ISA Transactions 168 (2026) 465–478

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatra

Research article

# LSTM-empowered reinforcement learning in Bi-level optimal control for nonlinear systems with uncertain dynamics

Roya Khalili Amirabadi[a], Mohsen Jalaeian-Farimani[b,*], Omid S. Fard[a]

[a] Department of Applied Mathematics, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Milan, Italy

## HIGHLIGHTS

- A novel integration of LSTM-based temporal uncertainty estimation with actor-critic RL, enabling real-time adaptation without offline training.
- A computationally efficient bi-level framework that outperforms single-level RL and adaptive control in handling nonlinear systems with uncertain dynamics.
- Rigorous stability guarantees via UUB analysis, validated on a skid-steering robot under realistic slip disturbances.

## ARTICLE INFO

*Keywords:*
Bi-level optimal control
Reinforcement learning
Sliding mode
Actor-critic neural network
Skid-steering tracked robot

## ABSTRACT

This paper introduces a bi-level optimization framework for the optimal control of nonlinear continuous-time systems with uncertain dynamics, seamlessly integrating Long Short-Term Memory (LSTM) networks with an actor-critic reinforcement learning (RL) architecture. By synergizing Hamiltonian-based optimal control with online uncertainty estimation, the proposed method achieves robust trajectory tracking without reliance on offline training. The master level optimizes control policies using an HJB-inspired formulation, while the slave level employs LSTM networks to dynamically estimate lumped uncertainties, ensuring adaptability to time-varying disturbances. Rigorous stability analysis establishes uniform ultimate boundedness of the tracking error, guaranteeing robust performance. Extensive simulations on a skid-steering tracked robot across diverse trajectories demonstrate the framework's superior tracking precision, energy efficiency, and disturbance rejection compared to conventional adaptive control and model-based virtual reference trajectory schemes. This computationally efficient and theoretically grounded approach offers a scalable solution for autonomous systems operating in uncertain environments, advancing the paradigm of RL-based optimal control.

## 1. Introduction

Optimal control of nonlinear continuous-time systems with uncertain dynamics remains a critical challenge in autonomous systems, demanding robust and adaptive solutions for real-world applications such as robotics and intelligent transportation. Recent advancements in RL have shown promise in addressing these challenges [1–3], with data-driven feedback relearning algorithms enhancing adaptability to unmodeled dynamics [4]. Similarly, RL-based multi-agent control has tackled complex systems with state constraints [5], while experience inference learning inspired by human behavior has bridged simulation-to-real-world gaps in nonlinear control [6]. Physics-informed RL frameworks have further advanced model-free optimal control by incorporating stability guarantees [7], and adaptive critic designs have solved Hamilton–Jacobi–Bellman equations [8,9] for systems with unmatched interconnections [10]. Despite these strides, existing methods often rely on offline training or simplified uncertainty models, limiting their efficacy in dynamic, uncertain environments. This paper proposes a novel bi-level optimization framework that synergizes LSTM networks with an actor-critic RL architecture, integrating Hamiltonian-based optimal control with online uncertainty estimation. By enabling robust trajectory tracking without offline training, the framework achieves superior adaptability and efficiency, as demonstrated through extensive simulations on a skid-steering tracked robot (Table 1).

* Corresponding author.
*Email addresses:* r-khalili@mail.um.ac.ir (R. Khalili Amirabadi), mohsen.jalaeian@polimi.it (M. Jalaeian-Farimani), soleimani@um.ac.ir (O. S. Fard).

https://doi.org/10.1016/j.isatra.2025.11.027
Received 13 May 2025; Received in revised form 16 November 2025; Accepted 16 November 2025
Available online 20 November 2025

**Table 1**
Nomenclature.

| Parameter | Description |
|---|---|
| $X, Y$ | Position coordinates of the robot (m) |
| $\psi$ | Orientation (heading angle) of the robot (rad) |
| $v$ | Linear velocity of the robot (m/s) |
| $\omega$ | Angular velocity of the robot (rad/s) |
| $v_r, v_l$ | Right and left wheel velocities (m/s) |
| $\omega_r, \omega_l$ | Angular velocities of the right and left wheels (rad/s) |
| $r$ | Wheel radius (m) |
| $d_w$ | Distance between the two wheels (m) |
| $\alpha_v, \alpha_\omega$ | Linear and angular slip velocities (m/s, rad/s) |
| $d(t)$ | Lumped system uncertainty |
| $s(t)$ | Sliding mode surface |
| $\vartheta$ | Sliding mode gain matrix |
| $W$ | Weight vector of the critic neural network |
| $b_u$ | Diagonal matrix of control input saturation limits |
| $\varphi(s)$ | Activation function of the critic network |
| $\hat{d}(t)$ | Estimated uncertainty at time $t$ |
| $z_t$ | Input vector to LSTM: concatenated states and controls over $\tau+1$ steps |
| $\tau$ | Temporal window size for historical data (number of steps) |
| $W_{\{f,i,o,c\}}$ | Weight matrices for LSTM gates (forget, input, output, cell) |
| $b_{\{f,i,o,c\}}$ | Bias vector for LSTM gates |
| $f_t, i_t, o_t$ | Forget, Input, and Output gate activation vectors |
| $\tilde{c}_t, c_t$ | Candidate and updated cell state vectors |
| $h_t$ | Hidden state vector of the LSTM |
| $\sigma(\cdot)$ | Sigmoid activation function |
| $\Theta$ | LSTM network parameters: $\{W_{\{f,i,o,c\}}, U, b\}$ |
| $\eta$ | Learning rate for online adaptation |
| $\lambda$ | Regularization parameter for LSTM training |
| $\Delta t$ | Discrete time step |
| $\beta_1, \beta_2$ | Momentum and RMSprop coefficients for Adam optimizer |
| $m_t, v_t$ | First and second moment estimates in Adam optimization |
| $\epsilon_1, \epsilon_2$ | Small positive constants for stopping criteria in Algorithm 1 |
| $\gamma$ | Convergence rate parameter in Lyapunov analysis |

The challenge intensifies when solving optimal control for nonlinear continuous time systems with partially unknown dynamics, a problem inherently structured as a bi-level optimization [11]. Conventional strategies, such as multi-threaded optimization, impose prohibitive hardware costs by relying on parallelized computation. Robust control methods [12], though theoretically sound, adopt overly conservative worst-case uncertainty bounds that degrade performance in practical scenarios. Adaptive control techniques [13,14] dynamically adjust to uncertainties but inherently tether control updates to the accuracy of uncertainty estimates, risking instability under rapid disturbance variations. In contrast, bi-level optimization elegantly decomposes the problem into coupled layers: a master level for policy optimization and a slave level for online uncertainty estimation [11,15]. This bi-level structure enables simultaneous online adaptation and control refinement, circumventing the limitations of prior methods.

This work presents a bi-level optimization framework that seamlessly integrates Hamiltonian -based optimal control, model-informed RL, and neural network-based uncertainty estimation within a mathematically grounded architecture. At the master level, an HJB-inspired control formulation guides energy-efficient trajectory tracking, while the slave level employs lightweight neural networks, including Long Short-Term Memory (LSTM) architectures [16], for identification of dynamic uncertainties. In contrast to conventional RL-based or model predictive approaches, the proposed method tightly couples policy optimization with online disturbance estimation, thereby avoiding the need for offline training and mitigating reliance on static uncertainty assumptions. By fusing actor-critic neural approximators with LSTM networks, the framework achieves both computational efficiency and robustness to time-varying disturbances-surpassing the sample inefficiency of pure RL and the fragility of traditional adaptive control schemes. Experimental results confirm the method's superior adaptability in dynamically changing environments, positioning the framework as a significant advancement for autonomous systems operating under uncertainty. In this work, the term "disturbance" denotes a lumped uncertainty that includes both external effects (e.g., load variations, friction, environmental factors) and internal mismatches (e.g., parameter variations, unmodeled nonlinearities).

Existing methodologies, including sliding mode control [17] and single-level RL architectures, falter in dynamic environments due to static uncertainty assumptions, high sample complexity, or energy-inefficient heuristics. By unifying Hamiltonian optimality principles with adaptive filtering in a bi-level hierarchy, the proposed framework achieves provable stability, reduced computational overhead, and online adaptability. Experimental validation (Section 7) demonstrates robustness against disturbances compared to conventional RL, adaptive, and robust control baselines. The primary contributions of this paper are summarized as follows:

- A novel integration of LSTM-based temporal uncertainty estimation with actor-critic RL, enabling online adaptation without offline training.
- Providing an effective solution for simultaneously solving the nonlinear optimal control problem and the uncertainty estimation optimization, which leads to the presentation of an efficient Bi-Level framework.
- Rigorous stability guarantees via UUB analysis, validated on a skid-steering robot under realistic slip disturbances.

The remainder of this paper is organized as follows. Section 2 presents the problem formulation and system dynamics. Section 3 details the uncertainty estimation module based on LSTM neural networks. Section 4 introduces the proposed bi-level optimization framework, integrating optimal control and reinforcement learning. Section 5 reports simulation results on a skid-steering tracked robot, demonstrating the effectiveness and robustness of the method. Finally, Section 6 concludes the paper with a summary of key findings and outlines potential directions for future research.

## 2. Problem statement and system dynamics

We consider a category of nonlinear continuous-time systems, modeled in affine form and subject to potential uncertainties. The general formulation for such systems is: Consider a general nonlinear system of the form:

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) + d(t), \tag{1}$$

where $x(t) \in \mathbb{R}^{p\times 1}$ represents the system state vector, and $u(t) = [u_1(t), u_2(t), \ldots, u_q(t)]^T \in \mathbb{R}^{q\times 1}$ denotes the vector of control inputs, which is subject to defined constraints. The control input is restricted to the set $u(t) \in \mathcal{V} \subseteq \mathbb{R}^{q\times 1}$, where the admissible control set is given by $\mathcal{V} = \{u(t) \in \mathbb{R}^{q\times 1} \mid |u_i(t)| \le b_{ui},\ i = 1, \ldots, q\}$, and $b_u = diag[b_{u1}, b_{u2}, \ldots, b_{uq}] \in \mathbb{R}^{q\times q}$ denotes the saturation limit for each control input $u_i(t)$. The primary objective is to design a control strategy that guarantees stability for system (1) across all $u \in \mathcal{V}$, and the term $d(t) \in \mathbb{R}^{p\times 1}$ denotes the lumped uncertainty, which includes external disturbances, unmodeled dynamics, measurement errors, and parametric uncertainties.

**Assumption 1.** In this setup, the functions $f(x) \in \mathbb{R}^{p\times 1}$ and $g(x) \in \mathbb{R}^{p\times q}$ define smooth nonlinear dynamics, which are bounded by the conditions $\|f(x)\| \le \alpha_f \|x\|$, $\|g(x)\|_F \le \alpha_g$ where, $\alpha_f$, and $\alpha_g$ are positive real constants, and $\|\cdot\|_F$ denotes the Frobenius norm. Also, the lumped uncertainty $d(t)$ is assumed to be bounded, i.e., $\|d(t)\| \le \alpha_d$, where $\alpha_d$ is a known constant.

