Approximate Dynamic Programming for Constrained Piecewise Affine Systems With Stability and Safety Guarantees

He, Kanghui; Shi, Shengling; Boom, Ton van den; Schutter, Bart de

DOI
10.1109/TSMC.2024.3515645

Publication date
2025

Document Version
Final published version

Published in
IEEE Transactions on Systems, Man, and Cybernetics: Systems

Citation (APA)
He, K., Shi, S., Boom, T. V. D., & Schutter, B. D. (2025). Approximate Dynamic Programming for Constrained Piecewise Affine Systems With Stability and Safety Guarantees. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *55*(3), 1722-1734. https://doi.org/10.1109/TSMC.2024.3515645

Approximate Dynamic Programming for Constrained Piecewise Affine Systems With Stability and Safety Guarantees

Kanghui He, Shengling Shi, Ton van den Boom, and Bart De Schutter, *Fellow, IEEE*

***Abstract*—Infinite-horizon optimal control of constrained piecewise affine (PWA) systems has been approximately addressed by hybrid model predictive control (MPC), which, however, has computational limitations, both in offline design and online implementation. In this article, we consider an alternative approach based on approximate dynamic programming (ADP), an important class of methods in reinforcement learning. We accommodate nonconvex union-of-polyhedra state constraints and linear input constraints into ADP by designing PWA penalty functions. PWA function approximation is used, which allows for a mixed-integer encoding to implement ADP. The main advantage of the proposed ADP method is its online computational efficiency. Particularly, we propose two control policies, which lead to solving a smaller-scale mixed-integer linear program than conventional hybrid MPC, or a single convex quadratic program, depending on whether the policy is implicitly determined online or explicitly computed offline. We characterize the stability and safety properties of the closed-loop systems, as well as the suboptimality of the proposed policies, by quantifying the approximation errors of value functions and policies. We also develop an offline mixed-integer-linear-programming-based method to certify the reliability of the proposed method. Simulation results on an inverted pendulum with elastic walls and on an adaptive cruise control problem validate the control performance in terms of constraint satisfaction and CPU time.**

***Index Terms*—Approximate dynamic programming (ADP), constrained control, piecewise affine (PWA) systems, reinforcement learning (RL).**

I. Introduction

A. Backgrounds

THERE has been an increasing interest in control of piecewise affine (PWA) systems due to their capability of representing hybrid models and approximating nonlinear dynamics [1]. Many practical control problems can be modelled as PWA systems with constraints, including emergency evasive maneuvers [2], robotic manipulation that has multi-contact behaviors [3], and traffic control [4]. PWA systems are a special class of switched systems where the subsystem in each mode is affine and the transitions are based on the state belonging to different regions. Tractable controller design methods for PWA systems include synthesizing piecewise linear control laws via linear matrix inequalities [5], adaptive control [1], and model predictive control (MPC) [5]. The challenges of controlling a PWA system include ensuring stability and achieving optimality guarantees [5], as well as addressing both offline and online computational complexity [3], [6]. These difficulties primarily arise from the system's hybrid structure and inherent nonlinearity. For suboptimal control of PWA systems with constraints, MPC is widely applied. However, MPC for PWA systems still faces challenges in computational complexity [6], because it involves solving a mixed-integer linear programming (MILP) problem. The complexity of solving MILP MPC problems is in general dominated by the number of integers, which is proportional to the prediction horizon. Explicit MPC [6], an offline version of MPC, requires solving a parametric MILP problem, which is also suffering from computational complexity issues. These issues make MPC only suitable for slow PWA processes or for small scale problems [3].

In contrast to MPC, reinforcement learning (RL) can learn a policy that minimizes a finite-/infinite-horizon cost and could have a much lower online computational burden than MPC. In RL, two different methodologies can be distinguished: 1) policy search [7] and 2) dynamic programming [8]. Dynamic programming has the advantage over policy search is that it reduces the policy optimization problem to an one-step look-ahead problem. When applied to systems with continuous state and input spaces, approximate dynamic programming (ADP) has been developed [9]. In this article, we consider approximate value iteration (VI), the most basic and direct way to solve the Bellman equation. Moreover, we provide comprehensive performance guarantees for stability, safety and suboptimality of the developed ADP approach. In the context of RL, safety can have various definitions. In this article, we specifically address safety as ensuring that the state and input of the system satisfy predefined constraints throughout the entire system's evolution after the learning process is finished.

Received 6 January 2024; revised 18 August 2024; accepted 25 November 2024. Date of publication 24 December 2024; date of current version 19 February 2025. This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme-CLariNet under Grant 101018826. This article was recommended by Associate Editor X. Xu. *(Corresponding author: Kanghui He.)*

Kanghui He, Ton van den Boom, and Bart De Schutter are with the Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: k.he@tudelft.nl; a.j.j.vandenboom@tudelft.nl; b.deschutter@tudelft.nl).

Shengling Shi is with the Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: slshi@mit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2024.3515645.

Digital Object Identifier 10.1109/TSMC.2024.3515645

B. Related Work

To reduce the computational cost of MPC, two different types of approaches have been extensively studied: 1) approximate MPC and 2) RL under constraints.

Approximate MPC: Approximate MPC parameterizes a policy and then uses supervised learning or gradient-based methods to mimic a predictive control policy. The online computational cost is thus significantly reduced because approximate MPC directly computes control actions based on the learned parameters, rather than online solving an optimization problem. Some approximate MPC work considers linear systems, with different focuses on, e.g., stability [10] and constraint satisfaction [11]. Some approximate MPC approaches can handle nonlinear control problems with constraints, e.g., by using constraint tightening [12].

RL for Constrained Control: RL for constrained control can be roughly categorized into two groups: 1) policy-projection RL and 2) policy-optimization RL. For the first group, a predictive safety filter (PSF) [13], [14] can be adopted to modify the learned policy, which can be derived from any RL algorithm. For PWA systems with linear constraints, this group needs to solve mixed-integer convex problems online [14], and as a result it is not suitable for large-scale systems or for systems requiring fast computation. For the second group, constrained policy optimization [15], [16], [17] is often used. Most methods [17] consider constrained Markov decision process (MDP) problems, in which constraints are on expected cumulative costs. Recent developments have been made to transform instantaneous constraints into constraints on expected cumulative costs. The stability property of RL controllers for nonlinear systems has been investigated recently [18], [19], [20]. Nevertheless, these references consider unconstrained problems and do not address the suboptimality of the RL policy.

Based on these observations, for the policy optimization methods, one could know that no work has been done for PWA systems, and comprehensive performance of RL-based controllers regarding online computing convenience, stability, and constraint fulfillment cannot be achieved simultaneously.

Performance Verification of Learning-Based Controllers: In addition to controller design, there is some related work on performance analysis and verification of RL or any learning-based controllers for PWA systems, by using a learner/verifier framework or explicitly computing the range of trajectories in a finite horizon. However, for PWA systems with learning-based controllers, there is currently no systematic way to verify different properties, including practical and asymptotic stability as well as state constraint satisfaction.

C. Methods and Contributions of This Article

In conclusion, using RL to produce a reliable-learning-based controller for constrained PWA systems with performance guarantees and low online computational requirements is still an open problem. The main challenge is to concurrently ensure the stability, safety, and efficiency of the online computations. Existing work can either provide stability/safety guarantees [13], [14], [15], [20], [21], [22] or achieve low computational cost [23], [24]. In this article, we develop ADP algorithms under linear and union-of-polyhedra (UoP) constraints. We propose two formulations for the inclusion of PWA penalties in dynamic programming, i.e., adding penalties to the stage cost and integrating penalties into the cost-to-go. We then present two different controllers: 1) an implicit controller that is obtained online by solving an MILP problem that is much more simple than the one of implicit hybrid MPC and 2) an explicit controller that is learned offline by policy gradient. We provide rigorous analysis on the closed-loop stability, safety, as well as suboptimality of the controllers. We establish a systematic, MILP-based procedure that allows us to certify the reliability of the closed-loop system. This article contributes the state of the art as follows.

1) This work is the first research on designing policy optimization RL methods for constrained PWA systems. Systematic performance analysis on the feasibility, stability, and suboptimality of the RL-based controllers is provided. The analysis suggests several ways to employ the proposed algorithms in practice.
2) Compared to MPC, our method exhibits a superiority in terms of online computational simplicity. In particular, the resulting online policy optimization problem is either an MILP problem with significantly fewer integer variables than the hybrid MPC MILP problem, or a single convex quadratic programming (QP) problem.
3) We develop a mixed-integer-optimization-based framework to exactly verify the stability and safety of the closed-loop system. The framework extends the verification techniques of [10], [25], and [26] with a comprehensive scheme that addresses both practical and asymptotic stability properties and the enlargement of stable and safe regions.

II. Preliminaries

Notations: Let $\mathbb{R} = (-\infty, +\infty)$, $\mathbb{R}_{\geq 0} = [0, +\infty)$, and $\mathbb{R}_{>0} = (0, +\infty)$. The boundary of the set $\mathcal{S}$ is $\partial\mathcal{S}$, and $\text{int}(\mathcal{S})$ is the interior of $\mathcal{S}$. We utilize $A_{i\cdot}$ to represent the ith row of the matrix A. We define the sublevel set $\mathcal{B}(J, S)$ for a continuous function $J : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and a compact set $S \subseteq \mathbb{R}^n$ as $\mathcal{B}(J, S) \triangleq \{x \in \mathbb{R}^n | J(x) \leq \rho\}$, where $\rho = \min_{x\in\partial S} J(x)$. Denote by $\lceil a \rceil$ the smallest integer larger than or equal to a.

A. Optimal Control of PWA Systems

Our control objective is to solve the constrained infinite-horizon optimal control problem

$$\begin{aligned} J^*(x_0) = \min_{\pi,\mathbf{u},\mathbf{x}} \quad & \left\{ J_\pi(x_0) \triangleq \sum_{t=0}^{\infty} l(x_t, \pi(x_t)) \right\} \\ \text{s.t.} \quad & x_{t+1} = f_{\text{PWA}}(x_t, u_t),\ u_t = \pi(x_t) \\ & x_t \in X,\ u_t \in U,\ t = 0, 1, \ldots \end{aligned} \tag{1}$$

where we consider the discrete-time PWA system of the form

$$f_{\text{PWA}}(x, u) = A_i x + B_i u + f_i \quad \text{if} \quad \begin{bmatrix} x \\ u \end{bmatrix} \in \mathcal{C}_i. \tag{2}$$

In (2), $\{\mathcal{C}_i\}_{i=1}^s$ is a polyhedral partition of the state-input space $\mathcal{X} \times \mathcal{U}$. The matrices A_i, B_i, and the vectors f_i define the affine

dynamics in the regions $\mathcal{C}_i$. Including the offset vector f_i allows for the representation of an affine transformation instead of just a linear one in each region of the state space. In (1), $\mathbf{u} = \{u_t\}_{t=0}^{\infty} = \{\pi(x_t)\}_{t=0}^{\infty}$, $\mathbf{x} = \{x_t\}_{t=0}^{\infty}$, $\pi(\cdot) : \mathcal{X} \to \mathcal{U}$ is a control policy, $l : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$ is the stage cost. Besides, $x \in X$, $u \in U$ are the state and input constraints.[1] We assume that the dynamics, constraints, and stage costs satisfy the following assumption.

Assumption 1 (Dynamics): The function $f_{\text{PWA}}(\cdot, \cdot) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a continuous PWA function, with $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ and $\mathcal{U} \subseteq \mathbb{R}^{n_u}$. Besides, $f_{\text{PWA}}(0, 0) = 0$.

Constraints: The state constraint set $X = \bigcup_{i=1}^{r_0} X^{(i)}$ is a UoP, where $X^{(i)}$ is a polyhedron for each $i = 1, \ldots, r_0$ and r_0 is the number of polyhedra. The input constraint set U is a polyhedron. Moreover, $X \times U \subset \bigcup_{i=1}^{s} \mathcal{C}_i$.

