2 \quad x_{k1}^2 \quad x_{k2}^2 \quad x_{k3}^2 \quad x_{k4}^2]^T \quad (22)$$

so that $e_k = [e_{k1} \quad e_{k2} \quad e_{k3} \quad e_{k4}]^T = x_k - d_k$ is the tracking error. In this way, the Artificial Neural Network (ANN) to be implemented has 105 neurons.

The Q function takes the form

$$\hat{Q}_i(x_k, u_k, d_k, w_i) = w_{i,1}^T \varphi_1(z_k) + w_{i,21}^T \varphi_2(z_k) u_{k1} + w_{i,22}^T \varphi_2(z_k) u_{k2} + w_{i,31} u_{k1}^2 + w_{i,32} u_{k2}^2, \quad (23)$$

where $\phi(z_k) = [\varphi_1^T(z_k) \quad \varphi_2^T(z_k)u_{k1} \quad \varphi_2^T(z_k)u_{k2} \quad u_{k1}^2 \quad u_{k2}^2]^T$ follows from equation (21). Specifically, the elements that make up $\varphi_1(\cdot)$ and $\varphi_2(\cdot)$ are independent of u_{k1} and u_{k2}.

Applying equation (16) to determine the control policy, we have

$$\frac{\partial \hat{Q}_i(x_k, u_k, d_k, w_i)}{\partial u_{k1}} = w_{i,21}^T \varphi_2(z_k) + 2 w_{i,31} u_{k1} = 0$$

$$u_{k1} = -\frac{1}{2 w_{i,31}} w_{i,21}^T \varphi_2(z_k) \quad (24)$$

$$\frac{\partial \hat{Q}_i(x_k, u_k, d_k, w_i)}{\partial u_{k2}} = w_{i,22}^T \varphi_2(z_k) + 2 w_{i,32} u_{k2} = 0$$

$$u_{k2} = -\frac{1}{2 w_{i,32}} w_{i,22}^T \varphi_2(z_k). \quad (25)$$

reorganizing in matrix form, the control policy can be written as

$$h_i(x_k, d_k) = -\frac{1}{2} \begin{bmatrix} w_{i,31} & 0 \\ 0 & w_{i,32} \end{bmatrix}^{-1} \begin{bmatrix} \varphi_2^T(z_k) & \mathbf{0_{1\times 12}} \\ \mathbf{0_{1\times 12}} & \varphi_2^T(z_k) \end{bmatrix} w_{i,2}, \quad (26)$$

in which .$w_{i,2} = \begin{bmatrix} w_{i,21}^T & w_{i,22}^T \end{bmatrix}^T$

In reinforcement learning, the actor is the agent that generates the control policy, that is, the actor is described mathematically by equation (26). The critic is described by equation (23).

4.2 ACTOR-CRITIC ONLINE LEARNING ALGORITHM

One aspect related to the actor-Critic approach is that the estimates of the Q function of a given control policy are updated at each step of time k using observed data from the system (manipulator states). To this end, the iterative algorithm of recursive least squares (RLS) will be used to estimate the weight vector w_i. The efficiency of the RLS method in

online learning is mainly due to its robustness to deal with time variations in the regression parameters and the fast convergence speed (Ferreira; Rêgo; Neto, 2017).

Therefore, by applying the RLS algorithm, the estimation of the RNA weights, at each time step, k is given by

$$w_{k+1} = w_k + K_k \delta_k \tag{27}$$

$$K_k = \frac{P_k \phi(z_k)}{\lambda + \phi(z_k)^T P_k \phi(z_k)} \tag{28}$$

$$P_{k+1} = \frac{1}{\lambda}\left[P_k - \frac{P_k \phi(z_k)\phi(z_k)^T P_k}{\lambda + \phi(z_k)^T P_k \phi(z_k)}\right], \tag{29}$$

where λ, $0 < \lambda \leq 1$ is the forgetting factor and P_k is the inverse correlation matrix.

The reinforcement learning scheme employed in this work requires a stable initial control policy. The purpose is to keep the controller stable during the initial instants until the agent acquires enough experience (observing the environment) so that a new policy can be calculated. For simplicity, the neural network gains must be initialized in such a way as to result in a discrete PD (Proportional-Derivative) controller. This can be implemented by modifying the weights of the equation (26), where it is observed that:

$$\varphi_2(z_k) = [e_{k1} \quad e_{k2} \quad e_{k3} \quad e_{k4} \quad e_{k1}^2 \quad e_{k2}^2 \quad e_{k3}^2 \quad e_{k4}^2 \quad x_{k1}^2 \quad x_{k2}^2 \quad x_{k3}^2 \quad x_{k4}^2]^T, \tag{30}$$

that is, $h(x_k, d_k)$ depends directly on the position and speed errors of the links. That said, it can be seen that it is easy to obtain a PD control by establishing, for example,

$$\begin{aligned} w_{i,31} &= w_{i,32} = \frac{1}{2}, \\ w_{i,21} &= [K_{P_1} \quad 0 \quad K_{D_1} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T, \\ w_{i,22} &= [0 \quad K_{P_2} \quad 0 \quad K_{D_2} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T, \end{aligned} \tag{31}$$

where K_{P_1} and K_{D_1}, and K_{P_2} and K_{D_2} are the proportional and derivative gains, respectively, of joints 1 and 2. Adjustment of these parameters will be performed by trial and error.

A summary of the actor-Critic reinforcement learning algorithm implemented in this study is presented below.

Table 1

Actor-Critic RL Algorithm

Input: discount factor γ, learning factor α, initial value of the covariance matrix β, and the forgetting factor λ.