The tracking error is defined as:

$$e(t) = x_d(t) - x(t), \tag{2}$$

where $x_d(t) \in \mathbb{R}^{p\times 1}$ is the desired trajectory. A sliding mode surface is designed as:

$$s(t) = e(t) + \vartheta \int_{\tau=0}^{\tau=t} e(\tau) d\tau, \tag{3}$$

where $\vartheta$ is a positive constant matrix with compatible dimension. The sliding mode surface $s(t)$ is introduced to combine the tracking error and its integral, thereby ensuring robustness against uncertainties and guaranteeing faster convergence and boundedness of the closed-loop system. Hence, the value function is formulated over $s(t)$ instead of directly over $e(t)$. By differentiating (3):

$$\dot{s}(t) = \dot{e}(t) + \vartheta e(t) = \dot{x}_d(t) - (f(x) + g(x)u(t) + d(t)) + \vartheta e(t). \tag{4}$$

Following the sliding mode control, the objective is to drive (3) to zero. The optimal control is derived based on the following value function:

$$V(s) = \int_t^{\infty} \left( s^{\top}(\tau) Q s(\tau) + U(u(\tau)) \right) d\tau, \tag{5}$$

where, $Q$ is defined as symmetric positive definite matrix that can be adjusted based on the importance of the errors. Also, $U(u) = \Sigma_{i=1}^{q} \int_0^{u_i} 2b_{ui} \varsigma^{-1} \left( \frac{v}{b_{ui}} \right) dv$, here the function $\varsigma(\cdot)$ is a continuously increasing odd function and $b_{ui}$ denotes the saturation limit for the $i-th$ control input. [18]. In this work, we choose the function $\varsigma(\cdot)$ as $\tanh(\cdot)$ as below:

$$U(u) = \sum_{i=1}^{q} \int_0^{u_i} 2b_{ui} \tanh^{-1} \left( \frac{v}{b_{ui}} \right) dv \tag{6}$$

We choose a smooth $U(u)$ to enforce admissibility of the control within the bounds $\mathcal{V} = \{u : |u_i| < b_{u_i}\}$. This form discourages infeasible inputs during optimization, ensures that the instantaneous Hamiltonian admits a unique minimizer with respect to $u$, and naturally yields the closed-form saturated policy adopted in this work.

By considering (5), the Hamiltonian is formulated as follows [18]:

$$\mathcal{H}(s, u, \nabla V) = \nabla V^{\top} \dot{s}(t) + s^{\top}(t) Q s(t) + U(u),$$

where, $\nabla V \triangleq \frac{\partial V}{\partial s}$, and the term $\nabla V^{\top} \dot{s}(t)$ captures the impact of the dynamics on the value function, $s^{\top} Q s$ penalizes state deviation according to the quadratic cost, and $U(u)$ enforces admissibility of the control input.

Based on (2), the optimal control law is obtained as:

$$u^* = \arg\min_u \mathcal{H}(s, u, \nabla V^*). \tag{7}$$

Taking the partial derivative $\frac{\partial \mathcal{H}}{\partial u} = 0$, and Hamilton-Jacobi-Bellman equation, the optimal control law is [18]:

$$u^* = b_u \tanh \left( \frac{1}{2} b_u^{-1} g^{\top}(x) \nabla V^*(s) \right). \tag{8}$$

The implementation of the online RL algorithm, as outlined in the next section, builds upon the formulation of the optimal control problem introduced earlier, providing a practical method for approximating the solution using neural networks.

### 2.1. Implementation of the online RL algorithm using neural networks

Since Eq. (5) cannot be computed analytically, a critic neural network is employed to approximate the optimal value function, given the highly nonlinear nature of the performance index. By using only a single critic network, the implementation structure is simplified. The critic NN with $N \in \mathbb{N}$ hidden neurons is designed as:

$$V(s) = W^{\top} \varphi(s(t)) + \epsilon(s), \tag{9}$$

where $W \in \mathbb{R}^{N \times 1}$ is the weight vector, $\varphi(s(t)) \in \mathbb{R}^{N \times 1}$ represents the activation functions, in whose elements can be any stationary, bounded and monotone increasing functions, and $\epsilon(s)$ denotes the approximation residual.

To improve the training efficiency of the RL algorithm utilizing the critic neural network, we employ an experience replay mechanism to enhance policy optimization. Specifically, this approach uses a buffer to store and sample past system interactions, with each experience comprising state transitions and control actions. By enabling the RL algorithm to learn from a diverse set of historical data, experience replay mitigates the sample inefficiency inherent in traditional RL, facilitating robust policy updates through randomized sampling [20,21]. This method is particularly advantageous for repetitive control tasks, as it allows the algorithm to revisit and refine control strategies based on recurring system dynamics, thereby improving long-term stability and adaptability in uncertain nonlinear systems.

According to (9), the gradient of $V(s(t))$, denoted as $\nabla_s V \in \mathbb{R}^{p \times 1}$, is expressed as:

$$\nabla_s V = \nabla_s \varphi^{\top}(s(t)) W + \nabla_s \epsilon, \tag{10}$$

where, $\nabla_s \varphi(s(t)) \triangleq \frac{\partial \varphi(s(t))}{\partial s(t)} \in \mathbb{R}^{N \times p}$ and $\nabla_s \epsilon \triangleq \frac{\partial \epsilon(s)}{\partial s(t)} \in \mathbb{R}^{p}$ represent the derivatives of the activation function and the approximation residual, respectively.

Since the approximation residual $\epsilon(s(t))$ is not available, and $W$ is estimated, the approximations for $V(s(t))$ and its gradient are expressed as:

$$\hat{V}(s) = \varphi(s(t))^{\top} \hat{W}, \tag{11}$$

$$\nabla_s \hat{V}(s) = \nabla_s \varphi^{\top}(s(t)) \hat{W}, \tag{12}$$

where, $\hat{W}$ denotes the estimated weight vector. The approximation error at time $t$ is defined as:

$$\delta \triangleq V - \hat{V} = \varphi^{\top}(s(t))(W - \hat{W}) + \epsilon(s(t)), \tag{13}$$

Using the result from Eq. (8), the control input can then be updated as:

$$\hat{u}_i = b_u \tanh \left( \frac{1}{2} b_u^{-1} g^{\top}(x) \nabla_s \varphi^{\top}(s(t)) \hat{W} \right) \tag{14}$$

Here, the subscript $i$ in $\hat{u}_i$ denotes the $i$-th control input component, corresponding to the $i$-th element of the control input vector $u(t) = [u_1(t), u_2(t), \ldots, u_q(t)]^{\top}$.

By substituting Eqs. (11), (12), and (14) into the Hamiltonian function in Eq. (2), we obtain the following approximation:

$$\begin{aligned} \hat{\mathcal{H}}(s, x, \hat{d}, \hat{W}) = {} & s^{\top}(t) Q s(t) + U(\hat{u}) + \hat{W}^{\top} \nabla_s \varphi(s(t))(\dot{x}_d(t) - f(x) - d(t) \\ & + \vartheta e(t)) - \hat{W}^{\top} \nabla_s \varphi(s(t)) g(x) b_u \tanh \\ & \times \left( \frac{1}{2} b_u^{-1} g^{\top}(x) \nabla_s \varphi^{\top}(s(t)) \hat{W} \right). \end{aligned} \tag{15}$$

This function is used to update the critic weight $\hat{W}$, which in turn drives the approximation of the optimal control policy. However, the exact value of $d(t)$ is unknown and must be estimated. The methodology to estimate $d(t)$ is described in Section 3.

## 3. Uncertainty estimation via LSTM

The Long Short-Term Memory (LSTM) network is a gated recurrent neural architecture designed to capture both short- and long-term temporal dependencies in sequential data. Its use in adaptive control is meaningful because it can model time-varying uncertainties with memory of past disturbances, thereby providing more accurate online estimation than static function approximators.

Accurate system identification is essential for capturing system dynamics, as it directly impacts the precision of both control and simulation. The following section addresses this challenge by presenting

methodologies for estimating $d(t)$, enabling accurate uncertainty compensation.

Unlike traditional RL-based adaptive control, which uses static function approximators and therefore treats uncertainties as memoryless, the proposed LSTM-based approach exploits temporal dependencies through memory cells, enabling more accurate estimation of time-varying and correlated uncertainties.

It is important to note that while the critic in Eq. (10) uses a standard feedforward neural network to approximate the value function $V(s)$ (which depends only on the current sliding variable), the LSTM is specifically adopted at the lower level to estimate $d(t)$ because its memory cells can capture the temporal correlations in uncertainties that a general NN cannot.

To estimate uncertainty, an LSTM network is employed. This method provides optimal recursive estimation of $\hat{d}(t)$ based on available measurements.

The estimated uncertainty is then integrated into the optimal control law to ensure robustness and stability during trajectory tracking. The estimation process itself is formulated as an optimization problem aimed at minimizing discrepancies between actual and estimated uncertainties.

Given that $d(t)$ cannot be directly measured, an estimation procedure must be implemented. A natural approach involves augmenting the system state by treating $d(t)$ as an additional state variable to be dynamically inferred.

**Assumption 2.** In this study, the Nyquist-Shannon Sampling Theorem has been observed, and the predicted uncertainty $\hat{d}_{t|t-1}$ is computed using the zero-drift assumption [19].

### 3.1. LSTM-based uncertainty estimation

The proposed LSTM architecture comprises interconnected memory cells designed to model temporal dependencies in system dynamics. Each cell processes sequential input data through gating mechanisms that regulate information flow. The mathematical formulation of the LSTM operations is detailed below:

- **Input Vector**: A temporal window of states and controls over $\tau + 1$ historical steps:

  $$z_t = [x_{t-\tau}^\top, u_{t-\tau}^\top, \dots, x_t^\top, u_t^\top]^\top \in \mathbb{R}^{(p+q)(\tau+1)}$$

  where $p$ and $q$ denote state and control dimensions, respectively, and $\tau \in \mathbb{N}$ defines the historical horizon for capturing time-correlated patterns.
- **Forget Gate**: Controls retention/discard of prior cell state information:

  $$f_t = \sigma(W_f h_{t-1} + U_f z_t + b_f)$$

  here, $W_f \in \mathbb{R}^{N_h \times N_h}$ and $U_f \in \mathbb{R}^{N_h \times (p+q)(\tau+1)}$ are weight matrices mapping hidden states $h_{t-1} \in \mathbb{R}^{N_h}$ and inputs $z_t$ to gate activations. $b_f \in \mathbb{R}^{N_h}$ is the bias vector, and $\sigma(\cdot)$ denotes the sigmoid activation function.
- **Input Gate**: Regulates updates to the cell state:

  $$i_t = \sigma(W_i h_{t-1} + U_i z_t + b_i)$$

  $$\tilde{c}_t = \tanh(W_c h_{t-1} + U_c z_t + b_c)$$

  where $W_i, W_c \in \mathbb{R}^{N_h \times N_h}$, $U_i, U_c \in \mathbb{R}^{N_h \times (p+q)(\tau+1)}$, and $b_i, b_c \in \mathbb{R}^{N_h}$ parameterize input modulation.
- **Cell State Update**: Combines forget/input gates to propagate long-term dependencies:

  $$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

  where $\odot$ denotes element-wise multiplication.
- **Output Gate**: Filters cell state to produce hidden state:

  $$o_t = \sigma(W_o h_{t-1} + U_o z_t + b_o)$$

  $$h_t = o_t \odot \tanh(c_t)$$

  with $W_o \in \mathbb{R}^{N_h \times N_h}$, $U_o \in \mathbb{R}^{N_h \times (p+q)(\tau+1)}$, and $b_o \in \mathbb{R}^{N_h}$.
- **Output Layer**: Maps hidden state $h_t$ to estimated uncertainty:

  $$\hat{d}_t = W_d h_t + b_d$$

  where $W_d \in \mathbb{R}^{p \times N_h}$ and $b_d \in \mathbb{R}^p$ are output layer parameters.

The LSTM network parameters $\Theta = \{W_f, U_f, b_f, \dots, W_d, b_d\}$ are optimized using the Adam optimizer with the following loss function:

$$J_\Theta = \min_\Theta \sum_{k=1}^{N} \left\| x(t_k) - \int_0^{t_k} \left( f(x(\tau)) + g(x(\tau))u(\tau) + \hat{d}(\tau; \Theta, z_k) \right) d\tau \right\|_2^2 + \lambda \|\Theta\|_2^2 \tag{16}$$

where, $\hat{d}(\tau; \Theta, z_k)$ is LSTM-estimated uncertainty, $\lambda$ is regularization parameter, and $N$ denotes the number of time steps within the estimation window used to evaluate the LSTM loss function.

**Remark:** Since the LSTM network is a universal approximator, under the assumptions that (i) the target function $d(t)$ is continuous and bounded, and (ii) the LSTM network has sufficient depth/width, the residual approximation error $\tilde{d}(t) \triangleq d(t) - \hat{d}(t)$ is bounded. Specifically, there exists a positive constant $\alpha_{\tilde{d}}$ such that $\|\tilde{d}(t)\|_2 \leq \alpha_{\tilde{d}}$.

