IEEE Open Journal of Control Systems

Received 8 April 2025; revised 10 July 2025 and 2 September 2025; accepted 3 September 2025. Date of publication 9 September 2025; date of current version 22 October 2025. Recommended by Guest Editor Kyriakos G. Vamvoudakis.

Digital Object Identifier 10.1109/OJCSYS.2025.3607844

On the Representation of Piecewise Quadratic Functions by Neural Networks

DIETER TEICHRIB (Member, IEEE) AND MORITZ SCHULZE DARUP (Senior Member, IEEE)

(Intersection of Machine Learning with Control)

Control and Cyberphysical Systems Group, TU Dortmund University, 44227 Dortmund, Germany

CORRESPONDING AUTHOR: DIETER TEICHRIB (e-mail: dieter.teichrib@tu-dortmund.de).

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant 556837821.

ABSTRACT Neural networks (NNs) are commonly used to approximate functions based on data samples, as they are a universal function approximator for a large class of functions. However, choosing a suitable topology in terms of depth, width and activation function for NNs that allow for low error approximations is a non-trivial task. For the approximation of continuous piecewise affine (PWA) functions, this task has been solved by showing that for every PWA function, there exist NNs with rectified linear unit (relu) and maxout activation that allow an exact representation of the PWA function. This connection between PWA functions and NNs has led to some valuable insights into the representation capabilities of NNs. Moreover, the connection was used in control for approximating the PWA optimal control law of model predictive control (MPC) for linear systems. We show that a similar connection exists between NNs and continuous piecewise quadratic (PWQ) functions by deriving topologies for NNs that allow an exact representation of arbitrary PWQ functions with a polyhedral domain partition. Furthermore, we demonstrate that the proposed NNs can efficiently approximate the PWQ optimal value function for linear MPC.

INDEX TERMS Deep learning, machine learning, neural networks, optimization, predictive control.

I. INTRODUCTION

Neural networks (NNs) have been successfully applied in many fields, as they allow the approximation of a large class of functions with arbitrary accuracy [1]. However, the approximation quality of NNs heavily depends on their topology, e.g., depth, width, and activation. Choosing these hyperparameters is often done in a trial-and-error fashion due to the lack of design rules for many applications. However, for some applications, it is possible to exploit knowledge of the function to be approximated for choosing a suitable topology. For example, in model predictive control (MPC) for linear systems with quadratic cost and polyhedral constraints, it is well-known that the solution of the optimal control problem (OCP) leads to a control law that is a piecewise affine (PWA) function of the state [2]. There are several approaches [3], [4], [5] where the optimal control law is approximated by an NN to provide a fast evaluation without the need to solve an optimization problem (OP) online on the control device. In this case, it is possible to construct NNs with rectified linear unit (relu) or maxout activation that theoretically allow an exact representation of PWA functions [6], [7] and, thus, of the optimal control law [8], [9]. This connection between NNs and the optimal control law is used in [10] to derive topologies for NNs that are well-suited for the approximation of the optimal control law. The connection between NN and PWA functions was also recognized in [11], [12] and used to generate, represent, and approximate arbitrary PWA functions for circuit design. Thus, for PWA functions, the connection to NNs has led to some valuable insights into the representation capabilities and design of NNs. However, these results do not apply to other function types.

In extending the results from the PWA case, a natural next step could be to search for a link between NNs and continuous piecewise quadratic functions (PWQ). Finding such a link would enable the design of NNs that allow an exact representation of PWQ functions and provide design guidelines for choosing a suitable topology for the approximation of these functions. Apart from the pure theoretical value of a

connection between NNs and PWQ functions, such a result is also of practical interest in the context of control in cases where PWQ functions need to be approximated [13], [14], [15]. An application example is the use of model-based reinforcement learning (RL) or approximate dynamic programming (ADP) [16], [17], [18], [19] for MPC to derive a control policy. In model-based RL and ADP, an optimal value function (OVF) is learned online and offline, respectively, by some function approximator. The learned OVF allows for the acceleration of the online evaluation of MPC by reducing the prediction horizon to one without a significant loss in control performance [19], [20]. One of the first uses of ADP in control is presented [20], but only for unconstrained linear systems. In this case, the OVF is a quadratic function that can be learned by a recursive least squares algorithm [20, Sec. 4]. In MPC for linear systems with quadratic cost and polyhedral constraints, the OVF is a continuous and convex PWQ function of the state [2]. Thus, when applying ADP to constrained linear systems, a PWQ function has to be learned [17], [19]. In [17], this is done by first clustering sampled data from the OVF and then fitting a quadratic function to each cluster. Since the quadratic functions are fitted individually for each cluster, continuity of the resulting PWQ function is not guaranteed. This may lead to undesired behaviour of the controlled system. An alternative approach is presented in [19] where a NN with a PWQ structure is used to learn the OVF. The PWQ structure of the NN is ensured by squaring each neuron in the last layer of the NN. The drawback of this approach is that there are no theoretical results on the representation capabilities of the resulting NN. Thus, there may exist PWQ functions that cannot be represented by the NN form [19]. We will overcome the aforementioned issues by introducing NNs that inherently provide continuity and that can represent every PWQ function. Another example of PWQ functions in control is the use of model-free reinforcement learning methods [21], such as Q-learning [22], [23], where Q-functions are approximated. In MPC for linear systems with quadratic cost and polyhedral constraints, the Q-function is, as the OVF, a continuous PWQ function of the state [2]. Thus, in summary, a link between PWQ functions and NNs could be used to derive suitable topologies for ADP and reinforcement learning where the OVF and Q-function are PWQ, e.g., in MPC for linear systems. While there are results on NNs that allow an exact representation of PWA functions, PWQ functions have rarely been considered. There are some results [24], [25], [26], [27], [28] on PWQ functions for special cases. In [29] and [28], it is shown that for one-dimensional PWQ functions $\varphi : \mathbb{R} \to \mathbb{R}$, there exists relu and maxout NN, respectively, that allow an exact representation. Moreover, in [24], [25], [26], the authors show that PWQ functions can be represented in a form that can be interpreted as NN if the domain partition consists of polyhedral sets and satisfies some regularity conditions, which are detailed in Section II-B. However, to the best of our knowledge, there are no results for the general n-dimensional case with an arbitrary polyhedral domain partition. The main contribution of this paper is to address this lack by showing that every continuous PWQ function with a polyhedral domain partition can be represented exactly by relu and maxout NNs. Furthermore, we will experimentally validate that NNs that allow an exact representation of PWQ functions perform better in approximating OVFs than NNs that do not allow an exact representation.

The paper is organized as follows. In the rest of this section, we introduce relevant notations. In Section II, we briefly summarize relevant background on PWQ functions, NNs, MPC, and learning-based control. The theoretical basis for the main results of the paper is derived in Section III. In Section IV, we use the results from Section III to derive maxout and relu NNs that allow an exact representation of PWQ functions with a polyhedral domain partition. In Section V, we first illustrate the representation of PWQ functions by NNs using an exemplary PWQ function. Subsequently, we demonstrate the benefits of the proposed NNs for learning PWQ functions by using them to learn OVFs of linear MPC. The paper is concluded in Section VI with a conclusion and a summary.

A. NOTATION

We denote the set of natural, real and positive real numbers by $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{R}_+$, respectively. We refer to the i-th row of a matrix by $\boldsymbol{Q}_i$ and the element in the i-th row and j-th column by $\boldsymbol{Q}_{i,j}$. The identity matrix of dimension $n \in \mathbb{N}$ is denoted by $\boldsymbol{I}_n$.

II. PRELIMINARIES AND BACKGROUND

Throughout the paper, we deal with continuous PWQ functions $\varphi(\boldsymbol{x}) : \Omega \to \mathbb{R}$ of the form

$$\varphi(\boldsymbol{x}) := \begin{cases} \varphi^{(1)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \varphi^{(s)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{1}$$

with a bounded polyhedral domain $\Omega \subset \mathbb{R}^n$ that is partitioned according to $\cup_{i=1}^{s} \mathcal{R}^{(i)} = \Omega$ by s polyhedral regions

$$\mathcal{R}^{(i)} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^{(i)}\boldsymbol{x} + \boldsymbol{b}^{(i)} \leq \boldsymbol{0}\} \tag{2}$$

with $d^{(i)} \in \mathbb{N}$ hyperplanes specified by $\boldsymbol{h}^{(i)} \in \mathbb{R}^{d^{(i)} \times n}$ and $\boldsymbol{b}^{(i)} \in \mathbb{R}^{d^{(i)}}$ for all $i \in \{1, \dots, s\}$. The sets $\mathcal{R}^{(i)}$ have non-empty and pairwise disjoint interiors. We assume without loss of generality that the rows of the matrices $\boldsymbol{h}^{(i)}$ are normalized, i.e., $||\boldsymbol{h}_j^{(i)}||_2 = 1$ for all $j \in \{1, \dots, d^{(i)}\}$ and for all $i \in \{1, \dots, s\}$. We define the i-th local function as

$$\varphi^{(i)}(\boldsymbol{x}) := \boldsymbol{x}^\top \boldsymbol{Q}^{(i)}\boldsymbol{x} + \boldsymbol{l}^{(i)}\boldsymbol{x} + c^{(i)}$$

for all $i \in \{1, \dots, s\}$. Moreover, the function (1) is assumed to be continuous, i.e, $\varphi^{(p)}(\boldsymbol{x}) = \varphi^{(r)}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{R}^{(p)} \cap \mathcal{R}^{(r)}$ with $(p, r) \in \mathcal{N}$, where

$$\mathcal{N} := \{(i, j) \in \{1, \dots, s\}^2 \mid \dim(\mathcal{R}^{(i)} \cap \mathcal{R}^{(j)}) = n - 1\}$$

is the set of indices of neighbouring regions. The matrices $\boldsymbol{Q}^{(i)} \in \mathbb{R}^{n \times n}$, the row-vectors $\boldsymbol{l}^{(i)} \in \mathbb{R}^{1 \times n}$, and the constants $c^{(i)} \in \mathbb{R}$ reflect the quadratic, linear, and constant parts of the

various local functions. We assume that the matrices $\boldsymbol{Q}^{(i)}$ are symmetric. Note that this assumption is not restrictive. Even if there is a quadratic term $\boldsymbol{x}^\top \tilde{\boldsymbol{Q}}^{(i)} \boldsymbol{x}$ with a non-symmetric matrix $\tilde{\boldsymbol{Q}}^{(i)}$ it is always possible to construct a symmetric $\boldsymbol{Q}^{(i)} = 0.5(\tilde{\boldsymbol{Q}}^{(i)} + \tilde{\boldsymbol{Q}}^{(i)\top})$ with $\boldsymbol{x}^\top \boldsymbol{Q}^{(i)} \boldsymbol{x} = \boldsymbol{x}^\top \tilde{\boldsymbol{Q}}^{(i)} \boldsymbol{x}$. We further define the hyperplane separating the regions $\mathcal{R}^{(p)}$ and $\mathcal{R}^{(r)}$ with $(p, r) \in \mathcal{N}$ as

$$\mathcal{H}^{(p,r)} := \mathcal{R}^{(p)} \cap \mathcal{R}^{(r)} \subset \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)} = 0\}, \quad (3)$$

where $\boldsymbol{h}^{(p,r)}$ and $b^{(p,r)}$ are rows of the vectors $\boldsymbol{h}^{(p)}$ and $\boldsymbol{b}^{(p)}$, respectively. We thus have $\boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)} \leq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(p)}$ as well as $\boldsymbol{h}^{(r,p)} = -\boldsymbol{h}^{(p,r)}$ and $b^{(r,p)} = -b^{(p,r)}$. Finally, we denote the regions $\mathcal{R}^{(p)}$ and $\mathcal{R}^{(r)}$ as neighbouring if $(p, r) \in \mathcal{N}$. Although we are considering functions with a one-dimensional codomain, the results in the following sections can be extended to functions with a m-dimensional codomain by applying them to each dimension of the codomain individually.

We will frequently use the following relation from [25, Thm. 1] for continuous piecewise defined polynomials. For two neighbouring regions of the domain partition, i.e., $(p, r) \in \mathcal{N}$ of a continuous piecewise defined polynomial, there always exists a function $g^{(p,r)}(\boldsymbol{x})$ such that

$$\varphi^{(p)}(\boldsymbol{x}) - \varphi^{(r)}(\boldsymbol{x}) = e^{(p,r)}(\boldsymbol{x}) g^{(p,r)}(\boldsymbol{x}) \quad (4)$$

holds for all $\boldsymbol{x} \in \Omega$, where $\{\boldsymbol{x} \in \mathbb{R}^n \mid e^{(p,r)}(\boldsymbol{x}) = 0\}$ is the boundary separating the regions $\mathcal{R}^{(p)}$ and $\mathcal{R}^{(r)}$, i.e., $e^{(p,r)}(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(p)} \cap \mathcal{R}^{(r)}$. The relationship between $\varphi^{(p)}(\boldsymbol{x})$ and $\varphi^{(r)}(\boldsymbol{x})$ provided by (4) allows to derive some interesting facts. First of all, the order of the polynomial $e^{(p,r)}(\boldsymbol{x})$ describing the boundary is, at most, the order of the function $\varphi(\boldsymbol{x})$. Second, for the case considered in the paper, i.e, a PWQ function with a polyhedral domain partition, $e^{(p,r)}(\boldsymbol{x}) = \boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)}$ is an affine function and thus so is $g^{(p,r)}(\boldsymbol{x})$ [25]. We thus have

$$e^{(p,r)}(\boldsymbol{x}) = \boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)} \Rightarrow g^{(p,r)}(\boldsymbol{x}) = \boldsymbol{k}^{(p,r)} \boldsymbol{x} + d^{(i,j)}.$$

We will use this result in Section III to derive NNs for representing PWQ functions. Prior to this, we introduce our notion of NNs.

A. NEURAL NETWORKS

In general, a multi-layer or deep feed-forward-NN with $l \in \mathbb{N}$ hidden layers and w_i neurons in layer i can be written as a composition of the form

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{f}^{(l+1)} \circ \boldsymbol{g}^{(l)} \circ \boldsymbol{f}^{(l)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{\xi}). \quad (5)$$

Here, the functions $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{p_i w_i}$ for $i \in \{1, \ldots, l\}$ refer to preactivations, where the parameter $p_i \in \mathbb{N}$ allows to consider "multi-channel" preactivations as required for maxout (see [30]). The values of $\max_{1 \leq i \leq l} \{w_i\}$ and l are referred to as the width and depth of the NN. Moreover, $\boldsymbol{g}^{(i)} : \mathbb{R}^{p_i w_i} \to \mathbb{R}^{w_i}$ stand for activation functions and $\boldsymbol{f}^{(l+1)} : \mathbb{R}^{w_l} \to \mathbb{R}^{w_{l+1}}$ reflects postactivation. The functions $\boldsymbol{f}^{(i)}$ are typically affine, i.e.,

$$\boldsymbol{f}^{(i)}(\boldsymbol{y}^{(i-1)}) = \boldsymbol{W}^{(i)} \boldsymbol{y}^{(i-1)} + \boldsymbol{v}^{(i)}, \quad (6)$$

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i w_i \times w_{i-1}}$ is a weighting matrix, $\boldsymbol{v}^{(i)} \in \mathbb{R}^{p_i w_i}$ is a bias vector, and $\boldsymbol{y}^{(i-1)}$ denotes the output of the previous layer with $\boldsymbol{y}^{(0)} := \boldsymbol{\xi}$. We will summarize the parameters of an NN by $\boldsymbol{\theta} := \{\boldsymbol{W}^{(1)}, \boldsymbol{v}^{(1)}, \ldots, \boldsymbol{W}^{(l+1)}, \boldsymbol{v}^{(l+1)}\}$ and occasionally use the notation $\boldsymbol{\Phi}(\boldsymbol{\xi}, \boldsymbol{\theta})$ when we explicitly refer to the dependence of the NN on the weights and biases.