Initialize the neural network weights in a way that ensures a stable PD controller.
Measure initial states x_0 and trajectory erros e_0. Initialize the arrays $P_0 = \beta I$, Q_c, R and S arbitrarily and $i = 0$.

Repeat for each states sample $k = 0, 1, 2, \ldots$

▷ **Noise signal as an exploration component**

$\xi = [\,]$

▷ **Control Signal**

$$u_k = h_i(x_k, d_k) + \xi$$

Apply and measure states $u_k x_{k+1}$

$$u_{k+1} = h_i(x_{k+1}, d_{k+1})$$

$$e_{k+1} = x_{k+1} - d_{k+1}$$

$$r_k = e_k^T Q_c e_k + (u_{k+1} - u_k)^T S(u_{k+1} - u_k) + u_k^T R u_k$$

▷ **Recursive Least Squares - RLS**

$$\Delta_{objetivo} = r_k + \gamma \hat{Q}(x_{k+1},\, u_{k+1},\, e_{k+1})$$

$$\widehat{W}_k = w_k^T \phi_k$$

$$K_k = \frac{P_k \phi_k}{\lambda + \phi_k^T P_k \phi_k}$$

$$w_{k+1} = w_k + K_k\left(\Delta_{objetivo} - \widehat{W}_k\right)$$

$$P_{k+1} = \frac{1}{\lambda}\left(P_k - \frac{P_k \phi_k \phi_k^T P_k}{\lambda + \phi_k^T P_k \phi_k}\right)$$

If End of a learning period:

$$w_{i+1(ctrl)} = \alpha w_{k+1} + (1 - \alpha) w_{i(ctrl)}$$

▷**Updating the control policy**

$$h_{i+1} \leftarrow -\frac{1}{2}\begin{bmatrix} w_{i+1(ctrl),31} & 0 \\ 0 & w_{i+1(ctrl),32} \end{bmatrix}^{-1} \begin{bmatrix} \varphi_2^T(z_k) & \mathbf{0_{1\times 10}} \\ \mathbf{0_{1\times 10}} & \varphi_2^T(z_k) \end{bmatrix} w_{i(ctrl),2}$$

$$P_{k+1} = \beta I$$

$$i = i + 1$$

End up

up to meet the stop criterion

The algorithm starts with the RNA gains arbitrarily defined to produce the effect of a PD control. The initial condition of the inverse correlation matrix of the RLS given in the form $P_0 = \beta I$, where β is a constant with a sufficiently large value and I is the identity matrix, was considered. During the first momens, there is no update in the control policy to ensure stability during the initial learning, however the weight vector w_k is calculated at each step by applying equations (27) to (29). The control signal is obtained at each time k using (26). A noise signal ξ, known as exploration noise, is added to the control input for the purpose of online learning (Jiang; Jiang, 2017).

$$w_{i+1(ctrl)} = \alpha w_{k+1} + (1-\alpha) w_{i(ctrl)}, \quad (32)$$

where $w_{i(ctrl)}$ are the parameters of the controller implemented during the -ith cycle,$0 < \alpha \leq 1$ is the learning factor. At that moment, the matrix P is redefined. The controller weights are again kept unchanged until the current cycle has been completed. The process is repeated until the convergence of the network parameters. Once this goal is achieved, the controller operates at constant weights.

5 SIMULATION STRUCTURE

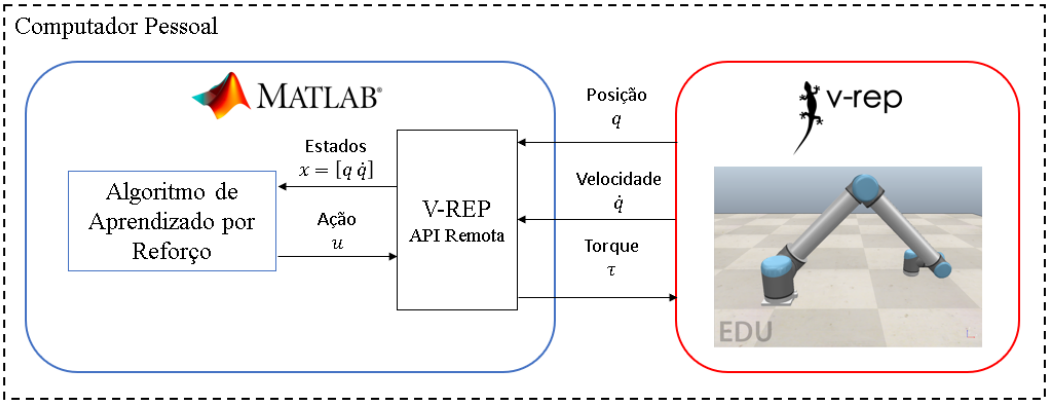
In order to provide a simulation structure that allows the development of the algorithms and the experiments, the *V-REP (*Virtual Robotics Experimentation Platform*) software was used* together with MATLAB (*Matrix Laboratory*). V-REP is a general-purpose robot simulator that provides multiple physics engines for the simulations, multiple robotic models, and multiple configurations of the environment. In this way, it is possible to customize all the objects in the scene, including the parameters of the sensors and actuators, thus allowing more faithful results to be achieved (Rohmer; Singh; Freese, 2013).

In the V-REP, different means of controlling the objects/models in the scene are available, either through embedded routines, ROS (*Robot Operating System) nodes* (Quigley; Gerkey; Smart, 2015), remote API (*Application Programming Interface*), a plugin or some custom solution. Controllers can be written in C/C++, Python, Java, Lua, and MATLAB (Shamshiri *et al*., 2018). In this study, the robotic model used in the simulator is controlled by an external routine developed on the MATLAB platform making use of the remote API. Figure 3 illustrates the communication between the controller and the simulation environment.