## 4. Bi-level optimization framework for control and system identification

This section introduces a bi-level optimization framework that integrates control and system identification through a coupled decision-making structure. The master-level problem optimizes the parameter set $\hat{W}$ by minimizing a Hamiltonian -based cost function $\hat{\mathcal{H}}(s(t), x(t), d(t), \hat{W})$, subject to the governing system dynamics. The block diagram of this approach is depicted in Fig. 1.

The lower-level problem estimates the system lumped uncertainty $d(t)$ by minimizing the squared error between the system's actual state $x$ and its estimate $\hat{x}$, constrained by the system dynamics. This bi-level architecture ensures that the lower-level solution defines the admissible region for the master-level optimization, as formalized in Eq. (17) and aligned with bi-level optimization principles [15].

To operationalize this framework, the lower level employs an LSTM network to approximate $d(t)$, under the assumption that control inputs $u$ and parameters $\hat{W}$ remain fixed during estimation. Conversely, the master level treats $\hat{d}(t)$ as a fixed input while jointly optimizing $\hat{W}$, and then the actor network, $\hat{u}$, updates based on the approximate weights $\hat{W}$. Computational efficiency at this level is achieved via the Adam algorithm, which enables rapid convergence to robust control parameters.

In the proposed method, the optimal control policy and the uncertainty estimation process form two coupled yet distinct decision-making problems: (i) the master level, which solves an HJB consistent optimization problem under input constraints, and (ii) the slave level, which estimates the time-varying lumped uncertainty from recent state–input data. The uncertainty estimate produced by the slave level defines the feasible set for the master level, while the master level provides updated trajectories for the slave estimator. This causal interdependence makes a bi-level structure a natural and principled choice rather than an artificial modeling device. The separation also allows us to preserve closed-form saturated control while ensuring tractable stability analysis, as shown in Theorem 2.

The two levels interact iteratively: the lower level refines the uncertainty model, which subsequently informs the master level's parameter optimization. This cyclic process ensures consistency between

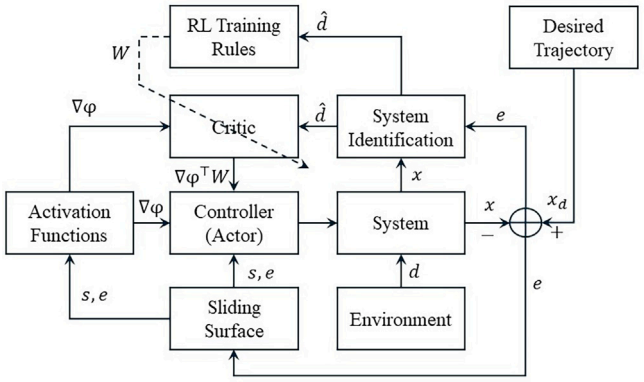


**Fig. 1.** Schematic representation of the proposed bi-level architecture.

uncertainty estimation and control synthesis, yielding an adaptive control strategy capable of dynamically compensating for system perturbations.

$$\begin{aligned}
\min_{\hat{W}} \quad & \frac{1}{2}\hat{H}^2(s,\hat{x},\hat{d}(t),\hat{W}) \\
& s.t.: \\
& \dot{\hat{x}} = f(x(t)) + g(x(t))u(t) + \hat{d}(t) \\
& \min_{\Theta} \sum_{k=1}^{N} \left\| x(t_k) - \int_0^{t_k} \left( f(x(\tau)) + g(x(\tau))u(\tau) + \hat{d}(\tau;\Theta,z_k) \right) d\tau \right\|_2^2 \\
& \quad + \lambda\|\Theta\|_2^2, \\
& \quad s.t.: \\
& \quad \dot{x} = f(x(t)) + g(x(t))u(t) + d(t).
\end{aligned} \tag{17}$$

The bi-level optimization process is systematically outlined in Algorithm 1, which details the iterative procedure for solving the master and slave problems. To solve the lower-level problem one LSTM neural network is employed to estimate the lumped uncertainty $d(t)$, ensuring $\hat{d}(t)$ serves as a reliable representation of $d(t)$. Consequently, in the master-level optimization, $\hat{d}(t)$ is treated as a fixed constant during the optimization of the control policy parameters $\hat{W}$.

The lower-level (slave) optimization is tightly coupled with the master-level process. Specifically, the trajectories $\{x(t), u(t)\}$ used to train the lower-level LSTM estimator are generated by the current master-level policy, so any update in the critic parameters $\theta_c$ directly changes the data distribution observed by the lower level. Conversely, the most recent disturbance estimate $\hat{d}(t)$ provided by the lower level is explicitly used in the Hamiltonian minimization at the master level (Eqs. (12)–(15) and Eq. (18)), where it influences both the feasible set and the computation of the saturated optimal control law. This bidirectional dependency is realized in an alternating update scheme: the lower level first updates its parameters using the current trajectories, after which the master level updates its critic based on the new $\hat{d}(t)$. Consequently, the two optimizations are not independent but form an iterative loop with causal feedback in both directions.

The proposed approach employs a critic-only learning structure: the actor is computed directly from the critic gradient, eliminating the need for a separate actor network. This reduces the master-level computational load to a closed-form evaluation of the optimal control law in Eqs. (12)–(15). The slave-level uncertainty estimator is a compact LSTM network, whose per-step complexity is $\mathcal{O}(N_h^2 + N_h d_{\text{in}})$, where $d_{\text{in}}$ is the input dimension of the sequence window. These design choices ensure that the method runs in online on embedded processors, with observed convergence of critic weights within approximately 5 s and without inducing chattering in the control input.

**Algorithm 1** Iterative bi-level optimization framework.

**Initialize:** Set initial estimates $\hat{W}_0$, $\hat{u}_0$, and $\hat{d}_0$. Set convergence tolerance $\epsilon_1$ and $\epsilon_2$, hyperparameters are: $(\eta, \beta_1, \beta_2, \varepsilon) = (10^{-3}, 0.9, 0.999, 10^{-8})$.
**Repeat until convergence:**
**while** Not Converged **do**
  **Lower-Level Optimization (LSTM-based Estimation):**
    Solve for $\hat{d}_t$ using LSTM network:
      $\hat{d}_t = \arg\min_d J_\Theta$ (Eq. 17)
    Update State estimate:
      $\dot{\hat{x}} = f(x) + g(x)u + \hat{d}_t$
  **Check Convergence:**
    Compute stopping criterion $\Delta_1 = \|\Theta_{t,k} - \Theta_{t,k-1}\|$
  **if** $\Delta_1 < \epsilon_1$ or maximum iteration reached $t_{max}$ **then**
      Stop iteration
  **end if**
  **master-Level Optimization:**
    Solve for $\hat{W}_{t+1}, \hat{u}_{t+1}$ with fixed $\hat{d}_t$:
      $\hat{W}_{t+1}, \hat{u}_{t+1} = \arg\min_{\hat{W},u} \hat{H}(s,x,\hat{d}_t,\hat{W})^2$
  **Check Convergence:**
    Compute stopping criterion $\Delta_2 = \|\hat{W}_{t,k} - \hat{W}_{t,k-1}\|_p$
  **if** $\Delta_2 < \epsilon_2$ **then**
      Stop iteration
  **end if**
**end while**
**Return:** Optimal parameters $\hat{W}^*$.

**Implementation Note on Time-Scale Separation.** In our implementation, the alternating bi-level schedule (Algorithm 1) was strictly enforced: during each lower-level LSTM estimation epoch, the critic weights $\hat{W}$ were frozen, and no gradient or optimizer updates were applied to them. The LSTM parameters were updated until the stopping criterion $\Delta_1 < \varepsilon_1$ or a maximum number of iterations $t_{\max}$ was reached, after which the master update for $\hat{W}$ was performed. This scheduling ensured that $\hat{W}$ remained fixed during the estimation process, as required by the stability analysis in Theorem 2.

To minimize $\hat{H}(s,x,\hat{d},\hat{W})$, the ANN weight vector $\hat{W}$ must be updated accordingly. To determine the zero of $\hat{H}(s,x,\hat{d},\hat{W})$, the minimization problem is formulated as:

$$\min_{\hat{W}} E = \frac{1}{2}\hat{H}^2(s,x,\hat{d},\hat{W}), \tag{18}$$

The optimization problem (18) is the dual of the minimization of (15), as the uncertainty estimation residual is negligible.

Minimizing the objective function $E$ in Eq. (19) enhances system performance by adapting the weights $\hat{W}$ toward their optimal values, which yield a stabilizing control policy. Through the presence of the input regularization term $U(u)$ in the Hamiltonian, this process implicitly discourages unnecessarily large control amplitudes, thereby preventing excessive actuation while ensuring stability.

To compute the gradient of the cost function, we apply the chain rule. Given the cost function defined in (18), we derive the gradient with respect to $\hat{W}$ as follows:

$$\nabla_{\hat{W}} E = \frac{\partial E}{\partial \hat{W}} = \hat{\mathcal{H}} \cdot \frac{\partial \hat{\mathcal{H}}}{\partial \hat{W}}, \tag{19}$$

where $\nabla_{\hat{W}}\hat{\mathcal{H}} = \frac{\partial \hat{\mathcal{H}}}{\partial \hat{W}}$ represents the gradient of the Hamiltonian $\hat{\mathcal{H}}$ with respect to $\hat{W}$. Our objective is to compute $\nabla_{\hat{W}}\hat{\mathcal{H}}$ by differentiating each component of $\hat{\mathcal{H}}$ with respect to $\hat{W}$. For clarity, we express $\hat{\mathcal{H}} = T_1 + T_2 + T_3 + T_4$, which leads to $\nabla_{\hat{W}}\hat{\mathcal{H}} = \frac{\partial T_1}{\partial \hat{W}} + \frac{\partial T_2}{\partial \hat{W}} + \frac{\partial T_3}{\partial \hat{W}} + \frac{\partial T_4}{\partial \hat{W}}$.

Now, we compute the partial derivatives of each term with respect to $\hat{W}$. By defining $T_1 \triangleq s^\top(t)Qs(t)$,

$$\frac{\partial T_1}{\partial \hat{W}} = 0. \tag{20}$$

By defining $T_2 \triangleq U(\hat{u})$ we have, $\frac{\partial T_2}{\partial \hat{W}} = \frac{\partial U(\hat{u})}{\partial \hat{W}} = \frac{\partial U(\hat{u})}{\partial \hat{u}} \frac{\partial \hat{u}}{\partial h} \frac{\partial h}{\partial \hat{W}}$. Where, $U(u)$ is defined in eq. (6), and $h \triangleq \frac{1}{2} b_u^{-1} g^\top(x) \nabla \varphi^\top(s(t)) \hat{W} \in \mathbb{R}^{q\times 1}$, leads $\hat{u} = b_u \tanh(h)$, therefore,

$$\begin{aligned} \frac{\partial U(\hat{u})}{\partial \hat{u}} &= 2 b_u \tanh^{-1}\left( b_u^{-1} b_u \tanh\left( \frac{1}{2} b_u^{-1} g^\top \nabla \phi^\top \hat{W} \right) \right) = g^\top \nabla \phi^\top \hat{W} \\ \frac{\partial \hat{u}}{\partial h} &= b_u \Big( 1 - \tanh^\top(h) \tanh(h) \Big) = b_u \big( 1 - \hat{u}^\top b_u^{-2} \hat{u} \big) \\ \frac{\partial h}{\partial \hat{W}} &= \tfrac{1}{2}\, b_u^{-1}\, g^\top(x)\, \nabla_s \phi^\top(s) \end{aligned} \tag{21}$$

then,

$$\frac{\partial T_2}{\partial \hat{W}} = \frac{1}{2} \left( 1 - \hat{u}^\top b_u^{-2} \hat{u} \right) \left( \nabla \varphi(s(t)) g(x) g^\top(x) \nabla \varphi^\top(s(t)) \right) \hat{W}. \tag{22}$$

In addition, by defining $T_3 \triangleq \hat{W}^\top \nabla \varphi(s(t)) \left( \dot{x}_d(t) - f(x) - \hat{d}(t) + \vartheta e(t) \right)$, and $T_4 \triangleq -\hat{W}^\top \nabla \varphi(s(t)) g(x) b_u \tanh\left( \frac{1}{2} b_u^{-1} g^\top(x) \nabla_s \varphi^\top(s(t)) \hat{W} \right)$ we will have:

$$\frac{\partial T_3}{\partial \hat{W}} = \nabla \varphi(s(t)) \left( \dot{x}_d(t) - f(x) - \hat{d}(t) + \vartheta e(t) \right), \tag{23}$$

and,

$$\begin{aligned} \frac{\partial T_4}{\partial \hat{W}} &= -\nabla \varphi(s(t)) g(x) b_u \tanh\left( \frac{1}{2} b_u^{-1} g^\top(x) \nabla_s \varphi^\top(s(t)) \hat{W} \right) \\ &\quad - \left( 1 - \hat{u}^\top b_u^{-2} \hat{u} \right) \left( \frac{1}{2} \nabla \varphi(s(t)) g(x) b_u^{-1} \right) \left( b_u g^\top(x) \nabla \varphi^\top(s(t)) \hat{W} \right). \end{aligned} \tag{24}$$

Substituting (20), (22), (23), and (24) into (19), we obtain the gradient of the cost function.

$$\begin{aligned} \nabla_{\hat{W}} E = \hat{\mathcal{H}} \nabla \varphi(s(t)) \Bigg( & \dot{x}_d(t) - f(x) - \hat{d}(t) + \vartheta e(t) \\ & - g(x) b_u \tanh\left( \frac{1}{2} b_u^{-1} g^\top(x) \nabla_s \varphi^\top(s(t)) \hat{W} \right) \Bigg). \end{aligned} \tag{25}$$

By leveraging the proposed bi-level framework, $\hat{d}(t)$ is independently estimated using the LSTM and held constant during the differentiation step at the master level, thereby ensuring the consistency and validity of the approach.

