Neural Networks for Control: Design and Analysis

Zur Erlangung des akademischen Grades eines

Dr.-Ing.

von der Fakultät Maschinenbau
der Technischen Universität Dortmund

genehmigte Dissertation von

Dieter Teichrib, M.Sc.

aus Pleschanowo

Tag der mündlichen Prüfung: 25.03.2026

1. Gutachter: Prof. Dr.-Ing. Moritz Schulze Darup
2. Gutachter: Prof. Dr. Filippo Fabiani

Lehrstuhl für Regelungstechnik und cyberphysische Systeme

Technische Universität Dortmund

2026

Vorwort

Die vorliegende Arbeit ist während meiner Zeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Regelungstechnik und cyberphysische Systeme der Technischen Universität Dortmund entstanden. Ich hatte die Gelegenheit, bereits in einer sehr frühen Phase der Lehrstuhlgründung dabei zu sein und dadurch den Aufbau und die Gestaltung der Forschung und Lehre von Anfang an erleben und mitgestalten zu können. Dafür und für die hervorragende Betreuung während meiner Zeit am Lehrstuhl bedanke ich mich zutiefst bei meinem Doktorvater, Prof. Dr. Moritz Schulze Darup, der immer zur richtigen Zeit mit einem passenden Rat und guten Impulsen maßgeblich zum erfolgreichen Gelingen beigetragen hat.

Des Weiteren bedanke ich mich bei Prof. Dr. Filippo Fabiani für sein Interesse an meiner Arbeit und für seine Teilnahme an der Prüfungskommission als 2. Gutachter.

Außerdem bedanke ich mich bei meinen Kollegen: Nils Schlüter, Manuel Klädtke, Philipp Binfet, Johannes van Randenborgh und Janis Adamek für die stets gute Zusammenarbeit und die interessanten Gespräche bei unzähligen Espresso sowie für die unvergesslichen und unterhaltsamen Konferenzbesuche.

Ein besonderer Dank gilt meiner Familie: Olga, Johann, Alexandra und meinem Bruder Andreas, ohne den ich vermutlich nie den Schritt ins Studium gewagt hätte. Abschließend bedanke ich mich bei meiner liebevollen Freundin Alina, die mir während der gesamten Zeit bei allen Herausforderungen zur Seite stand.

Dortmund, November 2025 Dieter Teichrib

Kurzfassung

Neuronale Netze (NN) sind ein vielseitiges Modell zur Approximation unterschiedlichster Funktionen. Es überrascht daher nicht, dass sie auch in der Regelungstechnik Anwendung finden. In diesem Kontext werden NN häufig eingesetzt, um komplexe Regler oder *Optimal Value Functions* zu approximieren. Die Flexibilität NN geht mit einer Vielzahl an Hyperparametern einher, die sowohl die Topologie als auch das Training parametrisieren und sorgfältig gewählt werden müssen. Darüber hinaus sind die üblichen Qualitätsmaße, die für des Training verwendet werden, wie z.B. *Mean Squared Error*, nicht ausreichend, um einen sicheren Betrieb NN-basierter Regler zu gewährleisten. Der Einsatz von NN in der Regelungstechnik erfordert daher Methoden, die zwei zentrale Herausforderungen adressieren. Die Erste ist die Wahl einer Topologie, die eine effiziente Approximation von in der Regelungstechnik relevanten Funktionen, z.B. stückweise affine oder stückweise quadratische (PWQ) Funktionen, ermöglicht. Die zweite liegt in der Gewährleistung eines sicheren Betriebs der durch NN geregelten Systeme im Hinblick auf die Einhaltung von Beschränkungen, Stabilität und Datenschutz. Diese Anforderungen werden in klassischen Einsatzgebieten von NNs nicht berücksichtigt. Deswegen werden in dieser Arbeit Methoden entwickelt, die es ermöglichen, NN zur exakten Darstellung stetiger PWQ Funktionen herzuleiten, welche in der modellprädiktiven Regelung eine zentrale Rolle spielen. Außerdem wird durch die Ausnutzung der Struktur dieser NN ein Verfahren vorgestellt, das eine globale Minimierung der während des Trainings betrachteten Kostenfunktion ermöglicht. Hinsichtlich der sicheren Anwendung NN-basierter Regler werden Erweiterungen und Vereinfachungen bestehender Methoden entwickelt, die es erlauben, Stabilität und die Einhaltung von Beschränkungen für lineare Systeme auf Basis des Approximationsfehlers des NNs gegenüber einem stabilisierenden Regler zu verifizieren. Ergänzend dazu werden Methoden zur Analyse NN-basierter Regler vorgestellt, die auf erreichbaren Mengen basieren. Diese Methoden erfordern keinen stabilisierenden Referenzregler und können auch auf nichtlineare Systeme erweitert werden. Zur Gewährleistung des Datenschutzes wird ein Verfahren zur Berechnung polynomialer Approximationen entwickelt, das eine verschlüsselte Auswertung von NN ermöglicht. Zusammenfassend werden verschiedene Methoden vorgestellt, die den sicheren Einsatz von NN in der Regelungstechnik ermöglichen. Gleichzeitig werden bestehende Herausforderungen aufgezeigt, die insbesondere in einer Effizienzsteigerung der auf gemischt-ganzzahliger Optimierung basierenden Methoden liegen, um ihre praktische Anwendbarkeit zu vereinfachen.

