

Technical report 24-002

Approximate Dynamic Programming for Constrained Linear Systems: A Piecewise Quadratic Approximation Approach*

K. He, S. Shi, T. van den Boom, and B. De Schutter

To cite this work, please refer to the published version:

K. He, S. Shi, T. van den Boom, and B. De Schutter, "Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach," *Automatica*, vol. 160, p. 111456, Feb. 2024. doi:[10.1016/j.automatica.2023.111456](https://doi.org/10.1016/j.automatica.2023.111456)

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: <https://www.dcsc.tudelft.nl>

* This report can also be downloaded via <https://dpub.eu/24-002>

Approximate Dynamic Programming for Constrained Linear Systems: A Piecewise Quadratic Approximation Approach [★]

Kanghui He, Shengling Shi, Ton van den Boom, Bart De Schutter

Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands

Abstract

Approximate dynamic programming (ADP) faces challenges in dealing with constraints in control problems. Model predictive control (MPC) is, in comparison, well-known for its accommodation of constraints and stability guarantees, although its computation is sometimes prohibitive. This paper introduces an approach combining the two methodologies to overcome their individual limitations. The predictive control law for constrained linear quadratic regulation (CLQR) problems has been proven to be piecewise affine (PWA) while the value function is piecewise quadratic. We exploit these formal results from MPC to design an ADP method for CLQR problems with a known model. A novel convex and piecewise quadratic neural network with a local-global architecture is proposed to provide an accurate approximation of the value function, which is used as the cost-to-go function in the online dynamic programming problem. An efficient decomposition algorithm is developed to generate the control policy and speed up the online computation. Rigorous stability analysis of the closed-loop system is conducted for the proposed control scheme under the condition that a good approximation of the value function is achieved. Comparative simulations are carried out to demonstrate the potential of the proposed method in terms of online computation and optimality.

Key words: Approximate dynamic programming; Reinforcement learning; Model predictive control; Value function approximation; Neural networks; Constrained linear quadratic regulation.

1 Introduction

Model-based reinforcement learning RL, also known as (approximate) dynamic programming ((A)DP) [1], has received much attention for the synthesis of controllers. Compared to its diverse industrial applications, the theoretical analysis on the stability and safety for RL faces great challenges. Thanks to Lyapunov stability theory, the stability issue of ADP has been comprehensively addressed, both for linear [2] and nonlinear systems [3, 4]. Although ADP approaches consider stability, safety is another important issue that needs further study. Safety means that the states and inputs of closed-loop systems satisfy some constraints. Techniques employed by RL or ADP approaches for dealing with constraints

can be grouped into two categories: policy-projection-based methods and policy-optimization-based methods. Policy-projection-based methods consider the RL formulation in the unconstrained case and involves a regulator to check and modify the policies that may violate the constraints [5, 6]. These indirect methods, however, sometimes fail to capture the optimal solution of the constrained problem and lack stability guarantees. In comparison, optimization-based methods intend to directly get the optimal value function for the constrained problems by solving the constrained Bellman equation. With the optimal value function available, the optimal control policy can thereby be produced by solving a constrained policy optimization problem. [7] was the first investigation of this kind of method, but it is limited to searching for the best linear feedback control law and needs an initial stabilizing policy. Following this direction, [8] explores an estimation approach to find an initial stabilizing policy even when there are uncertain nonlinear dynamics.

In constrained cases, neither the optimal policy nor the optimal value function is readily available, even for the most basic infinite-horizon linear quadratic regulation (LQR) problems. This to some extent restricts the de-

[★] This paper was not presented at any IFAC meeting. This paper is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101018826 - CLariNet). Corresponding author Kanghui He.

Email addresses: k.he@tudelft.nl (Kanghui He),
s.shi-3@tudelft.nl (Shengling Shi),
a.j.j.vandenBoom@tudelft.nl (Ton van den Boom),
b.deschutter@tudelft.nl (Bart De Schutter).

velopment of the policy-optimization-based RL methods for constrained control problems. In comparison, model predictive control (MPC) [9], an optimization-based control scheme widely adopted in the control community, has a mature stability and robustness theory as well as an inherent constraint handling. For infinite-horizon LQR problems, MPC can render the exact optimal control law due to the equivalence of finite and infinite optimal control as long as the horizon is sufficiently long [10]. With this fundamental property, infinite-horizon LQR problems can be solved by either implementing MPC online or explicit MPC [11], and the solution is proven to be piecewise affine (PWA) in the state space [11]. However, the computational complexity of both online MPC and explicit MPC grow dramatically with the increase of the problem size. This is one of the main drawbacks of MPC compared with RL or ADP.

Dedicated to overcoming this drawback, approximation methods of predictive control laws have received much attention in recent years. An emerging methodology is using specific function regression techniques such as PWA neural networks (NNs) [6, 12, 13]. Nevertheless, no guarantees of closed-loop stability are conveniently available, even through the final approximation error is small enough. Using NN-based controllers to warm start the solver in online MPC [14] can inherently guarantee stability, and learning Lyapunov functions to verify the stability of NN-based controllers [15] is also an alternative way. However, additional computation is required in the optimization or learning procedure.

Observing that MPC has computational limitations and approximation in the policy space lacks performance guarantees, we aim to attain a computationally inexpensive control scheme for linear systems with stability and feasibility guarantees. To this end, we approximate the value function via ADP and shorten the prediction horizon to one. We focus on the infinite-horizon LQR problem with state and input constraints. Different from the research on approximating the MPC policy [6, 12, 13], we propose a value function approximation scheme. The optimal value function that is characterized by explicit MPC, is approximated by a piecewise quadratic (PWQ) NN. The synthesis of the control law is conducted in a DP problem, where stability, feasibility, and sub-optimality can be guaranteed if a good approximation is obtained. Meanwhile, note that one disadvantage of approximating the value function is that it needs online policy optimization. To address this issue, we develop algorithms so that online optimization can be done efficiently.

The contributions of the paper are highlighted as follows:

(1) We propose a novel NN structure to approximate the solution of the constrained LQR problem based on ADP. The proposed NN structure can capture the PWQ and convex properties of the value function. Different from policy-based approximation approaches [6, 12, 16], our

approach has the important advantage that the resulting controller has safety and stability guarantees.

(2) We propose an efficient algorithm to solve the policy optimization problem, which is a convex piecewise quadratic program. In particular, this program is simplified to a collection of quadratic programs (QPs). Note that the main difficulty that prevents ones from considering more complex approximation structures is the increase in online computational time. We solve this problem in Algorithm 1. Complexity analysis and simulation results show that the proposed method requires much less online computation time than implicit MPC.

(3) Compared to [7], the first exploration of ADP in a constrained LQR setting, the proposed approach eliminates the restriction of searching for a linear feedback law and does not require an initially stabilizing policy nor an initial state belonging to an invariant set.

(4) We do a rigorous stability analysis, give stability conditions, and provide a tractable way to verify them.

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \dots\}$ and let $\lambda_{\max}(P)$ and $\lambda_{\min}(P)$ represent the maximum and minimum eigenvalues of a symmetric positive definite matrix P . The boundary of the set \mathcal{P} is $\partial\mathcal{P}$, and $\text{int}(\mathcal{P})$ stands for the interior of \mathcal{P} . We use $A_{i\cdot}$ to represent the i th row of the matrix A .

2.1 Infinite-horizon optimal control and MPC

We study the infinite-horizon constrained linear quadratic regulation (CLQR) problem

$$\begin{aligned} J_{\infty}^*(x) = \min_{U^{\infty}} \left\{ J_{\infty}(x_0, U^{\infty}) \triangleq \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \right\} \\ \text{s.t. } x_{k+1} = A x_k + B u_k, k = 0, 1, \dots, x_0 = x \\ x_k \in \mathcal{X}, u_k \in \mathcal{U}, k = 0, 1, \dots \end{aligned} \quad (1)$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, \mathcal{X} and \mathcal{U} are polyhedra that contain the origin in their interior, and $U^{\infty} = [u_0, u_1, \dots, u_{\infty}]$ is the infinite-dimensional decision variable. Matrices A and B are known. Like [6, 17], it is also assumed that

Assumption A1: (A, B) is stabilizable, $Q > 0$, and $R > 0$. Moreover, there exists an initial state, such that there exists a sequence of admissible input vectors u_0, u_1, \dots that can steer the state to the origin, i.e., $\bar{\mathcal{X}} \triangleq \{x \in \mathbb{R}^n | \exists U^{\infty} \text{ s.t. } x_k \in \mathcal{X}, u_k \in \mathcal{U}, \text{ and } J_{\infty}^*(x) < \infty, \forall k \in \mathbb{N}\}$ is not empty.