Stage Cost: The stage cost is based on the 1-/∞-norm: $l(x_t, u_t) = \|Qx_t\|_{q_1} + \|Ru_t\|_{q_2}$, where $q_1, q_2 \in \{1, \infty\}$ and Q, R have full column rank.

Similar assumptions can be found in other papers on control of PWA systems, e.g., [27] and [28]. In most literature [5], [27], [28], it is usually assumed that X is a polyhedron, while we generalize it to a UoP.

An optimal policy, denoted by $\pi^*(\cdot)$, minimizes $J_\pi(x_0)$ subject to the constraints in (1) for any initial state x_0 that makes (1) feasible. In this article, we denote by $\bar{X}$ the set of feasible initial states x_0 that make $J^*(x_0)$ finite.

Assumption 2: The set $\bar{X}$ is nonempty. Furthermore, for any $x_0 \in \bar{X}$, there exists a policy $\pi(\cdot)$ such that the system (2) with $u_t = \pi(x_t)$, starting from x_0, will reach the origin in a finite number of time steps.

Assumption 2 is a standard stabilizability assumption for discrete-time systems. Similar assumptions for PWA systems can be found, e.g., [5] and [27].

For any $x \in \mathcal{X}$, according to Bellman's Principle of Optimality [29], the value function $J^*(\cdot)$ and the optimal policy $\pi^*(\cdot)$ satisfy the following equations:

$$J^*(x) = \Gamma J^*(x) \triangleq \min_{u \in U} l(x, u) + J^*(f_{\text{PWA}}(x, u))$$
$$\pi^*(x) \in \arg\min_{u \in U} l(x, u) + J^*(f_{\text{PWA}}(x, u)) \quad (3)$$

where Γ is called the Bellman operator [29]. In (3), the domain of $J^*(\cdot)$ is the whole state space $\mathcal{X}$, which means that the value of $J^*(\cdot)$ goes to infinity outside $\bar{X}$. In general, the equation for $J^*(\cdot)$ in (3) may have multiple solutions. Nevertheless, it follows from [30, Proposition 1] that $J^*(\cdot)$ can be the unique solution that satisfies $J^*(0) = 0$ under Assumption 2.

B. Exact Value Iteration

Solving the Bellman equations is in general computationally prohibitive for nonlinear systems. Usually, an MPC problem with a finite horizon is solved online to approximate the infinite-horizon optimal policy. However, computational complexity also remains a hurdle in the application of MPC to PWA systems. For the PWA system, the equations in (3), on the other hand, can be solved by using an exact VI method [27], which solves multiple multiparametric linear programs (mp-LPs). To motivate our ADP methods, we summarize the exact VI method and discuss its limitations in this section. The exact VI algorithm starts from an initial value function $J_0(\cdot)$ that is either zero in $\mathcal{X}$ (case 1) or a control Lyapunov function defined on a subset of $\mathcal{X}$ (case 2). In case 2, the following assumption should be satisfied.

Assumption 3: A continuous and PWA control Lyapunov function $J_{\text{CL}}(\cdot) : X_{\text{CI}} \to \mathbb{R}_{\geq 0}$ in a polyhedral control-invariant set X_{CI} is available. In other words, $\min_{u \in U, f_{\text{PWA}}(x,u) \in X_{\text{CI}}} l(x, u) + J_{\text{CL}}(f_{\text{PWA}}(x, u)) - J_{\text{CL}}(x) \leq 0 \quad \forall x \in X_{\text{CI}}$.

Assumption 3 frequently appears in the stability analysis of MPC [31], where $J_{\text{CL}}(\cdot)$ is chosen as the terminal cost and X_{CI} is specified as the terminal constraint. To satisfy Assumption 3, it is sufficient to compute a stabilizing piecewise linear feedback law on X_{CI}, and then $J_{\text{CL}}(\cdot)$ can be computed by solving some nonlinear inequalities that contain 1-/∞-norm of some linear functions [5].

With the initialization $X_0 = X$ and $J_0(x) = 0 \quad \forall x \in X_0$ (case 1), or $X_0 = X_{\text{CI}}$ and $J_0(x) = J_{\text{CL}}(x) \quad \forall x \in X_0$ (case 2), the exact VI method iterates as follows:

$$J_k(x) = \min_{u \in U, f_{\text{PWA}}(x,u) \in X_{k-1}} l(x, u) + J_{k-1}(f_{\text{PWA}}(x, u)) \quad (4)$$

for $k = 1, 2, \ldots$. Here, $X_k = \text{Pre}(X_{k-1}) \cap X_0$, where $\text{Pre}(\mathcal{S}) = \{x \in \mathcal{X} | \exists u \in U \text{ s.t. } f_{\text{PWA}}(x, u) \in \mathcal{S}\}$ is the backward-reachable set to a set S. Even for PWA systems with polytopic state constraints, the backward-reachable set to a polyhedral set can be a nonconvex UoP because of the nonlinear dynamics (2), which means that $X_k, k = 1, 2, \ldots$ can be nonconvex UoPs [6].

The resulting $J^*(\cdot)$ and $\pi^*(\cdot)$ are both PWA functions sharing the same polyhedral partition of the feasible region $\bar{X}$ [27]. However, the complexity (i.e., the number of polyhedral regions or affine functions) of $J^*(\cdot)$ and $\pi^*(\cdot)$ is exponential in both the dimension of the system and the number of constraints in (1) [32], so that storing the affine functions and regions of $J^*(\cdot)$ and $\pi^*(\cdot)$ needs a huge amount of memory. Second, the number of mp-LPs that need to be solved per iteration also grows exponentially with the problem dimension. Moreover, the online implementation of [27] needs to search which polyhedron the measured state belongs to. For high-dimensional systems, these regions may have complex representations. Based on these observations, it is thus necessary to simplify both the procedure of solving the Bellman equation and the control policy, by using some approximation methods. However, the VI formulation in (4) is not suitable for approximation because the probably nonconvex constraint $f_{\text{PWA}}(x, u) \in X_{k-1}$ leads to too complex optimization problems when using sample-based approaches.

III. Value Iteration With Penalty Functions

To deal with this issue, we consider soft state constraints by defining a penalty function $P(\cdot, \cdot)$. Suppose that each $X^{(i)}$ of the UoP constraint set $X = \bigcup_{i=1}^{r_0} X^{(i)}$ has the half-space representation $X^{(i)} \triangleq \{x \in \mathbb{R}^{n_x} | E_X^{(i)} x \leq g_X^{(i)}\}$, where $E_X^{(i)} \in \mathbb{R}^{m_x^{(i)} \times n_x}$, $g_X^{(i)} \in \mathbb{R}^{m_x^{(i)}}$, and $m_x^{(i)}$ is the number of rows of $E_X^{(i)}$.

[1] All the results of this article also apply to the case when there is a coupled constraint: $[x^T \ u^T]^T \in D$ with D a polyhedron in $\mathcal{X} \times \mathcal{U}$, by letting X be the projection of D in $\mathcal{X}$ and by letting U be a time-varying set depending on x.

We design the penalty function $P(\cdot,\cdot)$ as the following min–max forms:

$$P(x,X)=p\min_i \max\left\{0,\left(E_X^{(i)}\right)_{1\cdot}x-\left(g_X^{(i)}\right)_1,\ldots \left(E_X^{(i)}\right)_{m_x^{(i)}\cdot}x-\left(g_X^{(i)}\right)_{m_x^{(i)}}\right\} \text{ or}$$

$$P(x,X)=p\min_i \sum_{j=1}^{m_x^{(i)}}\max\left\{0,\left(E_X^{(i)}\right)_{j\cdot}x-\left(g_X^{(i)}\right)_j\right\} \quad (5)$$

where $(g_X^{(i)})_j$ is the jth element of the vector $g_X^{(i)}$, $m_x^{(i)}$ is the number of rows of $E_X^{(i)}$, and $p>0$ is the constraint violation penalty weight. When X reduces to a polyhedron, i.e., $r_0=1$, the minimum operator in (5) will be removed.

An important property of $P(\cdot,\cdot)$ is that $P(\cdot,\cdot)$ is a PWA function w.r.t. its first argument. This means that adding such a penalty function into the cost function in (1) will not change the PWA properties of the optimal value function and the optimal policy. Note that we avoid barrier functions, such as the logarithmic barrier function because they can go to infinity in any compact constraint sets and will deprive the value function of the PWA property. Besides, we do not penalize the input constraint violation, because the input constraints are single polyhedral constraints that can be readily handled.

In case 2 of (4), in addition to enforcing a penalty for X, we need to reconstruct the initial value function since $J_{\mathrm{CL}}(\cdot)$ is undefined outside X_{CI}. To achieve this, we need to penalize $J_{\mathrm{CL}}(x)$ for x outside X_{CI} by finite values

$$J_0^{\mathrm{soft}}(x)=\begin{cases}J_{\mathrm{CL}}(x), & x\in X_{\mathrm{CI}}\\ J_{\mathrm{CL}}(\bar{z})+P(x,X_{\mathrm{CI}}), & x\notin X_{\mathrm{CI}}\end{cases} \quad (6)$$

where $\bar{z}(\cdot)$ is one of the optimizers of the following mp-LP:

$$\bar{z}(x)\in\arg\min_{z\in X_{\mathrm{CI}}}||z-x||_\infty. \quad (7)$$

In (6), $J_0^{\mathrm{soft}}(\cdot)$ is continuous on $\mathcal{X}$, which will be proven in Theorem 1. Based on the defined penalty function, a VI algorithm with penalty is developed as follows.

Algorithm 1 VI With Penalty

Output: A value function $J_{k-1}^{\mathrm{soft}}(\cdot):\mathcal{X}\to\mathbb{R}_{\geq 0}$.

1: Initialize the value function (option (a)) $J_0^{\mathrm{soft}}(x)\leftarrow 0\quad\forall x\in\mathcal{X}$, or (option (b)) by (6).
2: **for** $k=1,2,\ldots$ **do**
3: the value iteration $J_k^{\mathrm{soft}}(x)\leftarrow\Gamma_{\mathrm{p},\alpha}J_{k-1}^{\mathrm{soft}}(x)\quad\forall x\in\mathcal{X}$, where $\alpha=1$ if option 1 is chosen, or $\alpha=2$ if option 2 is chosen, and $\Gamma_{\mathrm{p},\alpha}$ is defined in (8) and (9).
4: If $J_k^{\mathrm{soft}}(x)=J_{k-1}^{\mathrm{soft}}(x)\quad\forall x\in\mathcal{X}$, **break**.
5: **end for**

In Algorithm 1, we consider two options for the VI, in which we define two Bellman operators for $J:\mathcal{X}\to\mathbb{R}$. The first one used in option 1 is

$$\Gamma_{\mathrm{p},1}J(x)\triangleq\min_{u\in U}\ l_p(x,u)+J(f_{\mathrm{PWA}}(x,u)),\ x\in\mathcal{X} \quad (8)$$

where $l_p(x,u)=l(x,u)+P(x,X)$. The second one used in option 2 is

$$\Gamma_{\mathrm{p},2}J(x)\triangleq\min_{u\in U}\ l(x,u)+J(f_{\mathrm{PWA}}(x,u))+P_{k-1}(f_{\mathrm{PWA}}(x,u))\quad x\in\mathcal{X} \quad (9)$$

where $P_0(x)=0\quad\forall x\in\mathcal{X}$ and $P_k(x)=P(x,X)\quad\forall x\in\mathcal{X}$ and $\forall k>0$. Since the state constraints are removed, the working region of VI is the whole state space $\mathcal{X}$. Besides, we also consider two options [options (a) and (b)] for the initialization of the value function. The combinations of above options result in four different options: options 1(a), 1(b), 2(a), and 2(b). In option 2 of the algorithm, we propose a novel scheme in which we add a penalty into the cost-to-go $J_{k-1}^{\mathrm{soft}}(f_{\mathrm{PWA}}(x,u))$. In the following theorem, we will analyze the PWA property and continuity of each $J_k^{\mathrm{soft}}(\cdot)$ as well as the convergence of the sequence $\{J_k^{\mathrm{soft}}(\cdot)\}_{k=0}^{\infty}$ to a fixed optimal value function in all options. The proof is given in Appendix A.

