Online Constrained Reinforcement Learning for Optimal Tracking ⋆

Hyochan Lee * **Kyunghwan Choi** *

* *CCS Graduate School of Mobility, Korea Advanced Institute of Science and Technology, Daejeon, 34051, Republic of Korea*
e-mail: hyochanlee@kaist.ac.kr, kh.choi@kaist.ac.kr

Abstract: This paper presents a constrained online reinforcement learning framework for the optimal tracking control of constrained nonlinear systems. While reinforcement learning provides powerful tools for optimal control, conventional implementations typically rely on unconstrained minimization strategies. Since this approach does not restrict the policy search space within the feasible region, it often drives the control policy toward unbounded actions, exacerbating the instability inherent in nonlinear function approximation. To address these issues, the proposed method reformulates the Bellman optimality equation as a constrained optimization problem where the control policy and value function are treated as joint decision variables. Crucially, this formulation allows for the explicit incorporation of system constraints directly into the learning process. A Lagrangian-based primal-dual scheme is then employed to find a Karush-Kuhn-Tucker solution, ensuring strict constraint satisfaction. Experimental validation on a differential-wheeled mobile robot demonstrates that the algorithm strictly enforces hard constraints during complex maneuvers while maintaining stable convergence of the value function.

Keywords: Online reinforcement learning, Constrained optimization, Nonlinear systems, Tracking control, Mobile robots.

1. INTRODUCTION

Reinforcement Learning (RL) and Approximate Dynamic Programming (ADP) have emerged as powerful paradigms for the optimal control of complex nonlinear systems (Wang et al. (2018); Köpf et al. (2017); Sutton and Barto (2018)). By iteratively approximating the solution to the governing optimality conditions—specifically, the Hamilton-Jacobi-Bellman (HJB) equation in continuous time or the Bellman Optimality Equation (BOE) in discrete time—these methods enable autonomous agents to learn optimal policies without requiring exact analytical solutions or offline computation. This capability is particularly advantageous for systems operating in uncertain environments where the system dynamics may be partially unknown (Wang et al. (2024, 2021); Kamalapurkar et al. (2016)).

However, the transition of these algorithms to safety-critical physical systems faces significant hurdles. A critical deficiency of the standard ADP formulation lies in the unconstrained nature of the minimization step, specifically within the critic learning phase. Conventional methods typically formulate the weight update as an unconstrained error minimization problem, often implemented via iterative gradient descent or least-squares estimation (Gu et al. (2024); Ye et al. (2025); Modares and Lewis (2014); Hailemichael et al. (2022a,b)). Since this approach does not restrict the policy search space within the feasible region, it prioritizes the numerical reduction of the temporal difference error over the physical realizability of the resulting control policy. Consequently, the value function approximation often evolves in a direction that implies unbounded control efforts, causing the derived control law u_k^* to frequently violate actuator saturation limits or safety boundaries. This issue is compounded in conventional control-oriented ADP frameworks—designed to approximate Policy Iteration (PI) or Value Iteration (VI) solutions—that rely on sample-based valuation techniques. The integration of these incremental update targets with nonlinear function approximation frequently leads to a loss of contraction properties or excessive variance, risking divergence (Bertsekas (2019); He et al. (2020); Moreno-Mora et al. (2023)). Such instability manifests as oscillatory convergence or complete divergence, making the system highly susceptible to variations in step sizes and distribution shifts.

To concurrently secure system feasibility and stability, this paper proposes a framework where the traditional incremental backup strategy is superseded by a constrained optimization formulation that equates to the Bellman optimality equation. Within this paradigm, both the value-function parameters and the control policy are formulated as joint decision variables. By embedding input and safety boundaries directly as constraints within the optimization problem, feasibility is enforced intrinsically, thereby obviating the need for ad hoc projections. The problem is resolved online via a Lagrangian-based primal–dual scheme, which iteratively seeks a Karush-Kuhn-Tucker (KKT) so-

⋆ This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00554087).

lution. This ensures that the Bellman error is driven to zero while strictly adhering to the imposed constraints.