Now, various activation functions have been proposed. We focus here on PWA activation functions as they provide the most promising basis for our purpose. Therefore, we consider the relu activation function

$$\boldsymbol{g}^{(i)}_{\text{relu}}(z^{(i)}) = \max\{\boldsymbol{0}, z^{(i)}\} := \begin{pmatrix} \max\{0, z_1^{(i)}\} \\ \vdots \\ \max\{0, z_{w_i}^{(i)}\} \end{pmatrix} \quad (7)$$

and the maxout activation function

$$\boldsymbol{g}^{(i)}_{\max}(z^{(i)}) = \begin{pmatrix} \max\limits_{1 \leq j \leq p_i} \{z_j^{(i)}\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1 \leq j \leq p_i w_i} \{z_j^{(i)}\} \end{pmatrix}, \quad (8)$$

where we use the shorthand notation $\max_{1 \leq j \leq p_i} \{z_j^{(i)}\} := \max\{z_1^{(i)}, \ldots, z_{p_i}^{(i)}\}$. We will refer to the resulting NNs as relu NNs and maxout NNs, respectively. For relu and maxout NNs, it is known that they are continuous PWA functions [6], [30] and thus not suitable for representing PWQ functions. However, in Section IV-A, we will modify these NNs such that they can be used to represent continuous PWQ functions.

B. REPRESENTATION OF PWQ FUNCTIONS

We aim for a representation of continuous PWQ functions in a form that can be interpreted as NN. Inspired by research on the representation of PWA and PWQ functions by NN [28], [29], [31], [32], we investigate under which conditions we can find a continuous PWA function $h(\boldsymbol{x}) : \Omega \to \mathbb{R}$ of the form

$$h(\boldsymbol{x}) := \begin{cases} h^{(1)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ h^{(s)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \quad (9)$$

with local functions $h^{(i)}(\boldsymbol{x}) := \boldsymbol{\beta}^{(i)} \boldsymbol{x} + \gamma^{(i)}$ for all $i \in \{1, \ldots, s\}$ such that

$$\varphi(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} - \max_{1 \leq i \leq s} \{h^{(i)}(\boldsymbol{x})\} \quad (10)$$

holds for all $\boldsymbol{x} \in \Omega$. Where we define

$$\tilde{\varphi}(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) \quad (11)$$

and $\tilde{\varphi}^{(i)}(\boldsymbol{x}) := \varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ for all $i \in \{1, \ldots, s\}$. The representation (10) is particularly well-suited for our purpose due to several reasons. First, the difference between two max-expressions (10) can be represented by a maxout NN.

Moreover, results on the connection between maxout NNs and relu NNs [6] can then be used to derive relu NN for representing (10). The connection between (10), relu NNs, and maxout NNs described in [6], [7], [32] motivates the focus on the activation functions (7) and (8). We discuss this connection in more detail in Section IV-A. Second, for certain functions, it is already known that a representation of the form (10) exists. For example, for PWA $\varphi(\boldsymbol{x})$, i.e., $\boldsymbol{Q}^{(i)} = \boldsymbol{0}$ for all $i \in \{1, \dots, s\}$, the function $\varphi(\boldsymbol{x})$ can be decomposed in the difference of two convex PWA functions [32]. Each of these convex PWA functions can be represented as the maximum of their local functions, which results in the end in a representation of the form (10). In Fig. 1(a), the previously described method for evaluating convex PWA functions is shown. The exemplary function in the figure is evaluated by evaluating all the local functions, potentially outside their respective domains, for a given point, here $x = -0.15$, and subsequently taking the maximum. For the example we have $\varphi(x = -0.15) = \max\{\varphi^{(1)}(-0.15),\ \varphi^{(2)}(-0.15),\ \varphi^{(3)}(-0.15)\} = \max\{0.575, 1, 0.425\} = 1$.

In principle, PWQ can also be decomposed into the difference of two convex PWQ functions [33]. However, for the PWQ case, it is not possible to represent the convex PWQ functions as the maximum of their local functions. In Fig. 1(b), this is illustrated for a convex PWQ function with three regions where we have $\varphi(x = -0.15) = 0.00225 \neq 1.3765 = \max\{0.6865,\ 0.00225,\ 1.3765\} = \max\{\varphi^{(1)}(-0.15), \varphi^{(2)}(-0.15), \varphi^{(3)}(-0.15)\}$. Thus, a straightforward extension of the approach used to represent PWA functions in the form (10) is not possible.

Nevertheless, there are some results for the representation of continuous PWQ functions. In [26], the authors showed that PWQ functions can be represented in a form that can be interpreted as an NN if the domain partition is non-degenerate. In a non-degenerate partition, the intersection of any k hyperplanes that partition the domain is of dimension less or equal to $n - k$ [34]. This also means that any intersection of more than n hyperplanes must be an empty set. The results we will derive in this article also hold for the case where the intersection of k hyperplanes that partition the domain is of dimension greater than $n - k$. Moreover, in [25], it is shown that PWQ functions can be represented as the sum of relu-like functions if the PWQ functions possess the so-called consistent variation property [34, Def. 5]. For our case this means that the function $g^{(\tilde{p},\tilde{r})}(\boldsymbol{x})$ from (4) is the same for all regions $\mathcal{R}^{(\tilde{p})}$ and $\mathcal{R}^{(\tilde{r})}$ that are separated by the hyperplane $\{\boldsymbol{x} \in \mathbb{R}^n \mid e^{(p,r)}(\boldsymbol{x}) = 0\}$. Or more precisely $g^{(p,r)}(\boldsymbol{x}) = g^{(\tilde{p},\tilde{r})}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and for all $(\tilde{p}, \tilde{r}) \in \mathcal{N}$ with $\mathcal{R}^{(\tilde{p})} \cap \mathcal{R}^{(\tilde{r})} \subset \{\boldsymbol{x} \in \mathbb{R}^n \mid e^{(p,r)}(\boldsymbol{x}) = 0\}$. Finally, in [35] and [28], results on the representation of piecewise polynomial and PWQ functions, respectively, for $n = 1$ are presented. In [28], it is additionally shown that for PWQ functions with $n = 1$, there exists a continuous and convex PWA $h(x)$ such that (10) holds.

In Section IV, we will extend the results from the literature by showing that every continuous PWQ function with arbitrary $n \in \mathbb{N}$ can be represented in the form (10). Thus, we will overcome the limitations from the known results as our representation does not require $n = 1$, a non-degenerate partition, or the consistent variation property.

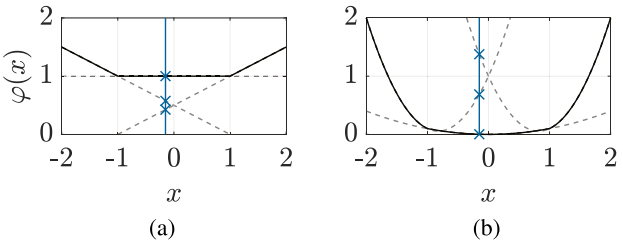

FIGURE 1. Exemplary convex PWA and convex PWQ function with three regions in (a) and (b), respectively. For both functions, the dashed grey lines represent the extension of the local functions $\varphi^{(1)}(\boldsymbol{x})$, $\varphi^{(2)}(\boldsymbol{x})$, and $\varphi^{(3)}(\boldsymbol{x})$ outside their respective regions $\mathcal{R}^{(1)} = \{\boldsymbol{x} \in \mathbb{R} \mid -2 \leq x \leq -1\}$, $\mathcal{R}^{(2)} = \{\boldsymbol{x} \in \mathbb{R} \mid -1 \leq x \leq 1\}$, and $\mathcal{R}^{(3)} = \{\boldsymbol{x} \in \mathbb{R} \mid 1 \leq x \leq 2\}$. The local functions are evaluated for $x = -0.15$, which is illustrated by the blue vertical line. Only for the convex PWA function, we have $\max\{\varphi^{(1)}(\boldsymbol{x}), \varphi^{(2)}(\boldsymbol{x}), \varphi^{(3)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x})$.

C. MODEL PREDICTIVE CONTROL

Classical MPC for linear systems builds on solving an OCP of the form

$$
\begin{aligned}
V_N(\boldsymbol{x}) := \min_{\substack{\boldsymbol{x}(0),\dots,\boldsymbol{x}(N)\\ \boldsymbol{u}(0),\dots,\boldsymbol{u}(N-1)}} \ & \ell_N(\boldsymbol{x}(N)) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\
\text{s.t.} \quad & \boldsymbol{x}(0) = \boldsymbol{x}, \\
& \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \quad \forall k \in \{0, \dots, N-1\}, \\
& (\boldsymbol{x}(k), \boldsymbol{u}(k)) \in \mathcal{X} \times \mathcal{U}, \qquad \forall k \in \{0, \dots, N-1\}, \\
& \boldsymbol{x}(N) \in \mathcal{T}
\end{aligned} \tag{12}
$$

in every time step $\kappa \in \mathbb{N}$ for the current state $\hat{\boldsymbol{x}}(\kappa) = \boldsymbol{x} \in \mathbb{R}^n$. Here, $N \in \mathbb{N}$ refers to the prediction horizon and

$$
\ell_N(\boldsymbol{x}) := \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \quad \text{and} \quad \ell(\boldsymbol{x}, \boldsymbol{u}) := \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}
$$

denote the terminal and stage cost, respectively, where the weighting matrices $\boldsymbol{P} \in \mathbb{R}^{n\times n}$, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ are positive (semi-) definite. The dynamics of the linear prediction model are described by $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$. The state and input constraints are given by the polyhedral sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$. Finally, the set $\mathcal{T} \subset \mathbb{R}^n$ is a terminal set that may be used to enforce closed-loop stability (see [36] for details). For the considered setup, it is well-known that the control policy $\boldsymbol{\pi} : \mathcal{F}_N \to \mathcal{U}$ with $\boldsymbol{\pi}(\boldsymbol{x}) := \boldsymbol{u}^*(0)$ is a continuous PWA function of the form (9) with domain $\Omega = \mathcal{F}_N$ where $\mathcal{F}_N \subseteq \mathcal{X}$ is the feasible set of (12) [2, Thm. 4]. Moreover, the OVF $V_N(\boldsymbol{x})$ is a continuous and convex PWQ function of the form (1) with domain $\Omega = \mathcal{F}_N$.

Typically, when MPC is applied to a system, the OCP (12) is solved in every time step κ for the current state $\hat{\boldsymbol{x}}(\kappa)$ and only the first element $\boldsymbol{u}^*(0)$ of the optimal input sequence is applied to the system. Thus, the OCP needs to be solved within the sampling period of the system, which may be intractable for systems with a short sampling period or if the

OCP has a large number of constraints, which may result from a long prediction horizon. Solutions to this problem are explicit MPC [2] or learning-based predictive control [10], where we exploit the PWA structure of the control policy or the PWQ structure of the OVF.

D. LEARNING-BASED PREDICTIVE CONTROL

In principle, the PWA function describing the control policy $\boldsymbol{\pi}(\boldsymbol{x})$ can be computed explicitly offline and stored on the control device. In this case, the computation of the optimal control input reduces to the evaluation of a PWA function. Thus, solving the OCP (12) online on the control device is no longer necessary. However, explicitly computing the control policy offline or even evaluating it online may be intractable for systems with large state dimensions or a long prediction horizon since the complexity in terms of the number of regions of the PWA function grows, in the worst case, exponentially with the number of constraints in the OCP (12) [2]. Different methods from machine learning (ML) have been used to tackle this challenge by approximating the solution of the OCP in different ways with NNs. The most direct approach is to use an NN of the form (5) for an approximation in the policy space. In this case, samples $(\boldsymbol{x}, \boldsymbol{\pi}(\boldsymbol{x}))$ of the PWA control policy $\boldsymbol{\pi}(\boldsymbol{x})$ are used to train an NN. Due to the involved multi-layer NN, approaches of this type are also known as deep MPC and have been considered in [3], [4], [10], [37], for linear systems and in [38] for nonlinear systems. Moreover, in [10], the authors derive NNs that allow an exact representation of the PWA control policy. Alternatively, one can also consider an approximation in the value space as in [13], [18], [19], where the PWQ OVF $V_N(\boldsymbol{x})$ or variants of it are approximated by NNs. In ADP [17], [19] the OVF is approximated based on samples $(\boldsymbol{x}, V_N(\boldsymbol{x}))$ for which the OCP (12) has been solved. Using the OVF an optimal input can be computed by $\boldsymbol{u}^*(\boldsymbol{x}) := \arg\min_{\boldsymbol{u}} \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u})$ [19]. However, this approach requires knowledge of the system parameters $\boldsymbol{A}$ and $\boldsymbol{B}$. There are also reinforcement learning-based approaches that do not require a model of the system. In these approaches, the so-called Q-function (see, e.g., [18, P. 13])

$$Q_N(\boldsymbol{x}, \boldsymbol{u}) := \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \tag{13}$$

is learned iteratively by an NN with, e.g., Q-learning [22], [23], which is a variant of temporal difference learning [39], [40]. The Q-function (13) consists of the cost for a first step $\ell(\boldsymbol{x}, \boldsymbol{u})$ with an undetermined input $\boldsymbol{u}$ followed by the optimal cost for a control sequence of length $N-1$. When learning the Q-function, the successor state $\boldsymbol{x}^+ := \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ is typically determined by measurement, i.e., without using the model parameters $\boldsymbol{A}$ and $\boldsymbol{B}$. Using the Q-function an optimal input can be computed by $\boldsymbol{u}^*(\boldsymbol{x}) := \arg\min_{\boldsymbol{u}} Q_N(\boldsymbol{x}, \boldsymbol{u})$. In MPC for linear systems as specified by (12), the OVF and, thus, the Q-function (13) are PWQ functions with a polyhedral domain partition [2, Thm. 4]. Thus, when approximating the OVF or the Q-function for MPC by an NN, the NN has to represent a PWQ function. Approximations of this type for MPC have been considered in, e.g., [13] and [14]. However, in these papers, PWA NN are used to approximately represent PWQ functions. There are also approaches with PWQ NN. In [19] the authors propose a "local-global" topology in the form $\boldsymbol{x}^\top \boldsymbol{P}^* \boldsymbol{x} + \Phi(\boldsymbol{x})$ (cf. [19, Eq. (5)]) where the local part $\Phi(\boldsymbol{x})$ is a NN in which all neurons of the last layer are squared and the global part $\boldsymbol{x}^\top \boldsymbol{P}^* \boldsymbol{x}$ is intended to promote an accurate representation of a PWQ OVF in a small region around the origin. However, there are no theoretical guarantees that the NN proposed in [19] can represent any PWQ function. The NNs we will propose do not require the separation into a local and global part, and they provably allow the exact representation of arbitrary PWQ functions.

We start our derivation in the following by introducing a representation of the form (10) that allows an exact representation of arbitrary PWQ functions and, thus, of the OVF and Q-function. Moreover, we will use the connection between PWQ functions and (10) to derive maxout and relu NNs for representing PWQ functions exactly. In the numerical examples, we will also show that NNs, which can represent PWQ functions exactly, perform better in approximating PWQ functions based on samples than classical NN, for which an exact representation of PWQ functions is not possible.