With Assumption A1, let $K \in \mathbb{R}^{m \times n}$ be a stabilizing gain matrix for the unconstrained plant $x_{k+1} =$

$Ax_k + Bu_k$. As in [10], we consider the admissible set of states under a given stabilizing control gain K as $\mathcal{X}_K = \{x \in \mathbb{R}^n \mid x \in \mathcal{X}, -Kx \in \mathcal{U}\}$.

If the constraints in (1) are removed, the problem admits a unique linear solution $u_k^{\text{LQR}} = -K^*x_k$, $k = 0, 1, \dots$ where $K^* = (R + B^T P^* B)^{-1} B^T P^* A$ and $P^* = P^{*T}$ is the unique positive-definite solution of the algebraic Riccati equation [2]. Then, $J_\infty^*(x) = x^T P^* x$ in the absence of constraints.

In the constrained case, if the state is close to the origin, the unconstrained LQR solution $u_k^{\text{LQR}} = -K^*x_k$, $k = 0, 1, \dots$ may not violate the constraints so that the solution to (1) is identical to the unconstrained LQR solution $-K^*x_k$ as if there is no constraint at all. This motivates the consideration of the maximal LQR invariant set $\mathcal{O}_\infty^{\text{LQR}} = \{x \in \mathbb{R}^n \mid (A - BK^*)^k x \in \mathcal{X}_{K^*}, \forall k \in \mathbb{N} \subseteq \mathbb{R}^n\}$ for the autonomous constrained linear system $x_{k+1} = (A - BK^*)x_k$, $x_k \in \mathcal{X}, \forall k \in \mathbb{N}$ [9, 10]. With this definition, the existing literature [10] considers using a finite-horizon problem:

$$J_N^*(x) = \min_{\{u_k\}_{k=0}^{N-1}} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + x_N^T P^* x_N$$

s.t. $x_{k+1} = Ax_k + Bu_k$, $k = 0, 1, \dots, N-1$, $x_0 = x$
 $x_k \in \mathcal{X}$, $u_k \in \mathcal{U}$, $k = 0, 1, \dots, N-1$ (2)

to approximate the infinite-horizon LQR problem (1). In (2), $U = [u_0, u_1, \dots, u_{N-1}]$ is the decision variable and $U^*(x)$ is the solution. The rationale behind this design is that after a sufficient long horizon, the resulting x_N will fall into $\mathcal{O}_\infty^{\text{LQR}}$ where the unconstrained LQR solution $u_N = -K^*x_N$ will not violate the constraints. As the result, the terminal cost $x_N^T P^* x_N$ is an exact representation for $J_\infty^*(x_N)$. The following theorem, proposed in [10], formally demonstrates the validity of this design.

Theorem 1 ([10]) *Let \bar{J} be an upper bound on $J_\infty^*(x)$. Suppose that Assumption A1 holds. For any $x \in \bar{X}$, if $N > (\bar{J} - p)/q$ where $0 < q \leq q_m \triangleq \inf_{x \notin \mathcal{O}_\infty^{\text{LQR}}} \{x^T Q x\}$ and $0 < p \leq p_m \triangleq \inf_{x \notin \mathcal{O}_\infty^{\text{LQR}}} \{x^T P x\}$, then $x_N \in \mathcal{O}_\infty^{\text{LQR}}$ and problem (2) is equivalent to problem (1) in the sense of $J_\infty^*(x) = J_N^*(x)$, $\forall x \in \bar{X}$.*

Throughout the paper, we assume that N in (2) is chosen such that the equivalence in Theorem 1 is satisfied.

By substituting the state update equation into $J_N^*(x)$ and the constraints, (2) can be reformulated as a multi-parametric quadratic program (mpQP). See [9, 18] for details. Explicit MPC aims to directly solve the mpQP for all possible x in a feasible set, and find the relationship between the optimizer $U^*(x)$ and x . The results from explicit MPC show that the optimizer $U^*(\cdot)$

is PWA. So, the first element $u_0^*(\cdot)$ of $U^*(\cdot)$ satisfies

$$u_0^*(x) = F_j x + g_j \quad \text{if } x \in \mathcal{R}_j, \quad j = 1, \dots, N_r \quad (3)$$

where the polyhedral sets $\mathcal{R}_j = \{x \in \mathbb{R}^n : H_j x \leq h_j\}$, $j = 1, \dots, N_r$ constitute a finite partition of a compact set of initial conditions $X_0 \subseteq \bar{X}$.

2.2 Problem formulation

The explicit controller is easy to compute offline in the case of a short horizon and a low-dimensional input vector. However, with the increase of the horizon and the system's dimension, the number of regions in (3) grows exponentially [11] and the representations of these regions become more complex, which may make the offline computation and online implementation intractable.

Artificial NNs with at least one hidden layer have the capability of universal approximation for any continuous function, provided that the hidden layer has enough units. Herein, we can use NNs to represent the explicit MPC law, without any need to identify the regions in (3). Related work is reported in [6, 12, 16], combined with supervised learning or policy gradient methods. The NN-based controllers proposed in [6, 12, 16] require an online feasibility certificate to determine whether the outputs of the NNs are safe. These NN-based policy approximators, however, inherently lack stability guarantees.

To address the computational burden of MPC and the lack of guarantees of policy-based approximation, we adopt value function approximation and shorten the MPC horizon to one. The main challenges are thereby (i) the design of the value function approximator, which is expected to output a function akin to the optimal value function, (ii) the relation between those guarantees and the quality of the approximation, and (iii) further reduction of online computation time concerning that the one-step problem contains a function approximator.

Before establishing our approximation structure, we concentrate on the properties of $J_\infty^*(\cdot)$.

Theorem 2 ([9]) *With Assumption A1 satisfied, in a compact polyhedral set of the initial conditions $X_0 \subseteq \bar{X}$, $J_\infty^*(\cdot)$ is continuous, convex, and PWQ over polyhedra:*

$$J_\infty^*(x) = J_i(x) = x^T P_i x + q_i^T x + v_i, \quad \text{if } x \in \mathcal{R}_i, i = 1, \dots, N_r \quad (4)$$

Moreover, if the mpQP problem (2) is not degenerate, then $J_\infty^*(\cdot)$ is continuously differentiable.

Theorem 3 ([19]) *Assume that (1) results in a non-degenerate mpQP. Let $\mathcal{R}_i, \mathcal{R}_j$ be two neighboring polyhedra and $\mathcal{A}_i, \mathcal{A}_j$ be the corresponding sets of active constraints at the optimum of (2). Then, (i) $P_i - P_j \leq 0$ if $\mathcal{A}_i \subset \mathcal{A}_j$, and (ii) $P^* - P_j \leq 0$, $\forall j \in \{1, 2, \dots, N_r\}$.*

For the detailed description of degeneracy, see [18, 20].

3 ADP design for CLQR problem

3.1 NN design for approximation in value space

According to Theorems 2 and 3, the value function approximator, denoted by $\hat{J}(\cdot, \theta)$ where θ refers to some parameters, is expected to have the following features: (F1) It can partition its input space into polyhedral regions. (F2) It can produce a convex and PWQ function partitioned by polyhedra. (F3) For x in a small region containing the origin, the approximator can provide the exact representation for the value function, i.e., $\hat{J}(x, \theta) = x^T P^* x$.

We will use a feed-forward NN to capture the relationship between the state x and $J_\infty^*(x)$. A feed-forward NN is composed of one or more hidden layers and an output layer, where each hidden layer contains an affine map $f_l(\kappa_{l-1}) = W_l \kappa_{l-1} + b_l$, followed by a nonlinear map $\kappa_l = g_l(f_l)$. Here, $W_l \in \mathbb{R}^{M_l \times M_{l-1}}$ and $b_l \in \mathbb{R}^{M_l}$ are the weights and biases, $g_l(\cdot) : \mathbb{R}^{M_l} \rightarrow \mathbb{R}^{M_l}$ is a nonlinear activation function, M_l is the width of the l th layer, referring to the number of units in the layer, and $M_0 = n$. Based on these definitions, an NN with L hidden layers and M_l units in the l th layer can be represented by $f_{\text{NN}}(x, \theta) = [f_{L+1} \circ g_L \circ f_L \circ \dots \circ g_1 \circ f_1](x)$, where θ contains all the weights and biases of the affine functions in all the layers, and the symbol \circ means the layers are connected in series.