The remainder of this paper is organized as follows. Section 2 formulates the problem. Section 3 details the proposed constrained optimization method and update laws. Section 4 presents the experimental validation on a mobile robot, and Section 5 concludes the paper.

2. PROBLEM FORMULATION

Consider a general discrete-time nonlinear system described by the following error dynamics:

$$e_{k+1} = f(e_k, d_k) + g(e_k)u_k \tag{1}$$

where $e_k \in \mathbb{R}^n$ represents the system state error, $u_k \in \mathbb{R}^m$ is the control input, and $f : \mathbb{R}^n \to \mathbb{R}^n$ represents the unknown drift dynamics of the system, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ represents the control input effectiveness. The term $d_k \in \mathbb{R}^n$ represents the unknown disturbance term, which accounts for unmodeled dynamics or discretization errors.

For the discrete-time error dynamics described in (1), the primary objective of the optimal control problem is to determine a policy $u_k = \pi(e_k)$ that minimizes the infinite-horizon cost function, defined as the accumulation of future costs:

$$J(e_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(e_i, u_i) \tag{2}$$

where $r(e_i, u_i)$ denotes the stage cost and $\gamma \in (0, 1]$ represents the discount factor.

In accordance with Bellman's principle of optimality (Lewis et al. (2012)), the solution is characterized by the optimal value function $J^*(e)$, which quantifies the minimum cost-to-go from a given state e. This function must satisfy the BOE:

$$J^*(e_k) = \min_{u_k} \left(r(e_k, u_k) + \gamma J^*(e_{k+1})\right) \tag{3}$$

Consequently, the optimal control policy $\pi^*(e_k)$ is defined as the argument that achieves the minimum of the right-hand side of the BOE:

$$\pi^*(e_k) = \arg\min_{u_k} \left\{r(e_k, u_k) + \gamma J^*(e_{k+1})\right\} \tag{4}$$

Assuming a quadratic cost function of the form $r(e_k, u_k) = e_k^T Q e_k + u_k^T R u_k$, the optimal control input can be derived analytically using the gradient of the value function:

$$u_k^* = -\frac{\gamma}{2} R^{-1} g_d(e_k)^T \nabla J^*(e_{k+1}) \tag{5}$$

However, the direct application of this unconstrained law is insufficient for systems with strict physical limitations. Therefore, the subsequent section introduces a method to solve (3) while explicitly satisfying inequality constraints.

3. PROPOSED METHOD

3.1 Neural Network Approximation

Since analytical solutions for the optimal value function $J^*(x)$ in the Bellman equation (3) are generally intractable for nonlinear systems, a function approximation scheme is necessary. Specifically, a neural network (NN) architecture is employed to approximate $J^*(x)$ over a compact set $\Omega \subset \mathbb{R}^n$.

Leveraging the universal approximation property of neural networks (Vrabie and Lewis (2009)), the optimal value function can be represented using a dense basis set as:

$$J^*(e_k) = W^{*T}\phi(e_k) + \varepsilon(e_k) \tag{6}$$

where $\phi(e_k) \in \mathbb{R}^p$ denotes the vector of p activation functions, and $\varepsilon(e_k)$ represents the reconstruction error. The ideal weight vector $W^* \in \mathbb{R}^p$ is defined as the parameter set that minimizes the supremum of the approximation error over Ω:

$$W^* = \arg\min_{W\in\mathbb{R}^p} \left\{ \sup_{e_k\in\Omega} \|J^*(e_k) - W^T\phi(e_k)\| \right\} \tag{7}$$

Theoretical results ensure that the error $\varepsilon(x_k)$ can be arbitrarily reduced by increasing the number of hidden layer neurons p.

