Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.074993

ARTICLE

Actor–Critic Trajectory Controller with Optimal Design for Nonlinear Robotic Systems

Nien-Tsu Hu[1,*], Hsiang-Tung Kao[1], Chin-Sheng Chen[1] and Shih-Hao Chang[2]

[1]Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei, 10608, Taiwan
[2]Computer Science and Information Engineering, National Taipei University of Technology, Taipei, 10608, Taiwan

*Corresponding Author: Nien-Tsu Hu. Email: nthu@ntut.edu.tw

Received: 22 October 2025; Accepted: 18 December 2025; Published: 10 February 2026

ABSTRACT: Trajectory tracking for nonlinear robotic systems remains a fundamental yet challenging problem in control engineering, particularly when both precision and efficiency must be ensured. Conventional control methods are often effective for stabilization but may not directly optimize long-term performance. To address this limitation, this study develops an integrated framework that combines optimal control principles with reinforcement learning for a single-link robotic manipulator. The proposed scheme adopts an actor–critic structure, where the critic network approximates the value function associated with the Hamilton–Jacobi–Bellman equation, and the actor network generates near-optimal control signals in real time. This dual adaptation enables the controller to refine its policy online without explicit system knowledge. Stability of the closed-loop system is analyzed through Lyapunov theory, ensuring boundedness of the tracking error. Numerical simulations on the single-link manipulator demonstrate that the method achieves accurate trajectory following while maintaining low control effort. The results further show that the actor–critic learning mechanism accelerates convergence of the control policy compared with conventional optimization-based strategies. This work highlights the potential of reinforcement learning integrated with optimal control for robotic manipulators and provides a foundation for future extensions to more complex multi-degree-of-freedom systems. The proposed controller is further validated in a physics-based virtual Gazebo environment, demonstrating stable adaptation and real-time feasibility.

KEYWORDS: Reinforcement learning; optimal control; actor–critic algorithm; trajectory tracking; nonlinear systems; robotic manipulator

1 Introduction

In recent years, intelligent and optimized methodologies for nonlinear system control have drawn significant attention. Owing to the complexity of nonlinear dynamics, traditional linear control approaches are often inadequate in practical applications. To address unknown dynamics and uncertainties, artificial neural networks (ANNs), especially radial basis function neural networks (RBFNNs), have been widely used for nonlinear system modelling and controller design. With Lyapunov-based adaptation schemes, such RBFNN-based controllers can guarantee that all closed-loop signals are semi-globally uniformly ultimately bounded (SGUUB) [1].

On the other hand, the mathematical foundation of optimal control is based on the Hamilton–Jacobi–Bellman (HJB) equation, which is generally intractable for nonlinear systems. Approximation methods such as Galerkin schemes have been developed to provide feasible solutions, laying the groundwork for the

integration of adaptive dynamic programming (ADP) and reinforcement learning (RL) into nonlinear control [2]. From a theoretical standpoint, the foundations of nonlinear analysis and stability are systematically presented in Khalil's Nonlinear Systems, which covers Lyapunov stability, backstepping, and input–output stability [3]. Similarly, the comprehensive framework of optimal control, extending from linear quadratic regulation (LQR) to nonlinear HJB approaches, is well established in Lewis's Optimal Control [4].

More recently, advanced RL-based methods have been introduced for robotic systems subject to complex constraints. For instance, Peng et al. developed an event-sampled critic learning strategy to solve the optimal tracking control problem of motion-constrained robot manipulators, reducing computational burden while ensuring stability [5]. Cheng and Dong proposed a single-critic neural network approach for robotic manipulators with input saturation and disturbances, with stability guaranteed through Lyapunov theory [6]. Hu et al. further extended actor–critic network designs to handle manipulators with dynamic disturbances, ensuring SGUUB stability and accurate trajectory tracking [7].

In addition, Wen et al. proposed optimized adaptive tracking and backstepping frameworks that integrate actor–critic RL into nonlinear control. In these works, both virtual and actual control inputs are designed as optimal solutions at each recursive step, and the resulting controllers simultaneously achieve trajectory tracking and performance optimization [8,9]. In the broader context of multi-agent systems (MASs), Wen et al. further introduced an identifier–actor–critic algorithm based on fuzzy logic systems for MASs with unknown nonlinear dynamics, addressing state coupling and ensuring stability under Lyapunov analysis [10]. These representative studies highlight how filtered or composite errors can be exploited to structure the optimal tracking problem in both single-system and multi-agent settings.

Beyond these developments, many tracking designs employ the reference error to streamline stability analysis and shape tracking dynamics [11]. For transient and steady-state guarantees, prescribed-performance control (PPC) provides Lyapunov-based performance envelopes that accommodate uncertainties and actuator limitations in robotic manipulators; recent works further strengthen PPC with Barrier Lyapunov Function (BLF)-based designs, assigned/appointed settling time, and neural adaptations [12–14]. On the optimal-learning side, actor–critic reinforcement learning has been applied to robotic manipulators to enhance practical tracking performance with stability-oriented designs, and RL-based controllers have also been developed for nonlinear syst ems such as single-link manipulators [15,16]. Moreover, model-based actor–critic learning has been exploited for robotic impedance control in complex interactive environments, where the actor–critic framework is used to learn impedance behavior for safe and efficient robot–environment interaction [17].

In parallel, integral and off-policy reinforcement learning techniques have been investigated to address optimal tracking under disturbances, input constraints, and model uncertainty. For perturbed discrete-time systems, H∞ tracking control frameworks based on On/Off policy Q-learning have been proposed, where model-free Q-learning algorithms are shown to guarantee convergence of the learned policy in the presence of external disturbances [18]. For perturbed bilateral teleoperators with variable time delay, nonlinear RISE-based integral RL algorithms have been developed by combining robust integral action with actor–critic structures to achieve both optimal performance and coordination tracking [19]. Off-policy integral RL-based optimal tracking schemes have also been proposed for nonzero-sum game systems with unknown dynamics, further relaxing model requirements while maintaining stability and disturbance attenuation [20].

Together, these contributions indicate that the integration of RBFNNs, Lyapunov stability analysis, HJB approximations, and actor–critic reinforcement learning provides an effective pathway for solving nonlinear optimal tracking problems. At the same time, most existing RL-based optimal tracking schemes still exhibit several limitations: many require partial model knowledge or rely on Persistent of Excitation (PE)-type excitation conditions to guarantee convergence of the actor–critic networks, and the transient dynamics

are usually determined implicitly by Lyapunov designs rather than by an explicitly tunable reference-error structure. In addition, the interaction between optimal cost design, actuator constraints, and reference-error shaping has not been fully explored in the context of nonlinear robotic systems.

To clarify how the proposed scheme is tailored to address the above gaps, we explicitly match each limitation with a corresponding design element of our method.

(i) Reliance on partial model knowledge: many RL/HJB/ADP controllers still require certain model components for policy/value formulation and stability analysis; our approach integrates an online actor–critic structure with RBFNN approximation to reduce dependence on precise model parameters while retaining Lyapunov-based stability analysis (see Sections 3 and 4).

(ii) Persistence of excitation (PE) requirements: theoretical convergence of adaptive/learning laws typically relies on PE; we adopt a standard PE condition for weight convergence analysis and provide implementation-oriented discussions/experiments to illustrate stable learning under practical excitation levels (Assumption 2 in Section 3.2; results in Sections 5 and 6).

(iii) Implicit transient tuning: transient behavior is often tuned implicitly through multiple gains without a clear performance knob; we introduce a structured reference-error design with an explicit gain to shape tracking dynamics in a transparent manner, while preserving the Lyapunov proof (see Sections 2 and 4; parameter study in Section 5).

(iv) Lack of a structured reference-error mechanism in optimal learning designs: by embedding the reference-error into the learning/control formulation, the proposed method establishes a one-to-one link between tracking-dynamics shaping and optimal-learning updates, which is validated in both numerical simulations and physics-based environments (see Sections 2, 3, 5 and 6).

Motivated by these developments and limitations, this work proposes a reinforcement learning–based optimal tracking control framework for robotic manipulators, with a particular focus on trajectory tracking under nonlinear dynamics. The key novelty of this work lies in augmenting the actor–critic optimal control framework with a tunable reference error structure, enabling direct adjustment of convergence speed while preserving Lyapunov stability guarantees and real-time implementability in both numerical and physics-based environments.

Contributions of this work are summarized as follows:

1. A structured reference-error formulation that explicitly shapes tracking transients with a tunable gain.
2. A Lyapunov-stability-oriented actor–critic learning design with online approximation.
3. Comparative studies and validations in both numerical simulations and physics-based environments.

2 Problem Formulation

2.1 System Model

The single-link robotic manipulator is widely used as a benchmark system to evaluate nonlinear control methods. Its dynamic equation can be expressed as

$$J\ddot{q}(t) + B\dot{q}(t) + Mglsin(q(t)) = \tau(t) \tag{1}$$

where $q(t)$ denotes the angular position of the manipulator, $\dot{q}(t)$ and $\ddot{q}(t)$ are the angular velocity and acceleration, respectively, and $\tau(t)$ is the input torque. The parameters are defined as follows: J represents the total rotational inertia, B is the damping coefficient, M is the mass of the link, l is the distance from the pivot to the center of mass, and g is the gravitational acceleration.

Following the definition of the manipulator dynamics in Eq. (1), the tracking error with respect to a desired trajectory $q_d(t)$ is introduced as

$$e(t) = q_d(t) - q(t) \tag{2}$$

To enforce convergence of the tracking error, reference error is defined as

$$r(t) = \dot{e}(t) + \Lambda e(t) \tag{3}$$

where Λ is a constant gain specifying the convergence rate.

Substituting Eq. (1) into the above, the closed-loop error channel can be rearranged as

$$J\dot{r}(t) = -Br(t) - \tau(t) + h(t) \tag{4}$$

where

$$h(t) = J(\ddot{q} + \Lambda\dot{e}) + B(\dot{q}_d + \Lambda e) + Mgl\sin(q) \tag{5}$$

and the augmented state is given by

$$\chi(t) = \begin{bmatrix} e(t) \\ r(t) \end{bmatrix} \tag{6}$$

Introducing the auxiliary input

$$u(t) = h(t) - \tau(t) \tag{7}$$

the error dynamics reduce to

$$\dot{r}(t) = -\frac{B}{J}r(t) + \frac{1}{J}u(t) \tag{8}$$

Combining with Eq. (3), the augmented system can be expressed in compact form as

$$\begin{aligned} \dot{\chi}(t) &= \begin{bmatrix} -\Lambda & I \\ 0 & -\frac{B}{J} \end{bmatrix}\chi(t) + \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix}u(t) \\ &= f(\chi) + g(\chi)u(t) \end{aligned} \tag{9}$$

Assumption 1: *The system dynamics $f(\chi)$ and the control gain $g(\chi)$ in Eq. (9) are locally Lipschitz and bounded on a compact set $\Omega \subset R^n$ that contains all closed-loop trajectories $\chi(t)$. Moreover, $g(\chi)$ is nonsingular for all $\chi \in \Omega$.*

Remark 1: *Although Eq. (9) is written in a compact state-space form that is linear in the error state vector χ and the control input u, this does not mean that the overall robotic system is linear. As shown in the original manipulator model in Eq. (1), the dynamics contain inherently nonlinear terms, including the inertia matrix, Coriolis/centrifugal forces, and gravity, which depend nonlinearly on the joint position and velocity (*e.g., *trigonometric functions). Eq. (9) is obtained by defining the tracking and reference errors and regrouping the dynamics into a convenient control-affine representation for controller design, without invoking any small-angle approximation or local linearization of the plant. In addition, the control policy generated by the actor network is a nonlinear function of the state, so the closed-loop dynamics obtained by substituting this policy*

into Eq. (9) remain nonlinear. Therefore, the proposed actor–critic reinforcement learning controller is still designed and analyzed in the context of nonlinear robotic systems, even though the error dynamics are expressed in a linear-like form in Eq. (9).