Theorem 1: Considering Algorithm 1, if Assumptions 1 and 2 hold in option (a) and Assumptions 1–3 hold in option (b), each $J_k^{\mathrm{soft}}(\cdot)$, $k<\infty$ is a continuous PWA function on $\mathcal{X}$ and the sequence $\{J_k^{\mathrm{soft}}(x)\}_{k=0}^{\infty}$ converges point-wise to

$$J^{\mathrm{soft}*}(x)=\min_{\{u_i,x_i\}_{i=0}^{\infty}}\ \sum_{i=0}^{\infty}l(x_i,u_i)+P(x_i,X)$$
$$\text{s.t.}\quad x_{i+1}=f_{\mathrm{PWA}}(x_i,u_i),\ u_i\in U,\ i=0,1,\ldots$$
$$x_0=x. \quad (10)$$

Option 2, incorporating penalties into the cost-to-go, yields the equivalent value function $J_k^{\mathrm{soft}}(\cdot)$ as option 1, which adds penalties to stage costs. Both options can alleviate the violation of state constraints by adding penalties to the overall infinite-horizon cost function. Detailed comparison between options 1 and 2 is given in Section IV-C.

After the optimal value function J_k^{soft} is obtained by Algorithm 1, the control policy is implicitly determined by the solution of the optimization problem (3) with J^* replaced by J_k^{soft}.

IV. Constrained ADP Algorithm

A. Algorithm Design

Continuity of the value functions, established in Theorem 1, is desired since it enables a universal approximation capability [33]. With Algorithm 1, a tractable ADP approach can be developed to approximate each $J_k^{\mathrm{soft}}(\cdot)$. In particular, a function approximator (critic) $\hat{J}_k(\cdot,\theta_k)$, which is parameterized by θ_k, is constructed to replace $J_k^{\mathrm{soft}}(\cdot)$. In each iteration k, a set $X_{\mathrm{s}}=\{x^{(i)}\}_{i=1}^{N_x}$ of state samples is collected from a compact region of interest $\Omega_k\subseteq\mathcal{X}$, according to some strategies, such as sampling from a uniform grid and random sampling [8]. Here, N_x is the number of samples. The update of the parameter θ_k minimizes $\sum_{i=1}^{N_x}[\Gamma_{p,\alpha}\hat{J}_{k-1}(x,\theta_{k-1})|_{x=x^{(i)}}-\hat{J}_k(x^{(i)},\theta_k)]^2$. The iterative procedure stops when the difference between θ_k and θ_{k-1} is small enough. The procedure is given in Algorithm 2.

In (13) of step 8 of Algorithm 2, $\rho_v(\cdot):\mathcal{X}\to\mathbb{R}_{>0}$ is the state relevance weighting function. In step 9, $\epsilon(\cdot):\mathcal{X}\to\mathbb{R}_{\geq 0}$ is a tolerance function, determining whether $\hat{J}_{k-1}(\cdot,\theta_{k-1})$ is

Algorithm 2 Constrained Approximate VI

1: Option (a): Initialize the value function $\hat{J}_0(\cdot,\theta_0) \leftarrow 0 \quad \forall x \in \Omega_0$, where $\Omega_0 = \mathcal{X}$ in option 1(a) or $\Omega_0 = X$ in option 2(a).
Option (b): Initialize the value function $\hat{J}_0(\cdot,\theta_0)$ by

$$\theta_0 \leftarrow \arg\min_{\theta} \sum_{x \in X_s \cap \Omega_0} \rho_v(x)\left(J_0^{\text{soft}}(x) - \hat{J}_0(x,\theta)\right)^2, \quad (11)$$

where $J_0^{\text{soft}}(\cdot)$ is from (6), and $\Omega_0 = \mathcal{X}$ in option 1(b) or $\Omega_0 = X_{\text{CI}}$ in option 2(b).
2: **for** $k = 1, 2, \ldots$ **do**
3: **if** option 1 is chosen, **then** let $\Omega_k \leftarrow \mathcal{X}$ and $\alpha \leftarrow 1$.
4: **end if**
5: **if** option 2 is chosen, **then** let $X_k \leftarrow \text{Pre}(X_{k-1}) \cap X$, $\Omega_k \leftarrow X_k$, and $\alpha \leftarrow 2$.
6: **end if**
7: Obtain the target value $v_k^{(i)}$ by

$$v_k^{(i)} \leftarrow \Gamma_{\text{p},\alpha}\hat{J}_{k-1}(x,\theta_{k-1})|_{x=x^{(i)}} \quad \forall x^{(i)} \in X_s \cap \Omega_k. \quad (12)$$

8: Find θ_k such that

$$\theta_k \leftarrow \arg\min_{\theta} \sum_{x^{(i)} \in X_s \cap \Omega_k} \rho_v\left(x^{(i)}\right)\left(v_k^{(i)} - \hat{J}_k(x^{(i)},\theta)\right)^2. \quad (13)$$

9: **if** $|\hat{J}_k(x,\theta_k) - \hat{J}_{k-1}(x,\theta_{k-1})| \le \epsilon(x) \quad \forall x \in \Omega_k \cap \Omega_{k-1}$, **then break** and output $\hat{J}_{k-1}(\cdot,\theta_k)$.
10: **end if**
11: **end for**

satisfactory. Both of $\rho_v(\cdot)$ and $\epsilon(\cdot)$ will be designed later in Section V-A. Besides, in practice one would also need a limit on the maximum number of iterations.

With $\hat{J}_{k-1}(\cdot,\theta_{k-1})$ available, a suboptimal control policy $\hat{\pi}^{\text{im}}(x)$ can be implicitly determined by

$$\hat{\pi}^{\text{im}}(x) \in \arg\min_{u \in U} \; l(x,u) + \hat{J}_{k-1}(f_{\text{PWA}}(x,u),\theta_{k-1}). \quad (14)$$

In step 1 of Algorithm 2, the function approximator is initialized by regressing $J_0^{\text{soft}}(\cdot)$, provided that the explicit form of $J_{\text{CL}}(\cdot)$ is known. If the explicit form of $J_{\text{CL}}(\cdot)$ is not available but a stabilizing and safe piecewise linear feedback law $\pi_{\text{PWL}}(\cdot) : \mathcal{X} \to \mathcal{U}$ on X_{CI} is known, we can initialize $\hat{J}_0(\cdot,\theta_0)$ as an approximation of $J_{\pi_{\text{PWL}}}(\cdot)$, which is also a control Lyapunov function, by doing a policy evaluation [8].

To carry out the iterative procedure in Algorithm 2 efficiently, we need to use a proper function approximator at each iteration. Since each $J_k^{\text{soft}}(\cdot)$ is a PWA function, it is preferable that the candidate approximator can also output a PWA function. Suitable choices are NNs with (leaky) rectifier linear units (ReLUs) as activation functions, difference of two max-affine functions, and so on. Detailed descriptions of these function approximators are given in [34, Appendix C].

Mixed-Integer Formulations of Problems (12) and (14): Problems (12) and (14) have similar forms. They can be transformed into MILP problems since both the PWA system and PWA function approximators are MILP representable, which is shown in [10]. Here, we say a function J is MILP representable if J can be represented by a set of mixed-integer linear equations and inequalities containing additional variables. As a result, we can obtain mixed-integer formulations of problems (12) and (14). He et al. [34] provided a detailed explanation of these formulations. MILP problems can be effectively solved by using the branch-and-bound approach [35], which is a global optimization algorithm.

Different from (12), Problem (14) is solved online. Even through it still belongs to an MILP problem, (14) can be solved more rapidly than a general hybrid MPC problem with a long horizon, as (14) in general results much fewer auxiliary and binary variables. That is one of main benefits of using RL.

Remark 1: When there are approximation errors, the convergence of $\hat{J}_k(\cdot,\theta_k)$ to $J^{\text{soft}*}(\cdot)$ is in general not guaranteed because the infinite-horizon cost (1) is undiscounted and does not induce a contraction property for the Bellman operator. In general, one can add a discount factor to (1) to ensure the convergence of the VI under approximation errors, but this may come out the cost of weakening the stability [36].

B. Approximating Explicit Policies

Since two PWA functions $f_{\text{PWA}}(\cdot,\cdot)$ and $\hat{J}_{k-1}(\cdot,\theta_{k-1})$ are coupled in (14), (14) may still have many auxiliary and binary variables, if, e.g., a multiple-layer (deep) NN is used. As (14) needs to be solved online, the advantage of low computational complexity brought by ADP is not obvious. To avoid solving complex MILP problems online, the policy $\hat{\pi}^{\text{im}}(\cdot)$ can also be represented explicitly, in which case it usually needs to be approximated by a second function approximator (actor). The actor is also recommended to having a PWA form since the optimal control policy $\pi^*(\cdot)$ is PWA.

As the optimizer $\hat{\pi}^{\text{im}}(\cdot)$ can be discontinuous and not unique, instead of using supervised learning methods to train the actor, we can directly construct a parameterized policy $\hat{\pi}^{\text{ex}}(\cdot,\omega)$ with parameter ω and update ω to minimize the expectation of the objective function in (14) w.r.t. the sample distribution d_s used in Algorithm 2. This results in the following policy optimization problem:

$$\begin{aligned}\omega^* \in \arg\min_{\omega} \; & \mathbb{E}_{x\sim d_s}[\rho_\pi(x)(l(x,\hat{\pi}^{\text{ex}}(x,\omega)) \\ & + \hat{J}_{k-1}(f_{\text{PWA}}(x,\hat{\pi}^{\text{ex}}(x,\omega)),\theta_{k-1}))] \\ \text{s.t. } & \mathbb{E}_{x\sim d_s}\left[\hat{\pi}^{\text{ex}}(x,\omega)\right] \in U \end{aligned} \quad (15)$$

where $\rho_\pi(\cdot) : \mathcal{X} \to \mathbb{R}_{>0}$ is another state relevance weighting function to be specified later in Section V-A. Similar to the critic, we specify $\hat{\pi}^{\text{ex}}(\cdot,\omega)$ as a PWA approximator.

To solve (15), the policy gradient method, combined with the Lagrangian multiplier methods [17], [24] for constraints handling, can be employed.

The above procedures are conducted offline. Ideally, if there are no approximation errors on both the critics and the actor, and the penalty weight p and the number of iterations are infinite, we have $X_\infty = \bar{X}$ and $\hat{\pi}^{\text{ex}}(\cdot,\omega^*) = \hat{\pi}^{\text{im}}(\cdot)$. Consequently, $\hat{\pi}^{\text{ex}}(\cdot,\omega^*)$ will always make the system satisfy all the constraints for the initial condition $x_0 \in \bar{X}$. However, due to approximation errors and the finite penalty weight, the

policy cannot always satisfy the state and input constraints. As the input constraints are usually hard constraints, in the online setting we project $\hat{\pi}^{\text{ex}}(\cdot, \omega^*)$ onto U when the current state x_t is received. This results in a convex quadratic program

$$\phi\left(u_t^{\text{ex}}\right) = \arg\min_{u \in U} ||u - u_t^{\text{ex}}||_2 \tag{16}$$

where $u_t^{\text{ex}} = \hat{\pi}^{\text{ex}}(x_t, \omega^*)$, and the function $\phi(\cdot) : \mathbb{R}^{n_u} \to \mathbb{R}^{n_u}$ maps the output of the actor to its projected value. Problem (16) can be treated as a parametric quadratic program with the parameter u_t^{ex}. Therefore, the optimizer $\phi(\cdot)$ of (16) is unique, PWA [31, Th. 6.7], and also MILP representable [10, Lemma 4]. Meanwhile, (16) defines a projected policy $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot) = \phi(\hat{\pi}^{\text{ex}}(\cdot, \omega^*))$ of $\hat{\pi}^{\text{ex}}(\cdot, \omega^*)$.