III. PWQ FUNCTIONS WITH POLYHEDRAL PARTITION

In this section, we derive the theoretical basis for the representation of PWQ functions as the difference of two max-expressions by first deriving sufficient conditions that ensure (10). In the second part of this section, we show that it is always possible to satisfy the derived conditions using a continuous and convex PWA function $h(\boldsymbol{x})$ as in (9).

A. SUFFICIENT CONDITIONS

We start the derivation of sufficient conditions that ensure (10) by noting that if

$$\max_{1\le i\le s} \{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) \text{ and} \tag{14}$$

$$\max_{1\le i\le s} \{h^{(i)}(\boldsymbol{x})\} = h(\boldsymbol{x}) \tag{15}$$

hold for all $\boldsymbol{x} \in \Omega$, then we have $\max_{1\le i\le s}\{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} - \max_{1\le i\le s}\{h^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) - h(\boldsymbol{x}) = \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ and thus we can represent the PWQ function $\varphi(\boldsymbol{x})$ in the desired form (10). Therefore, in the remainder of this section, we will investigate under which conditions the PWA function $h(\boldsymbol{x})$ is such that (14) and (15) hold. We first present an intermediate result, which we will use to prove that (14) holds for all $\boldsymbol{x} \in \Omega \subset \mathbb{R}^n$.

Lemma 1: Let the PWQ function $\varphi(\boldsymbol{x})$, the PWA function $h(\boldsymbol{x})$, and $\tilde{\varphi}(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ be defined as in (1), (9), and (11), respectively. Consider

$$d^{(p,r)} = \begin{cases} \frac{\tilde{c}^{(p)} - \tilde{c}^{(r)}}{b^{(p,r)}} & \text{if } b^{(p,r)} \neq 0, \\ \frac{\tilde{l}_t^{(p)} - \tilde{l}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} & \text{if } b^{(p,r)} = 0 \end{cases} \tag{16}$$

and

$$\boldsymbol{k}_q^{(p,r)} = \frac{1}{\boldsymbol{h}_t^{(p,r)}} \left(2\Delta \boldsymbol{Q}_{t,q}^{(p,r)} - \frac{\boldsymbol{h}_q^{(p,r)}}{\boldsymbol{h}_t^{(p,r)}} \Delta \boldsymbol{Q}_{t,t}^{(p,r)} \right) \tag{17}$$

for all $q \in \{1, \dots, n\}$ with $\Delta \boldsymbol{Q}^{(p,r)} := \boldsymbol{Q}^{(p)} - \boldsymbol{Q}^{(r)}$, $\tilde{\boldsymbol{l}}^{(p)} := \boldsymbol{l}^{(p)} + \boldsymbol{\beta}^{(p)}$, and $\tilde{c}^{(p)} := c^{(p)} + \gamma^{(p)}$ where $t \in \{1, \dots, n\}$ is such that $\boldsymbol{h}_t^{(p,r)} \neq 0$. Further, assume that

$$\begin{cases} \frac{\gamma^{(p)} - \gamma^{(r)}}{b^{(p,r)}} \leq \frac{c^{(r)} - c^{(p)}}{b^{(p,r)}} - \boldsymbol{k}^{(p,r)} \boldsymbol{v}_l{}^\top & \text{if } b^{(p,r)} \neq 0, \\ \frac{\boldsymbol{\beta}_t^{(p)} - \boldsymbol{\beta}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} \leq \frac{\boldsymbol{l}_t^{(r)} - \boldsymbol{l}_t^{(p)}}{\boldsymbol{h}_t^{(p,r)}} - \boldsymbol{k}^{(p,r)} \boldsymbol{v}_l{}^\top & \text{if } b^{(i,j)} = 0 \end{cases} \tag{18}$$

holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \dots, V\}$, where $\boldsymbol{v}_l \in \mathbb{R}^{1 \times n}$ are the $V \in \mathbb{N}$ vertices of the polyhedral domain Ω. Then,

$$\begin{aligned} \Delta \tilde{\varphi}^{(p,r)}(\boldsymbol{x}) &:= \tilde{\varphi}^{(p)}(\boldsymbol{x}) - \tilde{\varphi}^{(r)}(\boldsymbol{x}) \\ &= \left(\boldsymbol{x}^\top \boldsymbol{h}^{(p,r)^\top} + b^{(p,r)} \right) \left(\boldsymbol{k}^{(p,r)} \boldsymbol{x} + d^{(p,r)} \right) \end{aligned} \tag{19}$$

holds for all $\boldsymbol{x} \in \mathbb{R}^n$ and

$$\boldsymbol{k}^{(p,r)} \boldsymbol{x} + d^{(p,r)} \leq 0 \tag{20}$$

holds for all $\boldsymbol{x} \in \Omega$ and for all $(p, r) \in \mathcal{N}$.

Proof: According to [25, Thm. 1] for

$$\Delta \tilde{\varphi}^{(p,r)}(\boldsymbol{x}) = \boldsymbol{x}^\top \Delta \boldsymbol{Q}^{(p,r)} \boldsymbol{x} + \left(\tilde{\boldsymbol{l}}^{(p)} - \tilde{\boldsymbol{l}}^{(r)} \right) \boldsymbol{x} + \tilde{c}^{(p)} - \tilde{c}^{(r)} \tag{21}$$

with $(p, r) \in \mathcal{N}$ there exist a $\boldsymbol{k}^{(p,r)} \in \mathbb{R}^{1 \times n}$ and a $d^{(p,r)} \in \mathbb{R}$ with

$$\begin{aligned} &\boldsymbol{x}^\top \Delta \boldsymbol{Q}^{(p,r)} \boldsymbol{x} + \left(\tilde{\boldsymbol{l}}^{(p)} - \tilde{\boldsymbol{l}}^{(r)} \right) \boldsymbol{x} + \tilde{c}^{(p)} - \tilde{c}^{(r)} \\ &= \left(\boldsymbol{x}^\top \boldsymbol{h}^{(p,r)^\top} + b^{(p,r)} \right) \left(\boldsymbol{k}^{(p,r)} \boldsymbol{x} + d^{(p,r)} \right) \\ &= \boldsymbol{x}^\top \frac{1}{2} \left(\boldsymbol{h}^{(p,r)^\top} \boldsymbol{k}^{(p,r)} + \boldsymbol{k}^{(p,r)^\top} \boldsymbol{h}^{(p,r)} \right) \boldsymbol{x} \\ &\quad + \left(\boldsymbol{h}^{(p,r)} d^{(p,r)} + \boldsymbol{k}^{(p,r)} b^{(p,r)} \right) \boldsymbol{x} + b^{(p,r)} d^{(p,r)} \end{aligned} \tag{22}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$, where $\boldsymbol{h}^{(p,r)}$ and $b^{(p,r)}$ describe the separating hyperplane $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)} = 0\}$ of the neighbouring regions $\mathcal{R}^{(p)}$ and $\mathcal{R}^{(r)}$ according to (3). Now, by comparing the coefficients of the right-hand and left-hand side of the quadratic functions in (22), we can identify

$$\frac{1}{2} \left(\boldsymbol{h}^{(p,r)^\top} \boldsymbol{k}^{(p,r)} + \boldsymbol{k}^{(p,r)^\top} \boldsymbol{h}^{(p,r)} \right) = \Delta \boldsymbol{Q}^{(p,r)} \tag{23a}$$

$$\boldsymbol{h}^{(p,r)} d^{(p,r)} + \boldsymbol{k}^{(p,r)} b^{(p,r)} = \tilde{\boldsymbol{l}}^{(p)} - \tilde{\boldsymbol{l}}^{(r)} \tag{23b}$$

$$b^{(p,r)} d^{(p,r)} = \tilde{c}^{(p)} - \tilde{c}^{(r)}. \tag{23c}$$

With (23b) and (23c) we can write $d^{(p,r)}$ as (16). Note that for (16) there always exits at least one $t \in \{1, \dots, n\}$ with $\boldsymbol{h}_t^{(p,r)} \neq 0$. Because if we assume $\boldsymbol{h}_t^{(p,r)} = 0$ for all $t \in \{1, \dots, n\}$ then the corresponding hyperplane in (3) either leads to infeasibility for $b^{(p,r)} > 0$ and thus would be invalid or for $b^{(p,r)} \leq 0$ the hyperplane is redundant and thus can be removed. We now use $\tilde{\boldsymbol{l}}^{(p)} := \boldsymbol{l}^{(p)} + \boldsymbol{\beta}^{(p)}$, and $\tilde{c}^{(p)} := c^{(p)} + \gamma^{(p)}$ to reformulate (18) as

$$\begin{cases} \frac{\tilde{c}^{(p)} - \tilde{c}^{(r)}}{b^{(p,r)}} + \boldsymbol{k}^{(p,r)} \boldsymbol{v}_l{}^\top \leq 0 & \text{if } b^{(p,r)} \neq 0, \\ \frac{\tilde{\boldsymbol{l}}_t^{(p)} - \tilde{\boldsymbol{l}}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} + \boldsymbol{k}^{(p,r)} \boldsymbol{v}_l{}^\top \leq 0 & \text{if } b^{(i,j)} = 0 \end{cases}$$

and subsequently substitute the reformulation of $d^{(p,r)}$ by (16), which leads to

$$\boldsymbol{k}^{(p,r)} \boldsymbol{v}_l{}^\top + d^{(p,r)} \leq 0 \tag{24}$$

for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. Moreover, the elements of the vector $\boldsymbol{k}^{(p,r)}$ in (18) can be expressed in terms of the coefficients of the functions (1) and (9). By using (23a) we find $\boldsymbol{k}_t^{(p,r)} = \frac{\Delta \boldsymbol{Q}_{t,t}^{(p,r)}}{\boldsymbol{h}_t^{(p,r)}}$ and $\boldsymbol{h}_t^{(p,r)} \boldsymbol{k}_q^{(p,r)} + \boldsymbol{h}_q^{(p,r)} \boldsymbol{k}_t^{(p,r)} = 2\Delta \boldsymbol{Q}_{t,q}^{(p,r)}$ which results in (17). Now, since (24) holds for all vertices $\boldsymbol{v}_l$ of the polyhedron Ω we have (20) for all $\boldsymbol{x} \in \Omega$ and for all $(p, r) \in \mathcal{N}$, which completes the proof. ■

Now we can proceed and use Lemma 1 to prove that (14) holds. This requires to show

$$\Delta \tilde{\varphi}^{(i,j)}(\boldsymbol{x}) := \tilde{\varphi}^{(i)}(\boldsymbol{x}) - \tilde{\varphi}^{(j)}(\boldsymbol{x}) \geq 0 \tag{25}$$

for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$, for all $j \in \{1, \dots, s\} \setminus \{i\}$, and for all $i \in \{1, \dots, s\}$, which involves the verification of a quadratic inequality. However, if $\mathcal{R}^{(i)}$ and $\mathcal{R}^{(j)}$ are neighbouring regions, i.e., $(i, j) = (p, r) \in \mathcal{N}$ we can use (19) from Lemma 1 to decompose the quadratic function $\Delta \tilde{\varphi}^{(p,r)}(\boldsymbol{x})$ into two affine factors. Then, to prove (25), we have to show that both affine factors are negative, which is more convenient than working with the quadratic inequality. However, in general in the domain partition there are regions $\mathcal{R}^{(i)}$ and $\mathcal{R}^{(j)}$ that are not neighbouring, as for example $\mathcal{R}^{(2)}$ and $\mathcal{R}^{(4)}$ in Fig. 2. Thus, we cannot directly use Lemma 1 to prove (25) for all regions. Fortunately, it is possible to rewrite $\Delta \tilde{\varphi}^{(i,j)}(\boldsymbol{x})$ in terms of differences of neighbouring regions by

$$\Delta \tilde{\varphi}^{(i,j)}(\boldsymbol{x}) = \sum_{k=1}^{K} \Delta \tilde{\varphi}^{(p_k, r_k)}(\boldsymbol{x}) \tag{26}$$

with $(p_k, r_k) \in \mathcal{N}$ for all $k \in \{1, \dots, K\}$. This allows us to use (19) from Lemma 1 for every summand in (26). We define

$$\mathcal{C}^{(i,j)} := \{(p_1, r_1), \dots, (p_K, r_K)\}$$

as the set containing the K index pairs with $(p, r) \in \mathcal{N}$ of the involved neighbouring regions and by $(\mathcal{C}^{(i,j)})_k := (p_k, r_k)$ a specific element of this set. For (26) to be a valid reformulation of $\Delta \tilde{\varphi}^{(i,j)}(\boldsymbol{x})$, we need

$$p_K = i, \quad r_1 = j \quad \text{and} \quad p_k = r_{k+1} \tag{27}$$

for all $k \in \{1, \dots, K-1\}$. As illustrated in Fig. 2(a) and (b), there are several possible ways for choosing a valid $\mathcal{C}^{(i,j)}$ that satisfies (27). The paths from Fig. 2(a) and (b) can both be used to determine a valid $\mathcal{C}^{(4,2)}$ for rewriting $\Delta \tilde{\varphi}^{(4,2)}(\boldsymbol{x})$ by collecting the index pairs (p, r) of all hyperplanes crossed by the path. For the path from Fig. 2(a), this results in $\mathcal{C}^{(4,2)} =$

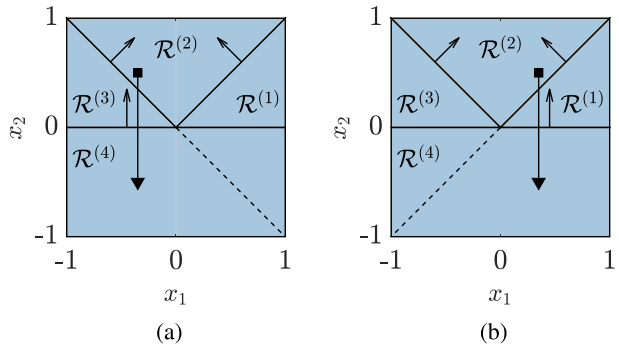

FIGURE 2. Illustration of an exemplary domain partition with paths from $\mathcal{R}^{(2)}$ to $\mathcal{R}^{(4)}$. Each path suggests a way of expressing the difference $\Delta\tilde{\varphi}^{(4,2)}(x) = \tilde{\varphi}^{(4)}(x) - \tilde{\varphi}^{(2)}(x)$ in terms of differences of neighbouring regions by (26). For the path in (a) we have $\mathcal{C}^{(4,2)} = \{(3,2),(4,3)\}$. For the path in (b) we have $\mathcal{C}^{(4,2)} = \{(1,2),(4,1)\}$. The arrows on the boundaries between the regions in (a) and (b) represent the normal vectors $h^{(1,4)}$, $h^{(2,1)}$, $h^{(2,3)}$, and $h^{(3,4)}$ of the hyperplanes crossed by the paths.