2.2 Optimal Control Design

To measure tracking performance and input usage, we introduce the performance index in terms of a running-cost surrogate:

$$J = \int_t^{\infty} L\left(\chi\left(\tau\right), u\left(\tau\right)\right) d\tau \tag{10}$$

with the running cost $L\left(\cdot\right)$ is defined as

$$L\left(\chi, u\right) \triangleq \chi^T Q \chi + u^T u \tag{11}$$

where $Q = g\left(\chi\right) g^T\left(\chi\right) \in R^{n\times n}$ is a positive definite matrix, ensuring that both state deviation and input magnitude are penalized.

Definition 1: *A control strategy u is considered to belong to the admissible set* Ω, *written as* $u \in \Psi\left(\Omega\right)$, *provided that u is continuous, meets the condition, and ensures stability of the augmented error system in Eq. (9), and renders the performance index in Eq. (10) finite for all* $\chi \in \Omega$.

Here, J in Eq. (10) represents the general performance index for an admissible policy $u\left(t\right)$. The optimal performance index is denoted as $J^*\left(\chi\right)$, which corresponds to the minimum cost achievable from the error state χ under the optimal policy u^*.

According to optimal control theory, the Hamilton–Jacobi–Bellman (HJB) equation for this problem can be written as

$$\begin{aligned} &H\left(\chi, u^*, \nabla J^*\right) = L\left(\chi, u^*\right) + \nabla J^{*\mathrm{T}}\left(\dot{\chi}\right) \\ &= \chi^T Q \chi + u^{*T} u^* + \nabla J^*\left(f\left(\chi\right) + g\left(\chi\right) u^*\right) = 0 \end{aligned} \tag{12}$$

where $\nabla J^*\left(\chi\right)$ is its gradient with respect to the error state.

By applying the optimality condition $\partial H/\partial u = 0$, the corresponding control law takes the form

$$u^*\left(t\right) = -\frac{1}{2} g^T\left(\chi\right) \nabla J^*\left(\chi\right) \tag{13}$$

Thus, the solution to the control problem requires finding the optimal policy $u^*\left(t\right)$ through the gradient $\nabla J^*\left(\chi\right)$. Since $\nabla J^*\left(\chi\right)$ is generally difficult to obtain in closed form for nonlinear systems, reinforcement learning with an actor–critic structure is adopted in the next section to approximate this term online.

3 Optimal Control with Reinforcement Learning

3.1 Radial Basis Function Neural Network (RBFNN) Approximation

To facilitate the construction of the optimal tracking control law, the optimal performance function $J^*\left(\chi\right)$ can be decomposed as

$$J^*\left(\chi\right) = \beta \left\|\chi\left(t\right)\right\|^2 - \beta \left\|\chi\left(t\right)\right\|^2 + J^*\left(\chi\right) = \beta \left\|\chi\left(t\right)\right\|^2 + J^0\left(\chi\right) \tag{14}$$

where $\beta > 0$ is a design constant and $J^0\left(\chi\right) = J^*\left(\chi\right) - \beta \parallel \chi\left(t\right) \parallel^2$.

Since neural networks have the capability to approximate continuous functions with arbitrary accuracy on a compact domain, the residual term $J^0(\chi)$ can be approximated by a neural network as

$$J^0(\chi) = W^{*T} B(\chi) + \varepsilon(\chi) \tag{15}$$

where $W^* \in R^m$ is the ideal weight vector, $B(\chi) \in R^m$ acts as the basis vector, m indicates the number of nodes. The function $\varepsilon(\chi)$ represents the approximation error.

By substituting the approximation Eq. (15) into the decomposition Eq. (14), the gradient of $J^*(\chi)$ can be expressed as

$$\nabla J^*(\chi) = 2\beta\chi(t) + \frac{\partial^{\mathrm{T}} B(\chi)}{\partial\chi} W^* + \frac{\partial\varepsilon(\chi)}{\partial\chi} \tag{16}$$

According to the optimal control policy derived in Eq. (13), the optimal input can be rewritten as

$$u^*(t) = -\beta g^T(\chi)\chi(t) - \frac{1}{2} g^T(\chi) \frac{\partial^{\mathrm{T}} B(\chi)}{\partial\chi} W^* - \frac{1}{2} g^T(\chi) \frac{\partial\varepsilon(\chi)}{\partial\chi} \tag{17}$$

By substituting Eqs. (16) and (17) into Eq. (12), the following result can be obtained:

$$\begin{aligned} &H(\chi, u^*, \nabla J^*) = -\left(\beta^2 - 1\right)\chi^T(t) Q(\chi)\chi(t) + 2\beta\chi^T(t) f(\chi) \\ &+ W^{*T} \frac{\partial B(\chi)}{\partial\chi^T} \left(f(\chi) - \beta Q(\chi)\chi(t)\right) - \frac{1}{4} W^{*T} \Theta(\chi) W^* + \omega(t) = 0 \end{aligned} \tag{18}$$

where

$$\Theta(\chi) = \frac{\partial B(\chi)}{\partial\chi^{\mathrm{T}}} Q(\chi) \frac{\partial^T B(\chi)}{\partial\chi} \in R^{m\times m}$$

$$\omega(t) = \frac{\partial\varepsilon(\chi)}{\partial\chi^T} \left(f(\chi) + g(\chi) u^* + \frac{1}{4} Q \frac{\partial\varepsilon(\chi)}{\partial\chi}\right)$$

Remark 2: *The number of radial basis function (RBF) nodes has a direct impact on both the approximation capability and the closed-loop performance. In this work, Gaussian RBFNNs are adopted for both the critic and actor, and the centers are uniformly distributed over the estimated reachable region of the state vector. Preliminary simulations with different node numbers* (e.g., *25, 36, and 49) indicate that 36 nodes provide a good trade-off between approximation accuracy and online computational complexity; therefore, this configuration is used in the final simulations.*

From the viewpoint of function approximation, too few basis functions lead to insufficient representation power and large residual errors, which may slow down learning and degrade tracking performance. On the other hand, using too many RBF nodes increases the network dimension and may cause overfitting and high-variance responses. In our simulations, when the number of nodes is excessively large, the learned control policy tends to exhibit sharper variations and high-frequency oscillations, resulting in "spiky" tracking trajectories and more aggressive control actions. A similar trade-off between approximation accuracy and model complexity has been reported in adaptive neural network control of robot manipulators, where an optimal number of hidden nodes is sought to avoid both under-fitting and over-fitting of the unknown dynamics [21].

3.2 Actor–Critic Implementation

In practice, the ideal weight vector W^* and the approximation error $\varepsilon(\chi)$ are unknown. To address this, an actor–critic reinforcement learning structure is employed.

The critic network approximates the performance index as

$$\nabla \hat{J}^*(\chi) = 2\beta\chi(t) + \frac{\partial^{\mathrm{T}} B(\chi)}{\partial \chi} \hat{W}_c^* \tag{19}$$

where $\nabla \hat{J}^*(\chi)$ denotes the estimated gradient of the optimal performance index $J^*(\chi)$, and $\hat{W}_c^*(t)$ is the critic weight vector adapted online.

Accordingly, the actor generates the control input using both the critic information and its own adaptive weights:

$$u(t) = -\beta g^T(\chi)\chi(t) - \frac{1}{2} g^T(\chi) \frac{\partial^{\mathrm{T}} B(\chi)}{\partial \chi} \hat{W}_a^*(t) \tag{20}$$

where $\hat{W}_a(t)$ is the actor weight vector updated during the learning process.

Therefore, the actor–critic pair collaborates such that the critic approximates the gradient $\nabla J^*(\chi)$, while the actor produces the near-optimal control input $u(t)$ in real time.

The Bellman residual is defined in terms of the Hamiltonian as

$$\phi(t) = H\left(\chi, u, \nabla \hat{J}^*\right) - H\left(\chi, u^*, \nabla J^*\right) \tag{21}$$

since the Hamiltonian at the optimal policy u^* satisfies the HJB condition, the above reduces to

$$\phi(t) = H\left(\chi, u, \nabla \hat{J}^*\right) \tag{22}$$

expanding Eq. (22), the Bellman residual can be written explicitly as

$$\begin{aligned} \phi(t) &= \chi^T Q \chi + \left\| \beta g^T(\chi)\chi(t) + \frac{1}{2} g^T(\chi) \frac{\partial B(\chi)}{\partial \chi} \hat{W}_a(t) \right\|^2 \\ &+ \left(2\beta\chi(t) + \frac{\partial B(\chi)}{\partial \chi} \hat{W}_c^T(t)\right)^T \left(f(\chi) - \beta Q\chi(t) - \frac{1}{2} Q \frac{\partial B(\chi)}{\partial \chi} \hat{W}_a(t)\right) \end{aligned} \tag{23}$$

to minimize the Bellman residual error $\phi(t)$, the positive function

$$\Phi(t) = \phi^2(t) \tag{24}$$

is introduced. For stability, it is required that $\dot{\Phi} \le 0$.

The critic weight updating rule is obtained by applying a normalized negative gradient descent:

$$\begin{aligned} \dot{\hat{W}}_c(t) &= -\frac{\alpha_c \varsigma(t)}{1 + \|\varsigma(t)\|^2} \frac{\partial \Phi(t)}{\partial \hat{W}_c(t)} \\ &= -\frac{\alpha_c \varsigma(t)}{1 + \|\varsigma(t)\|^2} \left(\varsigma^T(t)\hat{W}_c(t) - \left(\beta^2 - 1\right)\chi^T Q\chi + 2\beta\chi^T f(\chi) + \frac{1}{4}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\right) \end{aligned} \tag{25}$$

where $\alpha_c > 0$ is the critic learning rate, and

$$\varsigma(t) = \frac{\partial B(z)}{\partial \chi^T} \left(f(\chi) - \beta Q(\chi)\chi(t) - \frac{1}{2} Q(\chi) \frac{\partial^{\mathrm{T}} B(\chi)}{\partial \chi} \hat{W}_a(t)\right) \in R^{m \times 1}$$

The actor weight adaptation rule is derived from stability analysis to ensure that the optimized control input remains stable. The actor weights are updated according to

$$\dot{\hat{W}}_a(t) = \frac{1}{2}\frac{\partial B(\chi)}{\partial \chi^T} Q\chi(t) - \alpha\Theta(\chi)\hat{W}_a(t) + \frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\Theta(\chi)\hat{W}_a(t)\varsigma^T(t)\hat{W}_c(t), \tag{26}$$

where $\alpha > 0$ acts as the gain parameter used in the actor update process.