Abstract

The universal approximation property of neural networks (NN) makes them a popular tool for approximating various functions. Therefore, it is no surprise that they also found their way in control. In control applications, NNs are commonly used to approximate complex control laws or optimal value functions (OVF) and Q-functions, which allows for simplifying optimal control problems. However, the flexibility of NNs comes with a large number of hyperparameters associated with the topology and training of the NN that need to be chosen carefully. Moreover, typical quality measures used during the training of NNs, like mean squared error, are not sufficient to ensure a safe operation of NN-based controllers. Consequently, the use of NNs in control requires the development of tailored methods that address two main challenges. The first is how to choose a suitable NN topology that allows for efficiently approximating control laws or other functions of interest in control, e.g., PWQ OVFs. The second challenge is to ensure that systems controlled by NNs can be operated safely in terms of constraint satisfaction, stability, and privacy. These requirements are unique to control applications and thus typically not considered in classical NN applications, like image or language processing. Therefore, we address these problems by developing tailored methods that enable us to derive topologies for representing continuous PWQ functions, which play a crucial role in MPC. Furthermore, by exploiting the structure of the derived topologies, we present a training method for maxout NNs that allows us to compute parameters for the NN that globally minimize the cost function considered during training. Regarding the safety of NN-based controllers, we present extensions and simplifications for analysis methods that allow us to certify stability and constraint satisfaction for linear systems based on the approximation error of the NN with respect to a stabilizing baseline controller. We also develop methods for analyzing systems controlled by NNs, based on over approximations of reachable sets, which are used to compute robustly positively invariant sets. These methods do not require a stabilizing baseline controller and can be extended for the analysis of nonlinear systems controlled by NNs. Regarding privacy, we develop a method for computing polynomial approximations that enable an efficient encrypted evaluation of NNs. In summary, the methods presented in this thesis cover a wide range of problems of NNs in control and solve or mitigate many of them. At the same time, we highlight still existing challenges, which especially include increasing the computational efficiency of the methods based on mixed-integer programming for enabling a wide use in practice.

Table of Contents

I Preliminaries and Results

II Articles

Notation

In the following, we list the majority of symbols and acronyms used in this thesis. In particular, the notion used in Part I is covered, whereas the notation in the articles reprinted in Part II can slightly deviate or additional notation is introduced. Whenever such deviations arise, we remind the reader of these differences and point them out.