The settings to be followed for the proper functioning of the simulations using the platforms described above and within the context of reinforcement learning can be seen in detail in (Pluškoski; Ciganović; Jovanović, 2019).

Figure 3

V-REP control scheme by remote API via MATLAB



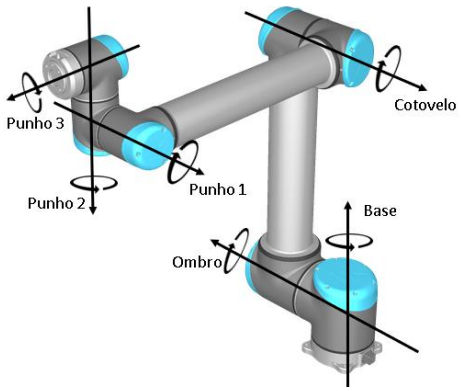
Source: Prepared by the authors.

6 SIMULATION RESULTS

In this section, the results of the simulations of the control scheme proposed in this work are presented and discussed. To perform these computational experiments, the UR10 robotic arm model available in the V-REP simulator was used. Since this articulator has six degrees of freedom, in these tests the torque generated by the control law will be applied only to the shoulder and elbow joints (Figure 4) while the other joints are deactivated and locked in their respective equilibrium positions (0º). The control was carried out using the remote API through routines implemented in the MATLAB platform.

Figure 4

UR10 manipulator joints



Source: Prepared by the authors.

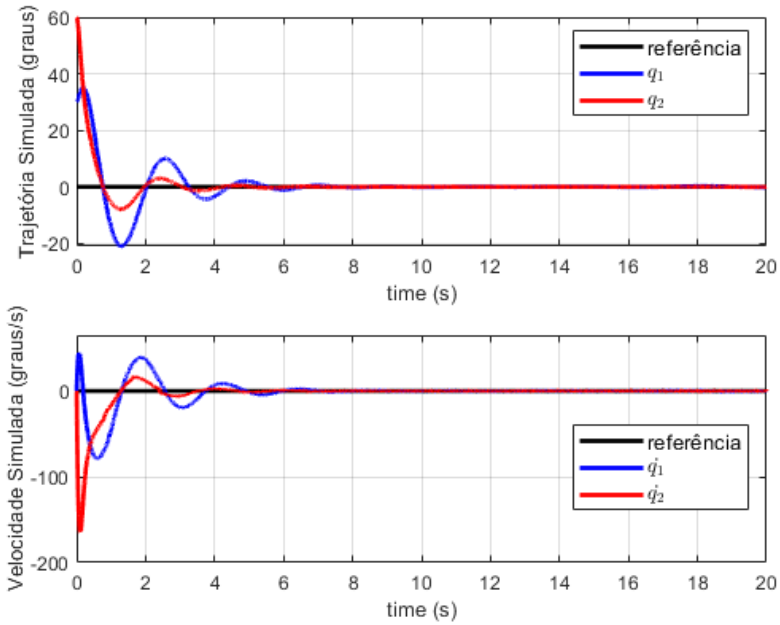
The evaluation of the control scheme via reinforcement learning will be done by analyzing the results of simulations of three tasks: regulation, trajectory following of a multi-step signal and a sinusoidal signal.

The behavior of the states for the regulation case is shown in Figure 5. The initial configuration of the joints has been set as $x_0 = [\pi/6 \quad \pi/3 \quad 0 \quad 0]^T$ and the controller

parameters have been adjusted to the following values $K_{P_1} = K_{P_2} = 150, K_{D_1} = K_{D_2} = 30, \gamma = 0{,}98, Q_c = diag(250,\ 250,\ 0{,}001,\ 0{,}001), \quad R = diag(0{,}0001,\ 0{,}0001), \quad \alpha = 0{,}2$ e $P_0 = 10^4 I_{78\times 78}$. The learning cycle for this simulation was 0.8 s. The control effort applied to the joints is shown in Figure 6 and the update of the actor's weights is shown in Figure 7.

Figure 5

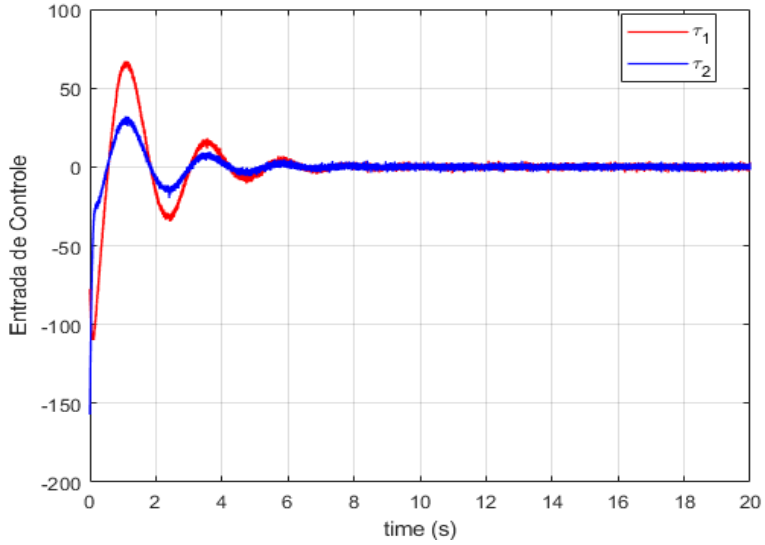
Trajectory of the states



Source: Prepared by the authors.