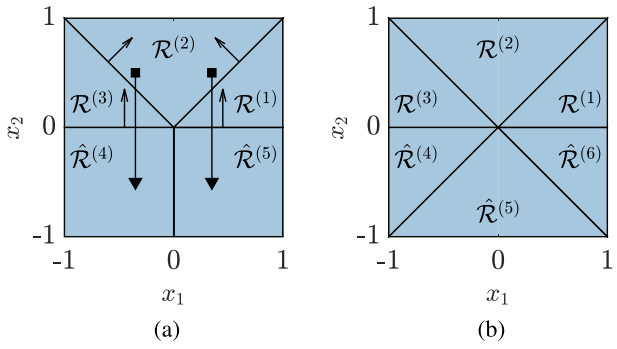

FIGURE 3. Illustration of two possible refinements of the partition shown in Fig. 2. The partitions in (a) and (b) are both such that Assumption 1 holds. The arrows on the boundaries between the regions in (a) represent the normal vectors of the hyperplanes crossed by the paths.

$\{(3,2),(4,3)\}$ and for the path from Fig. 2(b) in $\mathcal{C}^{(4,2)} = \{(1,2),(4,1)\}$. Thus, depending on the partition, the set $\mathcal{C}^{(i,j)}$ may not be unique. However, this does not pose a problem for the following results since, for our purpose, it is sufficient to have a partition for which the following assumption holds.

Assumption 1: For every $i \in \{1,\dots,s\}$ and for every $j \in \{1,\dots,s\} \setminus i$ there exists a $\mathcal{C}^{(i,j)}$ for which (26) and

$$\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0 \tag{28}$$

hold for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p,r) \in \mathcal{C}^{(i,j)}$.

Note that for a given PWQ function $\varphi(\boldsymbol{x})$, Assumption 1 may or may not hold. In Fig. 2 we have a partition which is such that for $i = 4$ and $j = 2$, there does not exists a $\mathcal{C}^{(4,2)}$ for which (26) and (28) holds for all $\boldsymbol{x} \in \mathcal{R}^{(4)}$ and for all $(p,r) \in \mathcal{C}^{(4,2)}$. As previously described, the paths in Fig. 2(a) and (b) illustrate the two possible ways for determining $\mathcal{C}^{(4,2)}$. For Fig. 2(a) and (b) we have $\mathcal{C}^{(4,2)} = \{(3,2),(4,3)\}$ and $\mathcal{C}^{(4,2)} = \{(1,2),(4,1)\}$, respectively. For the first case with $\mathcal{C}^{(4,2)} = \{(3,2),(4,3)\}$ we have $\boldsymbol{h}^{(3,2)}\boldsymbol{x} + b^{(3,2)} > 0$ for $x \in \mathcal{R}^{(4)}$ and above the dashed line in Fig. 2(a). Analogously, we have $\boldsymbol{h}^{(1,2)}\boldsymbol{x} + b^{(1,2)} > 0$ for $x \in \mathcal{R}^{(4)}$ and above the dashed line in Fig. 2(b) for the second case with $\mathcal{C}^{(4,2)} = \{(1,2),(4,1)\}$. For both variants, there is a $(p,r) \in \mathcal{C}^{(4,2)}$ for which (28) does not hold for all $\boldsymbol{x} \in \mathcal{R}^{(4)}$. Thus, Assumption 1 does not hold for the partition shown in Fig. 2.

However, if we add the hyperplane $\{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{x}_1 = 0\}$ in $\mathcal{R}^{(4)}$ as illustrated in Fig. 3(a) and create the new regions $\hat{\mathcal{R}}^{(4)}$ and $\hat{\mathcal{R}}^{(5)}$ with $\hat{\varphi}^{(4)}(\boldsymbol{x}) = \hat{\varphi}^{(5)}(\boldsymbol{x}) = \varphi^{(4)}(\boldsymbol{x})$, then the two paths starting in $\mathcal{R}^{(2)}$ from Fig. 2 now lead to the sets $\mathcal{C}^{(4,2)} = \{(3,2),(4,3)\}$ and $\mathcal{C}^{(5,2)} = \{(1,2),(5,1)\}$. As illustrated by the normal vectors and paths in Fig. 3(a), for both sets we have (26) and $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p,r) \in \mathcal{C}^{(i,j)}$ with $(i,j) = (4,2)$ and $(i,j) = (5,2)$, respectively. Moreover, the partition is now such that for all the remaining index pairs (i,j) with $i \in \{1,\dots,s\}$ and $j \in \{1,\dots,s\} \setminus i$ we are able to find a set $\mathcal{C}^{(i,j)}$ such that (26) and (28) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p,r) \in \mathcal{C}^{(i,j)}$. Thus, Assumption 1 holds for the modified partition.

In Fig. 3(b), an alternative refinement is shown, where all hyperplanes in the partition are extended to the whole domain. This refinement also leads to a partition for which Assumption 1 holds.

As shown in Figs. 2 and 3, if we have a PWQ function $\varphi(\boldsymbol{x})$ for which Assumption 1 does not hold, we can always modify the function $\varphi(\boldsymbol{x})$ by adding hyperplanes to the domain partition. The added hyperplanes will lead to a function with $\hat{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$, whose partition may satisfies Assumption 1. Adding of hyperplanes can be done as in Fig. 3(a), i.e., by carefully adding hyperplanes such that Assumption 1 holds for a specific partition. Fortunately, as we will show in the following proposition, the approach illustrated in Fig. 3(b), i.e., the extension of all hyperplanes to the whole domain, works without manually choosing hypeprlanes and will always lead to a modified partition for which Assumption 1 holds. For ease of reading, the proof is provided in the appendix.

Proposition 2: Assume that for every $(i,j) \in \{1,\dots,s\}^2$ and for every $k \in \{1,\dots,d^{(j)}\}$ either $\boldsymbol{h}_k^{(j)}\boldsymbol{x} + \boldsymbol{b}_k^{(j)} \leq 0$ or $\boldsymbol{h}_k^{(j)}\boldsymbol{x} + \boldsymbol{b}_k^{(j)} \geq 0$ holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. Then, Assumption 1 holds.

Intuitively, Proposition 2 states that Assumption 1 holds if all hyperplanes of the partition are extended to the whole domain, as shown in Fig. 3(b). This extension of all hyperlanes typically leads to a partition with a large number of regions s. However, according to Proposition 2 the extension of all hyperplanes is only sufficient for Assumption 1 to hold. As illustrated in Fig. 3(a), we may be able to find a refinement of the partition with fewer regions for which Assumption 1 holds. This will lead to a representation of the form (10) with a smaller s.

We will now use Lemma 1 to show that the factors (20) and (28) of each summand (19) in (26) are negative. This will help us to show that (25) holds, which is sufficient for (14).

Lemma 3: Let the PWQ function $\varphi(\boldsymbol{x})$, the PWA function $h(\boldsymbol{x})$, and $\tilde{\varphi}(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ be defined as in (1), (9), and

(11), respectively. Assume that (18) as in Lemma 1 holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. Then, (14) holds for all $\boldsymbol{x} \in \Omega$.

Proof: To ensure (14) we need

$$\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x}) \geq 0 \tag{29}$$

for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$, for all $j \in \{1, \dots, s\} \setminus \{i\}$, and for all $i \in \{1, \dots, s\}$. We can rewrite the left-hand side of (29) as the sum over differences of neighbouring regions using (26). According to Lemma 1 for the summands in (26) we have

$$\Delta\tilde{\varphi}^{(p,r)}(\boldsymbol{x}) = (\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)})(\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)}) \tag{30}$$

with (20) for all $\boldsymbol{x} \in \Omega$ and for all $(p, r) \in \mathcal{C}^{(i,j)} \subseteq \mathcal{N}$. Moreover, with $\mathcal{R}^{(i)} \subset \Omega$ for all $i \in \{1, \dots, s\}$, we can infer that (20) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(i,j)}$. We further have (28) for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(i,j)}$ by Assumption 1. Consequently, for all $j \in \{1, \dots, s\} \setminus \{i\}$ and for all $i \in \{1, \dots, s\}$ the inequality

$$\underbrace{(\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)})}_{\leq 0 \forall \boldsymbol{x} \in \mathcal{R}^{(i)}} \underbrace{(\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)})}_{\leq 0 \forall \boldsymbol{x} \in \mathcal{R}^{(i)}} \geq 0 \tag{31}$$

holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(i,j)}$. Finally, since for all $i \in \{1, \dots, s\}$ and for all $j \in \{1, \dots, s\} \setminus i$ there exists a $\mathcal{C}^{(i,j)}$ for which (26) holds, we have

$$\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x}) = \sum_{k=1}^{K} \Delta\tilde{\varphi}^{(p_k,r_k)}(\boldsymbol{x})$$

$$= \sum_{k=1}^{K} (\boldsymbol{h}^{(p_k,r_k)}\boldsymbol{x} + b^{(p_k,r_k)})(\boldsymbol{k}^{(p_k,r_k)}\boldsymbol{x} + d^{(p_k,r_k)}) \geq 0$$

for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$, for all $i \in \{1, \dots, s\}$, and for all $j \in \{1, \dots, s\} \setminus \{i\}$. In summary, we have $\tilde{\varphi}^{(i)}(\boldsymbol{x}) \geq \tilde{\varphi}^{(j)}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$, for all $j \in \{1, \dots, s\} \setminus \{i\}$, and for all $i \in \{1, \dots, s\}$ and thus $\tilde{\varphi}^{(i)}(\boldsymbol{x}) \leq \max_{1 \leq j \leq s}\{\tilde{\varphi}^{(j)}(\boldsymbol{x})\} \leq \tilde{\varphi}^{(i)}(\boldsymbol{x})$ if $\boldsymbol{x} \in \mathcal{R}^{(i)}$ or equivalently

$$\max_{1 \leq j \leq s}\{\tilde{\varphi}^{(j)}(\boldsymbol{x})\} = \tilde{\varphi}^{(i)}(\boldsymbol{x}) \text{ if } \boldsymbol{x} \in \mathcal{R}^{(i)}$$

for all $i \in \{1, \dots, s\}$. With $\tilde{\varphi}^{(i)}(\boldsymbol{x}) = \varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ for all $i \in \{1, \dots, s\}$ and the definitions (1), (9), and (11) we now have

$$\max_{1 \leq i \leq s}\{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}) + h(\boldsymbol{x})$$

for all $\boldsymbol{x} \in \Omega$, which proves the claim. ■

Remark 1: The condition (18), which is sufficient for a representation in the form (14), can be seen as a generalization of the known results for representing PWA functions as the maximum of their local affine functions. For continuous PWA functions, i.e. $\varphi(\boldsymbol{x})$ as in (1) with $\boldsymbol{Q}^{(i)} = \boldsymbol{0}$ for all $i \in \{1, \dots, s\}$, a representation in the form

$$\varphi(\boldsymbol{x}) = \max_{1 \leq i \leq s}\{\varphi^{(i)}(\boldsymbol{x})\} \tag{32}$$

is possible if $\varphi(\boldsymbol{x})$ is convex [32]. For this special case, Lemma 3 states that (32) holds for all $\boldsymbol{x} \in \Omega$ if

$$\begin{cases} \frac{c^{(p)} - c^{(r)}}{b^{(p,r)}} \leq 0 & \text{if } b^{(p,r)} \neq 0, \\ \frac{\boldsymbol{l}_t^{(p)} - \boldsymbol{l}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} \leq 0 & \text{if } b^{(i,j)} = 0 \end{cases}$$

holds for all $(p, r) \in \mathcal{N}$. With (16) the former is equivalent to

$$d^{(p,r)} \leq 0 \tag{33}$$

for all $(p, r) \in \mathcal{N}$. Using [25, Thm. 1] we find $\varphi^{(p)}(\boldsymbol{x}) = \varphi^{(r)}(\boldsymbol{x}) + d^{(p,r)}(\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)})$ and thus (33) holds if and only if $\varphi^{(p)}(\boldsymbol{x}) = \varphi^{(r)}(\boldsymbol{x}) + d^{(p,r)}(\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)}) \geq \varphi^{(r)}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{R}^{(p)}$ and for all $(p, r) \in \mathcal{N}$, which is the same as assuming that $\varphi(\boldsymbol{x})$ is convex. Thus, the PWA case is included as a special case in Lemma 3 and leads to the result known from the literature [32].

We will now show how to find a PWA function $h(\boldsymbol{x})$ such that (18) holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \dots, V\}$. In the end, this will allow us to show that PWQ functions with polyhedral domain partition can be represented as the difference of two max-expressions by (10).

B. FEASIBILITY OF THE CONDITIONS

In this section, we will show that it is always possible to find a continuous and convex PWA function of the form (9) such that the function (11) can be represented as the maximum of its local functions by (14). According to Lemma 3, a sufficient condition for such a representation is (18). Therefore, we will show in the following that it is always possible to construct a convex lifting [41] via $h(\boldsymbol{x})$ such that (18) and consequently (14) holds if the partition of the PWQ function $\varphi(\boldsymbol{x})$ is convexly lifatable. Where we denote a polyhedral partition as convexly liftable according to [41, Thm. III.9], i.e, if and only if the OP [41, Alg. 3] for computing a convex lifting is feasible. If we now take into account that a convex PWA $h(\boldsymbol{x})$ can always be represented as the maximum of its local affine functions, i.e, (15) holds, then, by combining (14) and (15), we can show that every PWQ function can be represented as the difference of two max-expressions by (10). To prove the next results, we assume from now on that the following assumption holds.

Assumption 2: The polyhedral domain partition $\cup_{i=1}^{s} \mathcal{R}^{(i)} = \Omega$ of $\varphi(\boldsymbol{x})$ as in (1) is convexly liftable.

Remark 2: The domain partition of a PWQ function $\varphi(\boldsymbol{x})$ may or may not be convexly liftable. However, if the domain partition is not convexly liftable, we can construct a modified PWQ function $\hat{\varphi}(\boldsymbol{x}) : \Omega \to \mathbb{R}$ with $\hat{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega \subset \mathbb{R}^n$ whose partition is convexly liftable by running the Algorithm [41, Alg. 4]. In the algorithm, all hyperplanes between neighbouring regions are extended to the whole domain. Wherever the extension of the hyperplanes lead to a splitting of a region $\mathcal{R}^{(i)}$ into two new regions $\hat{\mathcal{R}}^{(p)}$ and $\hat{\mathcal{R}}^{(r)}$ with $\hat{\mathcal{R}}^{(p)} \cup \hat{\mathcal{R}}^{(r)} = \mathcal{R}^{(i)}$, we choose the local functions as $\hat{\varphi}^{(p)}(\boldsymbol{x}) = \hat{\varphi}^{(r)}(\boldsymbol{x}) = \varphi^{(i)}(\boldsymbol{x})$. This procedure results in a PWQ

function $\hat{\varphi}(\boldsymbol{x})$ with $\hat{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ whose partition is convexly liftable [41, Thm. III.12], i.e., Assumption 2 holds. Moreover, the modified partition is such that for every $(i, j) \in \{1, \dots, s\}^2$ and for every $k \in \{1, \dots, d^{(j)}\}$ we either have $\boldsymbol{h}_k^{(j)}\boldsymbol{x} + \boldsymbol{b}_k^{(j)} \leq 0$ or $\boldsymbol{h}_k^{(j)}\boldsymbol{x} + \boldsymbol{b}_k^{(j)} \geq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ [41, Alg. 4]. Thus, Proposition 2 applies and consequently Assumption 1 holds. In summary, this means that for a given PWQ function $\varphi(\boldsymbol{x})$, we can always find a modified PWQ function $\hat{\varphi}(\boldsymbol{x})$ with $\hat{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ for which Assumptions 1 and 2 hold. Thus, we can pose both assumptions without loss of generality.

In the following, we show that the condition (18), which is sufficient for (14), can always be satisfied by using a continuous and convex PWA function $h(\boldsymbol{x})$.