In the context of online learning, the objective is to estimate the ideal weights iteratively. Let W_k denote the estimate at time step k. The approximated value function, implemented by the critic network, is thus given by:

$$\hat{J}(e_k, W_k) = W_k^T\phi(e_k) \tag{8}$$

This parametric structure serves as the foundation for the constrained optimization problem formulated in the subsequent section.

3.2 Constrained Optimization Formulation

Based on the Bellman optimality equation (3) and the neural network approximation (8), the learning process is reformulated as a constrained optimization problem at each time step k. Rather than minimizing the Bellman residual error by unconstrained least-squares, the control input u_k and the weight vector W_k are treated as joint decision variables. The Bellman equation is enforced as an equality constraint, while physical limits are handled via inequality constraints. The resulting optimization problem is defined as:

$$\begin{aligned} \min_{u_k, W_k} \quad & r(e_k, u_k) + \gamma\hat{J}(e_{k+1}, W_k) \\ \text{s.t.} \quad & \delta_k(u_k, W_k) = 0 \\ & c_j(u_k, \xi_k) \le 0, \quad j = 1, \dots, m \end{aligned} \tag{9}$$

where the equality constraint corresponds to the Bellman residual error:

$$\delta_k(u_k, W_k) \triangleq r(e_k, u_k) + \gamma\hat{J}(e_{k+1}, W_k) - \hat{J}(e_k, W_k) \tag{10}$$

and $c_j(u_k, \xi_k)$ denotes the set of system constraints.

To solve this constrained problem, the Lagrangian function L is constructed as follows:

$$\begin{aligned} L(u_k, W_k, \lambda) =& r(e_k, u_k) + \gamma\hat{J}(e_{k+1}, W_k) \\ &+ \lambda_{\delta,k}\delta_k(u_k, W_k) + \sum_j \lambda_{c,j,k} c_j(u_k, e_k) \end{aligned} \tag{11}$$

where $\lambda_{\delta,k}$ is the Lagrange multiplier associated with the Bellman residual, and $\lambda_{c,j,k}$ represents the multipliers for the system constraints. The optimal solution is determined by satisfying the Karush-Kuhn-Tucker (KKT) conditions.

3.3 Online Learning Laws

To ensure convergence to the optimal solution, gradient-based update laws are formulated to iteratively seek the saddle point of the Lagrangian, which corresponds to the KKT conditions.

Critic Weight Update The critic weights W_k are updated via gradient descent to minimize the Lagrangian:

$$W_{k+1} = W_k - \alpha_W \nabla_{W_k} L \tag{12}$$

The gradient is computed as:

$$\nabla_{W_k} L = \phi(e_{k+1}) + \lambda_{\delta,k}(\gamma\phi(e_{k+1}) - \phi(e_k)) \tag{13}$$

Here, the term $\phi(e_{k+1})$ originates from the objective function, driving the value function to reflect the future discounted cost, while the second term incorporating the Bellman multiplier ensures adherence to the BOE.

Lagrange Multiplier Updates The dual variables are updated via gradient ascent to strictly enforce the constraints:

Bellman Residual Multiplier This multiplier is updated to drive the Bellman error δ_k to zero:

$$\lambda_{\delta,k+1} = \lambda_{\delta,k} + \alpha_\delta \delta_k(u_k, W_k) \tag{14}$$

Constraint Multipliers The inequality constraints are enforced using a projected gradient method, ensuring non-negativity:

$$\lambda_{c,j,k+1} = \max\left(0, \lambda_{c,j,k} + \alpha_c c_j(u_k, e_k)\right) \tag{15}$$

Policy Update The optimal control input u_k is obtained by solving the stationarity condition, $\nabla_{u_k} L = 0$. For a quadratic stage cost formulation, the closed-form update law is derived as:

$$u_{k+1} = -\frac{\gamma}{2} R^{-1} \left(W_k^T \nabla\phi(e_{k+1}) g_d(e_k) + \frac{\sum \lambda_{c,j,k} \nabla_u c_j}{1 + \lambda_{\delta,k}} \right) \tag{16}$$

The overall computational procedure, which integrates these update laws with real-time policy generation, is summarized in Algorithm 1.