Assumption 2: *In this work, we adopt the standard concept of persistence of excitation (PE) as a requirement for ensuring sufficient richness in the regressor signal. Let $\varsigma(t)$ denote a regressor signal used in the critic and actor update laws. We assume that there exist positive constants $\xi_1 > 0$, $\xi_2 > 0$ and $T > 0$ such that, for all t, the quantity $\varsigma(t)\varsigma^T(t)$ computed over $[t, t+T]$ fulfills*

$$\xi_1 I_m \le \varsigma(t)\varsigma^T(t) \le \xi_2 I_m$$

where $I_m \in R^{m\times m}$ is the identity matrix. In other words, the PE condition requires that the regressor $\varsigma(t)$ be sufficiently rich in every time window of length T, so that all directions in the parameter space are repeatedly excited and the critic–actor weights can, in principle, be identified as in the classical actor–critic analysis of Vamvoudakis and Lewis [22].

4 Stability Analysis

In this section, the stability of the closed-loop system is analyzed using a Lyapunov-based method. Consider the following Lyapunov candidate function:

$$V(t) = \frac{1}{2}\chi^T(t)\chi(t) + \frac{1}{2}\tilde{W}_a^T(t)\tilde{W}_a + \frac{1}{2}\tilde{W}_c^T(t)\tilde{W}_c \tag{27}$$

where $\tilde{W}_a(t) = \hat{W}_a(t) - W_a^*$ and $\tilde{W}_c(t) = \hat{W}_c(t) - W_c^*$ denote the estimation errors of the actor and critic weight vectors, respectively.

The time derivative of V is

$$\begin{aligned}\dot{V}(t) &= \chi^T(t)\dot{\chi}(t) + \tilde{W}_a^T(t)\dot{\hat{W}}_a + \tilde{W}_c^T(t)\dot{\hat{W}}_c \\ &= \chi^T(t)\left(f(\chi) + g(\chi)u^*(t)\right) + \tilde{W}_a^T(t)\dot{\hat{W}}_a + \tilde{W}_c^T(t)\dot{\hat{W}}_c\end{aligned} \tag{28}$$

Applying Eqs. (20), (25) and (26) to Eq. (28), the following equation can be formulated as:

$$\begin{aligned}\dot{V}(t) = &-\beta\chi^T(t)Q(\chi)\chi(t) - \frac{1}{2}\chi^T(t)Q(\chi)\frac{\partial^{\mathrm{T}}B(\chi)}{\partial\chi}\hat{W}_a(t) + \chi^T(t)f(\chi) + \frac{1}{2}\tilde{W}_a^T(t)\frac{\partial B(\chi)}{\partial\chi^T}Q(\chi)\chi(t) \\ &- \alpha\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t) + \frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\varsigma^T(t)\hat{W}_c(t) \\ &- \tilde{W}_c^T(t)\left(\frac{\alpha_c\varsigma(t)}{1+\|\varsigma(t)\|^2}\left(\varsigma^T(t)\hat{W}_c(t) - \left(\beta^2-1\right)\chi^T(t)Q(\chi)\chi(t) + 2\beta\chi^T(t)f(\chi)\right.\right. \\ &\left.\left. + \frac{1}{4}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\right)\right)\end{aligned} \tag{29}$$

Based on $\tilde{W}_c(t) = \hat{W}_c(t) - W^*$, we obtain the expressions below:

$$\frac{1}{2}\tilde{W}_a^T(t)\frac{\partial B(\chi)}{\partial\chi^T}Q(\chi)\chi(t) - \frac{1}{2}\chi^T(t)Q(\chi)\frac{\partial^T B(\chi)}{\partial\chi}\hat{W}_a(t) = -\frac{1}{2}\chi^T(t)Q(\chi)\frac{\partial^T B(\chi)}{\partial\chi}W^* \tag{30}$$

$$-\alpha \tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t) = -\frac{\alpha}{2}\tilde{W}_a^T(t)\Theta(\chi)\tilde{W}_a(t) - \frac{\alpha}{2}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t) + \frac{\alpha}{2}W^{*T}\Theta(\chi)W^* \tag{31}$$

According to Young's and Cauchy inequalities [10], we obtain the results below:

$$\chi^{\mathrm{T}}(t)f(\chi) \le \frac{1}{2}\|\chi(t)\|^2 + \frac{1}{2}\|f(\chi)\|^2 \tag{32}$$

$$-\frac{1}{2}\chi^T(t)Q(\chi)\frac{\partial^T B(\chi)}{\partial \chi}W^* \le \frac{1}{4}\chi^T(t)Q(\chi)\chi(t) + \frac{1}{4}W^{*T}\Theta(\chi)W^* \tag{33}$$

Substituting into (29), the following equation follows:

$$\begin{aligned}\dot{V}(t) \le & -\chi^T(t)\left(\left(\beta - \frac{1}{4}\right)Q(\chi) - \frac{1}{2}I_n\right)\chi(t) + \frac{1}{2}\|f(\chi)\|^2 \\ & - \frac{\alpha}{2}\tilde{W}_a^T(t)\Theta(\chi)\tilde{W}_a(t) - \frac{\alpha}{2}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t) + \frac{2\alpha+1}{4}W^{*T}\Theta(\chi)W^* \\ & + \frac{\alpha_c}{4(1+\|\varsigma(t)\|^2)}\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\varsigma^T(t)\hat{W}_c(t) \\ & - \tilde{W}_c^T(t)\left(\frac{\alpha_c\varsigma(t)}{1+\|\varsigma(t)\|^2}\left(\varsigma^T(t)\hat{W}_c(t) - (\beta^2-1)\chi^T(t)Q(\chi)\chi(t)\right.\right. \\ & \left.\left. + 2\beta\chi^T(t)f(\chi) + \frac{1}{4}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\right)\right).\end{aligned} \tag{34}$$

Based on (18), we obtain the equation shown below:

$$\begin{aligned}& -\left(\beta^2-1\right)\chi^T(t)Q(\chi)\chi(t) + 2\beta\chi^T(t)f(\chi) \\ & = -W^{*T}\varsigma(t) - \frac{1}{2}W^{*T}\Theta(\chi)\hat{W}_a(t) + \frac{1}{4}W^{*T}\Theta(\chi)W^* - \omega(t)\end{aligned} \tag{35}$$

By inserting (35) into (34), the following result is obtained:

$$\begin{aligned}\dot{V}(t) \le & -\chi^T(t)\left(\left(\beta - \frac{1}{4}\right)Q(\chi) - \frac{1}{2}I_n\right]\chi(t) + \frac{1}{2}\|f(\chi)\|^2 \\ & - \frac{\alpha}{2}\tilde{W}_a^T(t)\Theta(\chi)\tilde{W}_a(t) - \frac{\alpha}{2}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t) + \frac{2\alpha+1}{4}W^{*T}\Theta(\chi)W^* \\ & + \frac{\alpha_c}{4(1+\|\varsigma(t)\|^2)}\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\varsigma^T(t)\hat{W}_c(t) \\ & - \tilde{W}_c^T(t)\left[\frac{\alpha_c\varsigma(t)}{1+\|\varsigma(t)\|^2}\left(\varsigma^T(t)\hat{W}_c(t) - \frac{1}{2}W^{*T}\Theta(\chi)\hat{W}_a(t)\right.\right. \\ & \left.\left. + \frac{1}{4}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t) + \frac{1}{4}W^{*T}\Theta(\chi)W^* - \omega(t)\right)\right).\end{aligned} \tag{36}$$

Using the fact that:

$$\frac{1}{4}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)-\frac{1}{2}W^*\Theta(\chi)\hat{W}_a(t)+\frac{1}{4}W^*\Theta(\chi)W^* = \frac{1}{4}\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)-\frac{1}{4}W^{*T}\Theta(\chi)\tilde{W}_a(t) \tag{37}$$

Eq. (36) is obtained:

$$\begin{aligned}\dot{V}(t)\leq&-\chi^T(t)\left(\left(\beta-\frac{1}{4}\right)Q(\chi)-\frac{1}{2}I_n\right)\chi(t)+\frac{1}{2}\|f(\chi)\|^2-\frac{\alpha}{2}\tilde{W}_a^T(t)\Theta(\chi)\tilde{W}_a(t)\\&-\frac{\alpha}{2}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)+\frac{2\alpha+1}{4}W^{*T}(t)\Theta(\chi)W^*-\frac{\alpha_c}{1+\|\varsigma(t)\|^2}\tilde{W}_c^T(t)\varsigma(t)\varsigma^T(t)\tilde{W}_c(t)\\&+\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\varsigma^T(t)\hat{W}_c(t)\\&-\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_c^T(t)\varsigma(t)\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\\&+\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_c^T(t)\varsigma(t)W^{*T}\Theta(\chi)\tilde{W}_a(t)+\frac{\alpha_c}{1+\|\varsigma(t)\|^2}\tilde{W}_c^T(t)\varsigma(t)\omega(t)\end{aligned} \tag{38}$$

Substituting the facts that

$$\begin{aligned}&\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\varsigma^T(t)\hat{W}_c(t)-\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_c^T(t)\varsigma(t)\tilde{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)\\&=\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_a^T(t)\frac{\partial B(\chi)}{\partial\chi^T}g(\chi)W^{*T}\varsigma(t)g^T(\chi)\frac{\partial^T B(\chi)}{\partial\chi}\hat{W}_a(t)\end{aligned} \tag{39}$$

$$\begin{aligned}&\frac{\alpha_c}{1+\|\varsigma(t)\|^2}\tilde{W}_c^T(t)\varsigma(t)\omega(t)\\&\leq\frac{\alpha_c}{2\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_c^T(t)\varsigma(t)\varsigma^T(t)\tilde{W}_c(t)+\frac{\alpha_c}{2\left(1+\|\varsigma(t)\|^2\right)}\omega^2(t)\end{aligned} \tag{40}$$