The activation function plays an important role in the approximation of NNs. In this paper, we consider the rectifier linear unit (ReLU), which is defined as $g_{\text{ReLU}}(x) = \max\{0, x\}$. An important property of the ReLU is that it can produce a series of PWA functions with polyhedral partitions, combined with an affine transformation [21]. Actually, the output of an NN with ReLUs as activation functions is PWA. However, as the optimal value function is PWQ, we are interested in producing a class of PWQ basis functions that can efficiently represent the value function.

Let us focus on the last hidden layer. All activation units in this layer can still be written as a continuous PWA function over the input space of the network [22]. The element-wise product of any two vector-valued continuous PWA functions with the same number of components is a continuous PWQ function. We therefore calculate the element-wise product of all the units κ_L in the last hidden layer

$$\varphi = p(\kappa_L) \triangleq \text{diag}(\kappa_L) \kappa_L \in \mathbb{R}^{M_L}$$

to generate a series of PWQ functions $\varphi(x) = [\varphi_1(x) \ \varphi_2(x) \ \dots \ \varphi_{M_L}(x)]^T$. This can be viewed as a layer in the NN, denoted by the product layer $p(\kappa_L)$.

The output layer is a weighted sum of the outputs of the previous layer $f_{L+1}(\varphi) = r^T \varphi = \sum_{i=1}^{M_L} r_i \varphi_i$.

To make the proposed approximator satisfy F3, we develop a ‘‘local-global’’ architecture, which decomposes the outputs of the approximator into two parts

$$\hat{J}(x, W, b, r) = x^T P^* x + [f_2 \circ p \circ g_1 \circ f_1](x) \quad (5)$$

with P^* the solution to the algebraic Riccati equation. The term $x^T P^* x$ is included to capture ‘‘global’’ aspects of $J_\infty^*(\cdot)$, while the neural-network-based term is exploited to identify the polyhedral partition in (4) and capture the local residuals $J_\infty^*(x) - x^T P^* x$. Since the known term $x^T P^* x$ that dominates the value function is extracted and fixed, using such a ‘‘local-global’’ architecture is possible to enhance the quality of approximation.

We hereafter denote M_1 by M , and $\hat{J}(\cdot, W, b, r)$ by $\hat{J}(\cdot, \theta)$, with all the parameters condensed in θ .

3.2 NN training and convexity analysis

Training data needs to be generated offline by solving (2) for N_x different initial states $\{x^{(i)}\}_{i=1}^{N_x}$, $x^{(i)} \in X_0$, and getting the state-value pairs $\{(x^{(i)}, J_\infty^*(x^{(i)}))\}_{i=1}^{N_x}$. Let $\{u_k^{(i)*}\}_{k=0}^{N-1}$ and $\{x_k^{(i)*}\}_{k=0}^N$ denote the solution to the MPC problem for the initial state $x^{(i)}$ and the corresponding trajectories of the closed-loop controlled system. We present an efficient sampling strategy by leveraging the equivalence in Theorem 1. Suppose that we have obtained the optimal control sequence $\{u_k^{(i)*}\}_{k=0}^{N-1}$ and the corresponding value $J_N^*(x^{(i)})$ for different $x^{(i)}$, consider the sub-problems whereby we start at $x_k^{(i)*}$, for $k = 1, \dots, N-1$ and wish to minimize $J_\infty(x_k^{(i)*}, U)$ in (1). According to the principle of optimality [23], the truncated optimal control sequence $\{u_j^{(i)*}\}_{j=k}^{N-1}$ is also optimal for these sub-problems. Hence, the optimal value functions for these subsequent trajectories $x_k^{(i)*}$, $k = 1, \dots, N-1$ can directly be computed as $J_\infty^*(x_k^{(i)*}) = J_N^*(x_k^{(i)*}) = J_N^*(x^{(i)}) - \sum_{j=0}^{k-1} x_j^{(i)*T} Q x_j^{(i)*} + u_j^{(i)*T} R u_j^{(i)*}$, with no need to solve (2) repeatedly.

With this design, we can offline generate $N_x N$ state-value pairs by only solving (2) N_x times. With the $N_x N$ state-value pairs available, the NN is trained so that its parameters approximate the solution to

$$\min_{b < 0, W, r \geq 0} \frac{1}{N N_x} \sum_{i=1}^{N_x} \sum_{k=0}^{N-1} e(x_k^{(i)*}, \theta) \quad (6)$$

where $e(x_k^{(i)*}, \theta) = (\hat{J}(x_k^{(i)*}, \theta) - J_\infty^*(x_k^{(i)*}))^2$ is the square of the approximation error for each training pair, and $x_0^{(i)*} = x^{(i)}$, $\forall i \in \{1, \dots, N_x\}$. The constraint $b < 0$ is introduced to guarantee that no units in the hidden layer are activated when x is near the origin, i.e., to fulfill F3, while the constraint $r \geq 0$ is responsible for maintaining convexity of $\hat{J}(\cdot, \theta)$.

Problem (6) is a nonlinear least-squares problem, which can be successfully solved by the gradient descent method [24]. The constraints on the NN parameters can be handled by constraint elimination, i.e., by letting $r = (\bar{r}_1^2, \bar{r}_2^2, \dots, \bar{r}_M^2)^T$, which can always guarantee $r \geq 0$, penalizing the constraint violation in the loss function, or reducing the number of hidden units if constraint violation is detected.

After the NN is trained, the system can be run and the control signals are computed by solving a DP problem. With a specific structure, our proposed NN allows to produce a convex function $\hat{J}(\cdot, \theta)$ so that any locally optimal point is also globally optimal. Suppose that the output of the proposed approximator has the following PWQ form:

$$\hat{J}(x, \theta) = \hat{J}^j(x) = x^T \hat{P}_j x + \hat{q}_j^T x + \hat{v}_j, \text{ if } x \in \hat{\mathcal{R}}_j, j=1, \dots, \hat{N}_t \quad (7)$$

where $\hat{\mathcal{R}}_j$ are polyhedra defined by the hyperplanes $\{W_{i,j} x + b_i = 0\}_{i=1}^M$. Define $\bar{R} = \text{diag}(r)$ and we can rewrite (5) as $\hat{J}(x, \theta) = x^T P^* x + \kappa^T \bar{R} \kappa$, where κ , the output of the hidden layer, is PWA w.r.t. x . In the following, we will show that any feasible solutions to (6) can ensure the positive semi-definiteness of \bar{R} .

Proposition 1 *Consider the PWQ NN (5). With a non-negative r and a negative b , the function $\hat{J}(\cdot, \theta)$ that the NN produces is continuously differentiable and convex w.r.t. its input.*

The proof of Proposition 1 and of subsequent theorems and lemmas can be found in the appendices of the paper.

3.3 Suboptimal control law based on DP

With a well-fitted $\hat{J}(\cdot, \theta)$ available, at each time step $t \in \mathbb{N}$, we can obtain a suboptimal control policy by solving the one-step DP problem

$$\begin{aligned} \min_{u_t} \hat{Q}(x_t, u_t) &\triangleq x_t^T Q x_t + u_t^T R u_t + \hat{J}(Ax_t + Bu_t, \theta) \\ \text{s.t. } u_t &\in \mathcal{U}, Ax_t + Bu_t \in \mathcal{C} \end{aligned} \quad (8)$$

where $\hat{Q}(x_t, u_t)$ can be viewed as the approximated optimal Q-function [23] for (1). Denote the solution to (8) by \hat{u}_t^* . Here, we use another subscript $(\cdot)_t$ to indicate that (8) is solved online at each time step t .

In (8), \mathcal{C} is chosen as \bar{X} or \mathbb{R}^n , depending on whether \bar{X} is computable. In particular, if the iterative algorithm for computing \bar{X} , e.g., Algorithm 10.3 in [9] does not terminate in finite time, we drop the constraint on $Ax_t + Bu_t$. This could happen, e.g., when there is no state constraint in (1). In both cases problem (8) is recursively feasible since \bar{X} is a control invariant set (CIS) [9].