Algorithm 1 Proposed Method

Initialize:
  Critic weights W_0, Multipliers $\lambda_{\delta,0}, \lambda_{c,j,0}$
  Learning rates $\alpha_W, \alpha_\delta, \alpha_c$
  Discount factor γ, Matrices Q, R
  Initial error e_0, Input $u_0 = 0$
  Sampling time T_s, Equality constraint tolerance ϵ
Start Control Loop:
for $k = 0, 1, 2, \ldots$ **do**
  Step 1: Actuation and Feedback
  Apply u_k to robot; wait T_s; measure e_{k+1}
  Step 2: Online Learning
  2a. Critic Weight Update
  $\nabla_{W_k} L \leftarrow \phi(e_{k+1}) + \lambda_{\delta,k}(\gamma\phi(e_{k+1}) - \phi(e_k))$
  $W_{k+1} \leftarrow W_k - \alpha_W \nabla_{W_k} L$
  2b. Multiplier Updates
  $\delta_k \leftarrow r(e_k, u_k) + \gamma W_k^T \phi(e_{k+1}) - W_k^T \phi(e_k)$
  $\lambda_{\delta,k+1} \leftarrow \lambda_{\delta,k} + \alpha_\delta \delta_k$
  for $j = 1 \rightarrow m$ **do**
    $\lambda_{c,j,k+1} \leftarrow \max\left(0, \lambda_{c,j,k} + \alpha_c c_j(u_k, e_k)\right)$
  end for
  Step 3: Policy Update
  Compute $g_d(e_k)$ and $\nabla\phi(e_{k+1})$
  Calculate optimal control input u_{k+1}:
  $u_{k+1} \leftarrow -\frac{\gamma}{2} R^{-1} \Big[W_{k+1}^T \nabla\phi(e_{k+1}) g_d(e_k) + \frac{\sum_{j=1}^m \lambda_{c,j,k} \nabla_u c_j}{1+\lambda_{\delta,k}} \Big]$
  Step 4: Update States and Parameters for Next Iteration
  $e_k \leftarrow e_{k+1}$
  $W_k \leftarrow W_{k+1}$
  $\lambda_{\delta,k} \leftarrow \lambda_{\delta,k+1}$
  $\lambda_{c,j,k} \leftarrow \lambda_{c,j,k+1}$
  $u_k \leftarrow u_{k+1}$
end for

4. EXPERIMENTAL VALIDATION

4.1 Case Study: Mobile Robot

The proposed control framework is validated using a physical differential-wheeled mobile robot. The kinematic evolution of the robot on a 2-D plane is governed by the following model:

$$\dot{\xi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u_\xi \tag{17}$$

where $\xi = [x, y, \theta]^T$ denotes the state vector comprising the position and orientation, and $u_\xi = [v, \omega]^T$ represents the control input vector consisting of linear and angular velocities.

To define the tracking objective, a reference trajectory is specified by the reference state $\xi_r = [x_r, y_r, \theta_r]^T$ and reference inputs $u_r = [v_r, \omega_r]^T$, satisfying:

$$\dot{\xi}_r = \begin{bmatrix} v_r \cos\theta_r \\ v_r \sin\theta_r \\ \omega_r \end{bmatrix} \tag{18}$$

where the reference velocities are approximated based on the trajectory derivatives as $v_r = \sqrt{\dot{x}_r^2 + \dot{y}_r^2}$ and $\omega_r = (\dot{x}_r \ddot{y}_r - \ddot{x}_r \dot{y}_r)/(\dot{x}_r^2 + \dot{y}_r^2)$.