Substituting Eqs. (39) and (40) into Eq. (38) yields:

$$\begin{aligned}\dot{V}(t)\leq&-\chi^T(t)\left(\left(\beta-\frac{1}{4}\right)Q(\chi)-\frac{1}{2}I_n\right)\chi(t)-\frac{\alpha}{2}\tilde{W}_a^T(t)\Theta(\chi)\tilde{W}_a(t)\\&-\frac{\alpha_c}{2\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_c^T(t)\varsigma(t)\varsigma^T(t)\tilde{W}_c(t)\\&+\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_a^T(t)\frac{\partial B(\chi)}{\partial\chi^T}g(\chi)W^{*T}\varsigma(t)g^T(\chi)\frac{\partial^T B(\chi)}{\partial\chi}\hat{W}_a(t)\\&+\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^2\right)}\tilde{W}_c^T(t)\varsigma(t)W^{*T}\Theta(\chi)\tilde{W}_a(t)\\&-\frac{\alpha}{2}\hat{W}_a^T(t)\Theta(\chi)\hat{W}_a(t)+C(t)\end{aligned} \tag{41}$$

where

$$C(t)=\left(\frac{2\alpha+1}{4}W^{*T}\Theta(\chi)W^{*}\right)+\frac{1}{2}\|f(\chi)\|^{2}+\frac{a_c}{2\left(1+\|\varsigma(t)\|^{2}\right)}\omega^{2}(t)$$

Using Young's and Cauchy inequalities, we obtain the bounds shown below:

$$\begin{aligned}&\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^{2}\right)}\tilde{W}_a^{T}(t)\frac{\partial B(\chi)}{\partial\chi^{T}}g(\chi)W^{*T}\varsigma(t)g^{T}(\chi)\frac{\partial^{T}B(\chi)}{\partial\chi}\hat{W}_a(t)\\&\le\frac{1}{32}\tilde{W}_a^{T}(t)\frac{\partial B(\chi)}{\partial\chi^{T}}g(\chi)W^{*T}\varsigma(t)\varsigma^{T}(t)W^{*}g^{T}(\chi)\frac{\partial^{T}B(\chi)}{\partial\chi}\tilde{W}_a(t)\\&+\frac{\alpha_c^{2}}{2}\hat{W}_a^{T}(t)\Theta(\chi)\hat{W}_a(t)\end{aligned}\tag{42}$$

$$\begin{aligned}&\frac{\alpha_c}{4\left(1+\|\varsigma(t)\|^{2}\right)}\tilde{W}_c^{T}(t)\varsigma(t)W^{*T}\Theta(\chi)\tilde{W}_a(t)\\&\le\frac{1}{32\left(1+\|\varsigma(t)\|^{2}\right)}\tilde{W}_c^{T}(t)\varsigma(t)W^{*T}\Theta(\chi)W^{*}\varsigma^{T}(t)\tilde{W}_c(t)\\&+\frac{\alpha_c^{2}}{2}\tilde{W}_a^{T}(t)\Theta(\chi)\tilde{W}_a(t)\end{aligned}\tag{43}$$

By substituting the inequalities in Eqs. (42) and (43) into Eq. (41), we obtain:

$$\begin{aligned}&\dot{V}(t)\le\chi^{T}(t)(\beta-\frac{1}{2})Q-I_n)\chi(t)\\&-\left(\frac{\alpha}{2}-\frac{\alpha_c^{2}}{2}-\frac{1}{32}W^{*T}\varsigma(t)\varsigma^{T}(t)W^{*}\right)\tilde{W}_a^{T}(t)\Theta(\chi)\tilde{W}_a(t)\\&-\frac{1}{1+\|\varsigma(t)\|^{2}}\left(\frac{\alpha_c}{2}-\frac{1}{32}W^{*T}\Theta(\chi)W^{*}\right)\tilde{W}_c^{T}(t)\varsigma(t)\varsigma^{T}(t)\tilde{W}_c\\&-\left(\frac{\alpha}{2}-\frac{\alpha_c^{2}}{2}\right)\tilde{W}_a^{T}(t)\Theta(\chi)\tilde{W}_a(t)+C(t)\end{aligned}\tag{44}$$

Let

$$c_1=\left(\beta-\frac{1}{2}\right)\lambda_Q^{\min}-1$$

$$c_2=\left(\frac{\alpha}{2}-\frac{\alpha_c^{2}}{2}-\frac{\xi_1}{32}\|W^{*}\|^{2}\right)\lambda_\Theta^{\min}$$

$$c_3=\left(\frac{\alpha_c}{2}-\frac{\lambda_\Theta^{\max}}{32}\|W^{*}\|^{2}\right)\xi_2$$

$$\gamma_1=\sup\{C(t)\}$$

where $\lambda_Q^{\min}$ denotes the smallest eigenvalue of $Q(\chi)$ and $\lambda_\Theta^{\max}$ represents the largest eigenvalue of $\Theta(\chi)$. Then Eq. (28) be expressed as

$$\dot{V}(t)\le-c_1\chi^{T}(t)\chi(t)-c_2\tilde{W}_a^{T}(t)\tilde{W}_a(t)-c_3\tilde{W}_c^{T}(t)\tilde{W}_c(t)+\gamma_1-\left(\frac{\alpha}{2}-\frac{\alpha_c^{2}}{2}\right)\hat{W}_a^{T}(t)\Theta(\chi)\hat{W}_a(t)\tag{45}$$

equivalently,

$$\dot{V}(t) \leq -c_L V(t) + \gamma_1 \tag{46}$$

where $c_L = \min\{c_1, c_2, c_3\} > 0$.

Lemma 1 [1]: *Consider a scalar function $V \in R$ that remains positive and continuous, and whose initial value $V(0)$ is bounded. Suppose $V(t)$ evolves according to $\dot{V} \leq -aV(t) + b$, where $a > 0$ and $b > 0$ are given constants.*

$$V(t) \leq e^{-at} V(0) + \frac{b}{a}\left(1 - e^{-at}\right). \tag{47}$$

By Lemma 1, Eq. (47) can be obtained:

$$V(t) \leq e^{-c_L t} V(0) + \frac{\gamma_1}{c_L}\left(1 - e^{-c_L t}\right). \tag{48}$$

Result. Inequality Eq. (32) implies that the errors $e(t)$, $\tilde{W}_c(t)$ and $\tilde{W}_a(t)$ satisfy the SGUUB property.

Finally, our proposed actor–critic trajectory controller with optimal design for nonlinear robotic systems is shown in Fig. 1.

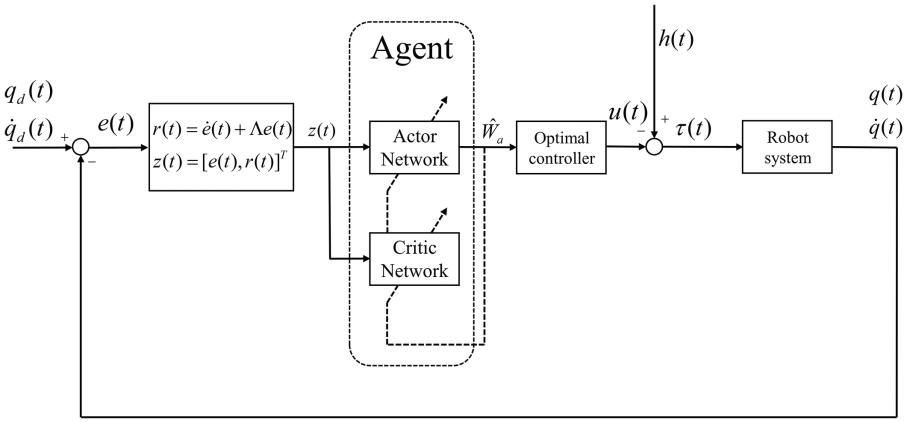


Figure 1: Our proposed actor–critic trajectory controller with optimal design for nonlinear robotic systems

5 Numerical Simulation Results

5.1 Link Manipulator

The effectiveness of the control scheme is assessed on a single-link robotic manipulator governed by Eq. (1). The system parameters follow Section 2.1 and are fixed as $J = 1$, $B = 2$, and $Mgl = 10$. The sampling period is $T = 0.02$ s, and $t_0 = 0$, $t_f = 20$ s. The initial condition is $q(0) = -5$, $\dot{q}(0) = 5$. The desired trajectory is specified by

$$q_d(t) = -\frac{200}{49}\sin(0.7t) + \frac{20}{7}t$$

For the neural network approximation, RBFNN with 36 units is adopted. The basis vector takes the form

$$B(e) = [B_1(e), B_2(e), \cdots, B_{36}(e)]^T$$

with each Gaussian basis function given by

$$B_i(e) = \exp\left(-\|e - \varsigma_i\|^2 / \mu_i^2\right), i = 1, \cdots, 36$$

where $\mu_i = 1$, and the centers ς_i are uniformly distributed over the range $[-6, 6]$.

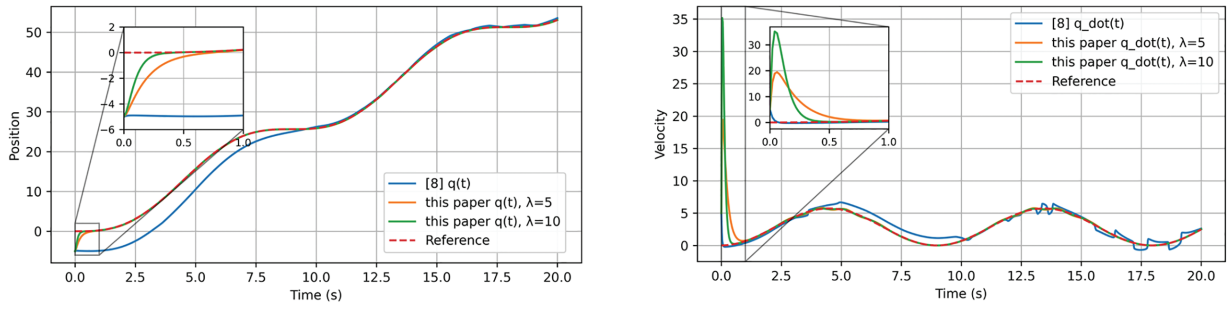
The control design parameters are selected as $\beta = 40.0$, critic learning rate $\alpha_c = 10.0$, and actor learning rate $\alpha = 5$. The initial conditions of the weights are set as

$$\hat{W}_c(0) = [0.3, \cdots, 0.3]^T$$
$$\hat{W}_a(0) = [0.5, \cdots, 0.5]^T$$

Two controllers are compared under identical conditions. The first follows the method in [8], which uses the tracking error $e(t) = q_d(t) - q(t)$ directly within the Optimal+RL framework. The second uses the method proposed in this paper, the key difference is the additional reference error defined in Eq. (3), with $\Lambda = 5$ and $\Lambda = 10$, after which the manipulator dynamics are reformulated in terms of the composite state in Eq. (6) for the actor–critic optimal-tracking design.