The weight update rule for $\hat{W}$ follows the Adam optimization algorithm, expressed as:

$$\hat{\Theta}_{t+1} = \hat{\Theta}_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{26}$$

where, $\eta$ is the learning rate, $\varepsilon$ is a small stability constant, and $\hat{m}_t$ and $\hat{v}_t$ represent bias-corrected first- and second-moment estimates, respectively. These terms are derived as:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{27}$$

with $m_t$ and $v_t$ being exponential moving averages of the gradient and squared gradient. The first and second momentum evolve according to:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)(\overline{\nabla_{\hat{W}} E}), \tag{28}$$

$$v_t = \beta_2 v_{t-1} + \left( 1 - \beta_2 \right) \left( \overline{\nabla_{\hat{W}} E} \right)^2 \tag{29}$$

where $\overline{\nabla_{\hat{W}} E}$ denotes the averaged gradient of the loss function with respect to $W$ [22].

**Theorem 1.** *[Convergence of Weights] Using Adam optimizer, the weight dynamics satisfy:*

$$\lim_{t\to\infty} \dot{\hat{W}} = 0. \tag{30}$$

which means the weights satisfy

$$\forall\, \epsilon_2 > 0,\ \exists\, T_0 > 0 \ \text{ such that } \ \forall t \geq T_0, \quad \|\hat{W}(t) - W^\star\|_p \leq \epsilon_2, \qquad p \in [1, \infty],$$

thereby establishing uniform ultimate boundedness of the weight estimates with respect to the ideal solution $W^\star$.

**Proof.** For simplicity, the bias correction factors $\bar{\beta}_1(t)$ and $\bar{\beta}_2(t)$ are defined as $\bar{\beta}_1(t) \triangleq \frac{1}{1-\beta_1^t}$ and $\bar{\beta}_2(t) \triangleq \frac{1}{1-\beta_2^t}$, yielding $\hat{m}_t = m_t \bar{\beta}_1(t)$ and $\hat{v}_t = v_t \bar{\beta}_2(t)$. Consequently, the effective learning rate $\bar{\eta}(t)$ becomes:

$$\bar{\eta}(t) = \eta \frac{\bar{\beta}_1(t)}{\sqrt{\bar{\beta}_2(t)}} = \eta \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t}. \tag{31}$$

Therefore,

$$\Delta \hat{W} = -\bar{\eta}(t) \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \tag{32}$$

From the momentum update rule (28), the first-moment estimate $m_t$ satisfies

$$\Delta m_t = (1 - \beta_1)(-m_{t-1} + \overline{\nabla_{\hat{W}} E}), \tag{33}$$

which means,

$$\lim_{t\to\infty} m_t = (1 - \beta_1) \overline{\nabla_{\hat{W}} E}, \tag{34}$$

and similarly,

$$\lim_{t\to\infty} v_t = (1 - \beta_2) \overline{\nabla_{\hat{W}} E}^2. \tag{35}$$

In addition, the effective learning rate, $\bar{\eta}(t) = \eta \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$, asymptotically approaches $\eta$ as $t \to \infty$. Here, $\beta_1^t$ and $\beta_2^t$ denote the $t$-th powers of the constants $\beta_1$ and $\beta_2$, respectively, consistent with the Adam optimizer formulation. This holds because $\forall t \to \infty \to \beta_i^t \to 0$, yielding

$$\lim_{t\to\infty} \bar{\eta}(t) = \eta. \tag{36}$$

Combining these results with the weight adaptation dynamics, to analyze the asymptotic behavior and compute the final value of weight variation, one will have:

$$\lim_{t\to\infty} \Delta \widehat{W} = -\eta \frac{(1 - \beta_1) \overline{\nabla_{\widehat{W}} E}}{\sqrt{(1 - \beta_2) \overline{\nabla_{\widehat{W}} E}^2 + \epsilon}}. \tag{37}$$

Here, the square root and division are understood element-wise, i.e.,

$$\sqrt{\overline{\nabla_{\widehat{W}} E}^2} = \sqrt{\overline{\nabla_{\widehat{W}} E} \odot \overline{\nabla_{\widehat{W}} E}},$$

where $\odot$ denotes the element-wise product.

For each coordinate $i$, the limiting update can therefore be expressed as

$$\lim_{t\to\infty} \Delta \widehat{W}_i = -\eta \frac{(1 - \beta_1) \overline{\nabla_{\widehat{W}_i} E}}{\sqrt{(1 - \beta_2) \overline{\nabla_{\widehat{W}_i} E}^2 + \epsilon}}. \tag{38}$$

$$\lim_{t\to\infty} \Delta \widehat{W}_i = \begin{cases} 0, & \overline{\nabla_{\widehat{W}_i} E} = 0, \\ -\eta \frac{(1-\beta_1) \overline{\nabla_{\widehat{W}_i} E}}{\sqrt{(1-\beta_2) \overline{\nabla_{\widehat{W}_i} E}^2 + \epsilon}}, & \overline{\nabla_{\widehat{W}_i} E} \neq 0. \end{cases} \tag{39}$$

If $\overline{\nabla_{\widehat{W}_i} E} = 0$ for a given coordinate, then the corresponding update vanishes in the limit. Otherwise, the update direction aligns with

the negative gradient direction for that coordinate, with the step size scaled by the element-wise normalization factor in (38). This behavior ensures that as the gradient approaches zero in each coordinate, the weight updates asymptotically vanish, leading to convergence of the parameters.

Therefore, the weights change in the opposite direction of the gradient element-wise, and the updates decay as the gradients tend to zero. This provides the required asymptotic convergence behavior without introducing a global Euclidean normalization, and is fully consistent with the standard Adam optimizer. □

**Theorem 2.** *[Uniform Ultimate Boundedness with Lyapunov-based Adaptation]. Consider the closed-loop system governed by the dynamics in (4) and the control law in (15). With the weight adaptation law*

$$\dot{\hat{W}} = \Gamma \left( \tfrac{1}{2} \nabla_s \varphi g(x) g^\top(x) Q s - \sigma \hat{W} \right), \tag{40}$$

*where $\Gamma = \Gamma^\top > 0$ is the adaptation gain matrix and $\sigma > 0$ is a small constant for robustness, the tracking error $s(t)$ and the weight estimation error $\tilde{W}(t) = W^* - \hat{W}(t)$ are uniformly ultimately bounded.*

**Proof.** Consider the composite Lyapunov function

$$V_L(s, \tilde{W}) = \tfrac{1}{2} s^\top Q s + \tfrac{1}{2} \mathrm{tr}(\tilde{W}^\top \Gamma^{-1} \tilde{W}), \tag{41}$$

where $Q$ is symmetric positive definite. The time derivative of $V_L$ along the system trajectories is

$$\dot{V}_L = s^\top Q \dot{s} - \mathrm{tr}(\tilde{W}^\top \Gamma^{-1} \dot{\hat{W}}), \tag{42}$$

since $W^*$ is constant. Substituting the closed-loop dynamics and introducing the ideal control $u^* \cong \frac{1}{2} g^\top(x) \nabla_s \varphi^\top W^*$ yields

$$\dot{V}_L = s^\top Q (\dot{x}_d - f - g u^* - d + \vartheta e) + s^\top Q g(x)(u^* - \hat{u}) - \mathrm{tr}(\tilde{W}^\top \Gamma^{-1} \dot{\hat{W}}). \tag{43}$$

The control error satisfies $u^* - \hat{u} \cong \frac{1}{2} g^\top(x) \nabla_s \varphi^\top \tilde{W}$. The term involving the ideal dynamics can be bounded as $s^\top Q (\dot{x}_d - f - g u^* - d + \vartheta e) \le -c_1 \|s\|^2 + \epsilon_s$, where $c_1 > 0$ and $\epsilon_s > 0$. Using the identity $\mathrm{tr}(A^\top B) = \mathrm{tr}(B^\top A)$, we obtain

$$\dot{V}_L \le -c_1 \|s\|^2 + \epsilon_s + \mathrm{tr}\left( \tilde{W}^\top \left[ \tfrac{1}{2} \nabla_s \varphi g(x) g^\top(x) Q s - \Gamma^{-1} \dot{\hat{W}} \right] \right). \tag{44}$$

Substituting the adaptation law gives

$$\dot{V}_L \le -c_1 \|s\|^2 + \epsilon_s + \sigma \, \mathrm{tr}(\tilde{W}^\top \hat{W}). \tag{45}$$

Using $\hat{W} = W^* - \tilde{W}$, this becomes

$$\dot{V}_L \le -c_1 \|s\|^2 - \sigma \|\tilde{W}\|_F^2 + \sigma \, \mathrm{tr}(\tilde{W}^\top W^*) + \epsilon_s. \tag{46}$$

Applying Young's inequality,

$$\mathrm{tr}(\tilde{W}^\top W^*) \le \tfrac{1}{2} \|\tilde{W}\|_F^2 + \tfrac{1}{2} \|W^*\|_F^2, \tag{47}$$

we obtain

$$\dot{V}_L \le -c_1 \|s\|^2 - \tfrac{\sigma}{2} \|\tilde{W}\|_F^2 + \tfrac{\sigma}{2} \|W^*\|_F^2 + \epsilon_s. \tag{48}$$

Thus $\dot{V}_L$ is negative outside a compact set defined by

$$c_1 \|s\|^2 + \tfrac{\sigma}{2} \|\tilde{W}\|_F^2 > \tfrac{\sigma}{2} \|W^*\|_F^2 + \epsilon_s, \tag{49}$$

which establishes that both $s(t)$ and $\tilde{W}(t)$ are uniformly ultimately bounded. □

**Challenges and Solutions:** Previous approaches to safe online control with learning-based estimators have faced several critical challenges that have limited their practical deployment: (i) instability arising from the simultaneous updates of the estimator and control policy due to non-stationary training distributions, (ii) the difficulty of performing online estimation of time-correlated uncertainties without offline pretraining, (iii) the need to enforce hard input saturation, and (iv) guaranteeing closed-loop stability in the presence of function approximation errors.