Remark 2: The number of decision variables and the number of constraints in the convex QP problem (16) are not larger than those in the MILP problem (14). It is known that convex QP problems are P problems while MILP problems are NP hard problems. Therefore, the consideration of the explicit policy $\hat{\pi}^{\text{ex}}(x_t, \omega^*)$ further enhances the online computational efficiency compared to (14).

C. Discussions on the ADP Method

Comparison Between Options 1 and 2 of Algorithm 2: The main differences between options 1 and 2 are the training region Ω_k and the way each $\hat{J}_{k-1}(\cdot, \theta_{k-1})$ iterates. These differences lead to differences in the adaptability and efficiency of options 1 and 2. Particularly, compared to option 1, option 2 is more efficient in sampling and can result in a better approximation accuracy, because the working region Ω_k in option 2 is in general much smaller than $\mathcal{X}$. We note that to implement option 1, one should choose a region of interest for sampling, and the region must be larger than X, so that the constraint violation can be penalized in the critic. However, the states that can be steered to the origin and have zero constraint violation are all contained in $\bar{X}$, which is much smaller than $\mathcal{X}$. Accordingly, in option 2, we concentrate on X_k, which converges to $\bar{X}$ as k goes to infinity.

On the other hand, option 2 needs to compute the k-step controllable set Ω_k (X_k) while option 1 does not. The property and computation of X_k under the UoP state constraint is given in [34, Lemma 1]. Therefore, for large-scale PWA systems, if the exact computation of each X_k is computationally very demanding, option 1 is preferable.

Comparison to RL With a Safety Filter: In the schemes of [13] and [14], RL policies are projected onto a safe set where both state and input constraints are considered. That design is motivated from the fact that the policies in [13] and [14] are derived from standard RL algorithms that do not account for constraints. In comparison, our proposed ADP algorithms incorporate the state constraints into the cost function by adding penalty terms for violating the constraints. Besides, input constraints are regarded as hard constraints in the optimization problems (12), (14), and (16). The optimal value function $J^{\text{soft}*}(\cdot)$ with penalties should be less than or equal to the optimal value function $J^*(\cdot)$ of the original constrained optimal control problem (1), as the optimal solution to (1) is consistently feasible for the unconstrained optimal control problem in Theorem 1. However, the potential over-optimality may result from minor violations of the state constraints. Therefore, in Section V we provide a tool to offline verify the state constraint satisfaction.

V. Performance Analysis and Verification

In this section, we will characterize the stability and safety of the closed-loop system with the policies $\hat{\pi}^{\text{im}}(\cdot)$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$, and also the suboptimality properties of these policies. First, we provide general conditions under which stability and safety hold. These conditions can guide the parameter tuning of Algorithm 2. Then, we give suboptimality guarantees, i.e., a bound on the mismatch between the infinite cost of the policies and real value functions. Finally, we develop verifiable stability and safety conditions. We say that a closed-loop system is safe in a set if its states and inputs satisfy the constraints for all trajectories starting from the set.

A. Stability and Safety Analysis

First, we state a useful lemma that gives some properties of the value function $J_k^{\text{soft}}(\cdot)$.

Lemma 1: Consider Algorithm 1. Suppose that Assumptions 1 and 2 hold in option (a) and Assumptions 1–3 hold in option (b).

(i) Then, there exists a positive constant $\gamma < \infty$ such that for all $k \geq 0$, $J_k^{\text{soft}}(x) \leq \gamma l(x, 0) \quad \forall x \in \bar{X}$;

(ii) there exists a finite $\bar{k} > 0$ such that $\forall k \geq \bar{k}$, we have

$$J_k^{\text{soft}}(x) - J_{k-1}^{\text{soft}}(x) \leq \beta l(x, 0) \quad \forall x \in \bar{X}, \text{ with } \beta \in (0, 1). \tag{17}$$

The proof of Lemma 1 is provided in [34]. The following theorem states the main result in this section.

Theorem 2: Consider Algorithm 2 and the proposed policies $\hat{\pi}^{\text{im}}(\cdot)$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$. Let Ω be a compact subset of X. Suppose that Assumptions 1 and 2 hold in option (a) and Assumptions 1–3 hold in option (b). Consider the follow conditions.

(C1): There exist a constant $\zeta \in (0, 1)$ and a positive integer k such that $|\hat{J}_{k-1}(x) - J_{k-1}^{\text{soft}}(x)| \leq \zeta J_{k-1}^{\text{soft}}(x) \quad \forall x \in \Omega$.

(C2): There exist a constant $e_p > 0$ and a positive integer k such that $\hat{J}_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) - \hat{J}_{\hat{\pi}^{\text{im}}}(x) \leq e_p l(x, 0) \quad \forall x \in \Omega$. Here, $\hat{J}_\pi(\cdot)$ is defined by $\hat{J}_\pi(x) \triangleq l(x, \pi(x)) + \hat{J}_{k-1}(f_{\text{PWA}}(x, \pi(x)))$.

As a result, we have the following.

(i) If C1 holds with $k \geq \bar{k}$ and

$$(1 + \zeta)(1 - \beta) > \max(2\zeta\gamma,\ 1) \tag{18}$$

where $\bar{k}$, β, and γ come from Lemma 1, the closed-loop system $x_{t+1} = f_{\text{PWA}}(x_t, \hat{\pi}^{\text{im}}(x_t)),\ t = 0, 1, \ldots,$ is asymptotically stable and safe in $\mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$.

(ii) If C1 and C2 hold with $k \geq \bar{k}$, and

$$(1 + \zeta)(1 - \beta) > \max\left(2\zeta\gamma + e_p,\ 1\right) \tag{19}$$

the closed-loop system $x_{t+1} = f_{\text{PWA}}(x_t, \hat{\pi}^{\text{ex}}_{\text{proj}}(x_t)),\ t = 0, 1, \ldots,$ is asymptotically stable and safe in $\mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$.

The proof of Theorem 2 is given in Appendix B.

Remark 3: Although Theorem 2 provides sufficient conditions for stability, some of them (e.g., C1 and C2) are difficult to verify. A method that can verify C1 and C2 in a probabilistic way is reported in [12]. On the other hand, Theorem 2 suggests several ways to design the parameters and function approximators in Algorithm 2. For practical verification of stability and safety, in Section V-C, we move beyond using Conditions C1 and C2. Instead, we propose an offline verification framework based on solving MILP problems. As will be demonstrated in the case study, this framework effectively verifies safe and stable regions.

1) All results in Theorem 2 require (17) to hold. The left-hand side of (17) is about the residual error of VI. It indicates that a suitable tolerance function $\epsilon(\cdot)$, which determines the stopping condition at step 9 of Algorithm 2, could be $\epsilon(x) = e_{\text{tole}} l(x, 0)$, for some $e_{\text{tole}} \in (0, 1)$.
2) Condition C1 limits the mismatch between $\hat{J}_{k-1}(\cdot)$ and $J^{\text{soft}}_{k-1}(\cdot)$, which further limits the approximation error $\upsilon_i(x) \triangleq \hat{J}_i(x) - \Gamma_{p,\alpha}\hat{J}_{i-1}(x)$, $i = 1, \ldots, k$ of VI. To make ζ as small as possible, which helps to fulfill (18) and (19), $\rho_v(\cdot)$ in (13) could be specified by $\rho_v(x) = 1/l^2(x, 0)$. To understand this, we consider the state trajectory x_0, $x_1, \ldots, x_k$ that is generated from the closed-loop system $x_{t+1} = f_{\text{PWA}}(x_t, \pi_{k-t}(x_t))$, $t = 0, \ldots, k-1$, where $\pi_i(\cdot)$ denotes the optimizer of $\Gamma_{p,\alpha} J^{\text{soft}}_{i-1}$. Then, we have $\hat{J}_k(x_0) - J^{\text{soft}}_k(x_0) \le \hat{J}_{k-1}(x_1) - J^{\text{soft}}_{k-1}(x_1) + \upsilon_k(x_0) \le \hat{J}_0(x_k) - J^{\text{soft}}_0(x_k) + \sum_{i=1}^{k} \upsilon_i(x_{k-i})$. Suppose that $|\hat{J}_0(x_k) - J^{\text{soft}}_0(x_k)| \le e_{\text{v}} l(x_k, 0)$ and $|\upsilon_i(x_{k-i})| \le e_{\text{v}} l(x_{k-i}, 0)$, $i = 1, \ldots, k$, where $e_{\text{v}} > 0$, we obtain
$$\hat{J}_k(x_0) - J^{\text{soft}}_k(x_0) \le e_{\text{v}} \sum_{i=0}^{k} l(x_i, 0) \le e_{\text{v}} J^{\text{soft}}_k(x_0). \quad (20)$$
Similar procedures can be applied to upper bound the value of $J^{\text{soft}}_k(x_0) - \hat{J}_k(x_0)$. From (11) and (13), we see that compared to letting $\rho_v(x) = 1$, our choice of $\rho_v(\cdot)$ is more likely to lead to a smaller e_{v}, which can contribute to the reduction of ζ, according to (20). Moreover, to circumvent the singularity of $\rho_v(\cdot)$ at the origin, we let $\rho_v(x) = 1/(l^2(x, 0) + \rho)$, with a small constant $\rho > 0$.
3) Condition C2 requires the policy approximation error $\hat{J}_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) - \hat{J}_{\hat{\pi}^{\text{im}}}(x)$ is constrained by the state cost $l(x, 0)$ scaled by a constant e_p and that the constant e_p is small enough. Based on this requirement, we take $\rho_\pi(x) = 1/(l(x, 0) + \rho)$ to make e_p small.

Remark 4: Different from the existing stability results on ADP [18], [19], [20], [37], Theorem 2 considers the effects of state constraints. Besides, condition C1 allows us to analyze the suboptimality of $\hat{\pi}^{\text{im}}(\cdot)$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$, which has not been addressed in existing research [18], [19], [20], [37].

B. Suboptimality Analysis

Based on Theorem 2, we can compute upper bounds on $J_{\hat{\pi}^{\text{im}}}(\cdot)$ and $J_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(\cdot)$, which are defined in (1).

Corollary 1: Consider Algorithm 2 and the proposed policies $\hat{\pi}^{\text{im}}(\cdot)$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$. Let Ω be a compact subset of X.

(i) Let the assumptions in (i) of Theorem 2 hold and $2\zeta\gamma < 1$. Then, for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$, we have the inequality
$$J^{\text{soft}*}(x) \le J_{\hat{\pi}^{\text{im}}}(x) \le \frac{1-\zeta}{1-2\zeta\gamma} J^{\text{soft}}_{k-1}(x). \quad (21)$$

(ii) Let the assumptions in (ii) of Theorem 2 hold and $2\zeta\gamma + e_p < 1$. Then, for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$, we have the inequality
$$J^{\text{soft}*}(x) \le J_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) \le \frac{1-\zeta}{1-2\zeta\gamma-e_p} J^{\text{soft}}_{k-1}(x). \quad (22)$$

It is almost impossible to get an optimal control policy from (15) due to the approximation error. However, (21) and (22) confirm the intuition that a smaller approximation error of the critic (and the actor) leads to tighter suboptimality guarantees. Namely, as $\zeta \to 0$, $e_p \to 0$, and $k \to \infty$, we have $J_{\hat{\pi}^{\text{im}}}(x) \to J^{\text{soft}*}(x)$ and $J_{\hat{\pi}^{\text{ex}}_{\text{proj}}}(x) \to J^{\text{soft}*}(x)$ for any $x \in \mathcal{B}(J^{\text{soft}}_{k-1}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$. Moreover, if option (a) of Algorithm 2 is used, $J^{\text{soft}}_{k-1}(x)$ in (21) and (22) can be replaced by $J^{\text{soft}*}(x)$ because $J^{\text{soft}}_{k-1}(x) \le J^{\text{soft}*}(x) \quad \forall x \in \mathcal{X}$.