Lemma 4: Let the PWQ function $\varphi(\boldsymbol{x})$ and $\tilde{\varphi}(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ be defined as in (1) and (11), respectively. Then, there exists a continuous and convex PWA function $h(\boldsymbol{x})$ as in (9) such that (18) holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$.

Proof: For a convexly liftable partition, [41, Thm. III.9] states that for all $\delta > 0$, there exists a continuous and convex PWA function of the form (9) with

$$\boldsymbol{\beta}^{(p)}\boldsymbol{x}^{(p)} + \gamma^{(p)} \geq \boldsymbol{\beta}^{(r)}\boldsymbol{x}^{(p)} + \gamma^{(r)} + \delta \tag{34}$$

for $\boldsymbol{x}^{(p)} \in \operatorname{int}(\mathcal{R}^{(p)})$ and for all $(p, r) \in \mathcal{N}$, where $\boldsymbol{x}^{(p)}$ is typically chosen as the Chebyshev center of $\mathcal{R}^{(p)}$. Furthermore, since the PWA function is continuous, there exists a $\rho^{(p,r)} \in \mathbb{R}$ with

$$(\boldsymbol{\beta}^{(p)} - \boldsymbol{\beta}^{(r)})\boldsymbol{x} + \gamma^{(p)} - \gamma^{(r)} = \rho^{(p,r)}(\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)}) \tag{35}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and for all $(p, r) \in \mathcal{N}$ according to [25, Thm. 1]. By comparing the coefficients of the right-hand and left-hand side of the affine functions in (35), we find

$$\boldsymbol{\beta}^{(p)} - \boldsymbol{\beta}^{(r)} = \rho^{(p,r)}\boldsymbol{h}^{(p,r)} \quad \text{and} \tag{36a}$$

$$\gamma^{(p)} - \gamma^{(r)} = \rho^{(p,r)}b^{(p,r)}. \tag{36b}$$

Thus, we can identify

$$\rho^{(p,r)} = \begin{cases} \frac{\gamma^{(p)} - \gamma^{(r)}}{b^{(p,r)}} & \text{if } b^{(p,r)} \neq 0, \\ \frac{\boldsymbol{\beta}_t^{(p)} - \boldsymbol{\beta}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} & \text{if } b^{(p,r)} = 0. \end{cases} \tag{37}$$

Substituting $\boldsymbol{\beta}^{(p)} = \boldsymbol{\beta}^{(r)} + \rho^{(p,r)}\boldsymbol{h}^{(p,r)}$ and $\gamma^{(p)} = \gamma^{(r)} + \rho^{(p,r)}b^{(p,r)}$ from (36a) and (36b), respectively in the left-hand side of (34) gives $\boldsymbol{\beta}^{(p)}\boldsymbol{x}^{(p)} + \gamma^{(p)} = (\boldsymbol{\beta}^{(r)} + \rho^{(p,r)}\boldsymbol{h}^{(p,r)})\boldsymbol{x}^{(p)} + \gamma^{(r)} + \rho^{(p,r)}b^{(p,r)}$ which results in

$$\begin{aligned}&(\boldsymbol{\beta}^{(r)} + \rho^{(p,r)}\boldsymbol{h}^{(p,r)})\boldsymbol{x}^{(p)} + \gamma^{(r)} + \rho^{(p,r)}b^{(p,r)} \\ &\geq \boldsymbol{\beta}^{(r)}\boldsymbol{x}^{(p)} + \gamma^{(r)} + \delta\end{aligned}$$

and thus

$$\begin{aligned}&\rho^{(p,r)} \underbrace{(\boldsymbol{h}^{(p,r)}\boldsymbol{x}^{(p)} + b^{(p,r)})}_{<0 \text{ for } \boldsymbol{x}^{(p)} \in \operatorname{int}(\mathcal{R}^{(p)})} \geq \delta > 0 \\ \Leftrightarrow\ &\rho^{(p,r)} \leq \frac{\delta}{\boldsymbol{h}^{(p,r)}\boldsymbol{x}^{(p)} + b^{(p,r)}} < 0\end{aligned} \tag{38}$$

is feasible for all $\delta > 0$. With (37) and (38) we can use

$$\begin{cases} \frac{\gamma^{(p)} - \gamma^{(r)}}{b^{(p,r)}} \leq \frac{\delta}{\boldsymbol{h}^{(p,r)}\boldsymbol{x}^{(p)} + b^{(p,r)}} < 0 & \text{if } b^{(p,r)} \neq 0, \\ \frac{\boldsymbol{\beta}_t^{(p)} - \boldsymbol{\beta}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} \leq \frac{\delta}{\boldsymbol{h}^{(p,r)}\boldsymbol{x}^{(p)} + b^{(p,r)}} < 0 & \text{if } b^{(p,r)} = 0. \end{cases} \tag{39}$$

to overestimate the left-hand side of (18). If we now take into account that the inequality (39) is feasible for all $\delta > 0$ and that the right-hand side in (18) is finite then we can always choose a $\delta > 0$ such that

$$\begin{cases} \frac{\delta}{\boldsymbol{h}^{(p,r)}\boldsymbol{x}^{(p)} + b^{(p,r)}} \leq \frac{c^{(r)} - c^{(p)}}{b^{(p,r)}} - \boldsymbol{k}^{(p,r)}\boldsymbol{v}_l{}^\top & \text{if } b^{(p,r)} \neq 0, \\ \frac{\delta}{\boldsymbol{h}^{(p,r)}\boldsymbol{x}^{(p)} + b^{(p,r)}} \leq \frac{l_t^{(r)} - l_t^{(p)}}{\boldsymbol{h}_t^{(p,r)}} - \boldsymbol{k}^{(p,r)}\boldsymbol{v}_l{}^\top & \text{if } b^{(p,r)} = 0 \end{cases} \tag{40}$$

holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \dots, V\}$. In combination with (39) this means that there exists a continuous and convex PWA function $h(\boldsymbol{x})$ such that (18) holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. ■

Remark 3: Lemma 4 states that it is always possible to compute a convex lifting via a PWA function $h(\boldsymbol{x})$ such that (18) holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. Such a lifting can be computed explicitly by solving the OP from [41, Alg. 3] with a $c := \delta$ for which $\delta > 0$ and (40) holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. This OP is always feasible for a convexly liftable partition.

With Lemma 3 and 4, we have shown that for every continuous PWQ function with a polyhedral domain partition, it is possible to compute a convex and continuous PWA function such that (14) holds for all $\boldsymbol{x} \in \Omega$. In the following section, we will use this result to prove that PWQ functions can be represented as the difference of two max-expressions.

IV. REPRESENTATION AND APPROXIMATIONS OF PWQ FUNCTIONS

In Section III, we derived the theoretical basis for the main result of this paper. Namely, the existence of a representation as the difference of two max-expressions by (10) for arbitrary continuous PWQ functions $\varphi(\boldsymbol{x})$ with polyhedral partition. The two results we will use to prove the existence are Lemmas 3 and 4. By combining the lemmas, we can show that there always exists a convex PWA function $h(\boldsymbol{x})$ such that $\varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ can be represented as the maximum of its local functions by (14). Moreover, since $h(\boldsymbol{x})$ is a convex PWA function, it can be represented as the maximum of its local affine functions by (15). Then, by combining (14) and (15), we can eventually prove the existence of (10).

Theorem 5: For every continuous PWQ function $\varphi(\boldsymbol{x})$ of the form (1) with a polyhedral domain partition as in (2), there exists a PWA function $h(\boldsymbol{x})$ as in (9) such that

$$\varphi(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\} - \max_{1 \leq i \leq s} \{h^{(i)}(\boldsymbol{x})\} \tag{41}$$

holds for all $\boldsymbol{x} \in \Omega$.

Proof: Lemma 4 states that for every PWQ function $\varphi(\boldsymbol{x})$ with a convexly liftable partition, there exists a continuous and convex PWA function $h(\boldsymbol{x})$ for which (18) holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. Then, we can deduce

from Lemma 3 that (14) holds for all $\boldsymbol{x} \in \Omega$. Moreover, for continuous and convex PWA functions $h(\boldsymbol{x})$, we have (15) for all $\boldsymbol{x} \in \Omega$ according to [32]. In summary,

$$\begin{aligned}&\max_{1\leq i\leq s}\{\varphi^{(i)}(\boldsymbol{x})+h^{(i)}(\boldsymbol{x})\}-\max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\}\\&=\varphi(\boldsymbol{x})+h(\boldsymbol{x})-h(\boldsymbol{x})=\varphi(\boldsymbol{x})\end{aligned}$$

holds for all $\boldsymbol{x} \in \Omega$. ■

The representation of PWQ functions by (41) will serve as a starting point for deriving NNs that allow an exact representation of arbitrary continuous PWQ functions with a polyhedral domain partition.

A. REPRESENTATION BY NEURAL NETWORKS

In this section, we derive different topologies for maxout and relu NNs that allow an exact representation of PWQ functions. We start with the derivation of maxout NNs by using Theorem 5. Then, based on the derived maxout NN, we use methods for decomposing max-expressions to derive suitable relu NNs.

1) MAXOUT NEURAL NETWORKS

Theorem 5 can directly be used to derive maxout NNs that allow an exact representation of PWQ functions by exploiting the structure of (41). For deriving such NNs, we use an extended input vector $\boldsymbol{\xi} \in \mathbb{R}^{\frac{n^2+3n}{2}}$ that includes all $\frac{n^2+n}{2}$ quadratic monomials of $\boldsymbol{x}$, i.e.,

$$\boldsymbol{\xi}(\boldsymbol{x}) := \begin{pmatrix}\boldsymbol{x}^\top & \boldsymbol{x}_1^2 & \boldsymbol{x}_1\boldsymbol{x}_2 & \dots & \boldsymbol{x}_1\boldsymbol{x}_n & \dots & \boldsymbol{x}_n^2\end{pmatrix}^\top. \tag{42}$$

Theorem 6: Every continuous PWQ function of the form (1) with a polyhedral domain partition as in (2) can be represented by a maxout NN with $l = 1$ hidden layer, $w_1 = 2$ neurons with $p_1 = s$, input vector $\boldsymbol{\xi}$ as in (42), postactivation weights $\boldsymbol{W}_1^{(2)} = 1$, $\boldsymbol{W}_2^{(2)} = -1$, and postactivtion bias $v^{(2)} = 0$.

Proof: According to (5), (6), and (8), the proposed maxout NN leads to

$$\begin{aligned}\Phi(\boldsymbol{\xi}) = \boldsymbol{W}_1^{(2)} &\max_{1\leq i\leq p_1}\{\boldsymbol{W}_i^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_i^{(1)}\}\\&+\boldsymbol{W}_2^{(2)}\max_{1+p_1\leq i\leq 2p_1}\{\boldsymbol{W}_i^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_i^{(1)}\}+v^{(2)}.\end{aligned}$$

If we use $p_1 = s, \boldsymbol{W}_1^{(2)} = 1, \boldsymbol{W}_2^{(2)} = -1$, and $v^{(2)} = 0$, the NN simplifies to

$$\Phi(\boldsymbol{\xi}) = \max_{1\leq i\leq s}\{\boldsymbol{W}_i^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_i^{(1)}\}-\max_{1\leq i\leq s}\{\boldsymbol{W}_{i+s}^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_{i+s}^{(1)}\}.$$

For all $i \in \{1, \dots, s\}$ the quadratic local functions $\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ and the linear local functions $h^{(i)}(\boldsymbol{x})$ in (41) are linear combinations of 1 and the monomials contained in $\boldsymbol{\xi}$ from (42). Thus, there exist weights $\boldsymbol{W}^{(1)}$ and biases $\boldsymbol{v}^{(1)}$ such that $\boldsymbol{W}_i^{(1)}\boldsymbol{\xi} + 1\boldsymbol{v}_i^{(1)} = \varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ and $\boldsymbol{W}_{i+s}^{(1)}\boldsymbol{\xi} + 1\boldsymbol{v}_{i+s}^{(1)} = h^{(i)}(\boldsymbol{x})$ for all $i \in \{1, \dots, s\}$. Then, using Theorem 5, we can infer that for every PWQ function $\varphi(\boldsymbol{x})$ there exists a NN with

$$\begin{aligned}\Phi(\boldsymbol{\xi}) &= \max_{1\leq i\leq s}\{\boldsymbol{W}_i^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_i^{(1)}\}-\max_{1\leq i\leq s}\{\boldsymbol{W}_{i+s}^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_{i+s}^{(1)}\}\\&=\max_{1\leq i\leq s}\{\varphi^{(i)}(\boldsymbol{x})+h^{(i)}(\boldsymbol{x})\}-\max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\}\\&=\varphi(\boldsymbol{x})\end{aligned}$$

for all $\boldsymbol{x} \in \Omega$, which completes the proof. ■

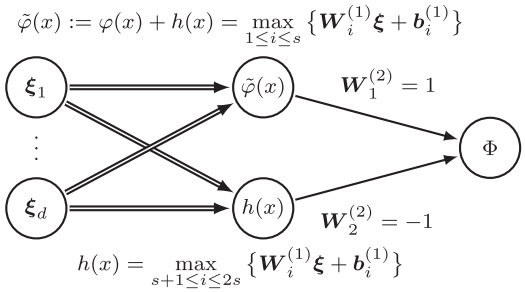

FIGURE 4. Maxout NN with two neurons and extended input vector (42) of dimension $d := \frac{n^2+3n}{2}$ for the exact representation of PWQ functions. The double arrows indicate the multi-channel preactivation with $p_1 = s$, i.e, the s elements in each max-expression (41).

Fig. 4 shows the structure of the maxout NN from Theorem 6 with the extended input vector (42) and two neurons. The first neuron represents (14), and the second represents (15).

2) RELU NEURAL NETWORKS

For deriving relu NNs that allow an exact representation of PWQ functions, we can use results from [6], where max-expressions are represented in terms of a relu NN. This can be directly used to represent the two max-expressions of the derived maxout NN by a relu NN.

Theorem 7: Every continuous PWQ Function of the form (1) with a polyhedral domain partition as in (2) can be represented by a relu NN with $l = \lceil \log_2(s) \rceil$ hidden layers, $w_i = 3 \times 2^{\lceil \log_2(s) \rceil}$ neurons in each layer $i \in \{1, \dots, l\}$, and input vector $\boldsymbol{\xi}$ as in (42).