The objective function in (8) is nonlinear and contains an NN. Standard solvers such as the ellipsoid algorithm or the interior-point algorithm require the computation of the Hessian $\nabla_u^2 \hat{Q}(x, u)$ or the gradient $\nabla_u \hat{Q}(x, u)$ in each iteration. Such computation can only be carried out by visiting all units in the hidden layers and extracting the activated ones. The advantage of low computational complexity brought by DP will then inevitably diminish.

In view of this, we intend to avoid frequently calculating $\nabla_u^2 \hat{Q}(x, u)$ and $\nabla_u \hat{Q}(x, u)$ by decomposing the Q-function $\hat{Q}(x_t, u_t)$ into some quadratic functions. In particular, we develop an optimization algorithm in which problem (8) is reduced to a QP problem in each iteration. For a given x_t , $t \in \mathbb{N}$, consider the set of activated ReLU units in $\hat{J}(Ax + Bu, \theta)$:

$$\bar{\mathcal{A}}(u) = \{i \in \{1, \dots, M\} \mid W_{i,\cdot}(Ax_t + Bu) + b_i > 0\} \quad (9)$$

With (9), $\hat{Q}(x_t, u)$ can thus be computed as

$$\hat{Q}(x_t, u) = u^T \bar{P}(\bar{\mathcal{A}}(u))u + \bar{q}^T(\bar{\mathcal{A}}(u))u + \bar{v}(\bar{\mathcal{A}}(u)) \quad (10)$$

where

$$\begin{aligned} \bar{P}(\bar{\mathcal{A}}(u)) &= R + B^T (P^* + \sum_{i \in \bar{\mathcal{A}}(u)} r_i W_{i,\cdot}^T W_{i,\cdot}) B \\ \bar{q}^T(\bar{\mathcal{A}}(u)) &= 2x_t^T A^T P^* B + 2(\sum_{i \in \bar{\mathcal{A}}(u)} r_i (W_{i,\cdot} Ax_t + b_i) W_{i,\cdot}) B \\ \bar{v}(\bar{\mathcal{A}}(u)) &= x_t^T (Q + A^T P^* A) x_t + \sum_{i \in \bar{\mathcal{A}}(u)} r_i (W_{i,\cdot} Ax_t + b_i)^2 \end{aligned} \quad (11)$$

Since $\bar{P}(\bar{\mathcal{A}}(u)) > 0$, the right-hand side of (10) is a convex quadratic function if $\bar{\mathcal{A}}(u)$ is fixed. An algorithm that can effectively solve general piecewise convex programs (PCP) is proposed in [25], called the PCP Algorithm. We adapt it to solving problem (8). For extra details see [18]. On the other hand, the PCP Algorithm is not the ideal choice for solving our DP problem, because it needs to iteratively compute the intersection of some sets and one of the auxiliary QPs contains numerous constraints if the NN has a large number of hidden units. This motivates us to consider the following design. In each iteration s , $s \in \mathbb{N}^+$, let $u^{(s)}$ be an initial input (for $s = 1$) or the input calculated from the last iteration (for $s > 1$). We compute the set of activated units $\bar{\mathcal{A}}(u^{(s)})$ at $u^{(s)}$, and thereby get $\bar{P}(\bar{\mathcal{A}}(u^{(s)}))$ and

$\bar{q}^T(\bar{\mathcal{A}}(u^{(s)}))$ from (11). Then, we solve the QP:

$$\begin{aligned} u^{(s+1)} &= \arg \min_u u^T \bar{P}(\bar{\mathcal{A}}(u^{(s)}))u + \bar{q}^T(\bar{\mathcal{A}}(u^{(s)}))u \\ \text{s.t. } u &\in \mathcal{U}, Ax_t + Bu \in \mathcal{C} \end{aligned} \quad (12)$$

which returns $u^{(s+1)}$ for the next iteration. In the next iteration, after computing $\bar{\mathcal{A}}(u^{(s+1)})$, we can terminate and output $u^{(s+1)}$ if $\bar{\mathcal{A}}(u^{(s+1)}) = \bar{\mathcal{A}}(u^{(s)})$. If a cycle occurs, i.e., $\bar{\mathcal{A}}(u^{(s+1)}) = \bar{\mathcal{A}}(u^{(k)})$, $\exists k \in \{1, \dots, s-1\}$, we arbitrarily choose another $\bar{\mathcal{A}}(u^{(s+1)})$ that has not been involved in previous iterations. The proposed method is summarized in Algorithm 1.

Algorithm 1 Decomposition algorithm for solving (8)

Input: State x_t at time step t , input u_{t-1} at the last time step $t-1$ (if $t > 0$), the optimal Q-function $\hat{Q}(\cdot, \cdot), \mathcal{C}$

Output: Control input u_t
Initialize a starting point $u^{(1)} \leftarrow u_{t-1}$

for $s = 1, 2, \dots$ **do** .

 Update the set of activated units $\bar{\mathcal{A}}(u^{(s)})$ by (9).

if $\bar{\mathcal{A}}(u^{(s)}) = \bar{\mathcal{A}}(u^{(s-1)})$ and $s > 1$ **then** let $s_m \leftarrow s$,

return $u_t \leftarrow u^{(s_m)}$, and **break**.

else

if $\bar{\mathcal{A}}(u^{(s)}) = \bar{\mathcal{A}}(u^{(k)})$, $\exists k \in \{1, \dots, s-2\}$ and

$s > 2$ **then** reset $\bar{\mathcal{A}}(u^{(s)})$ to be a new set of activated units that never occurred previously.

end if

 Compute the coefficients $\bar{P}(\bar{\mathcal{A}}(u^{(s)}))$ and $\bar{q}^T(\bar{\mathcal{A}}(u^{(s)}))$ associated with $\bar{\mathcal{A}}(u^{(s)})$ through (11).

 Update the policy through (12) and get $u^{(s+1)}$.

end if

end for

In Algorithm 1, the starting point $u^{(1)} = u_{t-1}$ is initialized with the last control input, which can be viewed as a warm start for the algorithm.

Compared to the PCP Algorithm, Algorithm 1 only needs to solve one QP per iteration, in which the constraints are the same as those in (8). Additionally, Algorithm 1 circumvents the calculations of some sets that are necessary in the PCP Algorithm. Meanwhile, Algorithm 1 can achieve finite termination as well as the optimality for problem (8).

Theorem 4 Consider the DP problem (8). For any x_t that makes (8) feasible, the decomposition algorithm (Algorithm 1) terminates in a finite number of iterations, i.e., there exists a finite s_m such that $\bar{\mathcal{A}}(u^{(s_m)}) = \bar{\mathcal{A}}(u^{(s_m-1)})$. If $\bar{\mathcal{A}}(u^{(s_m)}) = \bar{\mathcal{A}}(u^{(s_m-1)})$, then $u^{(s_m)}$ is the solution to (8).

In general, the amount of data needed in our method is in general less than that in policy-based methods [12] because the value function is a scalar, but the policy may be multi-dimensional.

4 Analysis of the proposed method

4.1 Stability analysis

Recursive feasibility of (8) is inherently guaranteed. In this section, we investigate the stability of the proposed control law. Since both $J_\infty^*(\cdot)$ and $\hat{J}(\cdot, \theta)$ are PWQ on polyhedra, the intersection of any \mathcal{R}_i and $\hat{\mathcal{R}}_j$, $i = 1, \dots, N_r$, $j = 1, \dots, \hat{N}_r$ is still polyhedral. Let $\mathcal{R}_{i,j}$ denote this intersection if such intersection represents a full-dimensional region, i.e., $\mathcal{R}_{i,j} \triangleq \mathcal{R}_i \cap \hat{\mathcal{R}}_j$, $i \in \{1, \dots, N_r\}$, $j \in \{1, \dots, \hat{N}_r\}$ and $\dim(\mathcal{R}_i \cap \hat{\mathcal{R}}_j) = n$. It is clear that all $\mathcal{R}_{i,j}$ constitute a partition of X_0 .

To certify the stability, we need to know the upper bound of the approximation error for all x belonging to X_0 , i.e., we will compute a positive constant ζ such that

$$|e(x)| \triangleq \left| \frac{\hat{J}(x, \theta)}{J_\infty^*(x)} - 1 \right| \leq \zeta, \quad \forall x \in X_0. \quad (13)$$

We propose to leverage the Lipschitz continuity of $\nabla \hat{J}(\cdot, \theta)$ and $\nabla J_\infty^*(\cdot)$. Before doing this, we need the following assumption to ensure the density of samples.