Abbreviations and Acronyms

AI	artificial intelligence
ADP	approximate dynamic programming
HE	homomorphic encryption
KKT	Karush–Kuhn–Tucker
LMI	linear matrix inequality
LP	linear program
LQR	linear quadratic regulator
MI	mixed-integer
MIMO	multi-input multi-output
MIP	mixed-integer programming
MILP	mixed-integer linear program
MIQP	mixed-integer quadratic program
ML	machine learning
MLD	mixed logical dynamical
MPC	model predictive control
mpQP	multi-parametric quadratic program
NN	neural network
OCP	optimal control problem
OVF	optimal value function
OP	optimization problem
PI	positively invariant
PWA	piecewise affine
PWQ	piecewise quadratic
QP	quadratic program
ReLU	rectified linear unit
RL	reinforcement learning
RPI	robustly positively invariant
SGD	stochastic gradient descent
SISO	single-input single-output
s.t.	subject to

Mathematical Notation

Set of Numbers

$\mathbb{N}_0$	set of natural numbers including 0
$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{R}$	set of real numbers

Matrices, Vectors, and Scalars

$\boldsymbol{A}, \boldsymbol{B}$	System matrices
N	Prediction horizon
$\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{R}$	Weighting matrices of a quadratic cost function
$\boldsymbol{u}$	Input vector
$\boldsymbol{v}^{(i)}$	Bias vector of layer i of a neural network
$\boldsymbol{W}^{(i)}$	Weighting matrix of layer i of a neural network
$\boldsymbol{x}$	State vector
$\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}$	Binary optimization variables
$\boldsymbol{\xi}$	Input vector of a neural network

Functions

$h(\boldsymbol{x})$	Continuous piecewise affine function
$h_{\mathcal{A}}(\boldsymbol{\eta})$	Support function of the polyhedral set $\mathcal{A}$ evaluated for the row vector $\boldsymbol{\eta}$
J	Cost function
$p(\boldsymbol{x})$	Polynomial function
$V(\boldsymbol{x})$	Optimal value function
$\boldsymbol{\pi}(\boldsymbol{x})$	Control law
$\varphi(\boldsymbol{x})$	Continuous piecewise quadratic function

Sets

$\mathcal{A}$	Set of active constraints
$\mathcal{A} \cup \mathcal{B}$	Union of the sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \cap \mathcal{B}$	Intersection of the sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{F}$	Feasible set
$\mathcal{P}$	Polyhedral set $\mathcal{P} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{\mathcal{P}}\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{P}}\}$
$\mathcal{P}^{(i)}$	Polyhedral sets partitioning the domain of a piecewise affine system
$\mathcal{R}^{(i)}$	Polyhedral sets partitioning the domain of a piecewise affine control law
$\mathcal{R}_k$	k-step reachable set
$s\mathcal{A}$	Scaling of a set with a scalar $s \in \mathbb{R}$, i.e., $s\mathcal{A} := \{s\boldsymbol{x} \mid \boldsymbol{x} \in \mathcal{A}\}$
$\mathcal{T}$	Terminal set
$\mathcal{X}, \mathcal{U}$	State and input constraints

Mathematical Notation

Multi Index Notation

$\boldsymbol{A}_i$	i-th row of the matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,j}$	Element in the i-th row and j-th column of the matrix $\boldsymbol{A}$
$\boldsymbol{A}_{\mathcal{A}}$	Submatrix of $\boldsymbol{A}$ containing the rows of $\boldsymbol{A}$ specified by the index set $\mathcal{A}$
$\emptyset$	Empty set

Symbols and Operations

$\mathbf{1}$	Column vector with all entries being one
$\boldsymbol{I}$	Identity matrix
$\mathrm{Dec}(c)$	Decryption of the cipher text c
$\mathrm{Enc}(p)$	Encryption of the plain text p
$\mathrm{int}(\mathcal{P})$	Interior of the polyhedral set $\mathcal{P}$, i.e., $\mathrm{int}(\mathcal{P}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{\mathcal{P}}\boldsymbol{x} < \boldsymbol{h}_{\mathcal{P}}\}$
$\max\{\boldsymbol{x}\}$	Largest element of the vector $\boldsymbol{x}$
$\left\|\boldsymbol{x}\right\|_p$	Vector p-norm of $\boldsymbol{x} \in \mathbb{R}^n$, i.e., $\left\|\boldsymbol{x}\right\|_p := \sqrt{\sum_{i=1}^{n} \left\|\boldsymbol{x}_i\right\|^p}$
$\left\|\boldsymbol{K}\right\|_p$	Norm of the matrix $\boldsymbol{K} \in \mathbb{R}^{m \times n}$ induced by the vector norm $\left\|\cdot\right\|_p$ with $p \in \{p \in \mathbb{R} \mid 1 \leq p < \infty\}$, i.e., $\left\|\boldsymbol{K}\right\|_p := \sup_{\left\|\boldsymbol{x}\right\|_p \leq 1} \left\|\boldsymbol{K}\boldsymbol{x}\right\|_p$
$\mathrm{ReLU}(x)$	Rectified linear unit $\mathrm{ReLU}(x) := \max\{0, x\}$
$\lfloor\cdot\rfloor$	Rounding down
$\lceil\cdot\rceil$	Rounding up
$\lfloor\cdot\rceil$	Rounding to the nearest integer