Proof: We first note that the maximum of two values can be implemented by a relu NN of depth 1 and width 3 using

$$\max\{\boldsymbol{x}_1, \boldsymbol{x}_2\} = \max\{\boldsymbol{x}_1 - \boldsymbol{x}_2, 0\} + \max\{\boldsymbol{x}_2, 0\} - \max\{-\boldsymbol{x}_2, 0\}.$$

Then, the function $\boldsymbol{f} : \mathbb{R}^{2^n} \to \mathbb{R}^{2^{n-1}}$ with

$$\boldsymbol{f}(\boldsymbol{x}) := (\max\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \max\{\boldsymbol{x}_3, \boldsymbol{x}_4\}, \dots, \max\{\boldsymbol{x}_{2^n-1}, \boldsymbol{x}_{2^n}\})^\top$$

and $n := \lceil \log_2(s) \rceil$ can be represented by a relu NN of depth 1 and width $3 \times 2^{n-1}$ since every of the 2^{n-1} components of $\boldsymbol{f}(\boldsymbol{x})$ is a relu NN with width 3. Now, we can compute the maximum $\max\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{2^n}\}$ by calling n-times the function $\boldsymbol{f}(\boldsymbol{x})$. According to [6, Lem. D.1], the composition

$$\underbrace{\boldsymbol{f} \circ \cdots \circ \boldsymbol{f}}_{n \text{ times}}(\boldsymbol{x}) = \max\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{2^n}\}$$

can be represented by a relu NN of depth n and width $3 \times 2^{n-1}$ since $\boldsymbol{f}(\boldsymbol{x})$ is the n-times composition of a function that can be

represented by a relu NN of depth 1 and width $3 \times 2^{n-1}$. This means that with (41) from Theorem 5 and $\boldsymbol{\xi}$ as in (42), every PWQ is the difference of two relu NN of depth n and width $3 \times 2^{n-1}$. This difference can be represented by a relu NN of depth $n = \lceil \log_2(s) \rceil$ and width $2 \times 3 \times 2^{n-1} = 3 \times 2^{\lceil \log_2(s) \rceil}$, according to [6, Lem. D.2]. ■

Theorem 7 proposes a "shallow and wide" NN for the exact representation of PWQ functions. Where shallow refers to the relatively small number of hidden layers $l = \lceil \log_2(s) \rceil$, which grow only logarithmic with s, i.e., with the number of regions of the PWQ function. Wide refers to the potentially large number of neurons w_i in each layer, which grows linearly with s. By using results from [7], we can derive an alternative "deep and narrow" NN, i.e., an NN with a large number of hidden layers and fewer neurons per layer.

Theorem 8: Every continuous PWQ Function of the form (1) with a polyhedral domain partition as in (2) can be represented by a relu NN with $l = 2s$ hidden layers, $w_i = \frac{n^2+3n}{2} + 3$ neurons in each layer $i \in \{1, \dots, l\}$, and input vector $\boldsymbol{\xi}$ as in (42).

Proof: The quadratic local functions $\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ and the linear local functions $h^{(i)}(\boldsymbol{x})$ in (41) are linear combinations of 1 and the $d := \frac{n^2+3n}{2}$ input variables $\boldsymbol{\xi}$ from (42). Thus, according to Theorem 5, for every PWQ function there exist parameters $\boldsymbol{K} \in \mathbb{R}^{s \times d}$, $\boldsymbol{k} \in \mathbb{R}^s$, $\boldsymbol{L} \in \mathbb{R}^{s \times d}$, and $\boldsymbol{l} \in \mathbb{R}^s$ such that

$$\begin{aligned} &\max_{1 \le i \le s} \{\boldsymbol{K}_i \boldsymbol{\xi} + \boldsymbol{k}_i\} - \max_{1 \le i \le s} \{\boldsymbol{L}_i \boldsymbol{\xi} + \boldsymbol{l}_i\} \\ = &\max_{1 \le i \le s} \{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\} - \max_{1 \le i \le s} \{h^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}). \end{aligned} \tag{43}$$

Moreover, since the domain Ω of the PWQ function (1) is polyhedral, the affine mapping

$$\boldsymbol{T}_j(\boldsymbol{\xi}) = \frac{\boldsymbol{\xi}_j - \min\limits_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j}{\max\limits_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j - \min\limits_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j}$$

with $j \in \{1, \dots, d\}$ and the inverse mapping

$$\boldsymbol{T}_j^{-1}(\boldsymbol{\zeta}) = \boldsymbol{\zeta}_j \left(\max_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j - \min_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j \right) + \min_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j$$

with $j \in \{1, \dots, d\}$ exist. We now consider the function $\hat{\varphi}(\boldsymbol{\zeta}) : [0, 1]^d \to \mathbb{R}_+$

$$\hat{\varphi}(\boldsymbol{\zeta}) = \varphi(\boldsymbol{T}^{-1}(\boldsymbol{\zeta})) - \min_{\boldsymbol{x} \in \Omega} \varphi(\boldsymbol{x}).$$

with $\boldsymbol{\zeta} = \boldsymbol{T}(\boldsymbol{\xi})$ and $\boldsymbol{\xi} = \boldsymbol{T}^{-1}(\boldsymbol{\zeta})$. Since $\hat{\varphi}(\boldsymbol{\zeta})$ is a composition of an affine function and (43), there exist parameters $\hat{\boldsymbol{K}} \in \mathbb{R}^{s \times d}$, $\hat{\boldsymbol{k}} \in \mathbb{R}^s$, $\hat{\boldsymbol{L}} \in \mathbb{R}^{s \times d}$, and $\hat{\boldsymbol{l}} \in \mathbb{R}^s$ such that

$$\hat{\varphi}(\boldsymbol{\zeta}) = \max_{1 \le i \le s} \{\hat{\boldsymbol{K}}_i \boldsymbol{\zeta} + \hat{\boldsymbol{k}}_i\} - \max_{1 \le i \le s} \{\hat{\boldsymbol{L}}_i \boldsymbol{\zeta} + \hat{\boldsymbol{l}}_i\}.$$

This function can be represented by a relu NN with $2s$ hidden layers of width $d + 3 = \frac{n^2+3n}{2} + 3$ according to [7, Thm. 2] since $\hat{\varphi}(\boldsymbol{\zeta})$ is a function that maps from $[0, 1]^d$ to $\mathbb{R}_+$. Moreover, we then have

$$\varphi(\boldsymbol{x}) = \varphi(\boldsymbol{\xi}(\boldsymbol{x})) = \varphi(\boldsymbol{T}^{-1}(\boldsymbol{\zeta})) = \hat{\varphi}(\boldsymbol{\zeta}) + \min_{\boldsymbol{x} \in \Omega} \varphi(\boldsymbol{x}).$$

Thus, the PWQ function $\varphi(\boldsymbol{x})$ can be written as a sum of the constant $\min_{\boldsymbol{x} \in \Omega} \varphi(\boldsymbol{x})$ and the function $\hat{\varphi}(\boldsymbol{\zeta}) = \max_{1 \le i \le s} \{\hat{\boldsymbol{K}}_i \boldsymbol{\zeta} + \hat{\boldsymbol{k}}_i\} - \max_{1 \le i \le s} \{\hat{\boldsymbol{L}}_i \boldsymbol{\zeta} + \hat{\boldsymbol{l}}_i\}$. According to [6, Lem. D.2], the sum describing the PWQ function $\varphi(\boldsymbol{x})$ can be represented by an NN with the same topology as the NN for $\hat{\varphi}(\boldsymbol{\zeta})$, i.e., a relu NN with $2s$ hidden layers of width $\frac{n^2+3n}{2} + 3$. This proves the claim of the theorem. ■

By comparing the number of neurons, i.e., the depth l multiplied by the width w_i, in the NNs proposed by Theorems 7 and 8, we can see that the NN from Theorem 7 is advantageous for PWQ functions with a small number of regions s since it has fewer neurons. The exact threshold from which the NN from Theorem 8 has fewer neurons depends on the dimension n. Assuming for simplicity that s only takes multiples of two, we have $w_i l = 3s \log_2(s)$ and $s(n^2 + 3n + 6)$ neurons in the NNs from Theorems 7 and 8, respectively. Thus, for small $s \le 2^{\frac{n^2}{3}+n+2}$, the NN from Theorem 7 has a smaller number of neurons. While for complex PWQ functions with a large number of regions $s > 2^{\frac{n^2}{3}+n+2}$, the deeper NN of Theorem 8 has fewer neurons. This observation is consistent with research in the deep learning field on the advantages of deep NN over NN with fewer hidden layers [42] for approximating complex functions.

B. APPROXIMATIONS OF PWQ FUNCTIONS

In Section IV-A, we showed that it is possible to represent PWQ functions by maxout and relu NNs if the inputs of the NNs are extended by all quadratic monomials as in (42). However, in practice, NNs are typically used to approximate functions based on samples rather than for an exact representation of a given function. However, the results on the exact representation can also be used for an efficient approximation of PWQ functions by using the NNs from Section IV-A for which, in theory, parameters exist that allow an exact representation. Thus, these NNs should lead to approximations with a low error. Moreover, if an upper bound on the number of regions s is known, we can use Theorems 6, 7, and 8 to give bounds on the depth and width an NN should have to provide a low error approximation.

When these NNs are used to approximate a function based on samples, the computation of NN parameters $\boldsymbol{\theta}$ that minimize some loss function can be done in different ways. The classical approach is to solve the nonlinear OP

$$e^* := \min_{\boldsymbol{\theta}} \sum_{i=1}^{N_D} \ell_{\text{NN}} \left(y^{(i)} - \Phi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \right) \tag{44}$$

with a convex loss function ℓ_{NN} approximately using a gradient-based optimization algorithm, e.g., stochastic gradient descent, RMSProb, or Adam [43] for a given dataset

$\mathcal{D} := \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid y^{(i)} = \varphi(\boldsymbol{x}^{(i)}) \forall i \in \{1, \dots, N_D\}\}$. In this approach, we can use Theorems 6, 7, and 8 to choose a suitable topology for which parameters exist that lead to an exact representation of the PWQ function by the NN and thus could provide an approximation with an error of zero, i.e., $e^* = 0$. In Section V-B, we will experimentally show that these NNs indeed lead to approximations with a lower error than classical PWA NNs when trained using a gradient-based optimization algorithm.

Another way to use the results of Section IV-A for computing low error approximations of PWQ functions is to use the fact that for quadratic loss functions ℓ_{NN} and the maxout NN from Theorem 6, the OP (44) can be reformulated as a mixed-integer quadratic program (MIQP) using the results from our preliminary study [31]. The MIQP can be solved globally using standard software, e.g., MOSEK [44] or Gurobi [45].

Corollary 9 ([31, Thm. 1]): For a maxout NN as in Theorem 6, the OP (44) with a quadratic loss function ℓ_{NN} is an MIQP.

Therefore, we can compute globally optimal parameters for maxout NNs, which can represent every PWQ exactly (cf. Theorem 6). This means that if the samples in the dataset $\mathcal{D}$ are from a PWQ function, i.e., $y^{(i)} = \varphi(\boldsymbol{x}^{(i)})$ with s regions and $p_1 \geq s$, then the MIQP for solving (44) will result in an approximation that is exact at the sampling points. Meaning we have $e^* = 0$ and thus $\varphi(\boldsymbol{x}^{(i)}) = \Phi(\boldsymbol{x}^{(i)})$ for all $i \in \{1, \dots, N_D\}$.

V. NUMERICAL EXAMPLES

We will provide different numerical examples. In the first example, we illustrate the representation (10) for an exemplary PWQ function with $n = 2$. In the second part of this section, we compare the in Section IV proposed NNs that allow an exact representation of PWQ functions with classical PWA NNs and alternative PWQ NNs for which an exact representation is not possible by using them to approximate a PWQ OVF (12) for different example systems.

A. EXACT REPRESENTATION OF PWQ FUNCTIONS

In the first example, we illustrate the theoretical results of Section III on the representation of PWQ functions for the continuous PWQ function

$$\varphi(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^\top \boldsymbol{Q}^{(1)} \boldsymbol{x} + \begin{pmatrix} -1 & -1 \end{pmatrix} \boldsymbol{x} + 0.5 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(2)} \boldsymbol{x} + \begin{pmatrix} -1 & -1 \end{pmatrix} \boldsymbol{x} + 0.5 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(2)}, \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(3)} \boldsymbol{x} + \begin{pmatrix} 1 & -1 \end{pmatrix} \boldsymbol{x} + 0.5 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(3)}, \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(4)} \boldsymbol{x} + \begin{pmatrix} 1 & 1 \end{pmatrix} \boldsymbol{x} + 0.5 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(4)}, \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(5)} \boldsymbol{x} + \begin{pmatrix} 1 & 1 \end{pmatrix} \boldsymbol{x} + 0.5 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(5)}, \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(6)} \boldsymbol{x} + \begin{pmatrix} -1 & 1 \end{pmatrix} \boldsymbol{x} + 0.5 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(6)}, \end{cases}$$

with

$$\boldsymbol{Q}^{(1)} = \boldsymbol{Q}^{(5)} = \boldsymbol{Q}^{(6)} = \begin{pmatrix} -2 & 2 \\ 2 & -2 \end{pmatrix},$$

$$\boldsymbol{Q}^{(2)} = \boldsymbol{Q}^{(4)} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}, \boldsymbol{Q}^{(3)} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix},$$

and regions $\mathcal{R}^{(i)}$ as in (2) specified by

$$\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(5)} = \begin{pmatrix} \boldsymbol{I}_2 \\ -\boldsymbol{I}_2 \\ -1 \quad 1 \end{pmatrix}, \boldsymbol{h}^{(2)} = \boldsymbol{h}^{(4)} = \begin{pmatrix} \boldsymbol{I}_2 \\ -\boldsymbol{I}_2 \\ 1 \quad -1 \end{pmatrix},$$

$$\boldsymbol{h}^{(3)} = \boldsymbol{h}^{(6)} = \begin{pmatrix} \boldsymbol{I}_2 & -\boldsymbol{I}_2 \end{pmatrix}^\top,$$

$$\boldsymbol{b}^{(1)} = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{b}^{(3)} = \begin{pmatrix} 0 \\ -1 \\ -1 \\ 0 \end{pmatrix}, \boldsymbol{b}^{(4)} = \begin{pmatrix} 0 \\ 0 \\ -1 \\ -1 \\ 0 \end{pmatrix}, \boldsymbol{b}^{(6)} = \begin{pmatrix} -1 \\ 0 \\ 0 \\ -1 \end{pmatrix},$$

$\boldsymbol{b}^{(2)} = \boldsymbol{b}^{(1)}$, and $\boldsymbol{b}^{(5)} = \boldsymbol{b}^{(4)}$. The function $\varphi(\boldsymbol{x})$ with the corresponding domain partition is shown in Fig. 5(a). We choose the function such that the domain partition is degenerate according to the definition from [26] and the description in Section II-B since at $\boldsymbol{x}^\top = (0 \quad 0)$, we have an intersection of $k = 3$ hyperplanes that is not an empty set. In addition, the consistent variation property (CVP) [25] does not hold here. The regions $\mathcal{R}^{(1)}$ and $\mathcal{R}^{(6)}$ as well as $\mathcal{R}^{(3)}$ and $\mathcal{R}^{(4)}$ are separated by the same hyperplane $\{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{x}_2 = 0\}$. However, $\boldsymbol{k}^{(p,r)}$ from (17) is $\boldsymbol{k}^{(1,6)} = (4 \quad -4) \neq (0 \quad 4) = \boldsymbol{k}^{(3,4)}$ for these regions and thus $g^{(1,6)}(\boldsymbol{x}) \neq g^{(3,4)}(\boldsymbol{x})$. For the CVP to hold, we would need $\boldsymbol{k}^{(1,6)} = \boldsymbol{k}^{(3,4)}$. Hence, the PWQ function considered in this section cannot be represented using methods from the literature, as they require either a non-degenerate partition or the CVP. The representation proposed by Theorem 5 does not assume a non-degenerate partition or the CVP. Thus, as we will show next, it is possible to represent the PWQ function as the difference of two max-expressions by (10). For determining the representation of $\varphi(\boldsymbol{x})$ in the form (10), we first compute a continuous and convex PWA function $h(\boldsymbol{x})$ such that (18) holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \dots, V\}$, which is according to Lemma 4 always possible. We compute the function $h(\boldsymbol{x})$ as described in Remark 3 by computing a convex lifting according to [41, Alg. 3] with a $c := \delta$ for which $\delta > 0$ and (40) holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. In this example, the conditions on δ are satisfied for $\delta = 3$. The convex lifting results in the PWA function

$$h(\boldsymbol{x}) = \begin{cases} \begin{pmatrix} 9.66 & 1.66 \end{pmatrix} \boldsymbol{x} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \begin{pmatrix} 1.66 & 9.66 \end{pmatrix} \boldsymbol{x} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(2)}, \\ \begin{pmatrix} -9.66 & 9.66 \end{pmatrix} \boldsymbol{x} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(3)}, \\ \begin{pmatrix} -9.66 & -1.66 \end{pmatrix} \boldsymbol{x} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(4)}, \\ \begin{pmatrix} -1.66 & -9.66 \end{pmatrix} \boldsymbol{x} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(5)}, \\ \begin{pmatrix} 9.66 & -9.66 \end{pmatrix} \boldsymbol{x} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(6)} \end{cases}$$