Figure 6

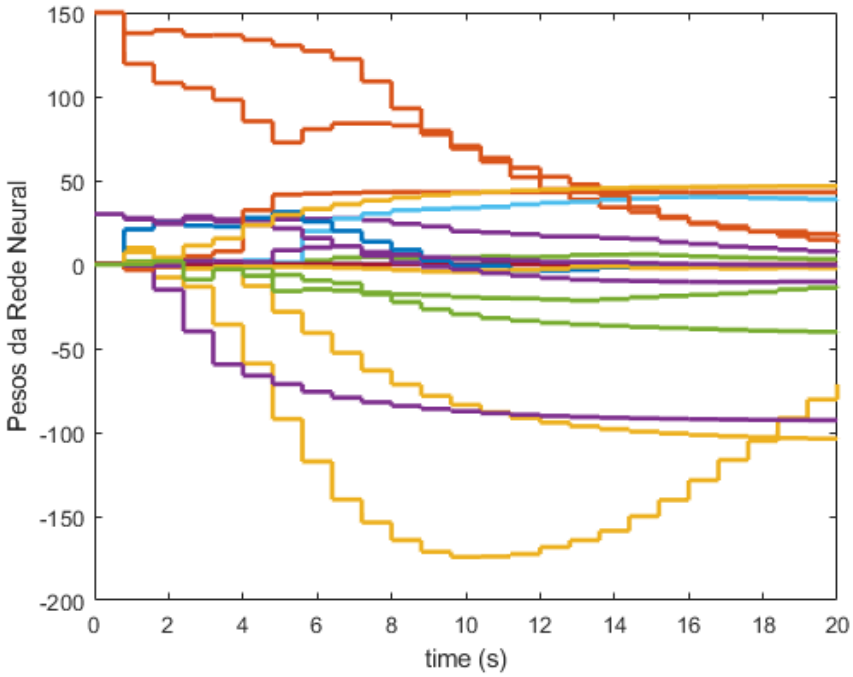
Control signal



Source: Prepared by the authors.

Figure 7

Actor network weight update



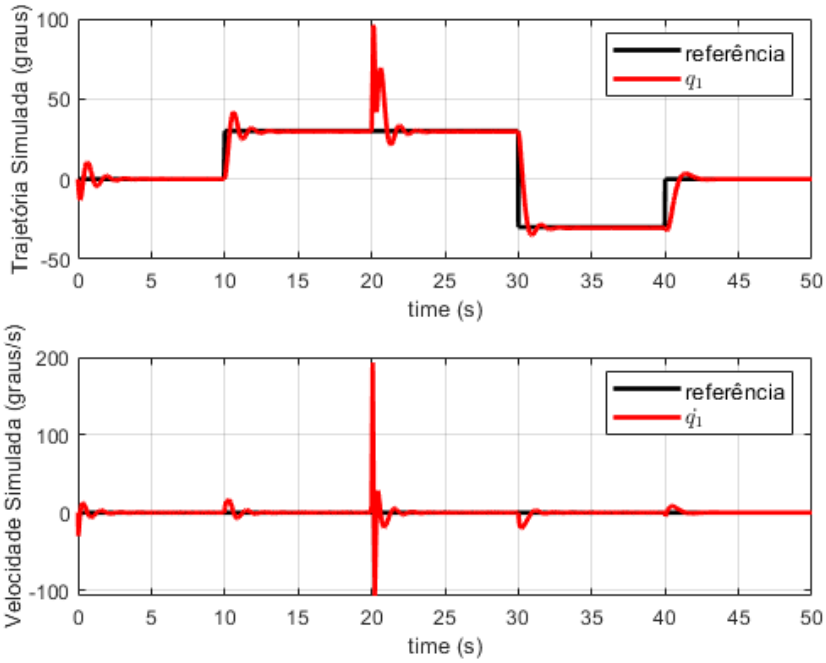
Source: Prepared by the authors.

In the second experiment suggested to validate the implemented controller, a multi-step reference signal was used, in order to simulate the pick and place task, commonly performed by manipulators. For this experiment, the initial state was set to $x_0 = [0 \quad 0 \quad 0 \quad 0]^T$. The control parameters were the same as those used for the regulation case, except for the following values $Q_c = diag(100, 100, 0{,}001, 0{,}001)$, $\alpha = 0{,}1$, $K_{P_1} = K_{P_2} = 500$, $K_{D_1} = K_{D_2} = 50$ the learning cycle changed to 2 s. Under these adjustments, the tracking response, the torque applied to the joints, and the update of the actor's net weights are presented in 8 through 11.

As seen, the joints are able to reach the reference signal with errors within acceptable limits and the stability of the system is maintained throughout the simulation time. It is also shown that at the time instant of 20 s there was an increase in the control signal causing an unwanted overtime, but in the following instants, from the 15th update of the policy (30 s), an improvement in tracking was observed in relation to the initial policy (first 2 s), a consequence of the learning acquired.

Figure 8

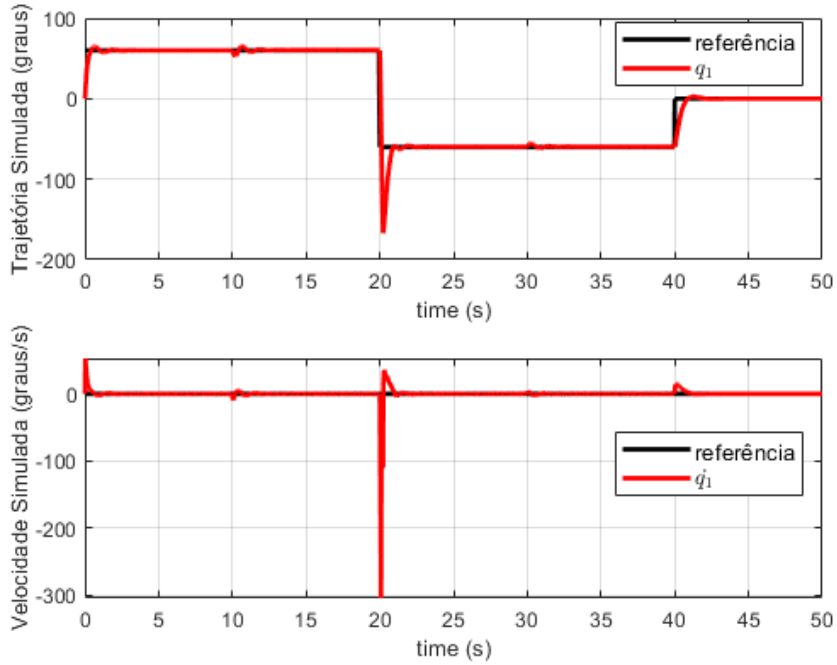
Trajectory tracking of the shoulder joint