Assumption A2: For the partition $\mathcal{R}_{i,j}$ of X_0 , there exists at least one training point in each $\text{int}(\mathcal{R}_{i,j})$.

Since we can compute the approximation error of the proposed NN at the training points, it is possible to obtain an upper bound of the approximation error for all x in X_0 . In the interior of each $\mathcal{R}_{i,j}$, both $J_\infty^*(\cdot)$ and $\hat{J}(\cdot, \theta)$ are quadratic and hence twice continuously differentiable. According to [9, Lemma 3.2], $\nabla_x J_\infty^*(\cdot)$ and $\nabla_x \hat{J}(\cdot, \theta)$ are locally Lipschitz on $\text{int}(\mathcal{R}_{i,j})$, i.e., there exist non-negative constants L_i and \hat{L}_j such that for all $(x, y) \in \text{int}(\mathcal{R}_{i,j}) \times \text{int}(\mathcal{R}_{i,j})$, we have

$$\begin{aligned} J_\infty^*(y) &\leq J_\infty^*(x) + \nabla^T J_\infty^*(x)(y-x) + \frac{L_i}{2} \|y-x\|_2^2 \\ \hat{J}(y, \theta) &\leq \hat{J}(x, \theta) + \nabla^T \hat{J}(x, \theta)(y-x) + \frac{\hat{L}_j}{2} \|y-x\|_2^2 \end{aligned} \quad (14)$$

where L_i and \hat{L}_j can be chosen as the largest eigenvalue of P_i and \hat{P}_j [9, Lemma 3.2], respectively.

We define a measure of the gradient error as $e_{\text{grad}}(x) \triangleq \frac{\|\nabla \hat{J}(x, \theta) - \nabla J_\infty^*(x)\|_2}{J_\infty^*(x)}$, and let \bar{e} and \bar{e}_{grad} stand for the maximum values of $|e(\cdot)|$ and $e_{\text{grad}}(\cdot)$ over all samples.

Let \mathcal{R}_1 refer to the polyhedron where no constraints in the mpQP derived from (2) are active, i.e., $\mathcal{R}_1 = \mathcal{O}_{\text{LQR}}^{\text{LQR}}$, and accordingly let $\hat{\mathcal{R}}_1$ represents the polyhedron where no ReLU units in $\hat{J}(\cdot, \theta)$ are activated. In the region

$\mathcal{R}_{1,1} = \mathcal{R}_1 \cap \hat{\mathcal{R}}_1$, we have $|e(x)| \equiv 0$ due to (5). For every $\mathcal{R}_{i,j}$ except $\mathcal{R}_{1,1}$, the following lemma presents an upper bound of $|e(x)|$

Lemma 1 *With Assumption A2 satisfied, for any $x \in \text{int}(\mathcal{R}_{i,j})$, $(i, j) \neq (1, 1)$, $|e(x)|$ is upper bounded by*

$$\zeta_{i,j} \triangleq \frac{1}{1 - \beta(y)d_{i,j}} \left(\bar{e} + \bar{e}_{\text{grad}}d_{i,j} + \frac{(\hat{L}_j + L_i)d_{i,j}^2}{2J_\infty^*(y)} \right) \quad (15)$$

where $y \in \text{int}(\mathcal{R}_{i,j})$ is the training or testing point closest to x , $\beta(y) = \|\nabla^T J_\infty^*(y)\|_2 / J_\infty^*(y)$, and $d_{i,j}$ denotes the maximum Euclidean distance between any nearest training or testing points in $\mathcal{R}_{i,j}$.

Finally, since $e(\cdot)$ is continuous on the closure of $\mathcal{R}_{i,j}$, the bound $\zeta_{i,j}$ applies to all x in $\mathcal{R}_{i,j}$. In addition, thanks to Assumption A2, the value of ζ in (13) can be determined by computing the right-hand side of (15) at all training/testing points and choosing the largest one.

With the property of boundedness for $e(\cdot)$ established, we can assess the stability for the closed-loop system with the approximate controller \hat{u}^* . Corresponding to the selection of \mathcal{C} in (8), we consider two cases: (1) $\mathcal{C} = \bar{X}$ and (2) $\mathcal{C} = \mathbb{R}^n$.

Theorem 5 *Let \hat{u}_t^* be the solution to problem (19) with $\mathcal{C} = \bar{X}$. Then, \hat{u}_t^* is recursively feasible for the initial condition $x_0 \in \bar{X}$. Furthermore, suppose that Assumptions A1-A2 hold with $X_0 = \bar{X}$. If ζ in (13) satisfies*

$$\frac{1 - \zeta^2}{\zeta} > 2 \sup_{x \in X_0 \setminus \{0\}} \frac{\hat{J}(x, \theta)}{x^T Q x}, \quad (16)$$

then the origin of the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \dots$ is asymptotically stable with domain of attraction \bar{X} .

In the case of $X_0 \subset \bar{X}$, where the set X_0 may not be invariant for the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \dots$, the corresponding stability conditions are described in the following corollary:

Corollary 1 *Let \hat{u}_t^* be the solution to problem (19) with $\mathcal{C} = \mathbb{R}^n$. For a given $X_0 \subset \bar{X}$, suppose that Assumptions A1-A2 hold. Define a compact set as $\Omega \triangleq \{x \in \mathbb{R}^n \mid \hat{J}(x, \theta) \leq \chi\}$ where $\chi \triangleq \inf_{x \in \partial X_0} \hat{J}(x, \theta)$. If ζ (13) satisfies (16), then \hat{u}_t^* is recursively feasible for the initial condition $x_0 \in \Omega$, and the origin of the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$, $t = 0, 1, \dots$ is asymptotically stable with domain of attraction Ω .*

According to Theorem 5 and Corollary 1, asymptotic stability is achieved if the condition in (16) holds. (16)

can be satisfied by making ζ small enough since the right-hand side of (16) is upper bounded. Besides, in view of (15), ζ is determined mainly by \bar{e} , \bar{e}_{grad} , and $d_{i,j}$. The condition (16) can thereby be guaranteed in two ways. One is to add more hidden units into the NN so that \bar{e} and \bar{e}_{grad} could be smaller according to the universal approximation theorem [26]. Another possibility is to involve more state-value/gradient data to test the upper bounds \bar{e} and \bar{e}_{grad} so that $d_{i,j}$ is reduced.

4.2 Complexity analysis

We analyze the offline storage requirement as well as the online computational complexity of the proposed control scheme, and compare them with other methods, such as implicit MPC, explicit MPC, and the policy approximation methods of MPC [12, 14]. Storage space is dominated by the number of regions and control laws (for explicit MPC), or the structure of the NN (for approximate MPC). The storage of some system's parameters, such as A , B , \mathcal{X} , \mathcal{U} , Q , and R , are neglected for consistency.

Explicit MPC requires the storage of N_r regions and affine feedback laws. Suppose that each region \mathcal{R}_j is defined by $n_c^{(j)}$ constraints. Then, explicit MPC needs to store $(n+1) \sum_{j=1}^{N_r} n_c^{(j)} + N_r(mn+m)$ real numbers. As for the proposed method, it needs to construct a PWQ NN before running the system. The NN contains 3 parameters: W , b , and r , so the storage of the proposed NN requires $nM + 2M$ real numbers in total. In addition, as for the policy approximation methods reported in [12, 14], the total storage demand of the NNs is $(n+n_0+1)M + (L-1)(M+1)M$, with $n_0 = m$ for [12] and $n_0 = N(m+2n+n_c+m_c)$ for [14], respectively. Here, L denotes the number of hidden layers, and n_c and m_c denote the number of constraints specified by \mathcal{X} and \mathcal{U} . In general, using an NN that requires much smaller storage space than explicit MPC can get an acceptable performance.

Online computation time will be evaluated in terms of floating point operations (flops) for the computations that should be performed online. Implicit MPC needs to solve the QP (2) at each time step. Solving (2) for a given x requires $f_{\text{QP}}(Nm, N(m_c+n_c))$ flops in the worst case ((2) has no redundant constraints). Here, $f_{\text{QP}}(n_D, n_I)$ represents the number of flops needed to solve a QP with n_D decision variables and n_I linear inequalities. So in an interior-point method, solving (2) requires $O(N^3m^3)$ flops per iteration. In comparison, the number of flops for explicit MPC is $2n \sum_{j=1}^{N_r} n_c^{(j)}$ [9].