C. Stability and Safety Verification

As mentioned in the previous section, conditions C1 and C2 in Theorem 2 can only be evaluated statistically. Moreover, the conditions in Theorem 2 are sufficient conditions for $\hat{J}_{k-1}(\cdot)$ and $J^{\text{soft}}_{k-1}(\cdot)$ to be Lyapunov functions, and thus these conditions can be conservative. Besides, sometimes only practical stability can be ensured for nonlinear systems with neural controllers [19]. In this section, we propose an offline verification framework to simultaneously verify the practical stability and safety of the system controlled by the projected policy $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ in a deterministic manner, based on MILP. A small adaption that verifies the asymptotic stability for $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ and $\hat{\pi}^{\text{im}}(\cdot)$ is provided at the end of this section.

The proposed verification procedure contains three steps. Different from [25] and [26] that directly verify asymptotic stability, for practical stability we need to first verify the convergence of the closed-loop system to a neighborhood containing the origin, and then verify the invariance of the neighborhood. These will be formulated as two MILP problems. Finally, to enlarge the inner-estimated region of attraction, which is a sublevel set of $\hat{J}_{k-1}(\cdot)$, the third MILP problem will be formulated.

After implementing Algorithm 2 and (15), we have the value function $\hat{J}_{k-1}(\cdot)$ and the explicit policy $\hat{\pi}^{\text{ex}}(\cdot, \omega^*)$ at our disposal. To verify the stability in any sublevel set $\mathcal{B}_{r_1} = \{x \in \mathcal{X} | \hat{J}_{k-1}(x) \le r_1,\ r_1 > 0\}$ that is contained in X, we formulate the following optimization problem:
$$\begin{aligned} a^*_1 = \max_{x,u} \quad & \hat{J}_{k-1}(f_{\text{PWA}}(x, u)) - \hat{J}_{k-1}(x) + c_1 l(x, 0) \\ \text{s.t.} \quad & u = \hat{\pi}^{\text{ex}}_{\text{proj}}(x),\ r_2 \le \hat{J}_{k-1}(x) \le r_1 \end{aligned} \quad (23)$$
where c_1 is a small positive parameter, and $r_2 \in (0, r_1)$. If $a^*_1 \le 0$, we can conclude that the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ is safe in $\mathcal{B}_{r_1}$, and that any trajectories starting in $\mathcal{B}_{r_1}$ will enter $\mathcal{B}_{r_2} = \{x \in \mathcal{X} | \hat{J}_{k-1}(x) \le r_2, \}$ in finite time. The formal

result is included in Theorem 3. If $a_1^* > 0$, we can reduce the values of r_1 and c_1. In this way, the objective function of problem (23) and the feasible region of x become smaller, which will make a_1^* smaller.

As all functions in (23) are PWA and thus MILP representable, problem (23) can be formulated as an MILP problem.

After the trajectories reach $\mathcal{B}_{r_2}$, we need to verify that they will always stay in $\mathcal{B}_{r_2}$, i.e., we need to prove the positive invariance of $\mathcal{B}_{r_2}$. This leads to the second MILP problem

$$\begin{aligned} a_2^* = \max_{x,u} \quad & \hat{J}_{k-1}(f_{\mathrm{PWA}}(x,u)) - c_2\hat{J}_{k-1}(x) - r_2 + r_2c_2 \\ \text{s.t.} \quad & u = \hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(x),\ 0 \le \hat{J}_{k-1}(x) \le r_2 \end{aligned} \tag{24}$$

where $c_2 \in [0, 1]$. One can directly take $c_2 = 0$ to minimize a_2^*. Clearly, $a_2^* \le 0$ implies the positive invariance of $\mathcal{B}_{r_2}$, which will be proven in Theorem 3. If $a_2^* > 0$, similarly we can make r_2 smaller.

However, the stable and safe region $\mathcal{B}_{r_1}$ derived from (23) and (24) is usually small, as $\mathcal{B}_{r_1}$ is a sublevel set of $\hat{J}_{k-1}(\cdot)$. In particular, if the weights in Q on different states vary greatly, the resulting $\mathcal{B}_{r_1}$ will be rather narrow and much smaller than the real region of attraction. It could also happen that the evolution of the closed-loop system does not make $\hat{J}_{k-1}(\cdot)$ decrease at the beginning, but will drive the state into $\mathcal{B}_{r_1}$ in a finite number of time steps. Besides, in most cases we are interested in the performance of a policy in a polyhedron (or a UoP), rather than a sublevel set.

Therefore, we further develop the third optimization problem that evaluates the range of trajectories of the closed-loop system in a finite number of time steps for all initial states in a polyhedron X_{in} or a UoP of interest. The problem is

$$\begin{aligned} &\text{Check if } x_t \in X,\ t = 1, \ldots, N-1 \quad \forall x_0 \in X_{\mathrm{in}} \\ &\text{and if } \hat{J}_{k-1}(x_N) \le r_1 \quad \forall x_0 \in X_{\mathrm{in}} \\ &\text{s.t. } u_t = \hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(x_t),\ x_{t+1} = f_{\mathrm{PWA}}(x_t, u_t),\ t = 0, \ldots, N-1 \end{aligned} \tag{25}$$

where N is the number of time steps and r_1 is such that it makes $a_1^* \le 0$ in (23). If (25) returns "Yes," we can conclude that for any initial state in X_{in}, the states of the closed-loop system will satisfy the constraints from $t = 0$ to $t = N - 1$, and the final state x_N will reach the stable and safe region $\mathcal{B}_{r_1}$ computed in (23). Similar to (23) and (24), (25) can be exactly expressed in an MILP form. If X_{in} is a UoP, we also need some additional binary variables to formulate the initial condition $x_0 \in X_{\mathrm{in}}$ (see Appendix C of the Arxiv paper [34]).

The integration of (23)–(25) constitutes the proposed verification framework, which computes the exact evolution of the closed-loop system. It does not need any sampling or statistical testing procedure. The effectiveness of the proposed verification framework is stated in the following theorem, of which the proof is provided in Appendix C.

Theorem 3: Consider the policy $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(\cdot)$ and the proposed verification framework consisting of (23)–(25). If $\hat{J}_{k-1}(0) = 0$, $a_1^* \le 0$, $a_2^* \le 0$, and (25) returns Yes, then the closed-loop system with $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(\cdot)$ is safe in X_{in}, and any trajectory starting from X_{in} will approach $\mathcal{B}_{r_1}$ in at most $N + \lceil (r_1 - r_2)\hat{\gamma}/(c_1 r_2) \rceil$ time steps and stay in $\mathcal{B}_{r_1}$ thereafter. Here, $\hat{\gamma}$ is a positive constant independent of the initial condition.

With the results in Theorems 2 and 3, we now make practical suggestions on how to implement Algorithm 2 and the proposed verification framework. After the explicit policy $\hat{\pi}^{\mathrm{ex}}(\cdot, \omega^*)$ is obtained, we first find r_2 that makes $a_2^* \le 0$. Then, one can use (24) to compute the safe and stable region and then use (25) to enlarge it. If $a_2^* \le 0$ but $a_1^* > 0$ whatever r_1 is chosen, one needs to apply (25) with r_1 replaced by r_2. The cost is that one may have to choose a large horizon N since r_2 is small. The complexity of (25) grows exponentially w.r.t. N. If $a_1^* > 0$ and $a_2^* > 0$ no matter what r_1 and r_2 are chosen, one have to refine the learning process. To this end, Theorem 2 implies that one can either: 1) increase the number of iterations by tightening the stopping condition, i.e., making e_{tole} smaller, or 2) improve the approximation quality of function approximators and restart Algorithm 2.

Remark 5: The proposed verification framework generalizes the verification methods in [10], [25], and [26], and is an extension of [10], [25], and [26] from linear systems to PWA systems and from asymptotic stability to practical stability. Specifically, if we remove (24) and (25) and fix $r_2 = 0$, the proposed method verifies asymptotic stability, which is stronger than the properties in Theorem 3, and it is then similar to the method in [25]. For the comparison with [26], we note that to guarantee state constraint satisfaction, [26] needs to find a positively invariant set. In comparison, X_{in} in (25) is not necessarily positively invariant. Besides, (25) includes the case when X and the searching region X_{in} are UoPs.

In addition, (23) can be adjusted to verify the asymptotic stability and safety of $\hat{\pi}^{\mathrm{im}}(\cdot)$. With $r_2 = 0$ and $c_1 l(x, 0)$ replaced by $c_1 l(x, u)$, $a_1^* \le 0$ tells that $\hat{J}_{k-1}(\cdot)$ is a control Lyapunov function, which further validates the asymptotic stability and safety of $\hat{\pi}^{\mathrm{im}}(\cdot)$ in $\mathcal{B}_{r_2}$.

D. Overall Design Procedure

We provide a roadmap (see Fig. 1) outlining the procedures for implementing the proposed methods. The flowcharts within the blue box depict offline processes, while the flowcharts in the pink box represent online steps. After Algorithm 2 is implemented, two options can be chosen. One can either learn the explicit policy via (15) or bypass this step. Then, the proposed verification tool in Section V-C is used to assess the performance of the policy. If the policy falls short of expectations, it becomes necessary to refine the learning process and restart Algorithm 2. After verifying the safety and stability of the policy $\hat{\pi}^{\mathrm{ex}}(\cdot, \omega)$ or $\hat{\pi}^{\mathrm{im}}(\cdot)$, in the online phrase we solve (14) or (16) to compute the control input $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(x_t)$ or $\hat{\pi}^{\mathrm{im}}(x_t)$ when the state measure x_t is received.

The control framework involves four types of parameters: 1) the hyperparameters of the learning models for the value function and the policy; 2) the penalty weight p; 3) the parameter e_{tole} of the tolerance function $\epsilon(\cdot)$ determining the stopping criterion of Algorithm 2; and 4) the parameters r_1, r_2, and N of the verification framework. Guidelines for setting hyperparameters of the learning models can be found in machine learning literature, such as [38]. The principle

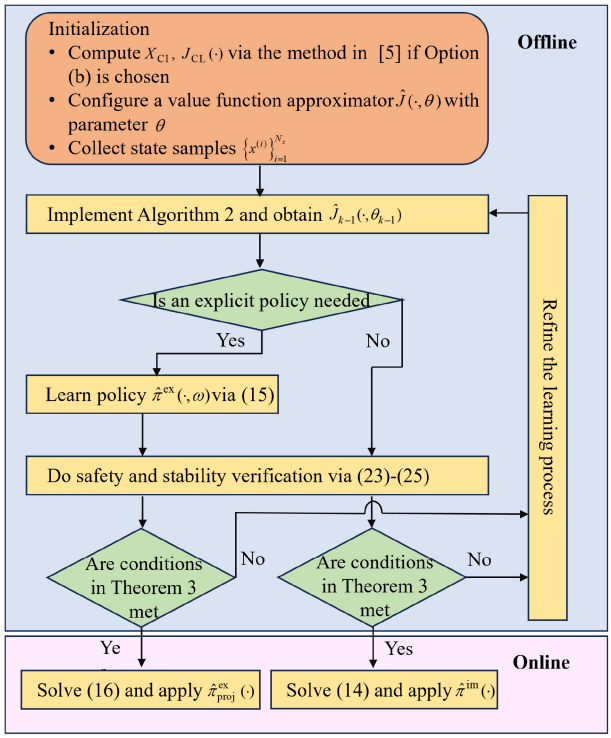

Fig. 1. Schematic of the design procedure.

for tuning r_1, r_2, and N has been provided after Theorem 3. To refine the learning process, one can adjust the first three kinds of parameters by the following steps: 1) improving the approximation quality of the learning models by tuning the hyperparameters or amplifying samples; 2) raising p; and 3) increasing the number of iterations by tightening the stopping condition (e_{tole} smaller).