Source: Prepared by the authors.

Figure 9

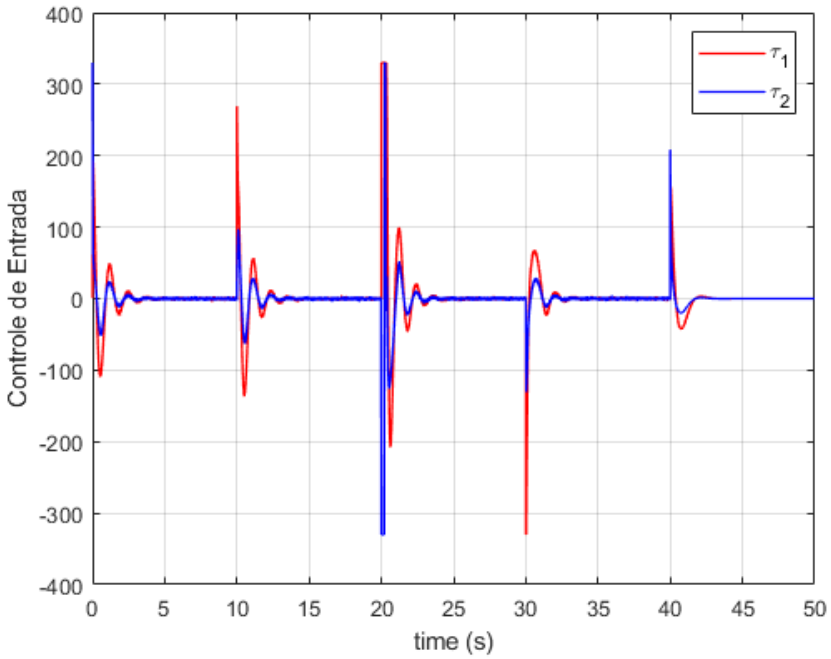
Trajectory tracking of the elbow joint



Source: Prepared by the authors.

Figure 10

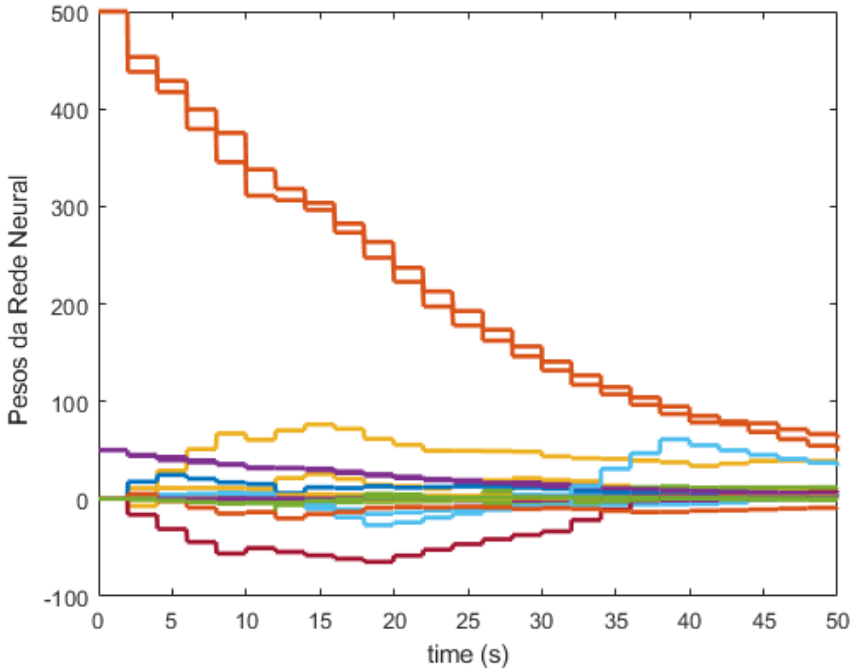
Control signal



Source: Prepared by the authors.