In our proposed control scheme, solving the DP problem (8) causes computational complexity. In the PCP Algorithm and Algorithm 1, the number of flops to determine $\bar{\mathcal{A}}(u^{(s)})$ and to compute the coefficients $\bar{P}(\bar{\mathcal{A}}(u^{(s)}), \bar{q}^T(\bar{\mathcal{A}}(u^{(s)})))$ is bounded by $f_{\text{act}} =$

$M(2n^2 + n + m^2 + 5m + 2mn + 2) + 2mn + m(m + 3)/2$ in total. Then, in each iteration the PCP Algorithm has to solve 2 different QPs, which need $f_{QP}(m, n_s)$ and $f_{QP}(m, M + n_c + l_c)$ flops. Here, n_s stands for the number of inequalities that define \mathcal{U}_s . Algorithm 1 needs only $f_{QP}(m, n_c + l_c)$ flops to find $u^{(s)}$ in each iteration. Besides, some other calculation includes the update of \mathcal{U}_s ($2m^2 + 2m - 1$ flops per iteration) and the comparison of $\bar{A}(u^{(s)})$ and $\bar{A}(u^{(s-1)})$ (M flops per iteration). Therefore, the total number of flops to implement the PCP Algorithm is $f_{\text{algo1}} = s_m(f_{\text{act}} + f_{QP}(m, M + n_c + l_c) + 2m^2 + 2m - 1) + \sum_{s=1}^{s_m} f_{QP}(m, n_s)$ and the number of flops to calculate the control input by using Algorithm 1 is $f_{\text{algo2}} = s_m(f_{\text{act}} + f_{QP}(m, n_c + l_c) + M)$, which can be much less than those in the PCP Algorithm. Theoretically, the maximum value of s_m can be the number of polyhedral regions in the partition for the NN, and thus can grow exponentially with the system's dimension. However, the above analysis largely overestimates the complexity, as it does not take into account that most regions are never visited, and that we use a warm start for the proposed algorithm.

5 Numerical example

A case study is presented to assess the feasibility and effectiveness of the proposed control scheme. The stability result in Corollary 1 is verified. In addition, some other methods including implicit MPC and the policy approximation method of MPC [12], are also compared with the proposed method. All the results have been obtained in MATLAB R2021a on an AMD Core R7-5800H CPU @3.20GHz machine.

Consider a 2-dimensional linear system with $A = \begin{bmatrix} 1 & 0.1 \\ -0.1 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 0.05 \\ 0.5 & 1 \end{bmatrix}$ and input constraints $\mathcal{U} = \{u \in \mathbb{R}^2 \mid \|u\|_\infty \leq 0.5\}$. We are interested in stabilizing the system at the origin and meanwhile minimizing the cost $\sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k$ with $Q = I_2$ and $R = 0.1I_2$. We choose the region of interest $\mathcal{X}_0 = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 3\}$. By applying the algorithm in [10], it can be verified that the vertices of \mathcal{X}_0 can be steered to an ellipsoidal subset of $\mathcal{O}_\infty^{\text{LQR}}$ with the horizon $N = 10$. By solving the explicit MPC problem with $N = 10$, the partition of the state space for the optimal control law on the region \mathcal{X}_0 can be obtained, and is depicted in Fig. 1(a).

To illustrate that the proposed PWQ NN has a good approximation performance, different types of NNs are compared in Fig. 1(c). Besides, a strictly global architecture without the quadratic term $x^T P^* x$ is also compared. 4410 state samples are selected to train the NNs. We use multi-start local optimization [27] with 20 starting points to reduce sub-optimality. Choosing the learning rate $\alpha = 0.1$, we compare the absolute mean square

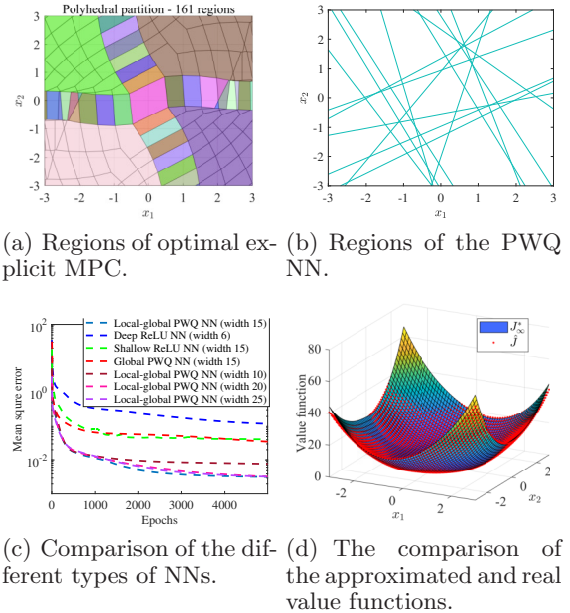


Fig. 1. Performance of the proposed PWQ NN.

errors in Fig. 1(c). From Fig. 1(c), it is observed that the training objectives for the NNs decrease consistently over epochs, while the proposed approach has the smallest mean square error during the training process. Additionally, increasing the width does not necessarily improve the approximation ability of the proposed NN.

Fig. 1(b) depicts the polyhedral partition for the proposed NN with width 15. It is noticed that the partition of the NN concentrates in the areas where the explicit MPC law changes rapidly. We also compare the output of the PWQ NN with the real optimal value function in Fig. 1(d), from which one can observe that the proposed method is able to closely approximate the optimal value function with a very simple network architecture.

To verify the stability of the closed-loop system, we note that the maximal stabilizable set $\bar{\mathcal{X}}$ is open and thereby not computable. So, we concentrate on checking (16) with X_0 replaced by Ω . The approximation error bounds at the samples are $\bar{e} = 0.122$ and $\bar{e}_{\text{grad}} = 0.395$. Based on (15), $\zeta = 0.188$. Next, the value of the right-hand side of (16) is 4.904. As a result, it can be readily verified that (16) holds.

In the closed-loop simulation, the proposed method, implicit MPC, and the policy-approximation method of [12] are compared. The implementation details are given in [18]. From Fig. 2(a), it can be seen that the proposed method can properly approximate the MPC controller, and that the corresponding trajectory is stabilized at the origin. In comparison, the policy-approximation method experiences some fluctuations when the state is close to the origin. This is probably because the value of the optimal control input is small if x is around the origin,

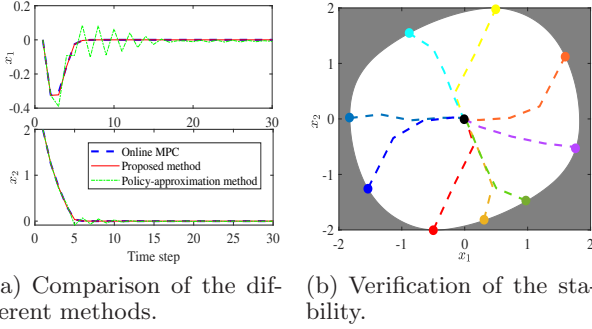


Fig. 2. Closed-loop simulation. In (b), the white region represents Ω , the colored lines refer to the trajectories, and the colored points are the starting state points.

so a slight approximation error of the policy may lead to drastic changes in the dynamic response. Besides, we can verify the stability by illustrating the invariance of the sub-level set $\Omega \triangleq \{x \in \mathbb{R}^n \mid \hat{J}(x, \theta) \leq \chi\}$. We select some initial states that are at the boundary of Ω , and plot the behavior of the closed-loop system with the proposed controller in Fig. 2(b). It is seen that the trajectories starting from the boundary of Ω will always stay in Ω , which is computed by Corollary 1.

To illustrate the computational performance of the proposed method on larger-size examples we consider the problem of regulating a system of oscillating masses [28], which is an 8-D system. For the detailed setup, see [18]. Table 1 shows the average CPU time of the different methods, as well as their total cost after running the system for 100 time steps. The policy-approximation method needs the least simulation time since it just needs to compute a single projection (solve a QP) at each time step. The proposed method with Algorithm 1 requires shorter computation time than online MPC. Besides, compared with the nonlinear solver and the PCP Algorithm, Algorithm 1 significantly reduces the computational complexity. In other aspects, the control law generated by the proposed method is nearly optimal since its total cost is very close to that of MPC, while the total cost of the policy-approximation method is the largest (about 140% of that of MPC), due to the fluctuations near the origin. Besides, one can refer to [18] for the comparison of the storage demand in both examples.