The tracking error $e = [x_e, y_e, \theta_e]^T$ is defined in the robot's local coordinate frame to facilitate controller design (Li et al. (2016)):

$$e = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} (\xi_r - \xi) \tag{19}$$

By differentiating (19) with respect to time and substituting the kinematic models, the error dynamics can be reformulated into the standard nonlinear control-affine form $\dot{e} = f(e, u_r) + g(e)u$:

$$\dot{e} = \begin{bmatrix} \omega_r y_e + v_r \cos\theta_e - v_r \\ v_r \sin\theta_e - \omega_r x_e \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & y_e \\ 0 & -x_e \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v - v_r \\ \omega - \omega_r \end{bmatrix} \tag{20}$$

For digital implementation, the continuous-time dynamics are discretized using Euler approximation with a sampling time T_s:

$$e_{k+1} = e_k + T_s \dot{e}_k = f_d(e_k, u_{r,k}) + g_d(e_k) u_k \tag{21}$$

where the discrete-time drift and input terms are given by $f_d(e_k, u_{r,k}) = e_k + T_s f(e, u_r)$ and $g_d(e_k) = T_s g(e)$, respectively.

4.2 Experimental Setup

The experimental validation is conducted within the environment depicted in Fig. 1. The detailed hardware speci-

fications of the mobile robot platform are summarized in Table 1.

The onboard computational architecture is powered by an Intel Core Ultra 5 125H processor (14 cores). The software ecosystem is founded on the Robot Operating System 2 (ROS 2) Humble Hawksbill framework, operating on Ubuntu 22.04 LTS. The proposed control algorithm is implemented within the MATLAB/Simulink 2024b environment. Real-time command execution and telemetry acquisition are facilitated via a ROS-Simulink communication bridge, ensuring a deterministic control sampling time (T_s) of 1 ms.

4.3 Experimental Results

To strictly assess the capability of the proposed method in handling actuator saturation and safety limits, a figure-eight trajectory tracking experiment was performed under hard constraints.

The reference trajectory is generated using the following sinusoidal functions:

$$\begin{aligned} x_r(t) &= 3\sin(2\pi\omega_{o1}t) \\ y_r(t) &= 1.5\cos(2\pi\omega_{o2}t) \end{aligned} \tag{22}$$

where the frequencies are set to $\omega_{o1} = 0.02$ Hz and $\omega_{o2} = 0.01$ Hz.

In this scenario, stringent constraints are imposed, particularly on the angular velocity, to induce saturation during complex maneuvering. The system constraints are defined as follows:

Fig. 1. Experimental environment and the mobile robot platform used for validation.

Table 1. Specifications of the Mobile Robot Platform

Parameter	Value
Dimensions ($L \times W \times H$)	$600 \times 600 \times 400$ mm
Weight	105 kg
Payload (Rated / Max)	35 kg / 200 kg
Max. Speed	1.8 m/s
Ground Clearance	38 mm
Wheel Diameter	139.7 mm
Wheelbase	490 mm
Motor Power	500 W
Encoder Resolution	4096 pulses/rev
Battery	24 V, 50 A
Operating Hours	$\approx$ 10 hours

$$\begin{aligned} c_1(u,\xi) &= v - 0.5 \le 0 \\ c_2(u,\xi) &= -v - 0.5 \le 0 \\ c_3(u,\xi) &= \omega - 0.5 \le 0 \\ c_4(u,\xi) &= -\omega - 0.5 \le 0 \end{aligned} \tag{23}$$

For the implementation of the proposed method, the weighting matrices for the quadratic stage cost are selected as $Q = I_3$ and $R = 0.5I_2$, where I_n denotes an $n \times n$ identity matrix. The discount factor is set to $\gamma = 0.95$. The critic network utilizes a quadratic basis function vector defined as $\phi(e) = [x_e^2, y_e^2, \theta_e^2, x_e y_e, x_e \theta_e, y_e \theta_e]^T$, initialized with weights $W_c(0) = [1, 1, 1, 1, 1, 1]^T$. The learning rates for the primal and dual updates are configured as $\alpha_W = 1 \times 10^{-6}$ for the critic weights, $\alpha_\delta = 1 \times 10^{-5}$ for the Bellman residual multiplier, and $\alpha_{c,j} = 0.6$ for the constraint multipliers ($j = 1, \ldots, 4$). Additionally, the tolerance for the Bellman residual equality constraint is set to $\epsilon = 0.08$.