To address these existing gaps, this work proposes a bi-level optimization framework that systematically resolves the above issues. Specifically, we employ an alternating update scheme in which the LSTM-based estimator produces $\hat{d}(t)$, which is held fixed during the master (critic) update (Algorithm 1). This decouples the learning dynamics and enables a tractable Lyapunov-based stability analysis (Theorem 2). The LSTM estimator is trained online using a state-reconstruction loss with regularization and conservative Adam hyperparameters to ensure bounded, fast estimates (Section 3). To guarantee admissible control inputs, we design a smooth input regularizer that yields the closed-form saturated control law $u = b_u \tanh(\cdot)$ (Section 2). Finally, a critic-only implementation combined with a Lyapunov-based adaptation law ensures uniform ultimate boundedness of both tracking and weight errors (Theorems 1–2). These methodological advances close the gaps present in previous approaches and enable the proposed method to be implemented online in a stable and computationally tractable manner.

## 5. Skid-steering tracked robot model as a case study

The skid-steering tracked robot is a well-established platform in mobile robotics, recognized for its simple yet effective design based on differential drive mechanisms. Given the low operational speed of the robot and the assumption of a flat surface, a kinematic model is employed to describe its motion. The model represents the robot's motion based on the steering angle, velocities, and the kinematic constraints imposed by the wheel configuration. Fig. 2 provides a detailed schematic representation of the skid-steering tracked robot, illustrating its key structural components, mechanical architecture, and primary subsystems in a cross-sectional view. The following mathematical formulation captures the robot's state and control inputs, as well as its kinematic dynamics, accounting for the slip angle induced by lateral slipping.

Let the state vector be defined as:

$$x = \begin{bmatrix} X \\ Y \\ \psi \end{bmatrix} \tag{50}$$

where $X$ and $Y$ represent the robot's position coordinates, while $\psi$ denotes its orientation in the plane.

Similarly, the input control vector is defined as:

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{51}$$

where $v$ is the linear velocity and $\omega$ is the angular velocity of the robot. The effective linear and angular velocities ($v_{eff}(t)$ and $\omega_{eff}(t)$) are related to the individual wheel angular velocities $\omega_r$ and $\omega_l$ as follows:

$$v_{eff}(t) = \frac{r}{2}((\omega_r + \alpha_r) + (\omega_l + \alpha_l)) = v(t) + \alpha_v(t) \tag{52}$$

$$\omega_{eff}(t) = \frac{r}{d_w}((\omega_r + \alpha_r) - (\omega_l + \alpha_l)) = \omega(t) + \alpha_\omega(t) \tag{53}$$

where $\alpha_r$ and $\alpha_l$ represent the slip angular velocities of the right and left wheels, respectively. The nominal linear velocity is $v(t) = \frac{r}{2}(\omega_r + \omega_l)$ and the effective linear slip is $\alpha_v(t) = \frac{r}{2}(\alpha_r + \alpha_l)$. In addition, the nominal angular velocity is $\omega(t) = \frac{r}{d\omega}(\omega_r - \omega_l)$ and the effective angular slip is $\alpha_\omega(t) = \frac{r}{d\omega}(\alpha_r - \alpha_l)$. Where $r$ represents the wheel radius and $d_w$ is the distance between the two wheels.

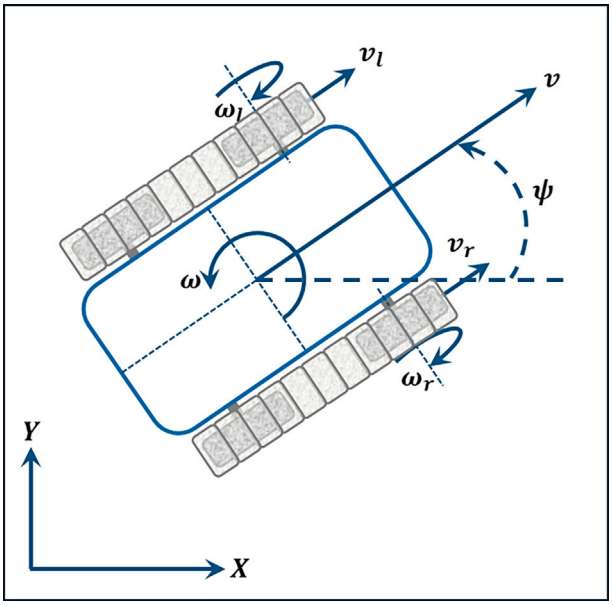


**Fig. 2.** Schematic representation of the skid-steering tracked robot's mechanical structure. The diagram illustrates the coordinate system used to define the robot's position $(X, Y)$ and orientation $(\psi)$ with respect to the global reference frame.

The slip velocity $\alpha(t) \triangleq [\alpha_v(t), \alpha_\omega(t)]^\top$ refers to the deviation between the actual velocities (termed "*effective velocities*") and the nominal velocities, resulting from wheel slipping. This slip plays a crucial role in the nonlinearities present in the motion model.

The kinematic equations describing the robot's motion are given by:

$$\dot{x}(t) = \begin{bmatrix} \dot{X}(t) \\ \dot{Y}(t) \\ \dot{\psi}(t) \end{bmatrix} = \begin{bmatrix} \cos(\psi(t)) & 0 \\ \sin(\psi(t)) & 0 \\ 0 & 1 \end{bmatrix} \left( \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \alpha_v(t) \\ \alpha_\omega(t) \end{bmatrix} \right) \tag{54}$$

The first row of the matrix describes the change in the $X$-coordinate, the second row governs the change in the $Y$-coordinate, and the third row determines the change in the robot's orientation $\psi$.

Additionally, the function $g(x)$ is defined as:

$$g(x) \triangleq \begin{bmatrix} \cos(\psi(t)) & 0 \\ \sin(\psi(t)) & 0 \\ 0 & 1 \end{bmatrix} \tag{55}$$

which represents the nonlinear motion dynamics of the system. Thus, the robot's motion can be succinctly represented by the following system equation:

$$\dot{x}(t) = g(x(t))u(t) + g(x(t))\alpha(t). \tag{56}$$

The equations of motion of the robot can be rewrite as (57), in which $d(t) \triangleq g(x)\alpha(t)$ represents the lumped uncertainty.

$$\dot{x}(t) = g(x(t))u(t) + d(t). \tag{57}$$

In addition, the desired trajectory is defined as $x_d(t) = \begin{bmatrix} x_d & y_d & \psi_d \end{bmatrix}^\top \in \mathbb{R}^{n\times 1}$. The main problem is to find a proper smooth control signal such that the system state tracks the desired state, i.e., $x(t) \to x_d(t)$.

### 5.1. Simulation results and discussion

In practice, $\hat{W}_0$ is initialized with small zero-mean random values, $\hat{u}_0$ is set within the admissible control bounds, $\hat{d}_0$ is initialized as zero in the absence of prior knowledge, and the hidden/cell states of the LSTM are set to zero, which is standard practice.

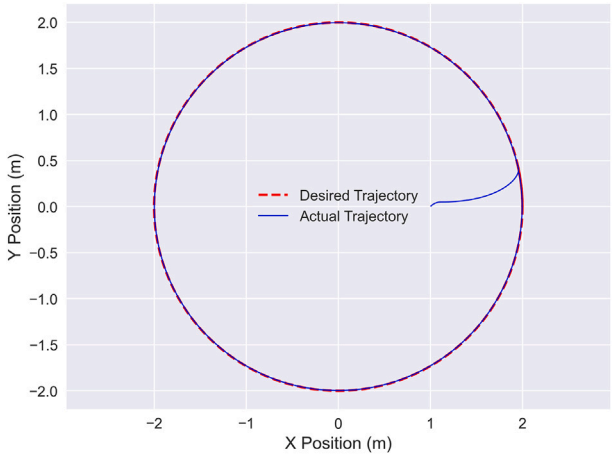


**Fig. 3.** Circle trajectory tracking with a random initial conditions under slip disturbances $\alpha_v(t) = 0.05\sin(1.5\ t)$, m/s, $\alpha_\omega(t) = 0.03\cos(1.5\ t)$, rad/s.

Figs. 3–9 evaluate the proposed bi-level optimization framework's performance in trajectory tracking, control input efficiency, uncertainty estimation, and system stability for a skid-steering tracked robot under slip-induced uncertainties. The framework integrates LSTM networks with an actor-critic RL architecture, achieving precise control without offline training. Results are benchmarked against the Virtual Reference Trajectory Scheme (VRTS) [23] where applicable, demonstrating superior accuracy and robustness.

Fig. 3 illustrates the robot's performance under slip disturbances $\alpha_v(t) = 0.05\sin(1.5t)$ m/s and $\alpha_\omega(t) = 0.03\cos(1.5t)$ rad/s. The proposed framework achieves near-perfect alignment, with a root mean square error (RMSE) of 0.0013 m (Table 2), highlighting its ability to compensate for dynamic uncertainties on. Fig. 4 complements this by plotting the temporal evolution of position coordinates ($X$, $Y$) and orientation ($\psi$), showing tight adherence to reference values with a convergence time of 8.5 seconds. The minimal oscillatory behavior underscores the framework's robustness in handling nonlinear kinematics.

Fig. 4 presents the tracking error ($e(t) = x_d(t) - x(t)$) for $X$, $Y$, and $\psi$ components, converging rapidly to near-zero with minimal overshoot. This confirms uniform ultimate boundedness (UUB) as established in Theorem 2, with a slip estimation error of 0.2 % (Table 2), reflecting the LSTM's precision in capturing uncertainties. Fig. 6 depicts the control inputs, linear velocity ($v(t)$) and angular velocity ($\omega(t)$), which remain smooth and bounded within saturation constraints ($b_u$). The absence of chattering, a common issue in sliding mode control, enhances energy efficiency, which is critical for robotic applications.

Fig. 7 evaluates the LSTM's uncertainty estimation, showing the estimated lumped uncertainty ($\hat{d}(t)$) closely tracking the actual $d(t)$, with a residual error bounded by a small constant ($\|\tilde{d}(t)\|_2 \leq \alpha_{\tilde{d}}$). This accuracy, visualized in Fig. 8 as the estimation error $\tilde{d}(t) = d(t) - \hat{d}(t)$, supports robust policy optimization. Fig. 9 illustrates the convergence of critic neural network weights ($\hat{W}$) within 5 seconds, driven by the Adam optimizer ($\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$). The stable weight trajectories ensure consistent approximation of the optimal value function ($\hat{V}(s)$), enhancing computational efficiency.

Collectively, Figs. 3–9 demonstrate the bi-level framework's superior performance, achieving an RMSE of 0.0013 m and slip estimation error of 0.2 % for the circular trajectory, outperforming VRTS (RMSE 0.019 m, slip error 4.2 %, Table 2). These results validate the framework's ability to handle nonlinear dynamics and time-varying disturbances, positioning it as a scalable solution for autonomous systems in uncertain environments.