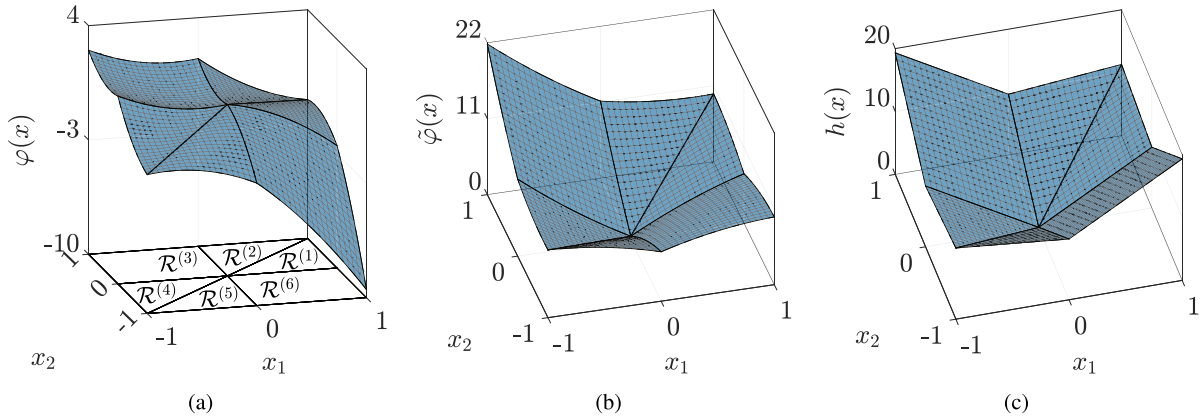

FIGURE 5. Exemplary PWQ function $\varphi(x)$ with six regions in (a). The functions $\tilde{\varphi}(x)$ in (b) and $h(x)$ in (c) are such that they can be represented in the form (14) and (15), respectively, i.e., we have $\varphi(x) = \tilde{\varphi}(x) - h(x) = \max_{1\leq i\leq s}\{\tilde{\varphi}^{(i)}(x)\} - \max_{1\leq i\leq s}\{h^{(i)}(x)\}$. The PWA function $h(x)$ in (c) is computed as described in Remark 3 by solving an OP to determine a convex lifting for the partition. For the sake of clarity, the six polyhedral regions of the domain partition are shown only in (a) but also apply to the functions in (b) and (c).

shown in Fig. 5(c). Since $h(\boldsymbol{x})$ is now such that (18) holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \dots, V\}$, we have $\tilde{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) = \max_{1\leq i\leq 4}\{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\}$ according to Lemma 3. Moreover, $h(\boldsymbol{x})$ is continuous, convex, and PWA. Consequently, we have $h(\boldsymbol{x}) = \max_{1\leq i\leq 4}\{h^{(i)}(\boldsymbol{x})\}$. Thus, as stated by Theorem 5, the function $\varphi(\boldsymbol{x})$ can be represented by

$$\varphi(\boldsymbol{x}) = \underbrace{\max_{1\leq i\leq 4}\{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\}}_{=\tilde{\varphi}(\boldsymbol{x})} - \underbrace{\max_{1\leq i\leq 4}\{h^{(i)}(\boldsymbol{x})\}}_{=h(\boldsymbol{x})}.$$

The decomposition of the function $\varphi(\boldsymbol{x})$ into $\tilde{\varphi}(\boldsymbol{x})$ and $h(\boldsymbol{x})$ is shown in Fig. 5.

B. LEARNING OF OPTIMAL VALUE FUNCTIONS

The aim of this section is to show that the proposed NNs, which allow an exact representation of PWQ functions, lead to better results when they are used for approximating PWQ functions compared to NNs for which an exact representation is not possible. For this purpose, we approximate the PWQ OVF given by (12) for different example systems using the NNs from Section IV-A. The considered example systems, together with the parameters for the OCP (12) and the number of regions of the corresponding OVF, are summarized in Table 1. The number of regions s in Table 1 is determined by explicitly computing the OVF of (12) using the multi-parametric toolbox 3 (MPT3) [48]. For each example system from Table 1, we solved the OCP (12) for different values of $\boldsymbol{x}$ on a regular grid with a step size of 0.1 to generate the data sets $\mathcal{D}_{\boldsymbol{\xi}} := \{(\boldsymbol{\xi}^{(i)}, y^{(i)}) \mid y^{(i)} = V_N(\boldsymbol{\xi}^{(i)}) \forall i \in \{1, \dots, N_D\}\}$ with an extended input vector $\boldsymbol{\xi}$ as in (42). This results in data sets with 6411, 10199, and 31690 data points for examples 1., 2., and 3., respectively, from Table 1. As is common in the machine learning literature [49], we normalized the data according to $\hat{y}^{(i)} := \frac{y^{(i)} - y_{\min}}{y_{\max} - y_{\min}}$ with $y_{\min} := \min_{1\leq j\leq N_D}\{y^{(i)}\}$, $y_{\max} := \max_{1\leq j\leq N_D}\{y^{(i)}\}$ and

$$\hat{\boldsymbol{\xi}}_j^{(i)} := \frac{\boldsymbol{\xi}_j^{(i)} - \min\limits_{1\leq j\leq N_D}\{\boldsymbol{\xi}_j^{(i)}\}}{\max\limits_{1\leq j\leq N_D}\{\boldsymbol{\xi}_j^{(i)}\} - \min\limits_{1\leq j\leq N_D}\{\boldsymbol{\xi}_j^{(i)}\}}$$

for all $j \in \{1, \dots, \frac{n^2+3n}{2}\}$. Afterwards, we trained the NNs listed in Table 2 with the corresponding normalized data set $\hat{\mathcal{D}}_{\boldsymbol{\xi}} := \{(\hat{\boldsymbol{\xi}}^{(1)}, \hat{y}^{(1)}), \dots, \hat{\boldsymbol{\xi}}^{(N_D)}, \hat{y}^{(N_D)})\}$ using the stochastic gradient descent optimizer from the Keras API [50] with a learning rate of 0.01 and a momentum of 0.9. For each case from Table 2, we ran the training 100 times and stopped each training run when there was no improvement in the mean absolute error (MAE) for 100 consecutive epochs. At the beginning of each training run, the weights of the NN are initialized randomly using the method described in [51]. For each of the 100 training runs, we documented the minimum MAE reached during the training. The mean of the MAE over all 100 training runs is shown in column seven of Table 2. The MAEs of the different cases from Table 2 are normalized on the range of the output data, i.e., divided by $y_{\max} - y_{\min}$ so that they are comparable. In the summary of the experiments in Table 2, we compare for each of the three considered OVFs the approximations of an NN with $l = 1$ layer, $w_1 = 2$ neurons with $r_1 = s$, and input vector $\boldsymbol{\xi} = \boldsymbol{x}$, against a NN with $l = 1$ layers, $w_1 = 2$ neurons with $r_1 = s$, and $\boldsymbol{\xi}$ as in (42). According to Theorem 6, the latter is an NN that can represent the considered PWQ function exactly, whereas the former NN is PWA and thus cannot represent PWQ functions. In total, this leads to the six test cases summarized in Table 2.

Now, the experiments offer several insights. First, the NNs with $\boldsymbol{\xi}$ as in (42) that allow an exact representation of the considered PWQ functions show the best results for all three example systems. Compared with the NNs that are not able to represent PWQ functions exactly, the relative improvement is at least $\frac{5.78}{5.7} \times 10 = 10.14$ between the test cases 3. and 4. and the highest observed improvement is $\frac{1.35}{9.9} \times 10^3 = 136.36$ between the test cases 1. and 2.. Thus, we have a significant

TABLE 1. Example systems from the literature.

Ex. No.	$\boldsymbol{A}$	$\boldsymbol{B}$	$\mathcal{X}$	$\mathcal{U}$	$\boldsymbol{Q}$	$\boldsymbol{R}$	N	s	Reference
1.	$\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}$	$\begin{pmatrix}0.5\\ 1\end{pmatrix}$	$\begin{matrix}\lvert\boldsymbol{x}_1\rvert \leq 25\\ \lvert\boldsymbol{x}_2\rvert \leq 5\end{matrix}$	$\lvert\boldsymbol{u}\rvert \leq 1$	$\boldsymbol{I}_2$	0.1	2	121	[46, Eqs. (2.8)–(2.9)]
2.	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\begin{pmatrix}2\\ 4\end{pmatrix}$	$\begin{matrix}\lvert\boldsymbol{x}_1\rvert \leq 5\\ \lvert\boldsymbol{x}_2\rvert \leq 5\end{matrix}$	$\lvert\boldsymbol{u}\rvert \leq 1$	$\boldsymbol{I}_2$	4.5	10	81	[47, Ex. 3]
3.	$\begin{pmatrix}1 & 0.5 & 0.125\\ 0 & 1 & 0.5\\ 0 & 0 & 1\end{pmatrix}$	$\begin{pmatrix}0.02\\ 0.125\\ 0.5\end{pmatrix}$	$\begin{matrix}\lvert\boldsymbol{x}_1\rvert \leq 20\\ \lvert\boldsymbol{x}_2\rvert \leq 3\\ \lvert\boldsymbol{x}_3\rvert \leq 1\end{matrix}$	$\lvert\boldsymbol{u}\rvert \leq 0.5$	$\boldsymbol{I}_3$	1	2	15646	[46, Rem. 4.8]

TABLE 2. Training results of NN approximating different OVFs.

Test case	Ex. No.	$\boldsymbol{\xi}$	l	w_1	r_1	MAE
1.	1.	$\boldsymbol{x}$	1	2	121	1.51×10^{-1}
2.	1.	(42)	1	2	121	4.43×10^{-3}
3.	2.	$\boldsymbol{x}$	1	2	81	5.78×10^{-2}
4.	2.	(42)	1	2	81	5.70×10^{-3}
5.	3.	$\boldsymbol{x}$	1	2	15646	1.35×10^{-1}
6.	3.	(42)	1	2	15646	9.90×10^{-4}

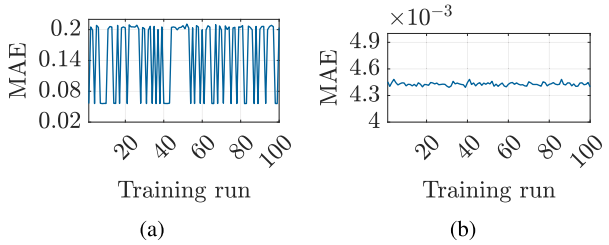

FIGURE 6. Minimum MAE reached in each of the 100 training runs when approximating the OVF for the first example system from Table 1 with the NN from the test cases 1. and 2. from Table 2 in (a) and (b), respectively. The weights of the NN are initialized randomly at the beginning of each training run.

improvement of a minimum of one order of magnitude when using the NNs proposed in Section IV-A.

For the first example system, we additionally illustrate the minimum MAE reached in each of the 100 training runs for test cases 1. and 2. in Fig. 6(a) and (b), respectively.

We can observe that during training with the classical PWA NN with input vector $\boldsymbol{\xi} = \boldsymbol{x}$, the MAE reaches in 36 of the training runs a local optimum at 5.62×10^{-2}, and in the remaining 64 training runs, a local optimum at 2.05×10^{-1}. Thus, with the classical NN, the training often stops at a poor local optimum. Furthermore, we can see in Fig. 6(a) that we have a high ratio of the MAEs between the worst and best training runs of 3.77. With the NN from Theorem 6, for which an exact representation of the OVF is possible, we can observe in Fig. 6(b) that there is only a small variation of the MAEs reached in the different training runs. The ratio of the MAEs between the worst and best training runs is only 1.02. This means that the MAE reaches approximately the value of 4.43×10^{-3} in all training runs. Thus, the training of the NN proposed in this work seems to be more robust with respect to the initialization of the weights.

The example systems from Table 1 all have a state dimension of $n \leq 3$ and an input dimension of $m = 1$, and until now, we have only compared our proposed PWQ NNs to classical PWA NNs. To provide a more competitive comparison, we approximated the OVF of the MIMO30 system from [52], which is a linear system with $n = 10$ and $m = 3$, with a PWQ NN from the literature [19]. For the MIMO30 system we consider the OCP (12) with weighting matrices $\boldsymbol{Q} = \boldsymbol{I}_{10}$, $\boldsymbol{R} = 0.25\boldsymbol{I}_3$, a prediction horizon of $N = 30$, as well as state and input constraints of the form $|\boldsymbol{x}_i| \leq 10$ for all $i \in \{1, \ldots, 10\}$ and $|\boldsymbol{u}_i| \leq 1$ for all $i \in \{1, \ldots, 3\}$, respectively. The resulting OCP is such that the MPT3 [48] fails to compute the explicit solution within a 6-hour time limit. However, we can still generate a data set for approximating the OVF by solving the OCP (12) for 10000 random values of $\boldsymbol{x} \in \mathcal{F}_N$, where $\mathcal{F}_n \subseteq \mathcal{X}$ is the feasible set of (12). We use this dataset to train two different relu NNs. The first relu NN has a topology as in Theorem 8. Note that, since we are not able to solve the OCP (12) explicitly in this example, the number of regions s of the OVF is unknown, and thus we cannot fully utilize the results of Theorem 8. However, some of the parameters are independent of s, e.g., the width w_i of the NN. Thus, we can partially use the results of Theorem 8 by choosing a relu NN with an input vector $\boldsymbol{\xi}$ as in (42) and $l = 1$ layer with $w_1 = \frac{n^2+3n}{2} + 3 = 68$ neurons. We compare this relu NN with the PWQ relu NN from [19] with the "local-global" topology as described in Section II-D. We choose the number of neurons as $w_1 = 380$ such that both NNs have approximately the same number of parameters, i.e., 4560 and 4557, respectively. For both NNs, we computed the mean of the MAE from 100 training runs with a training setup as for the examples in Table 2. The PWQ NN from [19] archives on average a MAE of 4.46×10^{-4} and the NN from Theorem 8 a MAE of 1.02×10^{-4}. Thus, even for a high-dimensional example where the results of Section IV-A can only be partially utilized, we observe a significant relative improvement of 4.37 compared to a baseline NN with PWQ structure from the literature.