The tracking performance under these hard constraints is presented in Fig. 2 and Fig. 3. Despite the reduced control authority resulting from strict angular velocity limits, the mobile robot successfully tracks the figure-eight reference path. As observed in Fig. 2(b) and Fig. 3, the tracking errors in the x-position, y-position, and orientation eventually converge. It is noted that the transient response exhibits slightly more aggressive behavior compared to unconstrained scenarios, a direct consequence of control input saturation.

The active enforcement of constraints is explicitly demonstrated in Fig. 4. While the linear velocity v remains within the feasible region, the angular velocity ω is frequently clipped at the imposed limits of ± 0.5 rad/s, particularly during the turning phases of the trajectory. This confirms that the computed optimal control policy strictly adheres to the hard safety bounds defined in (23).

Fig. 5 illustrates the learning progression, indicating that the estimated value function $\hat{J}$ and the critic weights W_c converge to stable values even under input saturation. This robustness is further corroborated by Fig. 6, where the Bellman residual error δ_k is driven toward zero, and the Bellman multiplier λ_δ reaches a steady state.

Finally, the dynamics of the constraint multipliers are depicted in Fig. 7 and Fig. 8. The multipliers associated with the angular velocity constraints (Fig. 8) exhibit sharp increases coincident with the angular velocity reaching its maximum or minimum limits. This dynamic activation provides evidence that the proposed primal-dual update law correctly identifies active constraints and adjusts the control policy to satisfy inequality constraints while maintaining optimality.

Crucially, these results highlight the algorithm's capability to simultaneously manage online learning and constraint enforcement. It is evident that while the constraint multipliers react aggressively to boundary violations to restore feasibility, this dynamic intervention does not compromise the stability of the critic learning process. The separation of concerns inherent in the Lagrangian formulation allows the critic weights to converge smoothly toward the optimal parameters, even in the presence of active constraints and fluctuating multipliers. This confirms that the proposed

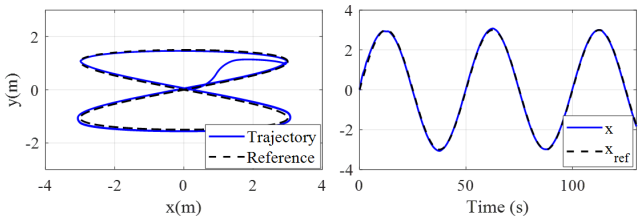

Fig. 2. Hard constrained control results: (a) Path tracking performance, (b) x-position tracking.

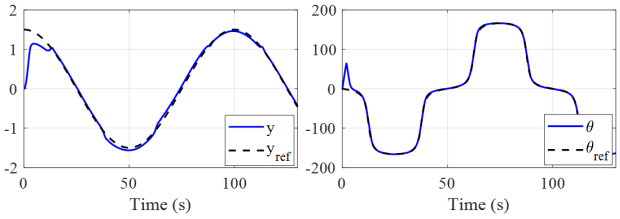

Fig. 3. Hard constrained control results: (a) y-position tracking, (b) Orientation (θ) tracking (deg).

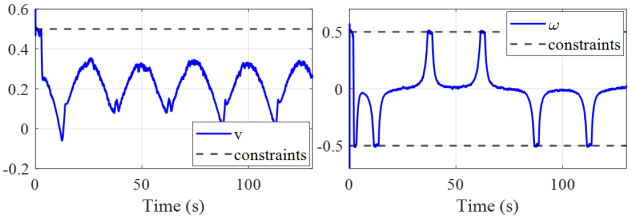

Fig. 4. Constrained control inputs: (a) Linear velocity v with active constraints, (b) Angular velocity ω with active constraints.