Figure 11

Actor network weight update



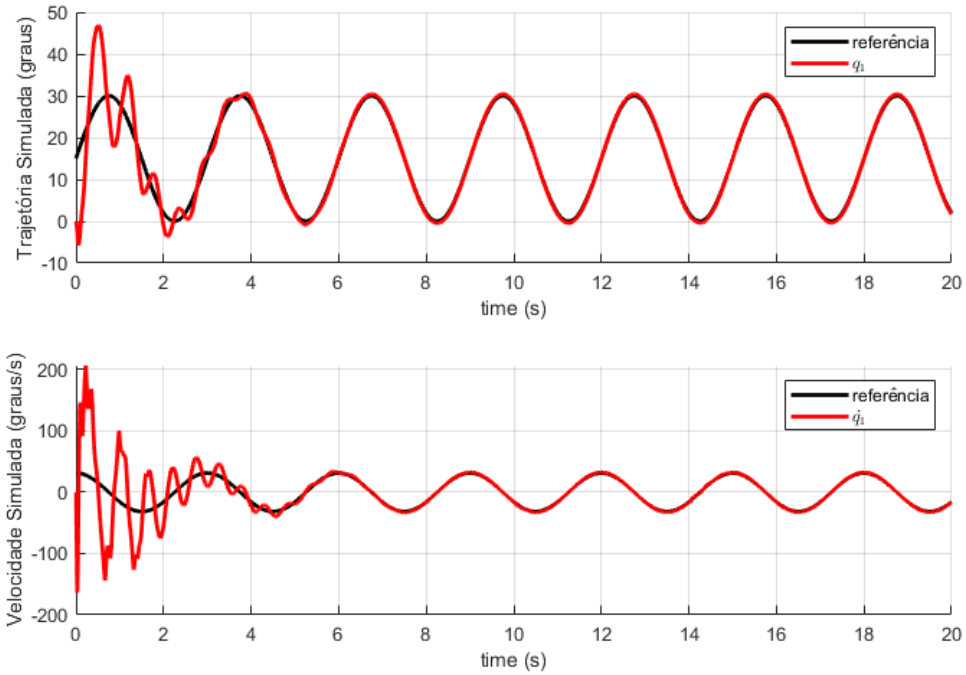
Source: Prepared by the authors.

In the last proposed experiment, a sinusoidal signal was established as a reference for the articulator joints under the following settings $Q_c = diag(200, 200, 0{,}001, 0{,}001)$, $K_{P_1} = 4000$, $K_{P_2} = 2000$, $K_{D_1} = 50$, $K_{D_2} = 20$ e $\alpha = 0{,}4$. The other parameters were set at the same

values as in experiment 2. The results of the simulation are observed in Figures 12 to 15. According to Figures 12 and 13, where the tracking performance is shown, it is observed that the trajectory following improves at the end of each learning cycle (2 s intervals). From the third cycle onwards, tracking errors stabilize within tolerable limits.

Figure 12

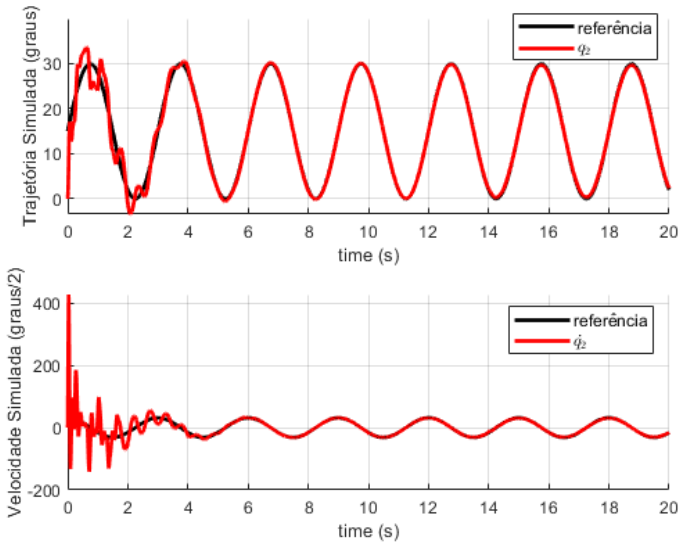
Trajectory tracking of the shoulder joint



Source: Prepared by the authors.

Figure 13

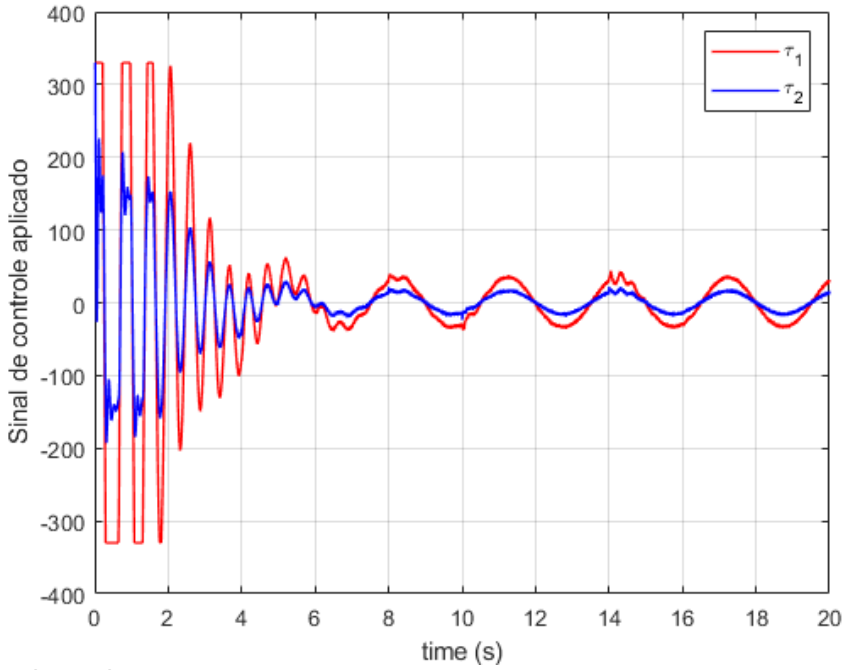
Trajectory tracking of the elbow joint



Source: Prepared by the authors.