Moreover, in [18], the stability conditions in Theorem 5 and the comparison between the proposed method and another ADP method in [7] are illustrated through another problem with both state and input constraints. The results in [18] can clearly show that the proposed controller is stabilizing and feasible even for those initial states on the boundary of \bar{X} . Combined with Algorithm 1, the proposed method takes less total computation time (0.142 s) than the ADP method (0.270 s) in [7] and implicit MPC (0.161 s).

Table 1
Comparison of different methods regarding the average CPU time and the total cost in the 8-D system.

Methods	CPU time	Total cost
Proposed + PCP Algorithm	1.2300	2004.2
Proposed + Algorithm 1	0.2225	2004.2
Proposed + nonlinear solver	0.7309	2004.2
Online MPC	1.7279	1857.3
Policy-approximation method	0.1845	2607.5

6 Conclusions

We have developed an ADP control framework for infinite-horizon optimal control of linear systems subject to state and input constraints. Compared to some common NNs such as ReLU NNs, the proposed NN maintains the PWQ property and convexity of the real value function and has a much better approximation performance. These properties and superiority contribute to the reduction of the online computation as well as the construction of explicit stability criteria. Therefore, advantages of our method include low computational requirements, stability assurance, and excellent approximation of the optimal control law.

A Proof of Proposition 1

In the interior of any $\hat{\mathcal{R}}_j, j = 1, \dots, \hat{N}_r$, continuous differentiability of $\hat{J}(\cdot, \theta)$ is clear since $\hat{J}(\cdot, \theta)$ has a quadratic form, and convexity of $\hat{J}(\cdot, \theta)$ follows from the positive semi-definiteness of R . Then, we have $\hat{P}_j - P^* \geq 0$ for all $j = 1, \dots, \hat{N}_r$. At the boundary of any neighboring $\hat{\mathcal{R}}_i$ and $\hat{\mathcal{R}}_j$, without loss of generality, suppose that $\hat{\mathcal{R}}_i, \hat{\mathcal{R}}_j$ are partitioned by the hyperplane $W_{1,\cdot}x + b_1 = 0$. It follows from (7) that $\hat{J}^j(x) = \hat{J}^i(x) + r_1(W_{1,\cdot}x + b_1)^2$. Differentiating both sides yields $\nabla_x \hat{J}_j(x, \theta) = \nabla_x \hat{J}_i(x, \theta), \forall x \in \{x \in \mathbb{R}^n \mid W_{1,\cdot}x + b_1 = 0\}$ which proves continuous differentiability of $\hat{J}(x, \theta)$ at the boundary.

Furthermore, since $\hat{J}(x, \theta)$ is differentiable, convexity of $\hat{J}(x, \theta)$ at the boundary can be checked through the first-order condition [29]. Without loss of generality, let $x_1 \in \hat{\mathcal{R}}_i$ and $x_2 \in \hat{\mathcal{R}}_j$, then we have

$$\begin{aligned} & \hat{J}(x_2, \theta) - \hat{J}(x_1, \theta) - \nabla_x \hat{J}(x_1, \theta)^T (x_2 - x_1) \\ &= x_2^T P_i x_2 + q_i^T x_2 + r_1 (W_{1,\cdot}x_2 + b_1)^2 \\ & \quad - x_1^T P_i x_1 + q_i^T x_1 - (x_1^T P_i + q_i^T) (x_2 - x_1) \\ &= (x_2 - x_1)^T P_i (x_2 - x_1) + r_1 (W_{1,\cdot}x_2 + b_1)^2 \geq 0 \end{aligned}$$

which demonstrates that $\hat{J}(\cdot, \theta)$ satisfies the first-order condition at the boundary. \square

B Proof of Theorem 4

We first show that if Algorithm 1 stops, it outputs a solution to (8). According to Algorithm 1, $u^{(s_m)}$ minimizes $\bar{Q}_{u^{(s_m)}}(x_t, u) \triangleq u^T \bar{P}(\bar{\mathcal{A}}(u^{(s_m)}))u + \bar{q}^T(\bar{\mathcal{A}}(u^{(s_m)}))u + \bar{v}(\bar{\mathcal{A}}(u^{(s_m)}))$ subject to $u \in \mathcal{U}, Ax_t + Bu \in \mathcal{C}_\infty$ if $\bar{\mathcal{A}}(u^{(s_m)}) = \bar{\mathcal{A}}(u^{(s_m-1)})$. In this case, the inequality

$$\nabla_u^T \bar{Q}_{u^{(s_m)}}(x_t, u^{(s_m)})(u - u^{(s_m)}) \geq 0 \quad (\text{B.1})$$

holds for all $u \in \mathcal{U}_1 \triangleq \{u | u \in \mathcal{U}, Ax_t + Bu \in \mathcal{C}\}$. Using the fact that $\nabla_u^T \bar{Q}_{u^{(s_m)}}(x_t, u^{(s_m)}) = \nabla_u^T \hat{Q}(x_t, u^{(s_m)})$, the gradient inequality for the convex PWQ function $\hat{Q}(x_t, \cdot)$ can be applied at $u^{(s_m)}$: $\hat{Q}(x_t, u) \geq \hat{Q}(x_t, u^{(s_m)}) + \nabla_u^T \hat{Q}_{u^{(s_m)}}(x_t, u^{(s_m)})(u - u^{(s_m)})$, which, combined with (B.1), shows that $\hat{Q}(x_t, u) \geq \hat{Q}(x_t, u^{(s_m)}) \forall u \in \mathcal{U}_1$. Thus, optimality of $u^{(s_m)}$ is proven.

If a cycle occurs, i.e., $\exists k \in \{1, 2, \dots, s-2\}$ such that $\bar{\mathcal{A}}(u^{(s)}) = \bar{\mathcal{A}}(u^{(k)})$ for some s , according to Algorithm 1, we select another set of activated units that has never been considered in problem (12). As the number of combinations of activated units is finite, the algorithm must stop in a finite number iterations. \square

C Proof of Lemma 1

Consider the following three cases:

Case 1: $e(x) \geq 0, \forall x \in \text{int}(\mathcal{R}_{i,j})$. In this case, for any $x \in \text{int}(\mathcal{R}_{i,j})$, let $y \in \text{int}(\mathcal{R}_{i,j})$ denote the training or testing point closest to x . Then (14) results in

$$\begin{aligned} |e(x)| &\leq \frac{\hat{J}(y, \theta) + \nabla^T \hat{J}(y, \theta)(x - y) + \hat{L}_j \|x - y\|_2^2 / 2}{J_\infty^*(y) + \nabla^T J_\infty^*(y)(x - y)} - 1 \\ &\leq \frac{\hat{J}(y, \theta) - J_\infty^*(y) + (\nabla \hat{J}(y, \theta) - \nabla J_\infty^*(y))^T (x - y) + \hat{L}_j \|x - y\|_2^2 / 2}{J_\infty^*(y) - \|\nabla^T J_\infty^*(y)\|_2 \|x - y\|_2} \\ &\leq \frac{|e(y)| + e_{\text{grad}}(y) \|x - y\|_2 + \hat{L}_j \|x - y\|_2^2 / (2J_\infty^*(y))}{1 - \beta(y) \|x - y\|_2} \end{aligned}$$

where $\beta(y) = \|\nabla^T J_\infty^*(y)\|_2 / J_\infty^*(y)$. The first inequality is true due to the second inequality of (14) and the convexity of $J_\infty^*(\cdot)$. Using the fact that $|e(y)| \leq \bar{e}, e_{\text{grad}}(y) \leq \bar{e}_{\text{grad}}$, we have

$$|e(x)| \leq \frac{1}{1 - \beta(y)d_{i,j}} \left(\bar{e} + \bar{e}_{\text{grad}}d_{i,j} + \frac{\hat{L}_j d_{i,j}^2}{2J_\infty^*(y)} \right) \quad (\text{C.1})$$

holds for any $x \in \text{int}(\mathcal{R}_{i,j})$. Note that $\beta(y)$ is bounded on all $\mathcal{R}_{i,j}$ except $\mathcal{R}_{1,1}$, since $J_\infty^*(\cdot)$ can only equal 0 at origin, which is in $\mathcal{R}_{1,1}$. Therefore, one can always make $d_{i,j}$ sufficiently small so that $1 - \beta(y)d_{i,j} > 0$.