VI. Case Study

We validate the proposed methods on an active inverted pendulum between elastic walls [see Fig. 2(a)]. We illustrate the convergence of different options of the proposed ADP method, the effectiveness of the verification method, as well as the online computational advantage of the proposed controllers.

By linearizing the dynamics around the vertical configuration $q = \dot{q} = 0$ and discretizing the system with a sampling time 0.05 s, we obtain a discrete-time PWA model with 4 modes, where the system matrices are given by

$$A_i = \begin{bmatrix} 1 & 0.05 \\ \alpha_i & 1 \end{bmatrix}, B_i = \begin{bmatrix} 0 \\ 0.05 \end{bmatrix}, f_i = \begin{bmatrix} 0 \\ \beta_i \end{bmatrix}, i = 1, 2, 3, 4$$

with $\alpha_1 = -29.5$, $\alpha_2 = -14.5$, $\alpha_3 = 0.5$, $\alpha_4 = -24.5$, $\beta_1 = -3.3$, $\beta_2 = -1.5$, $\beta_3 = 0$ and $\beta_4 = 2.5$, and where the partition $\{\mathcal{C}_i\}_{i=1}^4$ is given by $\mathcal{C}_1 = \{(x,u) \mid [1\ \ 0]x \le -0.12\}$, $\mathcal{C}_2 = \{(x,u) \mid -0.12 \le [1\ \ 0]x \le -0.1\}$, $\mathcal{C}_3 = \{(x,u) \mid -0.1 \le [1\ \ 0]x \le 0.1\}$ and $\mathcal{C}_4 = \{(x,u) \mid [1\ \ 0]x \ge 0.1\}$. The detailed description of the system is given in [34].

The inverted pendulum system is supposed to satisfy the constraints $[-0.15\ \ -1]^T \le x \le [0.15\ \ 1]^T$ and $-4 \le u \le 4$. The stage cost is $l(x,u) = ||\mathrm{diag}([20\ \ 1])x||_\infty + ||u||_\infty$. The overall control objective is to solve the infinite-horizon optimal control problem (1). The offline procedure for computing the proposed control policies $\hat{\pi}^{\mathrm{im}}$ and $\hat{\pi}^{\mathrm{ex}}$ includes solving several optimization problems. In particular, implementing Algorithm 2 requires solving the MILP problem (12) and the nonlinear regression problem (13). Computing $\hat{\pi}^{\mathrm{ex}}$ requires solving the policy optimization problem (15). Besides, the verification step contains solving three MILP problems (23)–(25).

Starting from a zero value function and after 10 iterations, the closed-loop behavior of the system with $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(\cdot)$ is illustrated in Fig. 2(c) and (d). Fig. 2(c) plots the trajectories of the closed-loop system (controlled by $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(\cdot)$) starting from some vertices of $\bar{X}$ in the state space, while Fig. 2(d) displays the time-domain responses corresponding to the state-space trajectories of Fig. 2(c). Although states starting from these vertices in general have the largest $J^*(\cdot)$ in their neighbors, and thus they are the most difficult to regulate to the origin, they converge rapidly under $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(\cdot)$. Meanwhile, the state constraints could be slightly violated: the trajectory depicted by the purple curves violates the constraints by about 5 percent. To avoid this, one can tighten the state constraints. The dashed purple curve describes the trajectory starting from the purple vertex with 10 percent constraint tightening.[2] We can observe that constraint violation is avoided.

The optimal value function $J^*(\cdot)$ is computed by the MPT3 toolbox [39] in 9.6 h. The learning processes of different versions of Algorithm 2 are compared in Fig. 2(b), in which the mean square errors between $\hat{J}_{k-1}(x^{(i)},\theta_k)$ and $J^*(x^{(i)})$ are depicted. The value function approximation in option (b) has a much faster convergence rate than that in option (a), and option 2 results in lower approximation errors than option 1. The accelerated convergence rate in option (b) is mainly attributed to its initial value function approximation $\hat{J}_0(\cdot,\theta_0)$ being considerably closer to $J^{\mathrm{soft}*}(\cdot)$ compared to option (a). The ultimately reduced approximation errors in option 2 likely stem from a more densely sampled state space.

The conditions for the stability and safety of the policy $\hat{\pi}^{\mathrm{ex}}_{\mathrm{proj}}(\cdot)$ are verified in Fig. 2(e). Fig. 2(e) depicts the safe and stable regions that are analyzed by the proposed verification framework. The input constraints are always satisfied in the simulation because the optimization problems (14) and (16) contain the input constraints as hard constraints. The blue region represents $\mathcal{B}_{r_1}$, which is computed by (23) with $r_1 = 18$, $c_1 = 0.1$, $r_2 = 3$. We note that for some states in $\mathcal{B}_{r_2}$, which is colored red, the objective function in (23) becomes positive, so the verification method in [25] fails. However, (24) outputs a negative a_2^* with $c_2 = 0.1$, which means that any trajectory of the closed-loop system starting from $\mathcal{B}_{r_1}$ will reach in the neighborhood $\mathcal{B}_{r_2}$ containing the origin in finite number of time steps. Furthermore, the safe and stable region $\mathcal{B}_{r_1}$ is enlarged to X_{in} (the yellow region) by (25) with $N = 3$. However, we observe that the verified safe and stable polytope X_{in} may be conservative compared to our trajectory simulation

[2]We leverage a backtracking strategy to incrementally increase the constraint tightening factors until the safety performance is successfully verified by our proposed verification method.

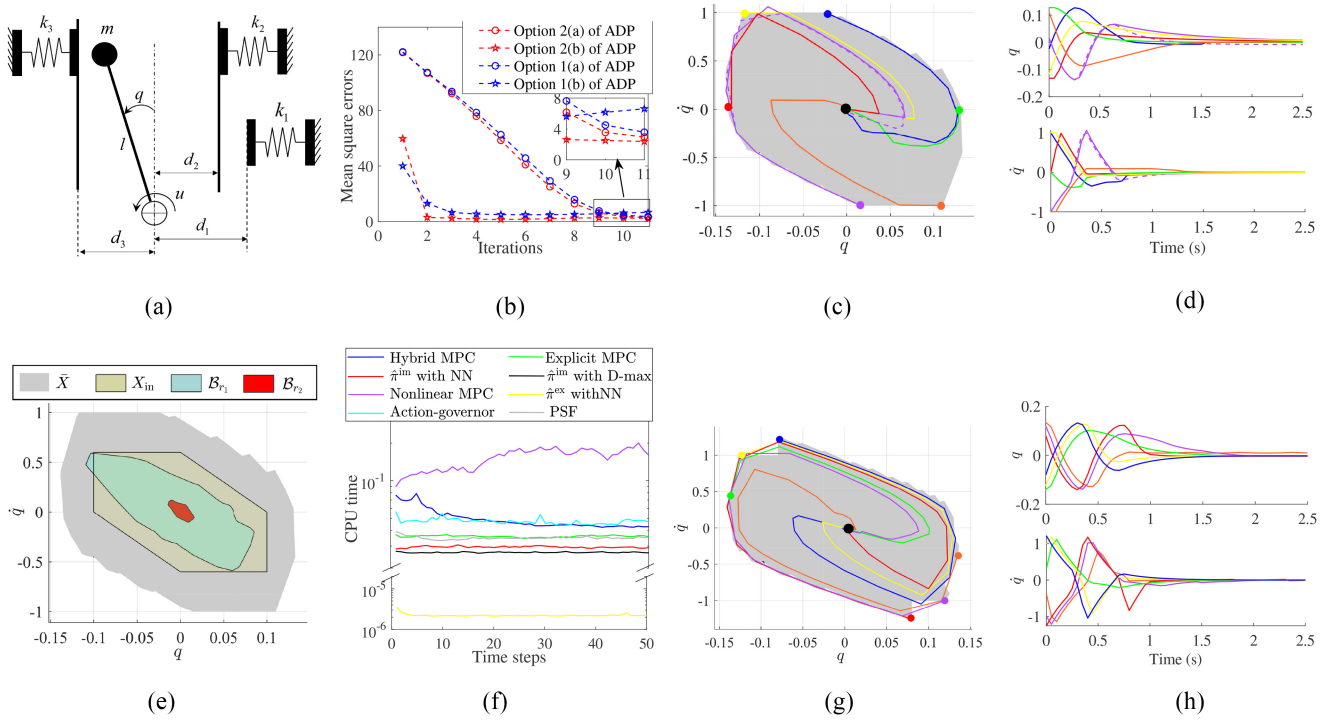

Fig. 2. Simulation results. (a) Inverted pendulum with elastic walls. (b) Learning process of the ADP algorithm with different options. (c) Closed-loop trajectories. The shaded region is $\bar{X}$. (d) Time-domain system responses. (e) Safe and stable regions verified by the verification framework. (f) Statistical analysis of different methods regarding CPU times. (g) Closed-loop trajectories under UoP state constraints. (h) Time-domain system responses under UoP state constraints.

in Fig. 2(c). Suppose that $X_{\text{in}} \triangleq \{x \in \mathbb{R}^{n_x} | E_{X_{\text{in}}} x \leq g_{X_{\text{in}}}\}$. The conservatism is primarily attributed to the naive choice of $E_{X_{\text{in}}}$. Practical strategies to mitigate this conservatism involve using a UoP X_{in} and extending the horizon N.

We compare the CPU time of running the ADP-based controllers, hybrid MPC, explicit MPC computed by the MPT3 toolbox, nonlinear MPC that uses the nonlinear model, action-governor RL in [14], and the PSF in [13]. The horizons of MPC and the PSF are taken as 8. The implementation of hybrid MPC and the PSF requires to solve an MILP problem. In comparison, for $\hat{\pi}^{\text{im}}(\cdot)$ with the value function approximator chosen as the ReLU NN (or the difference of two max-affine functions that has 15 and 5 terms in the first and second max blocks, respectively), one should solve a smaller MILP problem than the ones of hybrid MPC and the PSF. For statistical analysis, we randomly select 100 initial states and run the system for 50 time steps. According to the results in Fig. 2(f), the computation of $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ requires the least amount of CPU time, which is around 2.2×10^{-6} s per time step. Besides, $\hat{\pi}^{\text{im}}(\cdot)$ performs better than hybrid MPC, explicit MPC, action-governor RL, and PSF, regarding online computation time, both when using the ReLU NN (about 0.028 s) or the difference of two max-affine functions (about 0.026 s) for value function approximation.

We further consider the case when the state constraints are a UoP, namely, $x \in X^{(1)} \cup X^{(2)}$ with $X^{(1)} = \{x | [-0.15 \quad -1]^T \leq x \leq [0.15 \quad 1]^T\}$ and $X^{(2)} = \{x | [-0.08 \quad -1.5]^T \leq x \leq [0.08 \quad 1.5]^T\}$. After implementing option 2(a) of Algorithm 2 in 10 iterations, the trajectories of the closed-loop system with $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ starting from some vertices of $\bar{X}$ are plotted in Fig. 2(g)–(h), from which one can find that the proposed scheme is still valid even when the state constraints are a UoP.