Figure 14

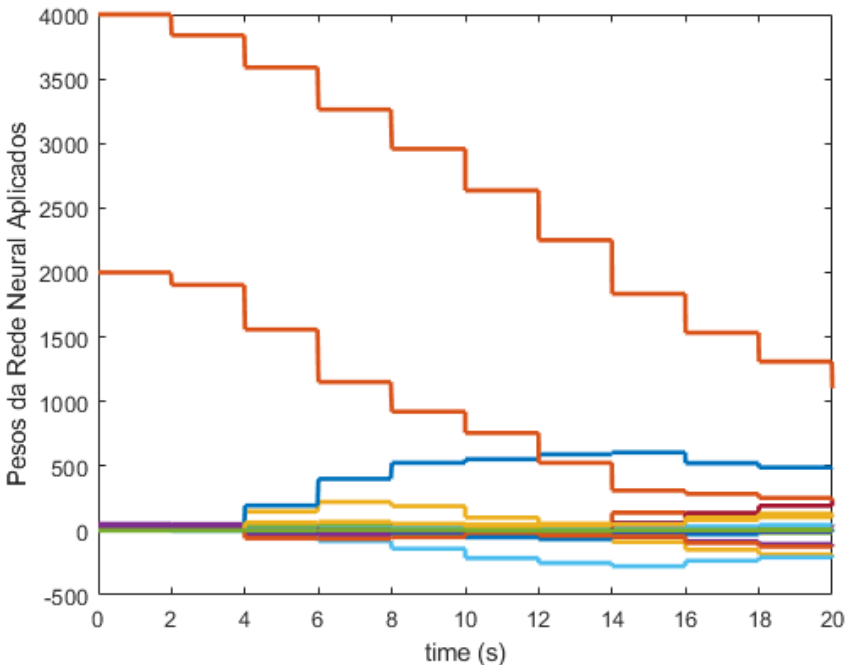
Control signal



Source: Prepared by the authors.

Figure 15

Actor network weight update



Source: Prepared by the authors.

7 CONCLUSION

In this work, a control scheme based on reinforcement learning applied to a robotic manipulator was proposed, using an actor-Critic approach. In this design, only one neural

network was trained to approximate the Q function using only the actual system measurements via the RLS estimator. In order to provide robustness to the scheme, the update of the control policy, obtained by minimizing the Q function, occurs at the end of a fixed number of iterations (learning cycle), remaining constant during this interval. To approximate the action-value function, a polynomial neural network was used, proving to be adequate to learn the nonlinearities of the manipulator. Computational experiments with the presented controller were carried out using the UR10 robot model in the V-REP simulator. The simulations included the task of regulating and tracking the trajectory of the sinusoidal and multi-step signals. In the simulated results, it was observed the stability of the state variables during the entire simulation time and the ability to track the reference signals, even without explicit knowledge of the manipulator dynamics.

ACKNOWLEDGMENTS

The present work was carried out with the support of the Dean of Research and Graduate Studies - PPG of the State University of Maranhão - UEMA.

REFERENCES

Abbas, Z. (2018). Motion control of robotic arm manipulator using PID and sliding mode technique [Tese de doutorado, Capital University of Science and Technology].

Al-Olimat, K. S., & Ghandakly, A. A. (2002). Multiple model reference adaptive control algorithm using on-line fuzzy logic adjustment and its application to robotic manipulators. In Conference Record of the 2002 IEEE Industry Applications Conference. 37th IAS Annual Meeting (pp. 1463–1466). IEEE.

Alquadi, B., et al. (2016). Model reference adaptive impedance control for physical human-robot interaction. Control Theory and Technology, 14, 68–82.

Bhatnagar, S., et al. (2009). Natural actor-critic algorithms. Automatica, 45(11), 2471–2482.

Borase, R. P., et al. (2021). A review of PID control, tuning methods and applications. International Journal of Dynamics and Control, 9, 818–827.

Cao, S., et al. (2023). Reinforcement learning-based fixed-time trajectory tracking control for uncertain robotic manipulators with input saturation. IEEE Transactions on Neural Networks and Learning Systems, 34(8), 4584–4595.

Chen, L., Dai, S.-L., & Dong, C. (2024a). Adaptive optimal tracking control of an underactuated surface vessel using actor–critic reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems, 35(6), 7520–7533.

Chen, L., Dong, C., & Dai, S.-L. (2024b). Adaptive optimal consensus control of multiagent systems with unknown dynamics and disturbances via reinforcement learning. IEEE Transactions on Artificial Intelligence, 5(5), 2193–2203.

Chen, W.-D. (2005). Experimental study of robot manipulators based on robust adaptive control. In International Conference on Machine Learning and Cybernetics (pp. 18–21).

Clegg, A. C., Dunnigan, M. W., & Lane, D. M. (2001). Self-tuning position and force control of an underwater hydraulic manipulator. In Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (pp. 3226–3231). IEEE.

Craig, J. J. (2021). Introduction to robotics: Mechanics and control (4th ed., Global ed.). Pearson.

Dubowsky, S., & Desforges, D. T. (1979). The application of model-referenced adaptive control to robotic manipulators. Journal of Dynamic Systems, Measurement, and Control, 101(3), 193–200.

Fateh, S., & Fateh, M. M. (2019). Adaptive fuzzy control of robot manipulators with asymptotic tracking performance. Journal of Control, Automation and Electrical Systems, 31, 52–61.

Ferreira, E. F. M., Rêgo, P. H. M., & Neto, J. V. F. (2017). Numerical stability improvements of state-value function approximations based on RLS learning for online HDP-DLQR control system design. Engineering Applications of Artificial Intelligence, 63, 1–19.

Freire, E. O., Rossomando, F. G., & Soria, C. M. (2018). Self-tuning of a neuro-adaptive PID controller for a SCARA robot based on neural network. IEEE Latin America Transactions, 16(5), 1364–1374.

Guo, X., Yan, W., & Cui, R. (2020). Reinforcement learning-based nearly optimal control for constrained-input partially unknown systems using differentiator. IEEE Transactions on Neural Networks and Learning Systems, 31(11), 4713–4725.