Case 2: $e(x) \leq 0, \forall x \in \text{int}(\mathcal{R}_{i,j})$. Similarly to Case 1, we can readily get almost the same expression of the upper bound as the right-hand side of (C.1), and the only difference is that \hat{L}_j is replaced by L_i .

Case 3: $\exists x_1, x_2 \in \text{int}(\mathcal{R}_{i,j})$ such that $e(x_1) > 0$ and $e(x_2) < 0$. In this case, for all $x \in \text{int}(\mathcal{R}_{i,j})$ subject to $\hat{J}(x, \theta) \geq J_\infty^*(x)$, we can get the same upper bound for $|e(x)|$ as in (C.1), and for all $x \in \text{int}(\mathcal{R}_{i,j})$ subject to $\hat{J}(x, \theta) < J_\infty^*(x)$, (C.1) holds with \hat{L}_j replaced by L_i .

D Proofs of Theorem 5 and Corollary 1

In Theorem 5, $\mathcal{C} = \bar{X}$. First consider the solution to (2). For every $x_t, t = 0, 1, \dots$, the MPC control law u_t^* , which contains the first m elements of $U^*(x_t)$, will be applied to the system. From the equivalence in Theorem 1, $\{u_t^*\}_{t=0}^\infty$ is also a solution to the (1). As a result, $Ax_t + Bu_t^* \in \bar{X}$ for any $x_t \in \bar{X}$, (else, $J_\infty^*(Ax_t + Bu_t^*) = \infty$ and $J_\infty^*(x_t) = \infty$, which contradicts the claim that $x_t \in \bar{X}$).

Suppose that $x_t \in \bar{X} \setminus \{0\}$. Applying the Bellman Optimality Equation [23] for (1) leads to

$$\begin{aligned} J_\infty^*(Ax_t + Bu_t^*) &= J_\infty^*(x_t) - x_t^T Q x_t - u_t^{*T} R u_t^* \\ &< J_\infty^*(x_t) - x_t^T Q x_t \end{aligned} \quad (\text{D.1})$$

Combining (D.1) with (13), we have

$$\begin{aligned} &x_t^T Q x_t + \hat{u}_t^{*T} R \hat{u}_t^* + \hat{J}(Ax_t + B\hat{u}_t^*, \theta) \\ &\leq x_t^T Q x_t + u_t^{*T} R u_t^* + \hat{J}(Ax_t + Bu_t^*, \theta) \\ &\leq x_t^T Q x_t + u_t^{*T} R u_t^* + J_\infty^*(Ax_t + Bu_t^*) + \zeta J_\infty^*(Ax_t + Bu_t^*) \\ &< J_\infty^*(x_t) + \zeta J_\infty^*(x_t) - \zeta x_t^T Q x_t \\ &< \hat{J}(x_t, \theta) + 2\zeta J_\infty^*(x_t) - \zeta x_t^T Q x_t \end{aligned}$$

The first inequality is true since \hat{u}_t^* is a minimizer of (8). The second and last inequalities hold due to (13), and the third line is true thanks to the optimal Bellman equation in (D.1). Then, combining (13) and (16) yields

$$\frac{1 + \zeta}{\zeta} > 2 \sup_{x \in \bar{X} \setminus \{0\}} \frac{\hat{J}(x, \theta)}{(1 - \zeta)x^T Q x} > 2 \sup_{x \in \bar{X} \setminus \{0\}} \frac{J_\infty^*(x)}{x^T Q x}$$

which implies

$$\begin{aligned} &\hat{J}(Ax_t + B\hat{u}_t^*, \theta) - \hat{J}(x_t, \theta) \\ &< -(1 + \zeta)x_t^T Q x_t - \hat{u}_t^{*T} R \hat{u}_t^* + 2\zeta J_\infty^*(x_t) < 0, \forall x_t \in \bar{X} \setminus \{0\} \end{aligned}$$

Let $x_0 \in \bar{X} \setminus \{0\}$ and x_1, x_2, \dots be the trajectory of the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*, t = 0, 1, \dots$. The sequence $\hat{J}(x_0, \theta), \hat{J}(x_1, \theta), \dots$ is strictly decreasing. Besides, it is easy to check that $\hat{J}(0, \theta) = 0, \hat{J}(x, \theta) > 0, \forall x \in \bar{X} \setminus \{0\}$, and $\hat{J}(\cdot, \theta)$ is continuous at the origin,

finite in \bar{X} . Then, $\hat{J}(\cdot, \theta)$ is therefore a Lyapunov Function according to [9, Theorem 7.2]. So, the asymptotic stability of the origin for any $x \in \bar{X}$ follows. \square

For the proof of Corollary 1, from the definition of Ω and the continuity of $\hat{J}(\cdot, \theta)$, we know that $\Omega \subseteq X_0$. Following the proof of Theorem 5, we can show that for all $x_t \in \Omega$, $\hat{J}(Ax_t + B\hat{u}_t^*, \theta) - \hat{J}(x_t, \theta) < 0$ holds. As a result, the set Ω is positively invariant w.r.t. the closed-loop system $x_{t+1} = Ax_t + B\hat{u}_t^*$. The recursive feasibility thus follows from $\Omega \subseteq X_0 \subseteq \mathcal{X}$. Similar to the proof of Theorem 5, it can be shown that $\hat{J}(\cdot, \theta)$ is a Lyapunov function and the stability of the origin follows. \square

References

- [1] P. Werbos, "Approximate dynamic programming for realtime control and neural modelling," *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pp. 493–525, 1992.
- [2] D. Kleinman, "On an iterative technique for riccati equation computations," *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 114–115, 1968.
- [3] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 943–949, 2008.
- [4] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [5] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.
- [6] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 1520–1527.
- [7] A. Chakrabarty, R. Quirynen, C. Danielson, and W. Gao, "Approximate dynamic programming for linear systems with state and input constraints," in *2019 18th European Control Conference (ECC)*, 2019, pp. 524–529.
- [8] A. Chakrabarty, D. K. Jha, G. T. Buzzard, Y. Wang, and K. G. Vamvoudakis, "Safe approximate dynamic programming via kernelized lipschitz estimation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 405–419, 2020.
- [9] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
- [10] D. Chmielewski and V. Manousiouthakis, "On constrained infinite-time linear quadratic optimal control," *Systems & Control Letters*, vol. 29, no. 3, pp. 121–129, 1996.
- [11] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.
- [12] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020.
- [13] E. T. Maddalena, C. G. de Silva Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit mpc controllers from data," *arXiv preprint arXiv:1911.10789*, 2019.
- [14] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets," *Automatica*, vol. 135, p. 109947, 2022.
- [15] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for piecewise affine systems with neural network controllers," *arXiv preprint arXiv:2008.06546*, 2020.
- [16] E. T. Maddalena, C. G. de Silva Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit MPC controllers from data," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 362–11 367, 2020.
- [17] E. C. Kerrigan and J. M. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 5, 2000, pp. 4951–4956.
- [18] K. He, T. van den Boom, and B. De Schutter, "Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach," *arXiv preprint arXiv:2205.10065*, 2022.
- [19] M. Baotić, F. Borrelli, A. Bemporad, and M. Morari, "Efficient on-line computation of constrained optimal control," *SIAM Journal on Control and Optimization*, vol. 47, no. 5, pp. 2470–2489, 2008.
- [20] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.
- [21] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.
- [22] S. Fahandezh-Saadi and M. Tomizuka, "In proximity of ReLU DNN, PWA function, and explicit MPC," *arXiv preprint arXiv:2006.05001*, 2020.
- [23] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.
- [24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [25] F. V. Louveaux, "Piecewise convex programs," *Mathematical Programming*, vol. 15, no. 1, pp. 53–62, 1978.
- [26] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [27] A. Rinnooy Kan and G. Timmer, "Stochastic global optimization methods part II: Multi level methods," *Mathematical Programming*, vol. 39, no. 1, pp. 57–78, 1987.
- [28] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2009.
- [29] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.