Besides, a case study on a centralized adaptive cruise control problem, which contains a 6-D PWA system with 8 modes, is provided in [34]. We train the proposed policies $\hat{\pi}^{\text{im}}(\cdot)$ and $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ using ReLU neural networks with various sizes. These policies are compared with MPC and the PSF in [13]. Performance metrics, including CPU time, total cost, and safety rate are considered. The results indicate that the proposed policies ensure the safety of the system with a 99% probability, and their total costs are only 5% to 10% higher than the MPC total costs. Besides, the average CPU time for $\hat{\pi}^{\text{ex}}_{\text{proj}}(\cdot)$ is about 8×10^{-4} s, significantly lower than that of both MPC and the PSF, which operate on the scale of 0.1 s.

VII. Conclusion and Future Work

We have proposed an ADP control scheme to deal with infinite-horizon optimal control of PWA systems subject to linear and UoP constraints, based on MILP. With carefully designed PWA penalty functions, the probably nonconvex UoP constraints during the learning process are removed while the PWA properties of the value functions are maintained. We have formally analyzed the PWA properties and continuity of the value function, as well as the closed-loop stability and safety under the approximation errors. We have also designed an offline verification tool to make the proposed method reliable. Simulation results show that the ADP-based policies are near-optimal, and require much less online computational effort than conventional hybrid MPC. The limitations of the proposed ADP method are threefold. The performance of the policies depends heavily on the approximation accuracy of value

functions. Second, our method does not scale well to high-dimensional problems due to the dramatic growth of sampling complexity. In addition, our method is limited to the cases involving UoP state constraints and linear input constraints. Therefore, topics for future work include eliminating the reliance on value function approximation, exploring more efficient sampling strategies, and considering general convex multitime-step constraints.

Appendix A
Proof of Theorem 1

The proof contains three parts.

1) Convergence of the VI Sequence in Option 1: Option 1(a): For the nonlinear parametric problem in (8), U is closed, and the set $\{u \in U |\ J_k^{\text{soft}}(x) = l(x,u) + P(x,X) + J_{k-1}^{\text{soft}}(f_{\text{PWA}}(x,u))\}$ is nonempty for any x in $\mathcal{X}$. Then, according to [40, Th. 4.2.1], $J_k^{\text{soft}}(\cdot)$ is lower-semicontinuous on $\mathcal{X}$. Hence, $\{x \in \mathcal{X} | J_k^{\text{soft}}(x) \le \lambda\}$ with $\lambda \in \mathbb{R}$ are closed. As $l_p(\cdot,\cdot)$ and $f_{\text{PWA}}(\cdot,\cdot)$ are continuous on $\mathcal{X} \times \mathcal{U}$, the set $U_k(x,\lambda) = \{u \in U |\ l(x,u) + P(x,X) + J_{k-1}^{\text{soft}}(f_{\text{PWA}}(x,u)) \le \lambda\}$ is closed and thus compact for all $x \in \mathcal{X}$, $\lambda \in \mathbb{R}$, and $k \ge 1$. Therefore, the compactness assumption in [30] is satisfied and $\{J_k^{\text{soft}}(\cdot)\}_{k=0}^{\infty}$ converges point-wise to $J^{\text{soft}*}(\cdot)$ [30, Proposition 2].

Option 1(b): We define $J_0^{\infty}(x) = \begin{cases} J_{\text{CL}}(x), & x \in X_{\text{CI}} \\ \infty & x \notin X_{\text{CI}} \end{cases}$. The point-wise convergence of the VI $J_k^{\infty}(x) = \Gamma_{\text{p},1} J_{k-1}^{\infty}(x)$ to $J^{\text{soft}*}(x)$ is always guaranteed by [30, Proposition 2] since $J_0^{\infty}(x) \ge J^{\text{soft}*}(x) \quad \forall x \in \mathcal{X}$. By applying the monotonicity of the Bellman operator $\Gamma_{\text{p},1}$ [29], we get the convergence of the VI sequence $\{J_k^{\text{soft}}(\cdot)\}_{k=0}^{\infty}$ to $J^{\text{soft}*}(\cdot)$ in option 1(b).

2) Continuity and PWA Property of the Value Function in Option 1: By recursively iterating (8), it is observed that $J_k^{\text{soft}}(x)$ can be derived via the following batch approach:

$$\begin{aligned} J_k^{\text{soft}}(x) = \min_{u_0,\dots,u_{k-1},\, x_0,\dots,x_k} & \sum_{i=0}^{k-1} l_p(x_i,u_i) + J_0^{\text{soft}}(x_k) \\ \text{s.t.} \quad & x_{i+1} = f_{\text{PWA}}(x_i,u_i), x_0 = x \\ & u_i \in U, i = 0,\dots,k-1. \end{aligned} \tag{26}$$

Option 1(a): The proof of continuity follows from the proof of [31, Corollary 17.2], because the objective function in (26) is continuous and there is no state constraint. The proof of the PWA property of $J_k^{\text{soft}}(\cdot)$ follows a similar approach to that of [31, Th. 17.3], with detailed exposition given in [34].

Option 1(b): We can prove that the initial value function $J_0^{\text{soft}}(\cdot)$ is continuous and PWA on $\mathcal{X}$ (the detailed proof is given in [34]). As a result, the remainder of the proof of the continuity and PWA property of $J_k^{\text{soft}}(\cdot)$ is similar to that in option 1(a). Thus, we have completed the proof of the statements of Theorem 1 in option 1.

3) Equivalence of the VI Sequences for Options 1 and 2: In option 2, by iterating $J_k^{\text{soft}}(x) = \Gamma_{\text{p},2} J_{k-1}^{\text{soft}}(x)$ from k to 0, we can get the expression of $J_k^{\text{soft}}(\cdot)$ via the batch approach

$$J_k^{\text{soft}}(x) = \text{the optimal value of (26)} - P(x,X). \tag{27}$$

From (27), we notice that the difference between the value functions in options 1 and 2 at the same iteration is $P(x,X)$, which is always continuous in x and equals zero if $x \in X$. Combining (27) with the first and second parts of the proof proves the statements of Theorem 1.

Appendix B
Proofs of Theorem 2 and Corollary 1

For each $k \ge \bar{k}$, we define the policy $\pi_k(\cdot)$ as (one of) the optimizer(s) of $\Gamma_{p,\alpha} J_{k-1}^{\text{soft}}(\cdot)$, $\alpha = 1$ or 2. For every $x \in \mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \subseteq X$, according to (2)

$$J_k^{\text{soft}}(x) = \Gamma_{p,\alpha} J_{k-1}^{\text{soft}}(x) \ge l(x,\pi_k(x)) + J_{k-1}^{\text{soft}}(f_{\text{PWA}}(x,\pi_k(x))) \tag{28}$$

holds in both options. Together with (17), (28) yields

$$\begin{aligned} J_{k-1}^{\text{soft}}(f_{\text{PWA}}(x,\pi_k(x))) - J_{k-1}^{\text{soft}}(x) &\le -(1-\beta) l(x,\pi_k(x)) \\ &\le -(1-\beta) l(x,0) \end{aligned} \tag{29}$$

which means that for every $x \in \mathcal{B}(J_{k-1}^{\text{soft}}, \Omega)$, the policy $\pi_k(\cdot)$ will make $f_{\text{PWA}}(x,\pi_k(x)) \in \mathcal{B}(J_{k-1}^{\text{soft}}, \Omega)$.

Now, we show that $J_{k-1}^{\text{soft}}(\cdot)$ and $\hat{J}_{k-1}(\cdot)$ are Lyapunov functions for the system with $\hat{\pi}^{\text{im}}(\cdot)$. In particular, for any $x \in \mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$, if C1 and (18) hold, we have

$$\begin{aligned} & l\left(x, \hat{\pi}^{\text{im}}(x)\right) + \hat{J}_{k-1}\left(f_{\text{PWA}}(x, \hat{\pi}^{\text{im}}(x))\right) \\ & \le l(x,\pi_k(x)) + \hat{J}_{k-1}(f_{\text{PWA}}(x,\pi_k(x))) \\ & \le l(x,\pi_k(x)) + (1+\zeta) J_{k-1}^{\text{soft}}(f_{\text{PWA}}(x,\pi_k(x))) \\ & \le (1+\zeta) J_{k-1}^{\text{soft}}(x) + (1-(1+\zeta)(1-\beta)) l(x,\pi_k(x)) \\ & \le \hat{J}_{k-1}(x) + 2\zeta J_{k-1}^{\text{soft}}(x) + (1-(1+\zeta)(1-\beta)) l(x,\pi_k(x)). \end{aligned} \tag{30}$$

In (30), the first inequality is true since $\hat{\pi}^{\text{im}}(\cdot)$ is an optimizer of problem (14); the second and the last inequalities hold because x and $f_{\text{PWA}}(x,\pi_k(x))$ all in Ω, in which C1 holds; and the third inequality is correct owning to (29). Since $1-(1+\zeta)(1-\beta) < 0$, considering Lemma 1, (30) results in

$$\hat{J}_{k-1}\left(x^+\right) - \hat{J}_{k-1}(x) \le (2\zeta\gamma - (1+\zeta)(1-\beta)) l(x,0) \tag{31}$$

with $x^+ = f_{\text{PWA}}(x, \hat{\pi}^{\text{im}}(x))$. The right-hand side of (31) is strictly negative except for $x = 0$ according to (18). Therefore, we get that $\forall x \in \mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$, $f_{\text{PWA}}(x, \hat{\pi}^{\text{im}}(x)) \in \mathcal{B}(\hat{J}_{k-1}, \Omega)$ holds. Condition C1 implies that $\hat{J}_{k-1}(x) > 0 \quad \forall x \in (\mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)) \backslash \{0\}$ and $\hat{J}_{k-1}(0) = 0$. This shows that $J_{k-1}^{\text{soft}}(\cdot)$ is a Lyapunov function for the system $x_{t+1} = f_{\text{PWA}}(x_t, \hat{\pi}^{\text{im}}(x_t))$.

Similarly to (30) and (31), we can get the following inequality for $J_{k-1}^{\text{soft}}(\cdot)$:

$$\begin{aligned} & l\left(x, \hat{\pi}^{\text{im}}(x)\right) + J_{k-1}^{\text{soft}}\left(f_{\text{PWA}}(x, \hat{\pi}^{\text{im}}(x))\right) \\ & \le \frac{(1+\zeta) J_{k-1}^{\text{soft}}(x) + (1-(1+\zeta)(1-\beta)) l(x,\pi_k(x)) - \zeta l\left(x, \hat{\pi}^{\text{im}}(x)\right)}{1-\zeta}. \end{aligned} \tag{32}$$

With (18) and Lemma 1, (32) readily leads to

$$J_{k-1}^{\text{soft}}\left(x^+\right) - J_{k-1}^{\text{soft}}(x) \le \frac{2\zeta\gamma - (1+\zeta)(1-\beta)}{1-\zeta} l(x,0) \tag{33}$$

which means that $J_{k-1}^{\text{soft}}(\cdot)$ is strictly decreasing from any $x \in (\mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega))\backslash\{0\}$ to the next state. Combining (31) and (33), we note that $\mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega)$ is a positively invariant set. This together with the Lyapunov functions $\hat{J}_{k-1}(\cdot)$ and $J_{k-1}^{\text{soft}}(\cdot)$ leads to the asymptotic stability and safety of the closed-loop system.

Next, we analyze the behavior of the closed-loop system with $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$. With C1 and C2, $\hat{J}_{k-1}(x^+) - \hat{J}_{k-1}(x)$ with $x^+ = f_{\text{PWA}}(x, \hat{\pi}_{\text{proj}}^{\text{ex}}(x))$ is upper bounded by the right-hand side of (31) plus $e_p l(x, 0)$, and $J_{k-1}^{\text{soft}}(x^+) - J_{k-1}^{\text{soft}}(x)$ is also upper bounded by the right-hand side of (33) plus $e_p l(x, 0)$. Together with (19) this results in the asymptotic stability and safety of the closed-loop system with policy $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$.