List of Figures

List of Tables

Part I:

Preliminaries and Results

Chapter 1

Introduction

In this chapter, the focus of the thesis is motivated by discussing the specific challenges that arise when applying machine learning (ML) techniques in control. To situate the work within the broader research landscape, related contributions are reviewed, and their limitations are discussed. Building upon this foundation, we highlight the significance of the problems addressed in the thesis and formulate the central research questions that will guide the research presented in the subsequent chapters.

1.1 Motivation

Over the last few decades, artificial intelligence (AI) has generated a lot of research interest, which has led to the development of numerous methods that can be broadly categorized under the term AI, e.g., (artificial) neural networks (NN) [1], imitation learning [2], reinforcement learning (RL) [3], or more recently transformer networks [4]. Remarkably, almost all methods have been used in the context of control. For example, NNs are used in system identification to approximate complex system dynamics [5], imitation learning to approximate stabilizing but potentially computationally demanding control laws [6, 7], and reinforcement learning for model-free online learning of optimal value functions (OVF) or Q-functions [8] for model predictive control (MPC) [9]. However, control applications typically impose restrictions such as computational limitations, process constraints, stability, and, potentially, also privacy of process data. These restrictions complicate the use of AI methods in control, as they are not considered in classical AI applications like image or language processing. Thus, enabling a reliable use of AI methods in control requires the development of tailored methods that either directly comply with the requirements or that allow for verifying whether the restrictions imposed by the control application are met.

While the aforementioned AI methods are very diverse, many of the methods relevant for a potential application in control involve the use of NNs. For example, to learn Q-functions in RL [10, 11], as a surrogate model in system identification [12], and as a controller in learning based methods [6, 7, 13, 14]. Due to the versatile applications of NNs in control and the open challenges

regarding their safe application, in this thesis, we will focus on NNs. Their flexibility allows for all the different applications, but also requires a careful choice of numerous hyperparameters related to the topology of the NNs and their training. Moreover, each application comes with its very own challenges that need to be addressed for safe employment in real-world systems. These challenges can be categorized into two classes. The first one is the design of suitable NNs that allow for low error approximations of complex functions while enabling a fast evaluation. The second problem class is the analysis of NNs that are used for control.

1.2 Related Work

In this section, we give a brief overview of related work relevant to this thesis. The discussion is intended to situate the presented work within the broader context of current developments in NN-based methods for control. Related articles that are only relevant for specific articles included in the thesis are discussed in detail within the corresponding sections of Chapter 3.

Design of Neural Networks: One of the first applications of NNs in control was the use as a surrogate controller to replace complex control laws, thereby providing an accelerated evaluation [6, 7, 15, 16]. This use of NNs is particularly of interest in the context of optimization-based control methods, e.g., MPC (see, e.g., [17]). These methods require repeatedly solving a mathematical optimization problem (OP) within the sampling period of the system. In the OP-based methods, the solution of the OP can be replaced by an NN evaluation, which can be performed efficiently and in parallel [18]. However, to maximize the speed-up gained from using NNs as a controller instead of MPC, the NN should be as simple as possible while still providing an accurate approximation. Thus, we aim to find suitable NN topologies that comply with these requirements. The topology of an NN is determined by numerous so-called hyperparameters, i.e., the number of neurons and layers and the activation function. Due to a lack of suitable design guidelines for selecting the topology in the context of control, hyperparameters are often chosen through trial and error or the design problem is neglected [6]. Systematic approaches to selecting an appropriate NN topology include heuristic optimization algorithms [19], which search for sets of hyperparameters that yield sufficiently low approximation errors by training NNs with varying topologies. The approaches based on heuristic optimization algorithms offer the benefit of being applicable to a wide range of applications. However, these approaches require the time-consuming training of a large number of NNs with different topologies, which we intend to avoid. Our aim is to find suitable hyperparameters a priori, i.e., before the training is started. Such approaches are already partly available for the case when NNs are used to approximate the optimal control