He, W., et al. (2021). Reinforcement learning control of a flexible two-link manipulator: An experimental investigation. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(12), 7326–7336.

Hu, Q., Xu, L., & Zhang, A. (2012). Adaptive backstepping trajectory tracking control of robot manipulator. Journal of the Franklin Institute, 349(3), 1087–1105.

Hu, Y., & Si, B. (2018). A reinforcement learning neural network for robotic manipulator control. Neural Computation, 30(7), 1983–2004.

Jiang, Y., & Jiang, Z.-P. (2017). Robust adaptive dynamic programming. John Wiley & Sons.

Kamboj, A., et al. (2020). Discrete-time Lyapunov based kinematic control of robot manipulator using actor-critic framework. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1–7). IEEE.

Khan, S. G., et al. (2011). A Q-learning based Cartesian model reference compliance controller implementation for a humanoid robot arm. In 2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM) (pp. 214–219). IEEE.

Khan, S. G., et al. (2012). Reinforcement learning and optimal adaptive control: An overview and implementation examples. Annual Reviews in Control, 36(1), 42–59.

Khan, S. G., et al. (2019). Reinforcement learning based compliance control of a robotic walk assist device. Advanced Robotics, 33(24), 1281–1292.

Kiumarsi, B., et al. (2018). Optimal and autonomous control using reinforcement learning: A survey. IEEE Transactions on Neural Networks and Learning Systems, 29(6), 2042–2062.

Konstantopoulos, G. C., & Baldivieso-Monasterios, P. R. (2020). State-limiting PID controller for a class of nonlinear systems with constant uncertainties. International Journal of Robust and Nonlinear Control, 30, 1770–1787.

Maliotis, G. A. (1991). Hybrid model reference adaptive control/computed torque control scheme for robotic manipulators. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 205(3), 215–221.

Moosavi, S. K. R., Zafar, M. H., & Sanfilippo, F. (2022). Forward kinematic modelling with radial basis function neural network tuned with a novel meta-heuristic algorithm for robotic manipulators. Robotics, 11(2), 1–17.

Pane, Y. P., et al. (2019). Reinforcement learning based compensation methods for robot manipulators. Engineering Applications of Artificial Intelligence, 78, 236–247.

Pane, Y. P., Nageshrao, S. P., & Babuška, R. (2016). Actor-critic reinforcement learning for tracking control in robotics. In 2016 IEEE 55th Conference on Decision and Control (CDC) (pp. 5819–5826). IEEE.

Peters, J., & Schaal, S. (2008a). Learning to control in operational space. International Journal of Robotics Research, 27(2), 197–212.

Peters, J., & Schaal, S. (2008b). Natural actor-critic. Neurocomputing, 71(7–9), 1180–1190.

Pluškoski, A., Ciganović, I., & Jovanović, M. D. (2019). Benefits of residual networks in reinforcement learning using V-Rep simulator. In 6th International Conference IcETRAN (pp. 1–6).

Qi, R., Tao, G., & Jiang, B. (2019). Adaptive control: A tutorial introduction. In Fuzzy system identification and adaptive control (pp. 55–74). Springer.

Quigley, M., Gerkey, B., & Smart, W. D. (2015). Programming robots with ROS: A practical introduction to the robot operating system. O'Reilly Media.

Rohmer, E., Singh, S. P. N., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. In IEEE/RSJ International Conference on Intelligent Robots and Systems.

Sasaki, M., et al. (2009). Self-tuning control of a two-link flexible manipulator using neural networks. In 2009 ICCAS-SICE (pp. 2468–2473).

Shah, H., & Gopal, M. (2009). Reinforcement learning control of robot manipulators in uncertain environments. In IEEE International Conference on Industrial Technology (pp. 1–6). IEEE.

Shamshiri, R. R., et al. (2018). Robotic harvesting of fruiting vegetables: A simulation approach in V-REP, ROS and MATLAB. In Automation in agriculture - Securing food supplies for future generations. IntechOpen.

Su, Y., et al. (2025). Fixed-time optimal trajectory tracking control for an unmanned surface vehicle via reinforcement learning. IEEE/ASME Transactions on Mechatronics, 1–12.

Sun, N., et al. (2020). Adaptive control for pneumatic artificial muscle systems with parametric uncertainties and unidirectional input constraints. IEEE Transactions on Industrial Informatics, 16(2), 969–979.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

Vrabie, D., Vamvoudakis, K. G., & Lewis, F. L. (2013). Optimal adaptive control and differential games by reinforcement learning principles. The Institution of Engineering and Technology.

Wang, Z., et al. (2025). Adaptive altitude control for underwater vehicles based on deep reinforcement learning. In 2025 8th International Conference on Transportation Information and Safety (ICTIS) (pp. 79–84).

Wu, L., Yan, Q., & Cai, J. (2019). Neural network-based adaptive learning control for robot manipulators with arbitrary initial errors. IEEE Access, 7, 180194–180204.

Yaghmaie, F. A., Gustafsson, F., & Ljung, L. (2023). Linear quadratic control using model-free reinforcement learning. IEEE Transactions on Automatic Control, 68(2), 737–752.

Yılmaz, B. M., et al. (2022). Self-adjusting fuzzy logic based control of robot manipulators in task space. IEEE Transactions on Industrial Electronics, 69(2), 1620–1629.

Zhang, D., & Wei, B. (2017). Design, analysis and modelling of a hybrid controller for serial robotic manipulators. Robotica, 35(9), 1888–1905.

Zhao, D., et al. (2025). Linear quadratic control of unknown nonlinear systems using model-free reinforcement learning. IEEE Transactions on Industrial Electronics, 72(12), 13751–13762.