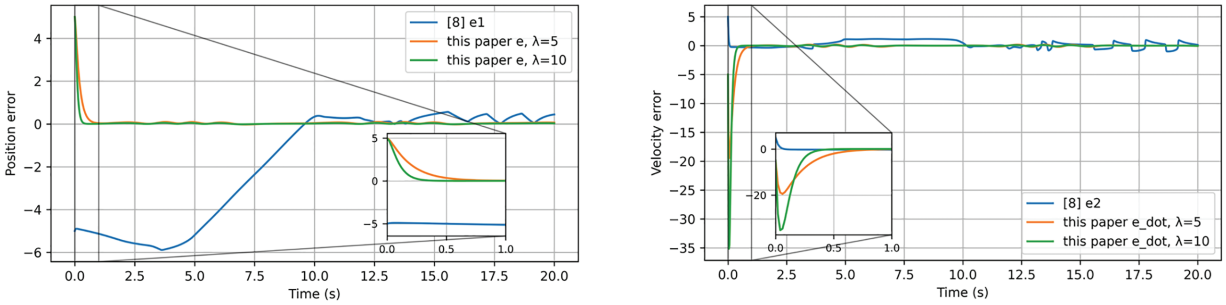
For the baseline controller based on [8], we reimplemented the algorithm under exactly the same settings as those described above for the proposed method. In particular, both controllers use the same single-link manipulator dynamics in Section 2, the same desired trajectory, the same parameters, and the same RBFNN structure (number of nodes, centers and widths) for the critic and actor. Moreover, the critic and actor learning rates and, the initial weight vectors, and the initial state of the system are chosen to be identical for the baseline and the proposed controller. Therefore, the performance differences observed in this subsection mainly originate from the structural modification introduced in this work, rather than from arbitrary retuning of the underlying parameters.

Fig. 2 presents position and velocity tracking results. The method proposed in this paper produces markedly faster transients; increasing Λ from 5 to 10 further accelerates the initial convergence without degrading steady-state tracking. Fig. 3 shows the position and velocity errors. Consistent with Fig. 1, the error trajectories of the proposed method reach zero much earlier and settle within a tighter band. Fig. 4 depicts the control torque, which remains bounded in all cases; the proposed method yields smoother steady-state actuation. Figs. 5 and 6 report the Euclidean norms of the actor and critic weights, confirming bounded adaptation. Fig. 7 plots the instantaneous cost and indicates faster cost decay for the proposed method.



(**a**) (**b**)

Figure 2: Position and velocity tracking of the single-link manipulator: (**a**) position comparison between the method in [8] and the proposed method; (**b**) velocity comparison between the method in [8] and the proposed method. All curves are generated from our own simulations based on the controller structure reported in [8]



(**a**) (**b**)

Figure 3: Tracking errors of the single-link manipulator: (**a**) position error; (**b**) velocity error

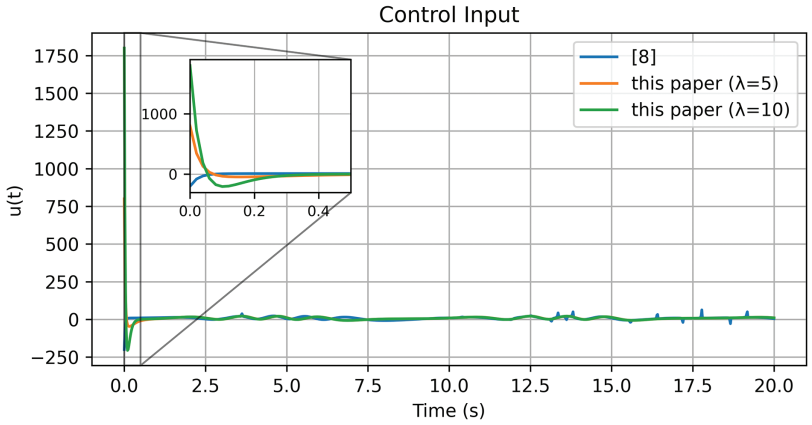


Figure 4: Control input

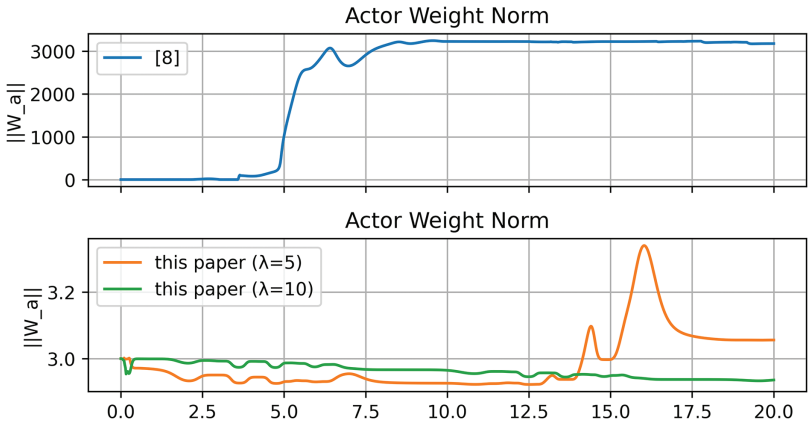


Figure 5: Norm of actor Neural Network (NN) weights

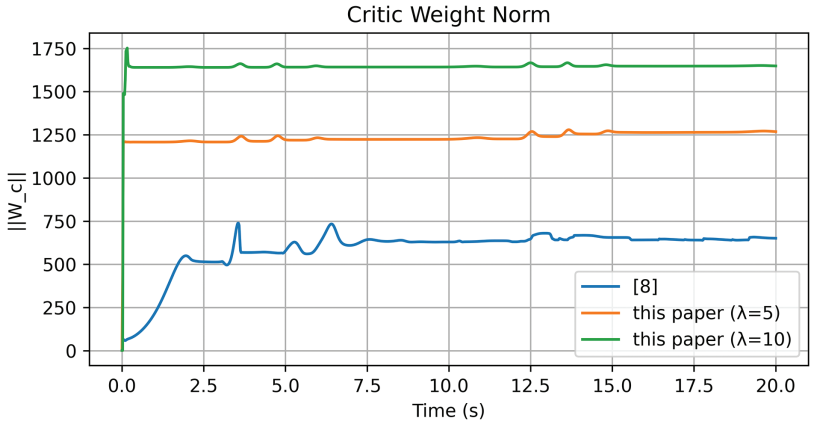


Figure 6: Norm of critic NN weights

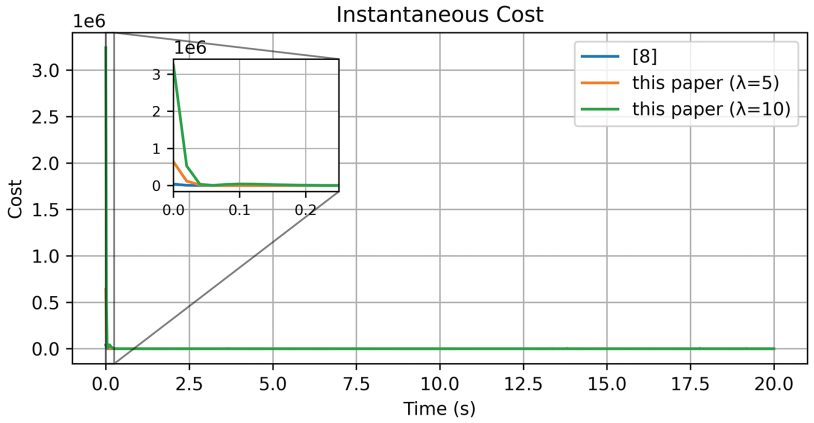


Figure 7: Cost function

These results, summarized in Tables 1 and 2, substantiate that incorporating the reference error into the Optimal+RL design substantially improves transient speed and reduces post-tracking error; moreover, Λ serves as a convergence gain that allows explicit shaping of the trajectory-tracking response.

Table 1: Settling time and steady-state error of position tracking for the single-link manipulator

Method	Settling time (s)	Steady-state error (rad)
[8]	~8.5–9.0	~0.30–0.50
This paper, Λ = 5	~0.35–0.40	~0.15–0.25
This paper, Λ = 10	~0.20–0.25	~0.05–0.10

Table 2: Settling time and steady-state error of velocity tracking for the single-link manipulator

Method	Settling time (s)	Steady-state error (rad/s)
[8]	~0.5–0.6	~0.80–1.20
This paper, Λ = 5	~0.30–0.35	~0.20–0.30

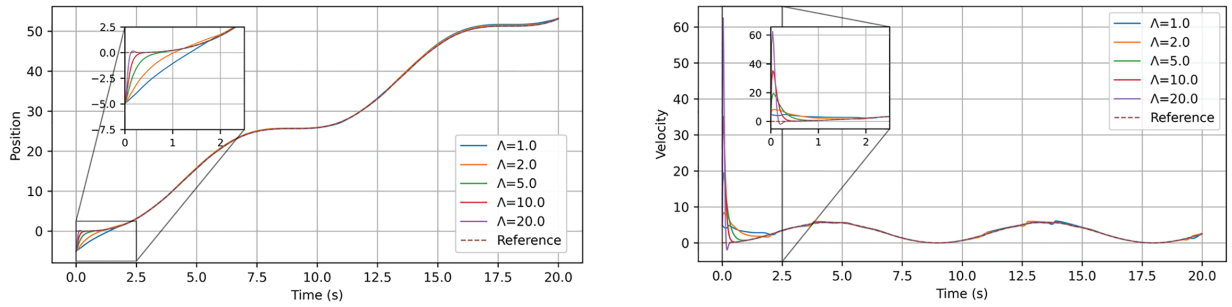
(Continued)

Table 2 (continued)

Method	Settling time (s)	Steady-state error (rad/s)
This paper, $\Lambda = 10$	~0.20–0.25	~0.05–0.10

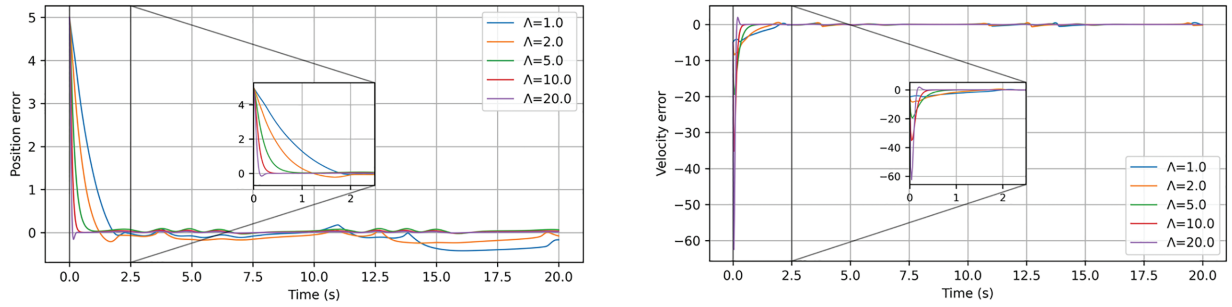
Influence of the Reference-Error Gain

To further investigate the influence of the reference-error gain Λ on the closed-loop behavior, an additional simulation study is carried out on the single-link manipulator with five different values, $\Lambda = 1, 2, 5, 10, 20$, while all other parameters are kept identical. The resulting position and velocity trajectories are shown in Fig. 8, where Fig. 8a depicts the joint position and Fig. 8b depicts the joint velocity for the five choices of Λ. The corresponding position and velocity tracking errors are plotted in Fig. 9a and b, respectively, and the associated control inputs are reported in Fig. 10.