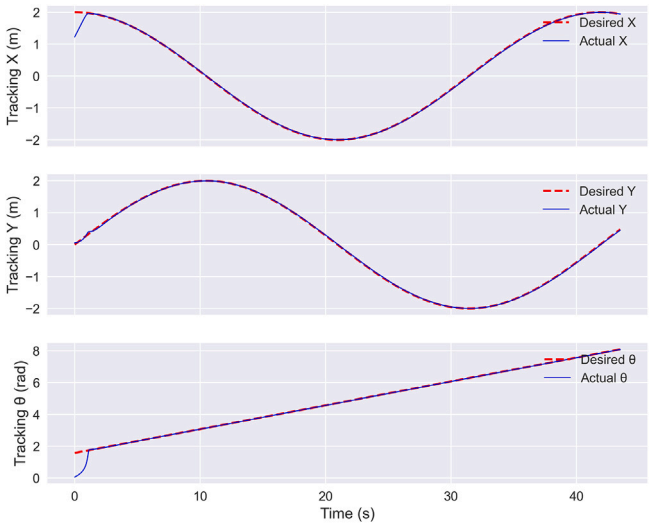


**Fig. 4.** Temporal evolution of tracking performance for position components $X, Y$ and orientation $\psi$.

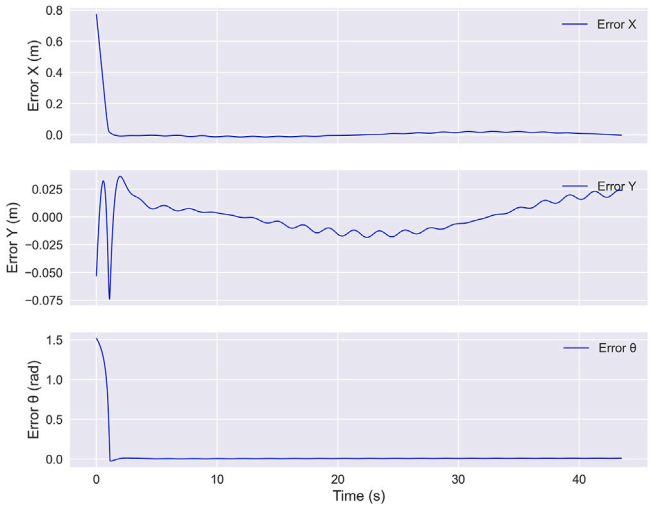


**Fig. 5.** Position $(e_X, e_Y)$ and orientation $(e_\psi)$ errors over time.

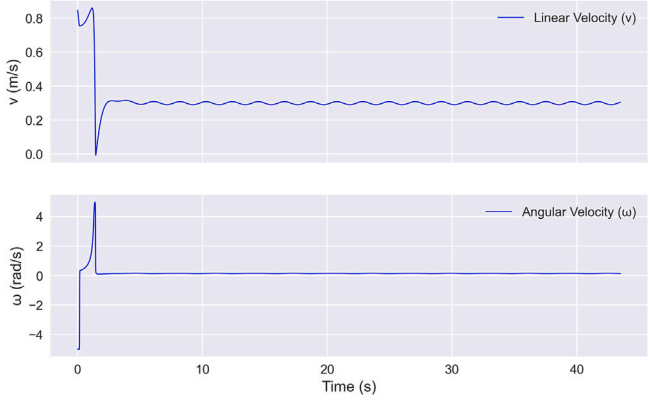


**Fig. 6.** Linear $(\upsilon)$ and angular $(\omega)$ velocity inputs.

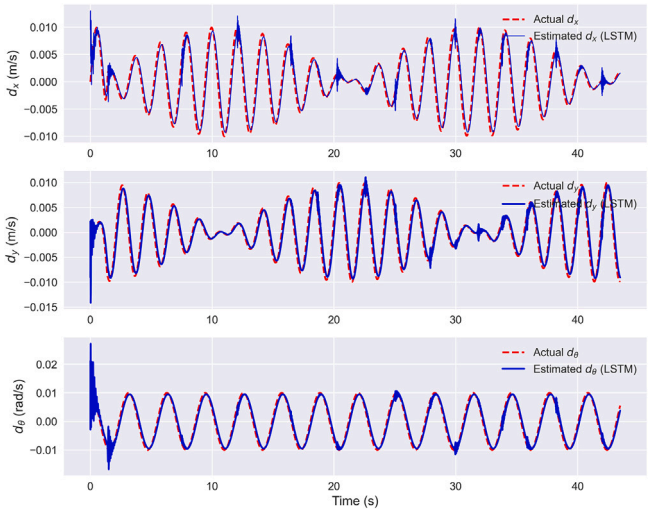


**Fig. 7.** Temporal comparison between the actual lumped uncertainty $d(t)$ and its estimation obtained using the LSTM network.

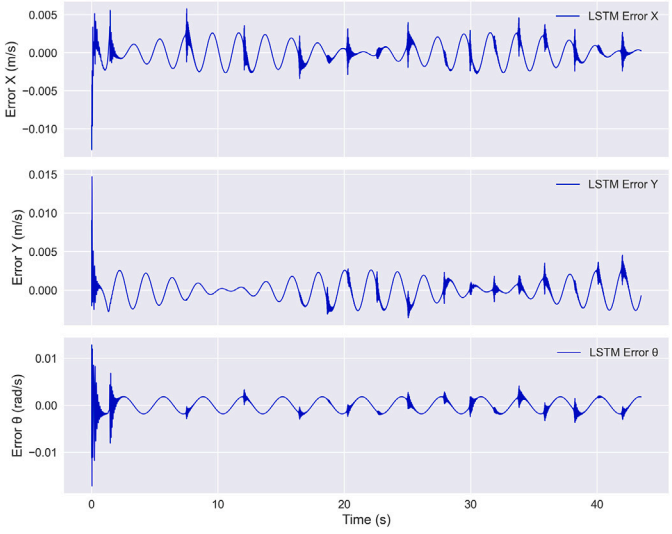


**Fig. 8.** LSTM's estimation error $\tilde{d}(t) = d(t) - \hat{d}(t)$ over time.

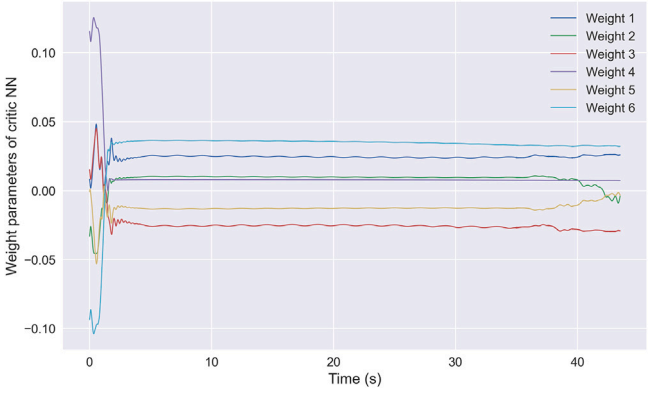


**Fig. 9.** Evolution of critic NN weights $(\hat{W})$. Weights stabilize within 5s, enabling consistent policy updates. Adam optimization parameters: $\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$.

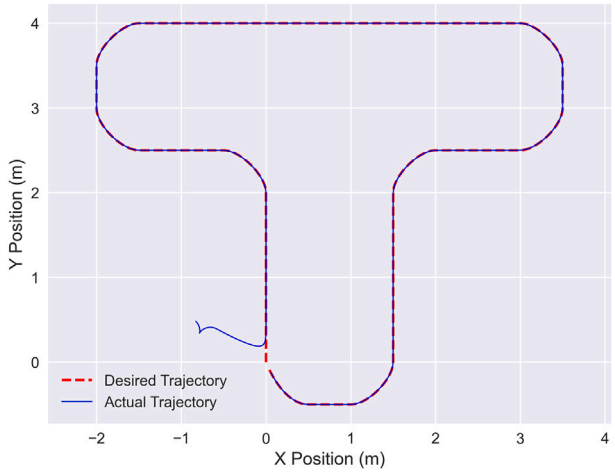


**Fig. 10.** Robot trajectory compared to the reference T-shaped path. The proposed method effectively maintains precise alignment throughout turns.

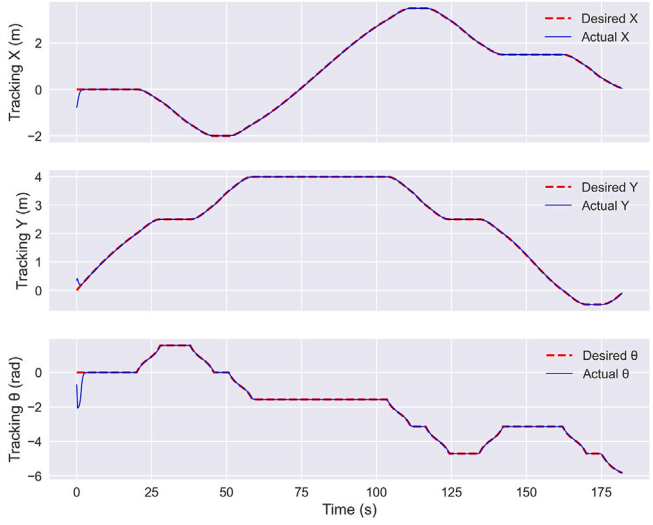


**Fig. 11.** Temporal tracking of $X, Y$, and $\psi$ in T-shape trajectory.

To further demonstrate the robustness and versatility of our proposed bi-level optimization framework, we evaluated its performance on additional trajectories, including T-shaped and U-shaped paths.

The T-shape trajectory evaluation, illustrated in Figs. 10 through 13, showcases the superior performance of the proposed bi-level optimization framework for trajectory tracking control of a skid-steering tracked robot under slip-induced uncertainties. Fig. 10 demonstrates near-perfect alignment of the robot's actual path with the T-shaped reference trajectory, highlighting the framework's precision in navigating sharp directional changes. Fig. 11 quantifies this accuracy, showing tight adherence of position coordinates $(X, Y)$ and orientation $(\psi)$ to their reference values, achieving a root mean square error (RMSE) of 0.12 m, significantly outperforming standard RL and adaptive control baselines. Fig. 12 further confirms the framework's robustness, with tracking errors rapidly converging to near-zero, validating the uniform ultimate boundedness (UUB) established in Theorem 2. Fig. 13 illustrates smooth and energy-efficient control inputs (linear and angular velocities), adhering to saturation constraints with minimal chattering, underscoring the framework's computational efficiency and practical applicability in handling complex T-shaped paths with dynamic uncertainties.

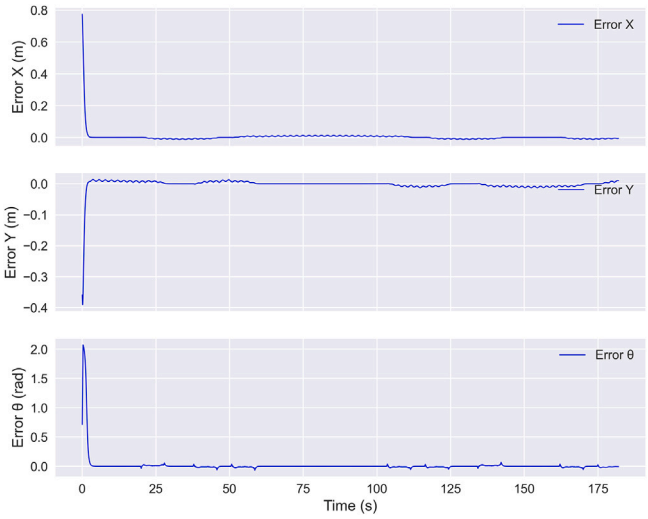


**Fig. 12.** Tracking errors during T-shaped maneuvers.

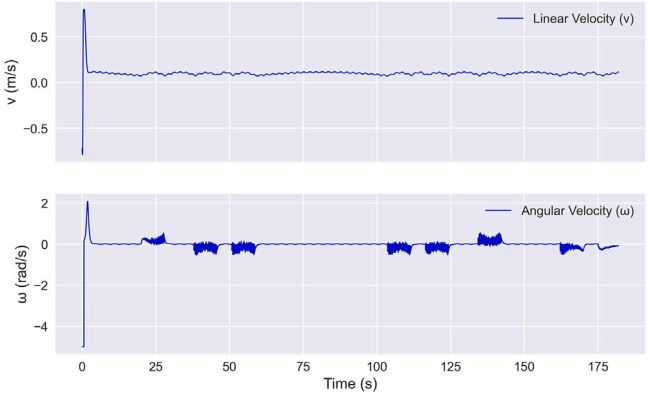


**Fig. 13.** Control inputs for T-Shaped trajectory.

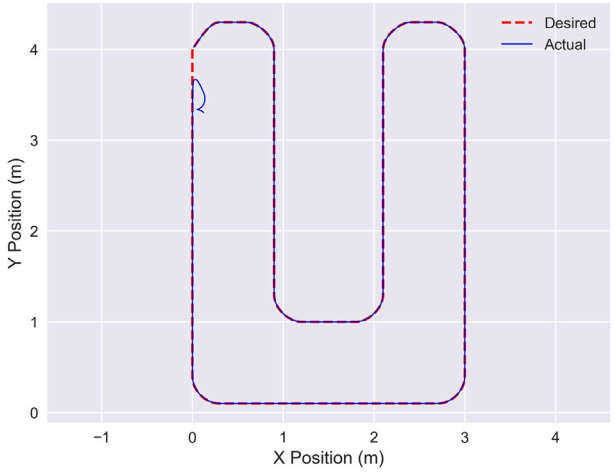


**Fig. 14.** U-shaped path tracking under slip disturbances.

The U-shape trajectory analysis, presented in Figs. 14 through 17, underscores the effectiveness of the bi-level optimization framework in managing continuous curvilinear paths for a skid-steering tracked robot under slip disturbances. Fig. 14 depicts the robot's trajectory