law of linear MPC, which is known to be a continuous PWA function of the system state [20]. Furthermore, PWA functions can be represented exactly by maxout [21, 22] and ReLU NNs [23, 24, 25]. This connection between NNs and PWA functions has been used in [6, 26] to derive topologies for NNs that allow, in principle, an exact representation of the optimal control law of linear MPC. Moreover, in [7], it has been experimentally demonstrated that NNs that theoretically allow for an exact representation of the function to be approximated lead to low-error approximations when they are used in the context of NN training. Thus, by deriving NNs for the exact representation of the function to be approximated, we can provide design guidelines for the NN topology without time-consuming training.

We aim to extend the results for PWA functions to another important application of NNs in control. Namely, to NNs that are used for an approximation in the value space [3], i.e., for approximating OVF or Q-functions, which are used in the context of RL for model-free learning [8] or approximate dynamic programming (ADP) [27]. The design of tailored NNs for this case has been considered only rarely. For the important system classes of linear and PWA systems, the exact OVFs or Q-functions are PWQ [20]. Thus, the results from the well-studied PWA case do not apply here and can not be used for the design of suitable NNs. Motivated by the success of approximations of control laws for linear MPC using NNs that can exactly represent PWA functions, we will develop NNs for the exact representation of PWQ functions and use them to approximate OVF or Q-functions. These results will fill a research gap, as current methods for representing PWQ functions using NNs are limited to specific function types [27] or impose further restrictions on the partition of PWQ functions [28, 29], thereby limiting their applicability.

Training of Neural Networks: The successful application of NNs in control not only depends on properly choosing the topology but also on the choice of the optimization algorithm and its hyperparameters, e.g., learning rate, batch size, and number of epochs, that is used to minimize a loss function during the training of the NN. As for the topology, an educated choice for these hyperparameters is often not available. Thus, they are often chosen by domain experts based on experience [30]. Moreover, common optimization algorithms for the training of NNs, e.g., stochastic gradient descent (SGD), RMSProp, and Adam [31], are gradient-based methods that typically converge to local optima of the loss function, with the quality of these solutions depending on the chosen hyperparameters. In addition, for some choices of the hyperparameters, the optimization algorithms may even fail to converge. Some approaches have been proposed to mitigate the challenges associated with training of NNs by reformulating the underlying optimization problem and solving it to global optimality [25, 32, 33, 34], thereby eliminating the issue of convergence to local optima. However, these methods are limited to certain types of NNs [25], i.e., ReLU NNs with one hidden layer, or allow only the approximation of func-

tions where the domain has a dimension of $n \in \{1, 2\}$ [33, 34]. We will present a method applicable to arbitrary domain dimensions that enables training a broader class of NNs to global optimality.

Safe Operation of Neural Network-Based Controllers: When NNs are used as an approximate controller in the context of MPC, the potentially complex optimal control problem (OCP) that needs to be solved in every time step is replaced by an evaluation of an NN, thereby avoiding the need to solve an OP in every time step. Thus, the use of NNs has the potential to simplify the evaluation of complex control methods. However, safety requirements such as constraint satisfaction, recursive feasibility, stability, and privacy of process data often prevent the use of NNs in practice.