(**a**) (**b**)

Figure 8: Position and velocity responses of the single-link manipulator under different values of the reference-error gain: (**a**) joint position; (**b**) joint velocity



(**a**) (**b**)

Figure 9: Position and velocity tracking errors of the single-link manipulator under different values of the reference-error gain: (**a**) position error; (**b**) velocity error

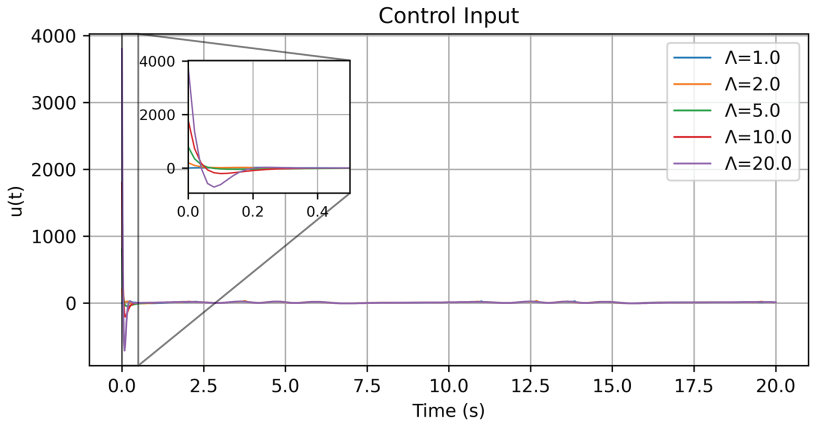


Figure 10: Control input of the single-link manipulator under different values of the reference-error gain

The cases $\Lambda = 1$ and $\Lambda = 2$ exhibit relatively slow convergence and longer settling times, although their control inputs in Fig. 10 are comparatively smooth and far from the saturation bounds. When Λ is increased to 5 and 10, the position and velocity errors decay much more rapidly, and the trajectories in Fig. 8 closely follow the desired motion with only moderate growth in the control effort. For $\Lambda = 20$, the tracking errors converge the fastest among all tested values, but the corresponding control signal in Fig. 10 becomes more aggressive and approaches the saturation limits more frequently, which may cause larger overshoot or high-frequency components in practical implementations.

These observations confirm that Λ provides a simple and effective tuning knob to balance transient speed and control effort in the proposed actor–critic controller. In the subsequent simulations, $\Lambda = 5$ and $\Lambda = 10$ are therefore adopted as representative "moderate" and "aggressive" settings, respectively: they clearly illustrate the trade-off between convergence rate and control activity, while avoiding the excessively sluggish response observed for very small Λ and the overly aggressive control behavior associated with excessively large Λ.

5.2 Nonlinear Mass–Spring–Damper

The second experiment considers a single-Degrees of Freedom (DOF) nonlinear oscillator described by

$$\ddot{q}(t) = -c\dot{q}(t) - kq(t) - k_3 q^3(t) + u(t)$$

with damping $c = 0.5$, linear stiffness $k = 2.0$, and cubic stiffness $k_3 = 0.5$. The sampling period is $T = 0.02$ s and the simulation horizon is $t_0 = 0$ and $t_f = 50$ s. The reference trajectory is chosen as

$$q_d(t) = \cos(t), \dot{q}_d(t) = -\sin(t)$$

the initial condition is $q(0) = \dot{q}(0) = 0$.

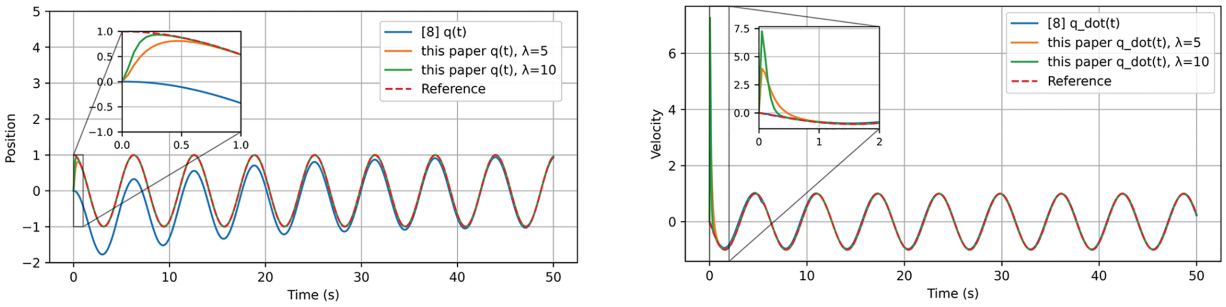
For function approximation, an RBFNN with 36 Gaussian nodes is used centers on $[-6, 6]$, the control design parameters are selected as $\beta = 40.0$, critic learning rate $\alpha_c = 5.0$, and actor learning rate $\alpha = 40$. The initial conditions of the weights are set as

$$\hat{W}_c(0) = [0.3, \cdots, 0.3]^T$$
$$\hat{W}_a(0) = [0.5, \cdots, 0.5]^T$$

Three tests are performed under identical settings: the method in [8], and the method proposed in this paper using Eq. (3) with $\Lambda = 5$ and $\Lambda = 10$. The tracking-time and post-tracking error metrics are defined as in Section 5.1.

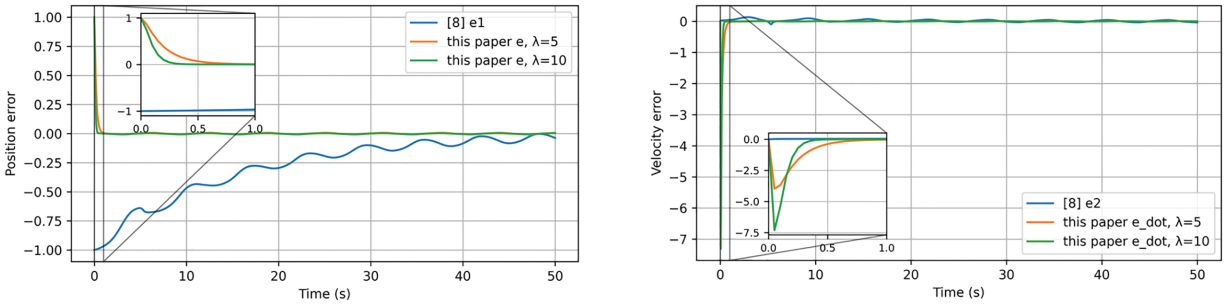
As in Section 5.1, the baseline controller from [8] in this example is also reimplemented using exactly the same controller parameters as the proposed method, so that the comparison is carried out under identical settings.

The position and velocity tracking responses of the three controllers are shown in Fig. 11, and the corresponding tracking errors are plotted in Fig. 12. The control inputs generated by the proposed method and the baseline are depicted in Fig. 13. The evolution of the actor and critic weight norms is illustrated in Figs. 14 and 15, and the associated cost function trajectories are reported in Fig. 16. The settling time and steady-state error of the position and velocity tracking are summarized in Tables 3 and 4, respectively.



(**a**) (**b**)

Figure 11: Position and velocity tracking of the nonlinear mass–spring–damper system: (**a**) position comparison between the method in [8] and the proposed method; (**b**) velocity comparison between the method in [8] and the proposed method. All curves are obtained from our own simulations based on the reference setup in [8]



(**a**) (**b**)

Figure 12: Tracking errors of the nonlinear mass–spring–damper system: (**a**) position error; (**b**) velocity error [8]

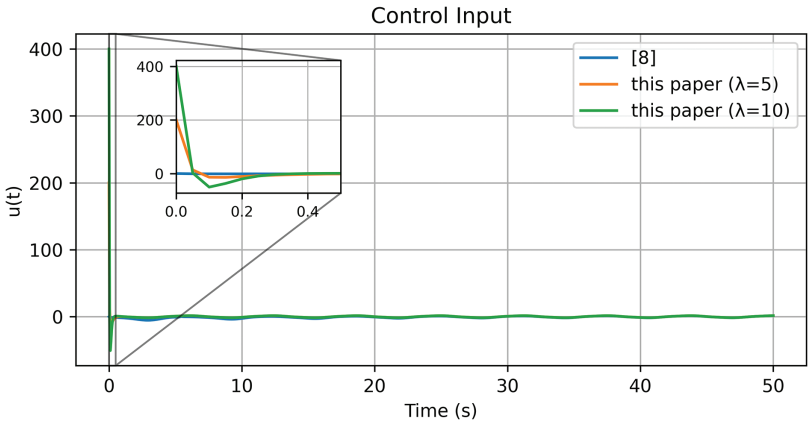


Figure 13: Control input [8]

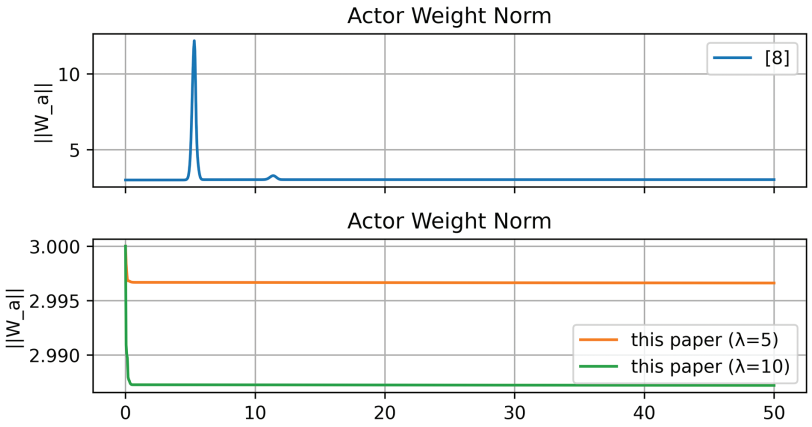


Figure 14: Norm of actor NN weights [8]

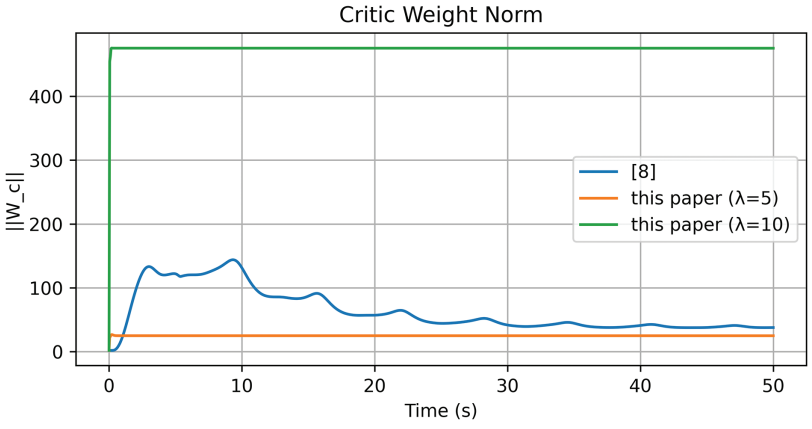


Figure 15: Norm of critic NN weights [8]