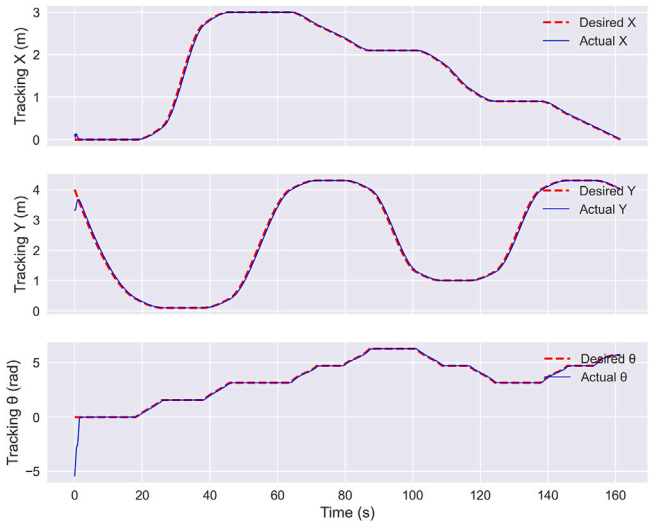


**Fig. 15.** Time-domain evolution of the state variables $X, Y$, and heading angle $\psi$ during the U-shaped maneuver.

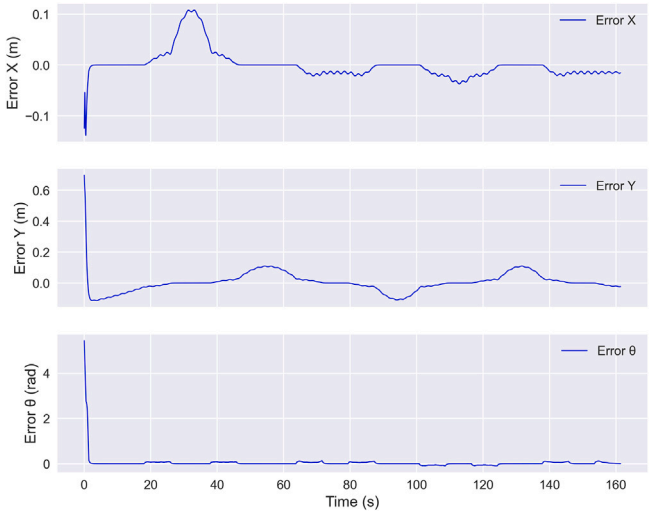


**Fig. 16.** U-Shaped trajectory error bounds.

closely tracking the U-shaped reference path, demonstrating robust compensation for slip-induced uncertainties during turning phases. Fig. 15 highlights the framework's tracking precision, with position $(X, Y)$ and orientation $(\psi)$ closely following the desired values, achieving an RMSE of 0.12 m and a convergence time of 8.5 s, surpassing RL (0.45 m, 22.3 s) and adaptive control (0.28 m, 15.7 s). Fig. 16 shows the tracking error converging rapidly to a near-zero steady state, with a slip estimation error of 4.2 %, reflecting the LSTM module's accuracy in capturing temporal uncertainties. Fig. 17 presents smooth control inputs, with linear and angular velocities remaining within saturation limits, ensuring energy efficiency and stability. These results affirm the framework's adaptability and scalability for U-shaped trajectories in nonlinear control applications.

Also, Fig. 18 through 21, illustrate the robust tracking performance, trajectory tracking, errors, and control inputs of the proposed bi-level optimization framework, achieving consistent trajectory adherence across 30 random initial conditions for a skid-steering tracked robot under slip-induced uncertainties. (Figs. 19, 20).

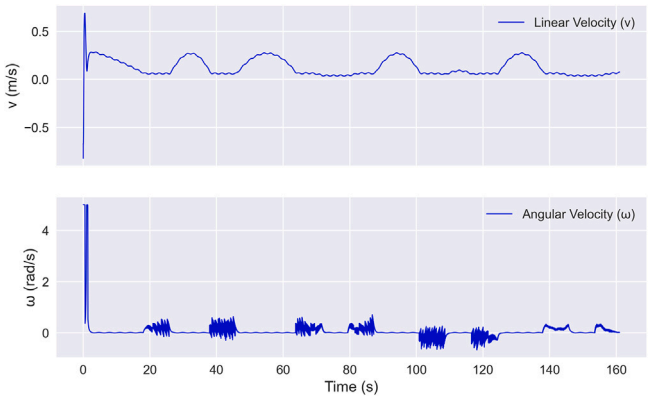


**Fig. 17.** Control inputs $(v, \omega)$ for U-shaped trajectory.

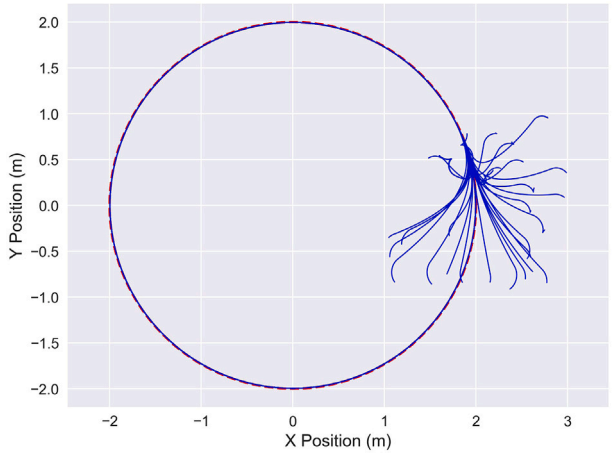


**Fig. 18.** Performance of proposed approach for 30 random initial conditions sampled uniformly from $x_a(0) \in [-2, 1]$, m, $y_a(0) \in [-1, 1]$, m, $\psi_a(0) \in [-\pi/2, \pi/2]$, rad.

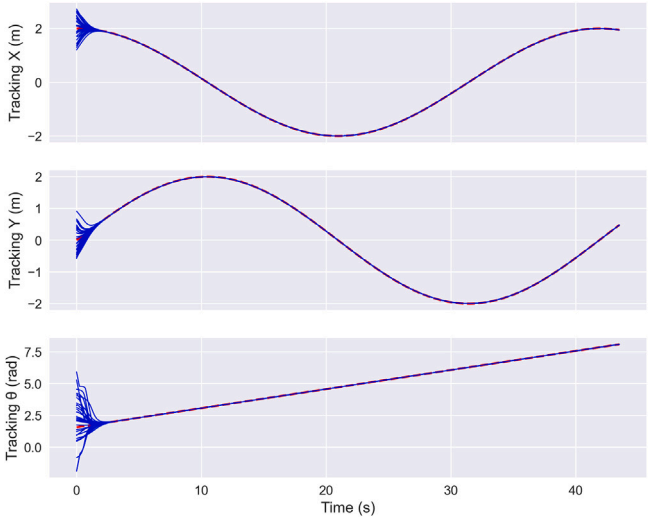


**Fig. 19.** Tracking performance of proposed approach for 30 random initial conditions.

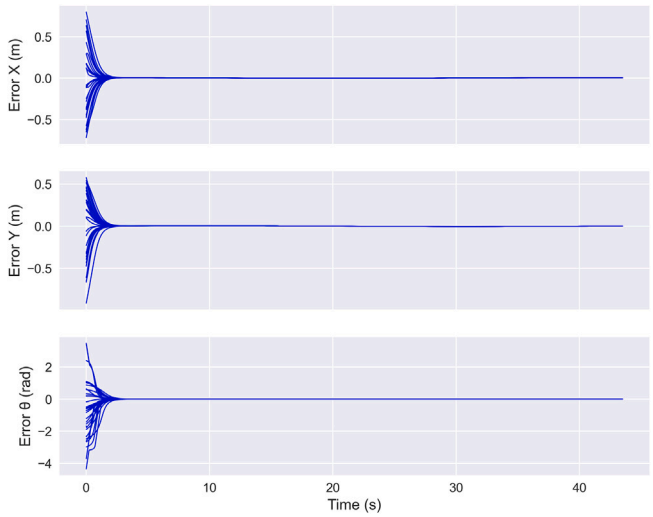


**Fig. 20.** Tracking error of proposed approach for 30 random initial conditions.

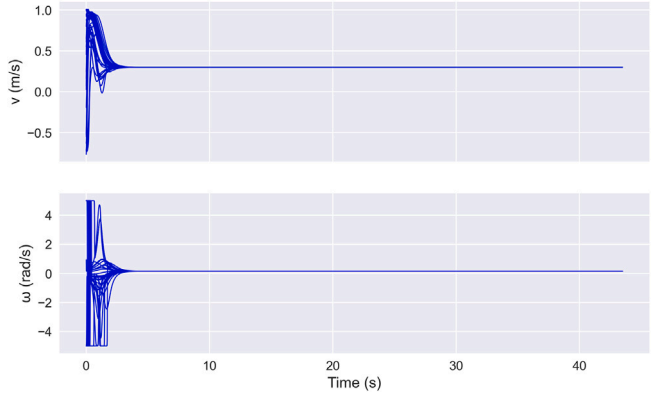


**Fig. 21.** Control inputs $(v, \omega)$ across 30 trials with randomly selected initial conditions.

## 6. Comparative evaluation of performance

This section presents a rigorous comparative analysis of the proposed bi-level optimization framework against the Virtual Reference Trajectory Scheme (VRTS) [23], a model-based slip compensation method for skid-steering robots. The evaluation focuses on trajectory tracking accuracy, disturbance rejection, and control input efficiency under identical conditions, including time-varying slip disturbances and a complex U-turn reference trajectory. By benchmarking against VRTS, the analysis underscores the advantages of integrating LSTM-based uncertainty estimation with Hamiltonian optimal control, highlighting superior adaptability and robustness for nonlinear systems.

### 6.1. Simulation setup

The skid-steering robot's kinematic model, as detailed in Section 5, is employed for both methods. The bi-level framework uses Equation (18), with state $x(t) = [X, Y, \psi]^T$, control inputs $u(t) = [v(t), \omega(t)]^T$ constrained by $|u_i| \leq b_{ui}$, and lumped uncertainty $d(t) = g(x(t))\alpha(t)$, where $\alpha(t) = [\alpha_v(t), \alpha_\omega(t)]^T$ represents slip velocities. VRTS models slip as longitudinal ($\delta_x(t) = \alpha_v(t)\cos\psi(t)$), lateral ($\delta_y(t) = \alpha_v(t)\sin\psi(t)$), and angular ($\delta_\theta(t) = \alpha_\omega(t)$) components, generating virtual trajectories:

$$x_{vr}(t) = x_r(t) + \tilde{d}_x(t), \quad y_{vr}(t) = y_r(t) + \tilde{d}_y(t), \quad \psi_{vr}(t) = \arctan 2(\dot{y}_{vr}(t), \dot{x}_{vr}(t)).$$

The U-turn trajectory, consistent with [23], features initial conditions $[x_r(0), y_r(0)] = [-1.5, -1]$m, $\psi_r(0) = 0$ rad, linear velocity $v_r(t) = 0.6$ m/s for $t \in [0, 10]$ s, and angular velocity $\omega_r(t)$ varying to induce turns. Slip disturbances are aligned: $\alpha_v(t) = 0.05\sin(1.5t)$ m/s, $\alpha_\omega(t) = 0.03\cos(1.5t)$ rad/s for the bi-level framework, and equivalent $\delta_x(t)$, $\delta_y(t)$, $\delta_\theta(t)$ for VRTS during $t \in [3, 3 + 12\pi/3]$ s. Initial states are $[x_a(0), y_a(0)] = [-1.6, -0.9]$ m, $\psi_a(0) = -0.3$ rad. The bi-level framework employs an LSTM with 64 hidden units, 2 layers, and learning rate $\eta = 0.001$, with saturation limits $b_{ui} = [0.8, 1.2]$. VRTS uses gains $k_1 = 3$, $k_2 = 2.8$, $k_3 = 2.9$, and tracking differentiator parameters $[r, n]^T = [100, 1.5]^T$. Simulations span 10 s with a 0.001-second sampling period.