VI. CONCLUSION AND SUMMARY

We presented a representation of PWQ functions with a polyhedral domain partition as the difference of two max-expressions in the form (10). The representation is based on constructing a convex lifting of the domain partition that satisfies certain conditions specified in Lemma 3, which ensure (14). We showed in Lemma 4 that these conditions

can always be satisfied by proving that the associated OP for the convex lifting is always feasible. This finally leads to one of the main results of the paper, which is Theorem 5, where we state the existence of a representation of the form (10) for PWQ functions. Based on this representation, we have derived different maxout and relu NN in the Theorems 6, 7 and 8. These NNs are the first for which an exact representation of arbitrary PWQ functions with a polyhedral domain partition is possible. The proposed representation of PWQ functions is illustrated in Section V-A using an exemplary two-dimensional PWQ function. Since the typical use case for NN is not the exact representation of given functions but the approximation of functions based on sample points, we investigated the behaviour of the proposed representation in the context of NN training. For this purpose, we approximated in Section V-B the PWQ OVF of linear MPC for different systems and demonstrated the benefits of the proposed NN compared to classical NN.

APPENDIX

Proof of Proposition 2: To show that for every $i \in \{1, \dots, s\}$ and for every $j \in \{1, \dots, s\} \setminus i$ an index set $\mathcal{C}^{(i,j)}$ as in Assumption 1 exists, we first note that for a fixed but arbitrary $r \in \{1, \dots, s\} \setminus i$, there exists at least one $p \in \{1, \dots, s\}$ with $(p, r) \in \mathcal{N}$ such that $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. We prove this claim by contradiction. Due to the assumption made in the proposition, we either have $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0$ or $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \geq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. We now assume that we have $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \geq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all p with $(p, r) \in \mathcal{N}$, i.e., $\boldsymbol{h}^{(r,p)}\boldsymbol{x} + b^{(r,p)} \leq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all p with $(p, r) \in \mathcal{N}$. This would mean that $\boldsymbol{h}^{(r)}\boldsymbol{x} + b^{(r)} \leq 0$ holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$, which is a contradiction since we have $r \neq i$. Thus, for every $r \in \{1, \dots, s\} \setminus i$ there exists at least one p with $(p, r) \in \mathcal{N}$ such that $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. This means that for every $r \in \{1, \dots, s\} \setminus i$, we can find a p for (19) such that (28) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. Thus, if we start with a fixed but arbitrary $r_1 = j \in \{1, \dots, s\} \setminus i$, we can find a p_1 such that (28) with $(p, r) = (\mathcal{C}^{(i,j)})_1 = (p_1, r_1 = j)$ holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. If $p_1 = i$, we would be done at this point since we then already have an index set $\mathcal{C}^{(i,j)} = \{(i, j)\}$ with the desired properties. For $p_1 \neq i$, we choose $(\mathcal{C}^{(i,j)})_2 = (p_2, r_2) = (p_2, p_1)$. In this case, we can again find a $p = p_2$ such that (28) with $(p, r) = (\mathcal{C}^{(i,j)})_2 = (p_2, p_1)$ holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. Proceeding in this way, we can show that as long as $p_k \neq i$, we can find a p_{k+1} such that (28) with $(p, r) = (\mathcal{C}^{(i,j)})_{k+1} = (p_{k+1}, r_{k+1}) = (p_{k+1}, p_k)$ and $(\mathcal{C}^{(i,j)})_1 = (p_1, j)$ holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. This means that we can find an index set $\mathcal{C}^{(p_k,j)}$ for rewriting $\Delta\tilde{\varphi}^{(p_k,j)}(\boldsymbol{x})$ in terms of (26) such that (28) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(p_k,j)}$. If we have $p_k = i$ for some $k \in \mathbb{N}$, we have a set $\mathcal{C}^{(p_k,j)} = \mathcal{C}^{(i,j)}$ with the desired properties. Thus, it remains to show that there exists a finite k such that $p_k = i$.

Every index pair $(p, r) \in \mathcal{C}^{(p_k,j)}$ represents a hyperplane with $\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. Due to $\boldsymbol{h}^{(p,r)} = -\boldsymbol{h}^{(r,p)}$ and $b^{(p,r)} = -b^{(r,p)}$, we further have $\boldsymbol{h}^{(r,p)}\boldsymbol{x} + b^{(r,p)} \geq 0$ for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(p_k,j)}$ and consequently $(r, p) \notin \mathcal{C}^{(p_k,j)}$ if $(p, r) \in \mathcal{C}^{(p_k,j)}$. Thus, we have $p_l \neq p_m$ and $r_l \neq r_m$ for all $l \in \{1, \dots, k\}$ and for all $m \in \{1, \dots, k\} \setminus l$. Because if we would have $p_l = p_m$ and $r_l = r_m$ for some $l \in \{1, \dots, k\}$ and for some $m \in \{1, \dots, k\} \setminus l$, then $(r, p) \in \mathcal{C}^{(p_k,j)}$, which is a contradiction. We thus have $\mathcal{C}^{(p_k,j)} \subseteq \mathcal{N}$, and therefore, the maximum number of elements in the set $\mathcal{C}^{(p_k,j)}$ is bounded by the number of elements in $\mathcal{N}$, i.e., $k \leq K_{\mathcal{N}} := |\mathcal{N}|$. Moreover, we have $\mathcal{R}^{(i)} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^{(i,r)}\boldsymbol{x} + b^{(i,r)} \leq 0 \ \forall (i, r) \in \mathcal{N}\}$, and thus, there exists at least one $(p, r) \in \mathcal{N}$ with $p = i$ for all $i \in \{1, \dots, s\}$.

In summary, we know that if $p_k \neq i$, we can find a p_{k+1} such that (28) holds for all $(p, r) \in \mathcal{C}^{(p_{k+1},j)}$ and for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$. Thus, as long as $p_k \neq i$ and $k \leq K_{\mathcal{N}}$, we can extend the index set $\mathcal{C}^{(p_k,j)}$ by one element to create the new set $\mathcal{C}^{(p_{k+1},j)}$. Now, we prove that a finite k exists such that $p_k = i$. We assume for contradiction that we have $p_k \neq i$ for all $k \leq K_{\mathcal{N}}$ and for a fixed but arbitrary $i \in \{1, \dots, s\}$. Then, we can extend the set $\mathcal{C}^{(p_k,j)}$ until $\mathcal{C}^{(p_k,j)} = \mathcal{N}$ with $k = K_{\mathcal{N}}$. Due to $(i, r) \in \mathcal{N} = \mathcal{C}^{(p_{K_{\mathcal{N}}},j)}$, we have $p_k = i$ for some $k \leq K_{\mathcal{N}}$, which contradicts our assumption. Thus, there exists a $k \leq K_{\mathcal{N}}$ with $p_k = i$ for all $i \in \{1, \dots, s\}$ and for all $j \in \{1, \dots, s\} \setminus i$.

Meaning that for every $i \in \{1, \dots, s\}$ and for every $r_1 = j \in \{1, \dots, s\} \setminus i$, we can find an index set $\mathcal{C}^{(i,j)} = \{(p_1, r_1) = (p_1, j), (p_2, p_1), (p_3, p_2), \dots, (p_k, p_{k-1}) = (i, p_{k-1})\}$ for rewriting $\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x})$ in terms of (26) such that (28) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(i,j)}$. This proves the claim of the proposition. ■

REFERENCES

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[2] 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.

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

[4] F. Fabiani and P. J. Goulart, "Reliably-stabilizing piecewise-affine neural network controllers," *IEEE Trans. Autom. Control*, vol. 68, no. 9, pp. 5201–5215, Sep. 2023.

[5] D. Teichrib and M. Schulze Darup, "Reachability analysis for piecewise affine systems with neural network-based controllers," in *Proc. 2024 Conf. Decis. Control*, 2024, pp. 894–900.

[6] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," 2018, *arXiv:1611.01491v6*.

[7] B. Hanin, "Universal function approximation by deep neural nets with bounded width and ReLU activations," 2017, *arXiv:1708.02691v3*.

[8] M. Schulze Darup, "Exact representation of piecewise affine functions via neural networks," in *Proc. Eur. Control Conf.*, 2020, pp. 1073–1078.

[9] D. Teichrib and M. Schulze Darup, "Error bounds for maxout neural network approximations of model predictive control," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 10113–10119, 2023.

[10] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3866–3878, Sep. 2020.

[11] S. Kang and L. Chua, "A global representation of multidimensional piecewise-linear functions with linear partitions," *IEEE Trans. Circuits Syst.*, vol. CS-25, no. 11, pp. 938–940, Nov. 1978.

[12] C. Wen and X. Ma, "A canonical piecewise-linear representation theorem: Geometrical structures determine representation capability," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 12, pp. 936–940, Dec. 2011.

[13] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, "Value function approximation and model predictive control," in *Proc. IEEE Symp. Adaptive Dynamic Program. Reinforcement Learn.*, 2013, pp. 100–107.
[14] M. Bhardwaj, S. Choudhury, and B. Boots, "Blending MPC & value function approximation for efficient reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2021.
[15] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for hybrid systems," in *Proc. 24th Int. Conf. Hybrid Syst. Computation Control*, 2021, pp. 1–11.
[16] J. M. Lee and J. H. Lee, "Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes," *Automatica*, vol. 41, no. 7, pp. 1281–1288, 2005.
[17] H. Nosair and J. M. Lee, "Parametric approximation of piecewise quadratic value functions for the control of complex systems," *IFAC Proc. Volumes*, vol. 41, no. 2, pp. 3252–3257, 2008.
[18] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*, Belmont, MA, USA: Athena Scientific, 2019.
[19] 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, 2024, Art. no. 111456.
[20] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, "Adaptive linear quadratic control using policy iteration," in *Proc. Amer. Control Conf.*, 1994, vol. 3, pp. 3475–3479.
[21] A. B. Kordabad, D. Reinhardt, A. S. Anand, and S. Gros, "Reinforcement learning for MPC: Fundamentals and current challenges," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5773–5780, 2023.
[22] M. Dawood, S. Pan, N. Dengler, S. Zhou, A. P. Schoellig, and M. Bennewitz, "Safe multi-agent reinforcement learning for behavior-based cooperative navigation," *IEEE Robot. Automat. Lett.*, vol. 10, no. 6, pp. 6256–6263, Jun. 2025.
[23] Y. Zheng et al., "Distributed q-learning model predictive control based on ADMM for continuous nonlinear systems," *Digit. Chem. Eng.*, vol. 16, 2025, Art. no. 100257.
[24] J.-N. Lin and R. Unbehauen, "On the quadratic extension of the canonical piecewise-linear network," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1992, vol. 1, pp. 316–319.
[25] J.-N. Lin and R. Unbehauen, "Canonical representation: From piecewise-linear function to piecewise-smooth functions," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 40, no. 7, pp. 461–468, Jul. 1993.
[26] W. Li, J.-N. Lin, and R. Unbehauen, "Canonical representation of piecewise-polynomial functions with nondegenerate linear-domain partitions," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 8, pp. 838–848, Aug. 1998.
[27] Y. Cui, T.-H. Chang, M. Hong, and J.-S. Pang, "A study of piecewise linear-quadratic programs," *J. Optim. Theory Appl.*, vol. 186, no. 2, pp. 523–553, 2020.
[28] D. Teichrib and M. Schulze Darup, "Tailored max-out networks for learning convex PWQ functions," in *Proc. Eur. Control Conf.*, 2022, pp. 2272–2278.
[29] D. Teichrib and M. Schulze Darup, "Tailored neural networks for learning optimal value functions in MPC," in *Proc. Conf. Decis. Control*, 2021, pp. 5281–5287.
[30] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, vol. 28, no. 3, pp. 1319–1327.
[31] D. Teichrib and M. Schulze Darup, "Piecewise regression via mixed-integer programming for MPC," in *Proc. 6th Annu. Learn. Dyn. Control Conf.*, (ser. Proceedings of Machine Learning Research), A. Abate, M. Cannon, K. Margellos, and A. Papachristodoulou, Eds., 2024, vol. 242, pp. 337–348.
[32] A. Kripfganz and R. Schulze, "Piecewise affine functions as a difference of two convex functions," *Optimization*, vol. 18, no. 1, pp. 23–29, 1987.
[33] L. Bittner, "Some representation theorems for functions and sets and their application to nonlinear programming," *Numerische Mathematik*, vol. 16, pp. 32–51, 1970.
[34] L. Chua and A.-C. Deng, "Canonical piecewise-linear representation," *IEEE Trans. Circuits Syst.*, vol. CS-35, no. 1, pp. 101–111, Jan. 1988.
[35] J. Caravantes, M. A. Gomez-Molleda, and L. Gonzalez-Vega, "A canonical form for the continuous piecewise polynomial functions," *J. Comput. Appl. Math.*, vol. 283, pp. 17–27, 2015.
[36] D. Q. Mayne, J. B. Rawlings, C. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.
[37] X. Li and K. You, "ReLU neural networks for approximating model predictive control: Complexity and stability guarantees," in *Proc. Amer. Control Conf.*, 2025, pp. 27–32.
[38] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3050–3057, Apr. 2020.
[39] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8387–8406.
[40] D. P. Bertsekas, "Model predictive control and reinforcement learning: A unified framework based on dynamic programming," *IFAC-PapersOnLine*, vol. 58, no. 18, pp. 363–383, 2024.
[41] N. A. Nguyen, M. Gulan, S. Olaru, and P. Rodriguez-Ayerbe, "Convex lifting: Theory and control applications," *IEEE Trans. Autom. Control*, vol. 63, no. 5, pp. 1243–1258, May 2018.
[42] M. Telgarsky, "Benefits of depth in neural networks," in *Proc. 29th Annu. Conf. Learn. Theory*, V. Feldman, A. Rakhlin, and O. Shamir, Eds., 2016, vol. 49, pp. 1517–1539.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.
[44] M. ApS, *The MOSEK Optimization Toolbox for MATLAB Manual*, Version 10.0., 2024. [Online]. Available: http://docs.mosek.com/latest/toolbox/index.html
[45] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com
[46] P.-O. Gutman and M. Cwikel, "An algorithm to find maximal state constraint sets for discrete-time linear dynamical systems with bounded controls and states," *IEEE Trans. Autom. Control*, vol. 32, no. 3, pp. 251–254, Mar. 1987.
[47] M. Schulze Darup and M. Cannon, "Some observations on the activity of terminal constraints in linear MPC," in *Proc. Eur. Control Conf.*, 2016, pp. 4977–4983.
[48] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-parametric Toolbox 3.0," in *Proc. Eur. Control Conf.*, 2013, pp. 502–510.
[49] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training DNNs: Methodology, analysis and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 10173–10196, Aug. 2023.
[50] F. Chollet et al., "Keras," 2015. [Online]. Available: https://keras.io
[51] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, vol. 9, pp. 249–256.
[52] M. Jost, G. Pannocchia, and M. Mönnigmann, "Simulation studies on online constraint removal with a Lyapunov function," 2015, *arXiv:1411.0981*.

DIETER TEICHRIB (Member, IEEE) received the B.Sc. and an M.Sc. degrees in electrical engineering from the Paderborn University, Paderborn, Germany, in 2019 and 2020, respectively. He is currently working toward the Ph.D. degree in control and cyberphysical systems group with the TU Dortmund University, Dortmund, Germany. His research focuses on tailoring machine learning methods for use in control.

MORITZ SCHULZE DARUP (Senior Member, IEEE) received the Diploma degree in mechanical engineering, the B.Sc. degree in physics, and the Ph.D. degree in control engineering from the Ruhr-Universität Bochum, Bochum, Germany, in 2008, 2010, and 2014, respectively. He became an Assistant Professor and Leader of an Emmy Noether Group for Encrypted Controlin 2019 with Paderborn University, Paderborn, Germany. Since 2020, he has been a Full Professor for Control and Cyberphyiscal Systems with TU Dortmund University, Dortmund, Germany. His research interests include secure, predictive, and data-driven control.