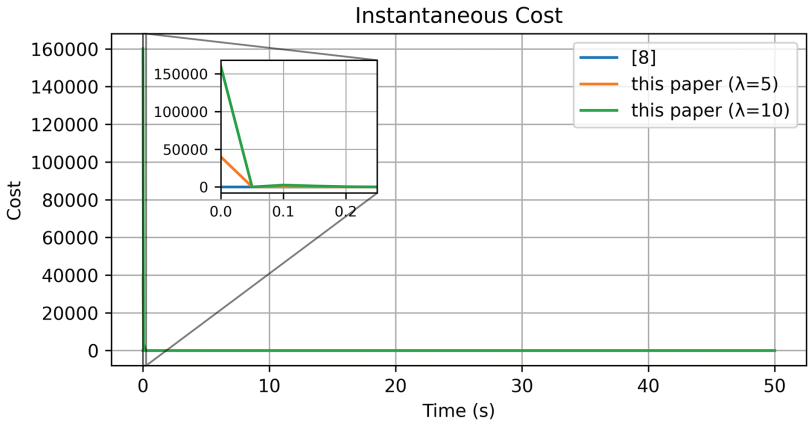


Figure 16: Cost function [8]

Table 3: Settling time and steady-state error of position tracking for the nonlinear mass–spring–damper system

Method	Settling time (s)	Steady-state error (rad)
[8]	Not reached within 50 s	~0.05–0.10
This paper, $\Lambda = 5$	~0.45–0.65	~0.02–0.04
This paper, $\Lambda = 10$	~0.18–0.25	~0.01–0.02

Table 4: Settling time and steady-state error of velocity tracking for the nonlinear mass–spring–damper system

Method	Settling time (s)	Steady-state error (rad/s)
[8]	~0.55–0.70	~0.05–0.10
This paper, $\Lambda = 5$	~0.35–0.45	~0.03–0.06
This paper, $\Lambda = 10$	~0.20–0.30	~0.02–0.04

From Figs. 11–16 and Tables 3 and 4, the method in [8] shows poor performance under the chosen nonlinear reference, whereas the proposed reference-error based design achieves faster convergence, smaller steady-state errors, and smoother control effort, as visible in the insets of the velocity and torque plots.

6 Simulation in Virtual Environment—Single-Link Manipulator

To further evaluate the proposed optimal reinforcement learning control strategy, a virtual simulation platform was constructed in the Gazebo environment, which provides physics-based visualization and real-time dynamic interaction. The manipulator model and control framework were configured following the experimental setup described in [15], where the same single-link robotic system was adopted as a benchmark for validation.

The developed Gazebo environment and manipulator model are illustrated in Fig. 17, which replicates realistic joint friction, gravitational effects, and torque constraints.

The plant dynamics follow the model of Eq. (1):

$$J\ddot{q} + B\dot{q} + Mgl\sin(q) = \tau$$

the parameters are set as $J = 1.0$, $B = 2.0$, and $Mgl = 10.0$.

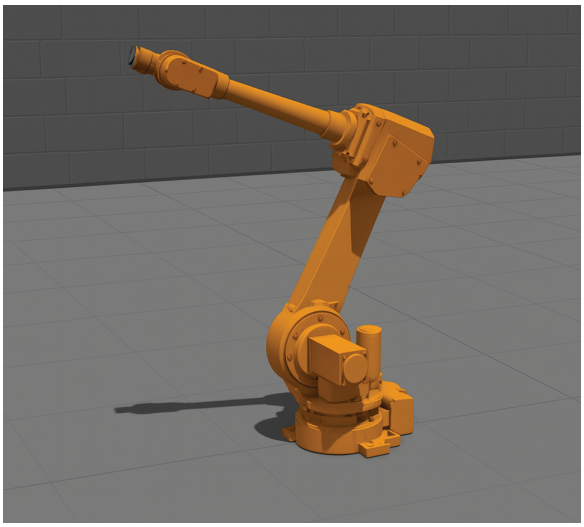


Figure 17: Gazebo simulation environment of the single-link robotic manipulator

The reference trajectory and its derivatives are defined as follows:

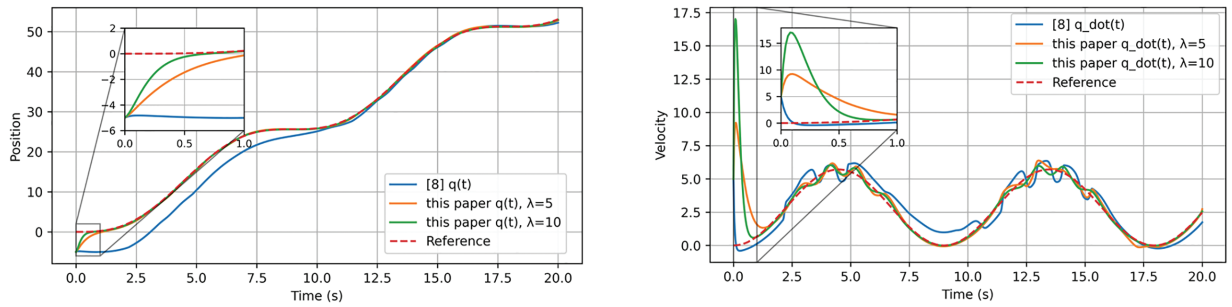
$$q_d(t) = -\frac{200}{49}\sin(0.7t) + \frac{20}{7}t$$

Simulation settings are summarized below: $\beta = 20.0$, $\mu = 1.0$, $\alpha = 5.0$, $\alpha_c = 5.0$, and 36 RBF nodes.

The time horizon is $t \in [0, 20]$ s, with 1000 uniformly spaced samples. The neural network centers are uniformly distributed within $[-6, 6]$.

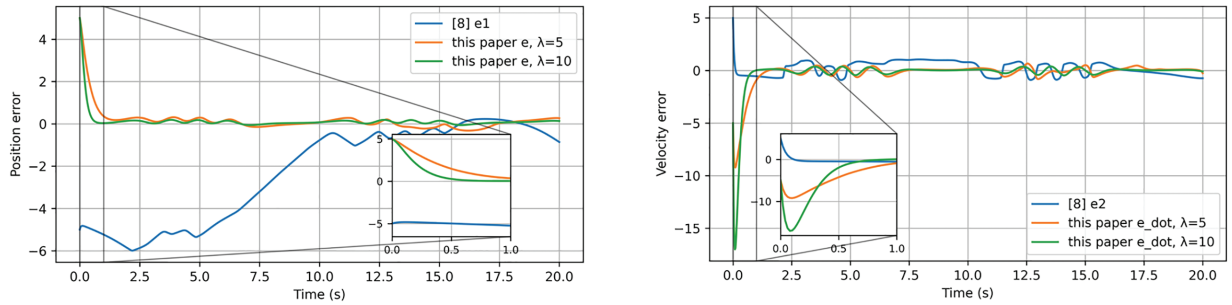
In this example, the baseline method of [8] is implemented with the same controller-related parameters as the proposed scheme, including the discount factor, learning rates, RBFNN structure, initial weight vectors and initial state, ensuring that both controllers are tested under the same conditions.

The results obtained in the Gazebo environment are presented in Figs. 18–23. The tracking response in Fig. 18a,b demonstrates that the proposed controller achieves accurate and stable motion tracking, closely following the desired trajectory under virtual dynamics. The position and velocity errors shown in Fig. 19a,b converge rapidly to zero, verifying that the controller maintains robustness and precision even with virtual actuator dynamics and friction. The control input shown in Fig. 20 remains smooth and bounded throughout the operation, indicating that the control effort required in the simulated environment is practical and stable. The evolution of actor and critic weights in Figs. 21–22 remains bounded, confirming the stability and learning consistency of the proposed actor–critic mechanism during the entire simulation.



(**a**) (**b**)

Figure 18: Position and velocity tracking of the single-link manipulator: (**a**) position comparison between the method in [8] and the proposed method; (**b**) velocity comparison between the method in [8] and the proposed method. All curves are generated from our own simulations based on the controller structure reported in [8]



(**a**) (**b**)

Figure 19: Tracking errors of the single-link manipulator: (**a**) position error; (**b**) velocity error [8]

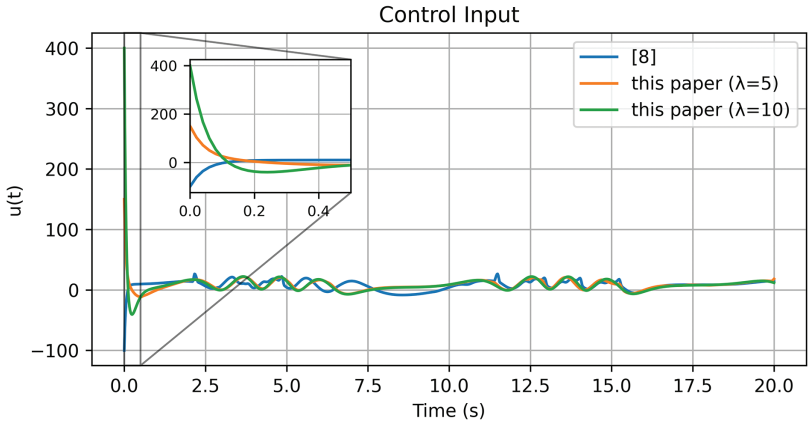


Figure 20: Control input [8]

The quantitative performance evaluation is summarized in Tables 5 and 6, showing that the proposed controller significantly improves both transient and steady-state accuracy compared with the benchmark approach [8]. The results verify that the actor–critic optimal control strategy can be effectively implemented in a physics-based virtual manipulator and maintains stable adaptation during real-time operation.