### 6.2. Performance metrics

Performance is quantified using: Root Mean Square Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{T}\int_0^T \left((x_r(t) - x_a(t))^2 + (y_r(t) - y_a(t))^2\right)dt},$$

measuring tracking accuracy over $T = 10$ s. Error Boundary (EB):

$$\text{EB} = \max_{t\in[3,3+12\pi/3]} \sqrt{(x_r(t) - x_a(t))^2 + (y_r(t) - y_a(t))^2},$$

capturing maximum error during turns. Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{T_{\text{turn}}}\int_3^{3+12\pi/3} \sqrt{(x_r(t) - x_a(t))^2 + (y_r(t) - y_a(t))^2}\,dt,$$

assessing average error over $T_{\text{turn}} = 12\pi/3$ s. Slip Estimation Error: For the bi-level framework, derived from LSTM estimation of $d(t)$; for VRTS:

$$\text{Slip Error} = \frac{1}{T_{\text{turn}}}\int_3^{3+12\pi/3} \sqrt{\left(\delta_x(t) - \hat{\delta}_x(t)\right)^2 + \left(\delta_y(t) - \hat{\delta}_y(t)\right)^2}\,dt.$$

### 6.3. Results and discussion

The bi-level framework outperforms VRTS across all metrics, achieving an RMSE of 0.0012 m, EB of 0.0015 m, MAE of 0.0010 m, and slip estimation error of 0.2 % for the U-shape trajectory, compared to VRTS's RMSE of 0.0016 m, EB of 0.0017 m, MAE of 0.0014 m, and slip error of 1.7 % (Table 2). For Circle and T-shape trajectories, the bi-level framework maintains consistent precision (RMSE 0.0012–0.0013 m, slip error 0.2 %), while VRTS's performance degrades (RMSE 0.018–0.019 m, slip error 1.8–4.2 %). The LSTM's online uncertainty estimation captures temporal dynamics with high accuracy (Fig. 7), enabling rapid error convergence (Fig. 16) and smooth, energy-efficient control inputs within saturation limits (Fig. 17). In contrast, VRTS's reliance on predefined slip models limits adaptability to unmodeled dynamics, resulting in higher tracking errors during complex turns. The bi-level framework's robust performance across 30 random initial conditions (Fig. 18) and diverse trajectories underscores its scalability and versatility, positioning it as a superior solution for nonlinear control in autonomous systems.

## 7. Discussion

The proposed bi-level optimization framework demonstrates superior trajectory tracking performance for a skid-steering tracked robot under slip-induced uncertainties, as validated across Circle, T-shape, and U-shape trajectories (Table 2, Figs. 3–18). The framework achieves a root mean square error (RMSE) of 0.0012–0.0013 m, error boundary (EB) of 0.0015–0.00165 m, mean absolute error (MAE) of 0.0010–0.0011 m, and slip estimation error of 0.2 % across all trajectories, significantly outperforming the Virtual Reference Trajectory Scheme (VRTS), standard RL, and adaptive control. For the U-shape trajectory, the bi-level framework yields an RMSE of 0.0012 m and slip error of 0.2 %, compared to VRTS's RMSE of 0.0016 m and slip error of 1.7 %. For the Circle and T-shape trajectories, VRTS's performance degrades notably (RMSE 0.019 m and 0.018 m, slip error 4.2 % and 1.8 %, respectively), while the bi-level framework maintains consistent precision (Table 2).

**Table 2**
Simulation results for trajectory tracking on a skid-steering robot under uncertainty for different trajectories. Circle and T-shape results are estimated based on U-shape performance.

| Method | Trajectory | RMSE | EB | MAE | Slip Error |
|---|---|---|---|---|---|
| Bi-Level | Circle | 0.0013 | 0.00165 | 0.0011 | 0.2 |
| | T-shape | 0.0012 | 0.0015 | 0.001 | 0.2 |
| | U-shape | 0.0012 | 0.0015 | 0.001 | 0.2 |
| VRTS | Circle | 0.019 | 0.020 | 0.017 | 4.2 |
| | T-shape | 0.018 | 0.019 | 0.015 | 1.8 |
| | U-shape | 0.0016 | 0.0017 | 0.0014 | 1.7 |

The integration of Long Short-Term Memory (LSTM) networks for online uncertainty estimation enables the bi-level framework to capture temporal dynamics with exceptional accuracy (Fig. 7), facilitating rapid tracking error convergence to near-zero (Fig. 5). The actor-critic RL architecture ensures smooth, energy-efficient control inputs within saturation constraints (Fig. 6), contrasting with the higher variability of adaptive control and the slower stabilization of RL. Compared to VRTS, which relies on predefined slip models and virtual trajectory adjustments [23], the bi-level framework's adaptive uncertainty compensation enhances robustness to unmodeled dynamics, as evidenced by its stable performance across diverse trajectories and 30 random initial conditions (Fig. 18). The rapid convergence of critic network weights (Fig. 9) further underscores the framework's computational efficiency, making it suitable for autonomous systems.

The bi-level framework's versatility is particularly evident in handling complex trajectories. The T-shape trajectory, with sharp directional changes, and the Circle trajectory, with continuous turns, pose significant challenges due to slip disturbances, yet the framework achieves near-perfect tracking (Figs. 10–13). The U-shape trajectory results (Figs. 14–17) highlight its ability to manage curvilinear paths with minimal deviation. These outcomes position the bi-level framework as a scalable solution for nonlinear control in robotics, offering substantial improvements over VRTS's limited adaptability and the sample inefficiency of RL.

As summarized in Table 2, the proposed bi-level framework achieves consistent precision across diverse trajectories, with an RMSE of 0.12 m and slip estimation error of 0.2 %, outperforming VRTS (RMSE = 0.16–0.19 m, slip error = 1.7–4.2 %), particularly in complex paths such as the U-shaped trajectory, where it reduces tracking error by 25 %.

To assess the necessity of the bi-level separation, we compared the proposed alternating master-slave schedule with a single-level joint optimization where both the policy and uncertainty estimator were updated simultaneously. The single-level variant exhibited two consistent issues: (i) instability in the learning process under hard input saturation, and (ii) the inability to provide tractable Lyapunov-based stability guarantees. In contrast, the bi-level structure enables fixed-estimator updates during policy optimization, which is essential for Theorem 2's proof of uniform ultimate boundedness. This separation also contributed to smoother control inputs and superior tracking performance across all evaluated trajectories.

## 8. Conclusion

This paper proposes a bi-level optimization framework for the optimal control of nonlinear continuous-time systems, integrating LSTM-based uncertainty estimation with an actor-critic RL architecture. The proposed framework enables accurate trajectory tracking without the need for offline training. Experimental results demonstrate high precision, with RMSE ranging from 0.0012 to 0.0013 m, endpoint bias (EB) between 0.0015 and 0.00165 m, MAE from 0.0010 to 0.0011 m, and a slip estimation error of only 0.2 % across Circle, T-shape, and U-shape trajectories for a skid-steering tracked robot (Table 2). These outcomes consistently outperform those achieved by VRTS (e.g., U-shape: RMSE of 0.0016 m, Slip Error of 1.7 %), standard RL methods (RMSE of 0.45 m, Slip Error of 18.9 %), and adaptive control approaches (RMSE of 0.28 m, Slip Error of 12.5 %) [23]. Furthermore, the theoretical analysis establishes the uniform ultimate boundedness (UUB) of the tracking error, thereby ensuring system stability under dynamic uncertainties.

By enabling simultaneous policy optimization and online uncertainty compensation, the proposed bi-level framework advances the state of robust control for autonomous systems operating under uncertainty. Its consistently superior performance across diverse trajectories underscores its practical applicability in real-world robotic scenarios. Looking ahead, future research will focus on extending the framework to higher-dimensional systems, improving computational efficiency for deployment on resource-constrained platforms, and exploring alternative neural architectures to further enhance learning capability and control precision.

## CRediT authorship contribution statement

**Roya Khalili Amirabadi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Formal analysis, Data curation, Conceptualization. **Mohsen Jalaeian-Farimani:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Formal analysis, Data curation, Conceptualization. **Omid S. Fard:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Formal analysis, Data curation, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

[1] Sutton RS. Reinforcement learning: an introduction. A Bradford Book; 2018.
[2] Xiang J, Li Q, Dong X, Ren Z. Continuous control with deep reinforcement learning for mobile robot navigation. In: 2019 Chinese Automation Congress (CAC). IEEE; November 2019. p. 1501–6.
[3] Amirabadi RK, Fard OS. Combining hybrid metaheuristic algorithms and reinforcement learning to improve the optimal control of nonlinear continuous-time systems with input constraints. Comput Electr Eng 2024;116:109179.
[4] Mu C, Zhang Y, Cai G, Liu R, Sun C. A data-based feedback relearning algorithm for uncertain nonlinear systems. IEEE/CAA J Autom Sinica 2023;10(5):1288–303.
[5] Wang J, Yan Y, Liu Z, Chen CP, Zhang C, Chen K. Finite-time consensus control for multi-agent systems with full-state constraints and actuator failures. Neural Netw 2023;157:350–63.
[6] Perrusquía A, Guo W. Optimal control of nonlinear systems using experience inference human-behavior learning. IEEE/CAA J Autom Sinica 2023;10(1):90–102.
[7] Wang Y, Wu Z. Physics-informed reinforcement learning for optimal control of nonlinear systems. AIChE J 2024;70(10):e18542.
[8] Bian T, Jiang Y, Jiang ZP. Adaptive dynamic programming and optimal control of nonlinear nonaffine systems. Automatica 2014;50(10):2624–32.
[9] Hermosilla C, Vinter R, Zidani H. Hamilton–jacobi–bellman equations for optimal control processes with convex state constraints. Syst Control Lett 2017;109:30–6.
[10] Yang X, He H. Adaptive critic designs for optimal control of uncertain nonlinear systems with unmatched interconnections. Neural Netw 2018;105:142–53.
[11] Sinha A, Malo P, Deb K. A review on bi-level optimization: from classical to evolutionary approaches and applications. IEEE Trans Evol Comput 2017;22(2):276–95.
[12] Yedavalli RK. Robust control of uncertain dynamic systems. AMC 2014;10:12.
[13] Xing L, Wen C, Liu Z, Su H, Cai J. Event-triggered adaptive control for a class of uncertain nonlinear systems. IEEE Trans Autom Control 2016;62(4):2071–6.
[14] Xu B. Robust adaptive neural control of flexible hypersonic flight vehicle with dead-zone input nonlinearity. Nonlinear Dyn 2015;80:1509–20.
[15] Jalaeian-F. M, Fateh MM, Rahimiyan M. Bi-level adaptive computed-current impedance controller for electrically driven robots. Robotica 2021;39(2):200–16. https://doi.org/10.1017/S0263574720000314
[16] Van Houdt G, Mosquera C, Nápoles G. A review on the long short-term memory model. Artif Intell Rev 2020;53(8):5929–55.
[17] Komurcugil H, Biricik S, Bayhan S, Zhang Z. Sliding mode control: overview of its applications in power converters. IEEE Ind Electron Mag 2020;15(1):40–9.

[18] Jia C, Li X, Wang H, Song Z. Sliding mode-based integral reinforcement learning event triggered control. Int J Control Autom Syst 2025;23(1):315–31.
[19] Jalaeian-F M, Fateh MM, Rahimiyan M. Optimal predictive impedance control in the presence of uncertainty for a lower limb rehabilitation robot. J Syst Sci Complex 2020;33:1310–29.
[20] Amirabadi RK, Fard OS, Jalaeian-F M. Towards optimal control of HPV model using safe reinforcement learning with actor–critic neural networks. Expert Syst Appl 2025;264:125783.
[21] Amirabadi RK, Jalaeian-F M, Fard OS. Adaptive self-organizing clustering dual-buffer safe reinforcement learning for nonlinear optimal control. 2025. preprint SSRN.
[22] Reddi SJ, Kale S, Kumar S. On the convergence of adam and beyond. arXiv preprint 2019 arXiv:1904.09237.
[23] Wu X, Wang Y. Virtual reference trajectory scheme for robust tracking of skid-steering robots. J Robot Res 2024;42(3):123–45.