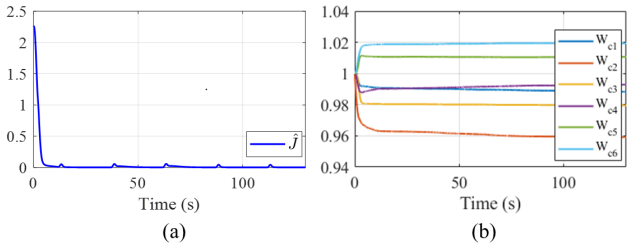

Fig. 5. Learning performance under hard constraints: (a) Estimated Cost Function $\hat{J}$, (b) Convergence of Critic Neural Network weights W_c.

method effectively balances the dual objectives of safety assurance and optimality learning within a unified real-time framework.

5. CONCLUSION

This paper presented a online constrained reinforcement learning framework for the optimal tracking control of constrained nonlinear systems, addressing the limitations of conventional unconstrained minimization strategies. The core contribution of this work lies in the reformulation

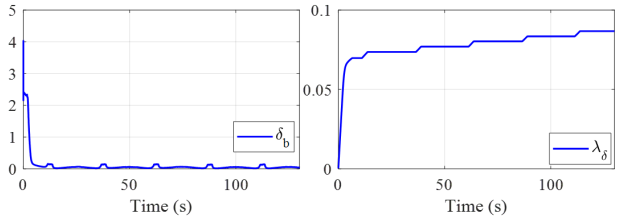

Fig. 6. Optimization performance under hard constraints: (a) Bellman residual error δ_k, (b) Bellman residual multiplier λ_δ.

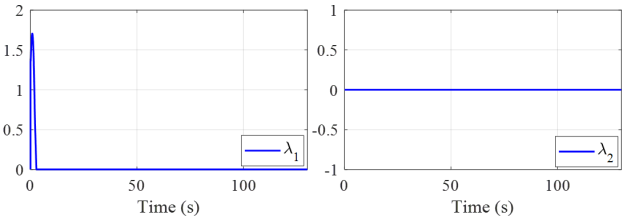

Fig. 7. Constraint multipliers dynamics under hard constraints: (a) Multiplier λ_1, (b) Multiplier λ_2.

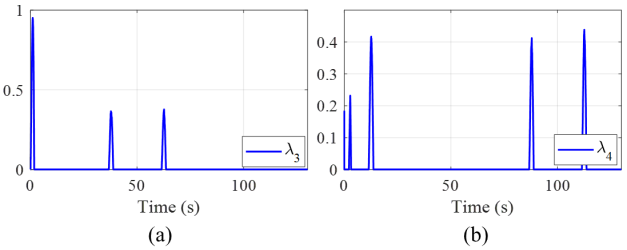

Fig. 8. Constraint multipliers dynamics under hard constraints: (a) Multiplier λ_3, (b) Multiplier λ_4.

of the Bellman optimality equation as a constrained optimization problem where the control policy and value function are treated as joint decision variables. Crucially, this framework allows for the explicit incorporation of system constraints directly into the learning process, ensuring physical realizability. To solve this problem, a Lagrangian-based primal-dual scheme is utilized to iteratively derive the Karush-Kuhn-Tucker solution, guaranteeing strict constraint satisfaction alongside learning stability. Experimental validation on a differential-wheeled mobile robot demonstrated the practical efficacy of the proposed approach, where the controller successfully enforced hard actuator limits during complex maneuvers while ensuring the stable convergence of the value function. Future research will focus on enhancing the framework's robustness against stochastic disturbances and dynamic obstacles in unstructured environments. Additionally, the proposed methodology will be extended to multi-agent systems to address distributed control problems involving complex task allocation and coordination.