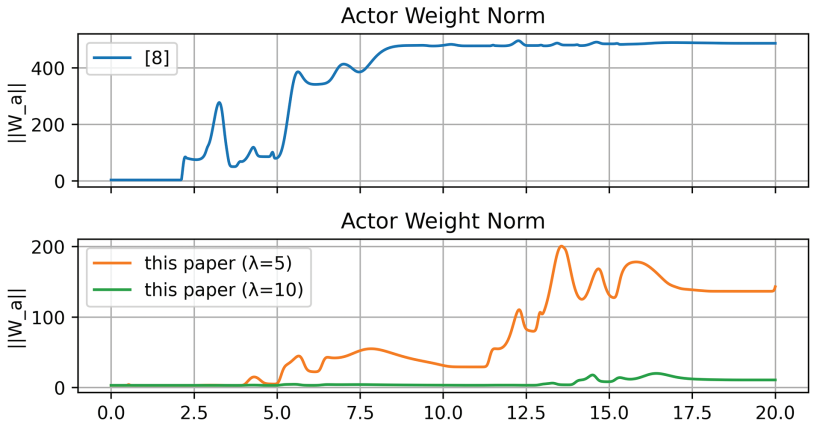


Figure 21: Norm of actor NN weights [8]

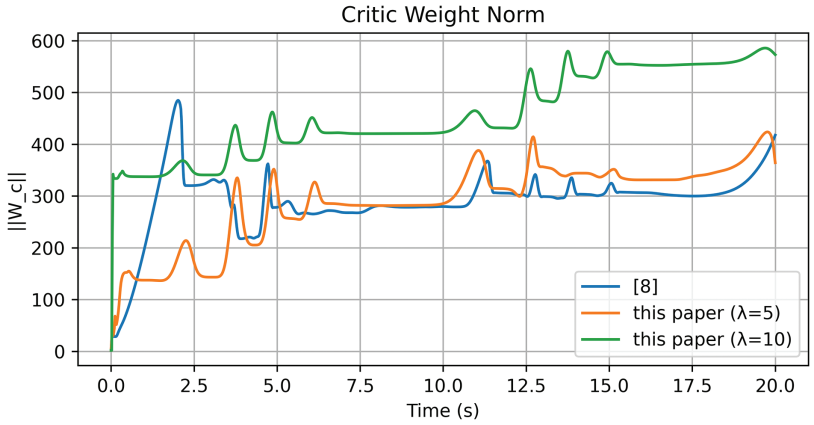


Figure 22: Norm of critic NN weights [8]

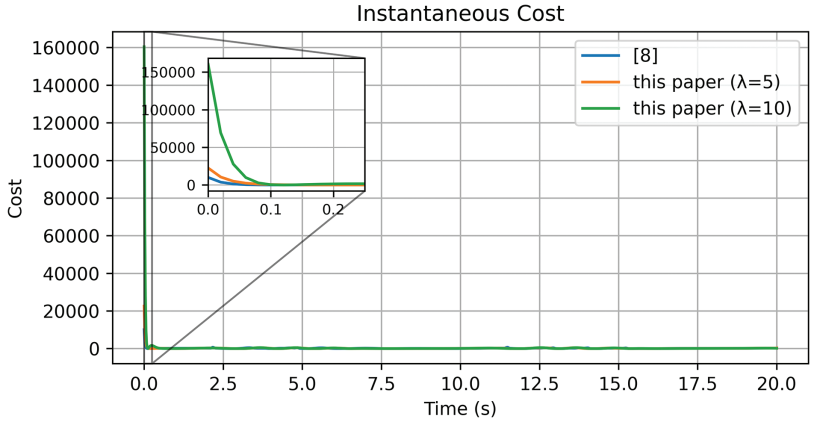


Figure 23: Cost function [8]

Table 5: Settling time and steady-state error of position tracking for the single-link manipulator

Method	Settling time (s)	Steady-state error (rad)
[8]	~10.5	~0.82
This paper, $\Lambda = 5$	~1.0	~0.18
This paper, $\Lambda = 10$	~0.6	~0.07

Table 6: Settling time and steady-state error of velocity tracking for the single-link manipulator

Method	Settling time (s)	Steady-state error (rad/s)
[8]	~10.5	~0.47
This paper, $\Lambda = 5$	~1.0	~0.06
This paper, $\Lambda = 10$	~0.6	~0.02

In summary, the Gazebo-based virtual validation confirms the effectiveness and practicality of the proposed controller. The system maintains stable adaptation, bounded control input, and high tracking precision under realistic simulation conditions. The cost function convergence further verifies that the control policy achieves near-optimal performance while ensuring smooth and efficient actuation. These findings demonstrate that the control algorithm can be reliably extended from numerical experiments to physical robotic platforms with consistent performance.

Quantitatively, the proposed method reduces settling time by approximately 90% and steady-state error by 85% compared to [8], confirming the improved real-time performance.

7 Conclusion

This paper presented an actor–critic optimal tracking framework for nonlinear systems that augments the classical tracking error with a reference error. By reformulating the plant dynamics in the composite error state, the proposed method enables explicit shaping of the convergence rate through the gain Λ while preserving a control-affine structure amenable to reinforcement learning. A Lyapunov-based design was adopted to derive the update laws and to ensure closed-loop boundedness of the tracking signals and adaptive weights.

Simulation studies on two benchmarks—a single-link robotic manipulator and a nonlinear mass–spring–damper—demonstrated clear performance gains over the reference approach in [8]. With identical neural approximators and learning gains, the proposed controller achieved substantially faster transients and smaller steady-state errors. On the manipulator, sub-second zero-crossing of the error was obtained for $\Lambda = 5$ and $\Lambda = 10$, whereas ref. [8] exhibited a much slower approach. On the oscillator, the proposed method tracked the periodic reference within fractions of a second and maintained a tighter residual band. These improvements were accompanied by bounded control inputs and weight norms, confirming stable adaptation. The trade-off is that larger Λ values yield sharper initial responses and higher early control demand; nevertheless, they provide a direct and tunable mechanism to accelerate convergence.

Overall, the results verify that incorporating the reference error into the Optimal+RL design offers a simple yet effective route to enhance tracking accuracy and convergence speed, without sacrificing stability guarantees. Future work will extend the method to multi-DOF manipulators and implement it on

a physical UR5 robot using Robot Operating System (ROS) 2 to evaluate robustness under sensor noise and actuator saturation.

Acknowledgement: Not applicable.

Funding Statement: This work was supported in part by the National Science and Technology Council under Grant NSTC 114-2221-E-027-104.

Author Contributions: Nien-Tsu Hu: Conceptualization, Formal analysis, Funding acquisition, Methodology, Project administration, Resources, Supervision, Validation, Writing—review and editing. Hsiang-Tung Kao: Data curation, Formal analysis, Investigation, Software, Validation, Writing—original draft, Writing—review and editing. Chin-Sheng Chen and Shih-Hao Chang: Supervision, Visualization. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets and source codes generated and analyzed during the current study are not publicly available at this time but are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Ge SS, Hang CC, Zhang T. Adaptive neural network control of nonlinear systems by state and output feedback. IEEE Trans Syst, Man, Cybern B. 1999;29(6):818–28. doi:10.1109/3477.809035.
2. Beard RW, Saridis GN, Wen JT. Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation. Automatica. 1997;33(12):2159–77. doi:10.1016/S0005-1098(97)00128-3.
3. Khalil HK. Nonlinear systems. 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall; 2002. 750 p.
4. Lewis FL, Vrabie DL, Syrmos VL. Optimal control. 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2012. 540 p.
5. Peng Z, Cheng H, Shi K, Zou C, Huang R, Li X, et al. Optimal tracking control for motion constrained robot systems via event-sampled critic learning. Expert Syst Appl. 2023;234:121085. doi:10.1016/j.eswa.2023.121085.
6. Cheng G, Dong L. Optimal control for robotic manipulators with input saturation using single critic network. In: Proceedings of the 2019 Chinese Automation Congress (CAC); 2019 Nov 22–24; Hangzhou, China. p. 2344–9. doi:10.1109/cac48633.2019.8996999.
7. Hu Y, Cui L, Chai S. Optimal tracking control for robotic manipulator using actor-critic network. In: Proceedings of the 2021 40th Chinese Control Conference (CCC); 2021 Jul 26–28; Shanghai, China. p. 1556–61.
8. Wen G, Philip Chen CL, Ge SS, Yang H, Liu X. Optimized adaptive nonlinear tracking control using actor–critic reinforcement learning strategy. IEEE Trans Ind Inf. 2019;15(9):4969–77. doi:10.1109/tii.2019.2894282.
9. Wen G, Ge SS, Tu F. Optimized backstepping for tracking control of strict-feedback systems. IEEE Trans Neural Netw Learn Syst. 2018;29(8):3850–62. doi:10.1109/TNNLS.2018.2803726.
10. Wen G, Philip Chen CL, Feng J, Zhou N. Optimized multi-agent formation control based on an identifier–actor–critic reinforcement learning algorithm. IEEE Trans Fuzzy Syst. 2018;26(5):2719–31. doi:10.1109/tfuzz.2017.2787561.
11. Kim YH, Lewis FL, Dawson DM. Intelligent optimal control of robotic manipulators using neural networks. Automatica. 2000;36(9):1355–64. doi:10.1016/S0005-1098(00)00045-5.
12. Bu X. Prescribed performance control approaches, applications and challenges: a comprehensive survey. Asian J Control. 2023;25(1):241–61. doi:10.1002/asjc.2765.
13. Ghanooni P, Habibi H, Yazdani A, Wang H, MahmoudZadeh S, Ferrara A. Prescribed performance control of a robotic manipulator with unknown control gain and assigned settling time. ISA Trans. 2024;145(2):330–54. doi:10.1016/j.isatra.2023.12.011.

14. Zhao K, Xie Y, Xu S, Zhang L. Adaptive neural appointed-time prescribed performance control for the manipulator system via barrier Lyapunov function. J Frankl Inst. 2025;362(2):107468. doi:10.1016/j.jfranklin.2024.107468.
15. Rahimi F, Ziaei S, Esfanjani RM. A reinforcement learning-based control approach for tracking problem of a class of nonlinear systems: applied to a single-link manipulator. In: Proceedings of the 2023 31st International Conference on Electrical Engineering (ICEE); 2023 May 9–11; Tehran, Iran. p. 58–63. doi:10.1109/icee59167.2023.10334874.
16. Rahimi Nohooji H, Zaraki A, Voos H. Actor–critic learning based PID control for robotic manipulators. Appl Soft Comput. 2024;151:111153. doi:10.1016/j.asoc.2023.111153.
17. Zhao X, Han S, Tao B, Yin Z, Ding H. Model-based actor–critic learning of robotic impedance control in complex interactive environment. IEEE Trans Ind Electron. 2022;69(12):13225–35. doi:10.1109/tie.2021.3134082.
18. Nam D, Huy D. H∞ tracking control for perturbed discrete-time systems using On/Off policy Q-learning algorithms. Chaos Solitons Fractals. 2025;197:116459. doi:10.1016/j.chaos.2025.116459.
19. Dao PN, Nguyen VQ, Duc HAN. Nonlinear RISE based integral reinforcement learning algorithms for perturbed Bilateral Teleoperators with variable time delay. Neurocomputing. 2024;605:128355. doi:10.1016/j.neucom.2024.128355.
20. Zhao JG, Chen FF. Off-policy integral reinforcement learning-based optimal tracking control for a class of nonzero-sum game systems with unknown dynamics. Optim Control Appl Meth. 2022;43(6):1623–44. doi:10.1002/oca.2916.
21. Liu C, Zhao Z, Wen G. Adaptive neural network control with optimal number of hidden nodes for trajectory tracking of robot manipulators. Neurocomputing. 2019;350:136–45. doi:10.1016/j.neucom.2019.03.043.
22. Vamvoudakis KG, Lewis FL. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. Automatica. 2010;46(5):878–88. doi:10.1016/j.automatica.2010.02.018.