Finally, to prove Corollary 1, we note that the first inequality in (21) and the first inequality in (22) directly follow from the optimality of $J^{\text{soft}*}(\cdot)$. We derive from (32) that

$$\frac{1-2\zeta\gamma}{1-\zeta} l\Big(x, \hat{\pi}^{\text{im}}(x)\Big) \le J_{k-1}^{\text{soft}}(x) - J_{k-1}^{\text{soft}}\Big(f_{\text{PWA}}(x, \hat{\pi}^{\text{im}}(x))\Big) \tag{34}$$

holds for any $x \in (\mathcal{B}(J_{k-1}^{\text{soft}}, \Omega) \cap \mathcal{B}(\hat{J}_{k-1}, \Omega))\backslash\{0\}$. Consider the trajectory $x_0, x_1, \ldots,$ that is generated by applying $\hat{\pi}^{\text{im}}(x_t)$ to system (2) at each time step t, $t = 0, 1, \ldots,$ Letting $x = x_t$ and summing up both sides of (34) from $t = 0$ to $t = \infty$, we get $J_{\hat{\pi}^{\text{im}}}(x_0) \le ([1-\zeta]/[1-2\zeta\gamma])(J_{k-1}^{\text{soft}}(x_0) - J_{k-1}^{\text{soft}}(x_\infty))$. The asymptotic stability in (i) of Theorem 2 indicates that $J_{k-1}^{\text{soft}}(x_\infty) = 0$, so (i) of Corollary 1 is proved. Similarly, we can upper bound the stage cost when applying $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$ by $([1-2\zeta\gamma-e_p]/[1-\zeta])l(x, \hat{\pi}_{\text{proj}}^{\text{ex}}(x)) \le J_{k-1}^{\text{soft}}(x) - J_{k-1}^{\text{soft}}(f_{\text{PWA}}(x, \hat{\pi}_{\text{proj}}^{\text{ex}}(x)))$. (ii) of Corollary 1 will be obtained by summing up the above inequality along the trajectory controlled by $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$.

APPENDIX C
PROOF OF THEOREM 3

If (25) returns Yes, the trajectories of the closed-loop system with $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$ with initial condition $x_0 \in X_{\text{in}}$ will be contained in $\mathcal{B}_{r_1}$ after N time steps. Then, for any initial state $x_0 \in \mathcal{B}_{r_1}$, suppose that the $t_{\text{f}} - 1$-step trajectory $x_0, x_1, \ldots, x_{t_{\text{f}}-1}$ of the closed-loop system with $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$ is not contained in $\mathcal{B}_{r_2}$. Since $a_1^* \le 0$, we have

$$\hat{J}_{k-1}(x_{t+1}) - \hat{J}_{k-1}(x_t) \le -c_1 l(x_t, 0), \ t = 0, \ldots, t_{\text{f}} - 1. \tag{35}$$

Summing (35) over time yields

$$\hat{J}_{k-1}\big(x_{t_{\text{f}}}\big) \le \hat{J}_{k-1}(x_0) - c_1 \sum_{t=0}^{t_{\text{f}}-1} l(x_t, 0). \tag{36}$$

Meanwhile, since both $\hat{J}_{k-1}(\cdot)$ and $l(x, 0)$ are continuous PWA functions on their domains, similarly to (i) of Lemma 1, there exists a positive and finite constant $\hat{\gamma}$ such that $\hat{J}_{k-1}(x) \le \hat{\gamma} l(x, 0)$ for all $x \in X$. As a result, (36) implies that $\hat{J}_{k-1}(x_{t_{\text{f}}}) \le r_1 - ([c_1 r_2 t_{\text{f}}]/\hat{\gamma})$. Specifying $t_{\text{f}} = \lceil (r_1 - r_2)\hat{\gamma}/(c_1 r_r) \rceil$, which is finite and does not depend on x_0, we have $\hat{J}_{k-1}(x_{t_{\text{f}}}) \le r_2$. Combining the above statements, we can conclude that any trajectory of the closed-loop system with $\hat{\pi}_{\text{proj}}^{\text{ex}}(\cdot)$ starting from X_{in} will reach $\mathcal{B}_{r_2}$ in less than $N + t_{\text{f}}$ time steps. Finally, the positive invariance of $\mathcal{B}_{r_2}$ is straightforward if $a_2^* \le 0$, since we have $\hat{J}_{k-1}(x) \le r_2 \Rightarrow \hat{J}_{k-1}(f_{\text{PWA}}(x, \hat{\pi}_{\text{proj}}^{\text{ex}}(x))) \le r_2$ from (24).

REFERENCES

[1] T. Liu, Y. Gao, and M. Buss, "Adaptive output tracking control of piecewise affine systems with prescribed performance," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 9, pp. 5398–5410, Sep. 2022.

[2] L. Gharavi, B. De Schutter, and S. Baldi, "Efficient MPC for emergency evasive maneuvers, part I: Hybridization of the nonlinear problem," 2023, *arXiv:2310.00715*.

[3] S. Sadraddini and R. Tedrake, "Sampling-based polytopic trees for approximate optimal control of piecewise affine systems," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 7690–7696.

[4] N. Groot, B. De Schutter, and H. Hellendoorn, "Integrated model predictive traffic and emission control using a piecewise-affine approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 587–598, Jun. 2013.

[5] M. Lazar, W. P. M. H. Heemels, S. Weiland, and A. Bemporad, "Stabilizing model predictive control of hybrid systems," *IEEE Trans. Autom. Control*, vol. 51, no. 11, pp. 1813–1818, Nov. 2006.

[6] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, 2005.

[7] F. Xiong et al., "Guided policy search for sequential multitask learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 216–226, Jan. 2019.

[8] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton, FL, USA: CRC Press, 2017.

[9] H. Jiang, H. Zhang, Y. Luo, and J. Han, "Neural-network-based robust control schemes for nonlinear multiplayer systems with uncertainties via adaptive dynamic programming," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 3, pp. 579–588, Mar. 2019.

[10] R. Schwan, C. N. Jones, and D. Kuhn, "Stability verification of neural network controllers using mixed-integer programming," 2022, *arXiv:2206.13374*.

[11] S. Chen et al., "Approximating explicit model predictive control using constrained neural networks," in *Proc. Annu. Amer. Control Conf. (ACC)*, 2018, pp. 1520–1527.

[12] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 543–548, Jul. 2018.

[13] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, Jul. 2021, Art. no. 109597.

[14] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, "Robust action governor for uncertain piecewise affine systems with non-convex constraints and safe reinforcement learning," 2022, *arXiv:2207.08240*.

[15] L. Beckenbach, P. Osinenko, T. Göhrt, and S. Streif, "Constrained and stabilizing stacked adaptive dynamic programming and a comparison with model predictive control," in *Proc. Eur. Control Conf. (ECC)*, 2018, pp. 1349–1354.

[16] J. Duan, Z. Liu, S. E. Li, Q. Sun, Z. Jia, and B. Cheng, "Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints," *Neurocomputing*, vol. 484, pp. 128–141, May 2022.

[17] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," 2018, *arXiv:1805.11074*.

[18] A. Heydari, "Stability analysis of optimal adaptive control using value iteration with approximation errors," *IEEE Trans. Autom. Control*, vol. 63, no. 9, pp. 3119–3126, Sep. 2018.

[19] R. Postoyan, M. Granzotto, L. Buşoniu, B. Scherrer, D. Nešsić, and J. Daafouz, "Stability guarantees for nonlinear discrete-time systems controlled by approximate value iteration," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, 2019, pp. 487–492.

[20] F. Moreno-Mora, L. Beckenbach, and S. Streif, "Predictive control with learning-based terminal costs using approximate value iteration," 2022, *arXiv:2212.00361*.

[21] S. Gros and M. Zanon, "Learning for MPC with stability & safety guarantees," *Automatica*, vol. 146, Dec. 2022, Art. no. 110598.

[22] K. He, T. V. D. Boom, and B. De Schutter, "Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach," 2022, *arXiv:2205.10065*.

[23] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 25636–25655.
[24] J. Li, D. Fridovich-Keil, S. Sojoudi, and C. J. Tomlin, "Augmented lagrangian method for instantaneously constrained reinforcement learning problems," in *Proc. 60th IEEE Conf. Decis. Control (CDC)*, 2021, pp. 2982–2989.
[25] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," 2021, *arXiv:2109.14152*.
[26] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, 2020, pp. 4947–4954.
[27] M. Baoti, F. J. Christophersen, and M. Morari, "Constrained optimal control of hybrid systems with a linear performance index," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1903–1919, Dec. 2006.
[28] J. Xu, T. van den Boom, L. Buşoniu, and B. De Schutter, "Model predictive control for continuous piecewise affine systems using optimistic optimization," in *Proc. Amer. Control Conf. (ACC)*, 2016, pp. 4482–4487.
[29] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Belmont, MA, USA: Athena Sci., 2019.
[30] D. P. Bertsekas, "Value and policy iterations in optimal control and adaptive dynamic programming," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 500–509, Mar. 2017.
[31] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2017.
[32] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, "An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems," in *Proc. Amer. Control Conf.*, vol. 6, 2003, pp. 4717–4722.
[33] B. Hanin, "Universal function approximation by deep neural nets with bounded width and ReLU activations," *Mathematics*, vol. 7, no. 10, p. 992, 2019.
[34] K. He, S. Shi, T. V. D. Boom, and B. De Schutter, "Approximate dynamic programming for constrained piecewise affine systems with stability and safety guarantees," 2023, *arXiv:2306.15723*.
[35] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*. Hoboken, NJ, USA: Wiley, 1999.
[36] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz, "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2736–2749, Jun. 2017.
[37] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, "Finite approximation error-based value iteration ADP," in *Adaptive Dynamic Programming with Applications in Optimal Control*. Cham, Switzerland: Springer, 2017, pp. 91–149.
[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[39] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-parametric toolbox 3.0," in *Proc. Eur. Control Conf. (ECC)*, Zürich, Switzerland, 2013, pp. 502–510. [Online]. Available: https://www.mpt3.org/
[40] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer, *Non-Linear Parametric Optimization*. Basel, Switzerland: Birkhäuser, 1983.

Kanghui He received the B.S. degree in mechanical engineering from the School of Mechanical Engineering and Automation and the M.S. degree (with Outstanding Graduation Thesis and Chinese National Scholarship) in dynamics and control from the Department of Flight Dynamics and Control, Beihang University, Beijing, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.

His research interests include learning-based control, model predictive control, hybrid systems, and their applications in mobile robots.

Shengling Shi received the B.Sc. degree in automotive technology from the Harbin Institute of Technology, Harbin, China, in 2013, the M.Sc. degree [with Distinction (great appreciation)] in automotive technology and the Ph.D. degree from the Control Systems Group from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2017 and 2021, respectively.

He is currently a Postdoctoral Researcher with the Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. From 2021 to 2024, he was a Postdoctoral Researcher with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His research interests include learning and system theory of networked systems.

Ton van den Boom received the M.Sc. and Ph.D. degrees in electrical engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1988 and 1993, respectively.

He is currently an Associate Professor with the Delft Center for Systems and Control Department, Delft University of Technology, CD, Delft, The Netherlands. His research focus is mainly in modeling and control of discrete event and hybrid systems, in particular max-plus-linear systems, max–min-plus-scaling systems, and switching max-plus-linear systems (both stochastic and deterministic), with applications in manufacturing systems and transportation networks.

Bart De Schuhtter (Fellow, IEEE) received the Ph.D. degree (summa cum laude) in applied sciences from KU Leuven, Leuven, Belgium, in 1996.

He is currently a Full Professor and the Head of Department with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His research interests include multilevel and multiagent control, model predictive control, learning-based control, and control of hybrid systems, with applications in intelligent transportation systems and smart energy systems.

Prof. de Schutter is a Senior Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.