REFERENCES

Bertsekas, D. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., and Knoll, A. (2024). A Review of Safe Reinforcement Learning: Methods, Theories, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12), 11216–11235. doi: 10.1109/TPAMI.2024.3457538.

Hailemichael, H., Ayalew, B., Kerbel, L., Ivanco, A., and Loiselle, K. (2022a). Safe Reinforcement Learning for an Energy-Efficient Driver Assistance System. *IFAC-PapersOnLine*, 55(37), 615–620. doi: 10.1016/j.ifacol.2022.11.250.

Hailemichael, H., Ayalew, B., Kerbel, L., Ivanco, A., and Loiselle, K. (2022b). Safety Filtering for Reinforcement Learning-based Adaptive Cruise Control. *IFAC-PapersOnLine*, 55(24), 149–154. doi: 10.1016/j.ifacol.2022.10.276.

He, S., Fang, H., Zhang, M., Liu, F., and Ding, Z. (2020). Adaptive Optimal Control for a Class of Nonlinear Systems: The Online Policy Iteration Approach. *IEEE Transactions on Neural Networks and Learning Systems*, 31(2), 549–558. doi: 10.1109/TNNLS.2019.2905715.

Kamalapurkar, R., Walters, P., and Dixon, W.E. (2016). Model-based reinforcement learning for approximate optimal regulation. *Automatica*, 64, 94–104. doi: 10.1016/j.automatica.2015.10.039.

Köpf, F., Inga, J., Rothfuß, S., Flad, M., and Hohmann, S. (2017). Inverse Reinforcement Learning for Identification in Linear-Quadratic Dynamic Games. *IFAC-PapersOnLine*, 50(1), 14902–14908. doi: 10.1016/j.ifacol.2017.08.2537.

Lewis, F.L., Vrabie, D.L., and Syrmos, V.L. (2012). *Optimal Control*. Wiley, 3rd edition.

Li, Z., Deng, J., Lu, R., Xu, Y., Bai, J., and Su, C.Y. (2016). Trajectory-Tracking Control of Mobile Robot Systems Incorporating Neural-Dynamic Optimized Model Predictive Approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6), 740–749. doi:10.1109/TSMC.2015.2465352.

Modares, H. and Lewis, F.L. (2014). Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica*, 50(7), 1780–1792. doi: 10.1016/j.automatica.2014.05.011.

Moreno-Mora, F., Beckenbach, L., and Streif, S. (2023). Predictive Control with Learning-Based Terminal Costs Using Approximate Value Iteration. *IFAC-PapersOnLine*, 56(2), 3874–3879. doi: 10.1016/j.ifacol.2023.10.1320.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition.

Vrabie, D. and Lewis, F. (2009). Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks*, 22(3), 237–246. doi:10.1016/j.neunet.2009.03.008.

Wang, D., Gao, N., Liu, D., Li, J., and Lewis, F.L. (2024). Recent Progress in Reinforcement Learning and Adaptive Dynamic Programming for Advanced Control Applications. *IEEE/CAA Journal of Automatica Sinica*, 11(1), 18–36. doi:10.1109/JAS.2023.123843.

Wang, N., Gao, Y., Zhao, H., and Ahn, C.K. (2021). Reinforcement Learning-Based Optimal Tracking Control of an Unknown Unmanned Surface Vehicle. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7), 3034–3045. doi:10.1109/TNNLS.2020.3009214.

Wang, Y., Velswamy, K., and Huang, B. (2018). A Novel Approach to Feedback Control with Deep Reinforcement Learning. *IFAC-PapersOnLine*, 51(18), 31–36. doi:10.1016/j.ifacol.2018.09.241.

Ye, J., Dong, H., Bian, Y., Qin, H., and Zhao, X. (2025). ADP-Based Optimal Control for Discrete-Time Systems With Safe Constraints and Disturbances. *IEEE Transactions on Automation Science and Engineering*, 22, 115–128. doi:10.1109/TASE.2023.3346876.