For ensuring the safety requirements of constraint satisfaction and stability, there are different methods to synthesize NNs that directly satisfy some of these requirements [6, 35, 36] and methods that allow for analyzing whether a given NN fulfills these requirements [13, 16, 37, 38, 39]. Among the analysis methods, there are some in which mixed-integer programming (MIP) is used for certifying stability and constraint satisfaction for linear systems controlled by NNs [13, 16, 38]. These methods build on modeling the approximation error of an NN with respect to a stabilizing baseline controller by an MILP [13, 16], which is used for computing the maximum approximation error and the minimum Lipschitz constant of the error. Thus, they require knowledge of the stabilizing controller that is used to train the NN. An alternative to methods based on approximation error are methods that use reachability analysis [38, 40]. In these methods, an MIP is formulated that describes a linear system controlled by a NN. Using the MIP, the behavior of the closed-loop system can be analyzed by computing reachable sets, which allow for the certification of stability and constraint satisfaction. Since these methods directly describe the closed-loop system by an MIP, a stabilizing baseline controller is not needed. This makes the reachability-based analysis more flexible and eventually allows extending the analysis to nonlinear systems controlled by NNs. We will consider in this thesis both methods, i.e., reachability-based and error-based methods, for the analysis of NNs in control. For the error-based methods, we will derive extensions and simplifications for the underlying MIPs. Furthermore, we will extend the reachability-based methods to an important system class, i.e, PWA systems with bounded additive disturbances, that has not been considered yet.

The final safety requirement for the practical use of NNs in control, namely data privacy, becomes particularly relevant when the control law is evaluated by a third party, such as a cloud computing provider [41]. In these cases, the previously discussed approximations of control laws by NNs enable an evaluation on encrypted input data without requiring an intermediate decryption step, thereby ensuring the privacy of the input data during controller evaluation. Due to certain restrictions of homomorphic encryption (HE) [42, 43],

which is used for a privacy-preserving evaluation, the activation functions of the NNs must be approximated by polynomials. Furthermore, the maximum degree of the used polynomials is limited [44]. This limitation on the degree also leads to a limited accuracy of the polynomial approximation. In this work, we introduce a method that increases the maximum degree, which enables more accurate polynomial approximations of NNs for use with HE.

1.3 Central Research Questions

As highlighted in the preceding discussion, the challenges associated with the use of NNs in control can be grouped into two categories, i.e., design and analysis. The design problem involves selecting suitable NN topologies and appropriately parameterizing the training, while the analysis problem focuses on understanding and verifying the behavior of systems controlled by NNs. The central research questions, formulated in the following, are intended to guide the research toward solving these challenges, which are crucial for enabling the use of NNs in safety-critical systems.

1.3.1 Design of Neural Networks for Control

Regarding the design problems associated with the use of NNs in control, we will focus on approximating functions relevant in control. Our aim is to answer the question of how to choose topologies of NNs that allow for low-error approximations without relying on heuristic optimization methods. This has already been answered for the approximation of certain function types by deriving tailored NNs that allow an exact representation of the function to be approximated. Such NNs have been proposed for the class of PWA functions, which include the optimal control law of linear MPC [6, 7, 26] by establishing a connection between PWA functions and NNs. Moreover, the effectiveness of these NNs, which allow for an exact representation of PWA control laws, has been demonstrated in [6, 7]. However, when NNs are used in the context of RL for model-free learning, PWQ functions, such as OVFs or Q-functions, often need to be approximated. For these cases, the results on the exact representation of PWA functions can not be utilized, and results concerning PWQ functions remain largely unexplored. This leads us to formulate our first research question as follows:
Can every continuous PWQ function with a polyhedral domain partition be exactly represented by an NN?

Answering this question allows us to derive design guidelines for choosing the topologies of NNs that are used to approximate PWQ OVFs or Q-functions in RL for control.

Another important problem within the domain of NN design for control emerges in the context of privacy-preserving evaluation of NNs. As discussed

in Section 1.2, HE can be used for a privacy-preserving evaluation. However, HE schemes typically support only encrypted additions and a limited number of consecutive encrypted multiplications, referred to as the multiplicative depth d. This restricts the functions that can be evaluated in an encrypted fashion to polynomials. Thus, non-polynomial functions need to be approximated by polynomials. Moreover, since the multiplicative depth d limits the degree of compatible polynomials and thereby the accuracy of the approximations, we aim to investigate the following question:
How to maximize the degree of polynomials implementable under a constant multiplicative depth?

An answer to this question would provide an efficient way for computing polynomial approximations of NNs or control laws tailored for use in HE.

1.3.2 Analysis of Neural Network-Based Controller

Using NNs in control imposes restrictions that are typically not considered or are less critical in classical applications of NN, such as regression, image processing, or classification. In these applications, it is often sufficient to find NNs that lead to a low error with respect to some chosen loss function. However, when NNs are used to approximate a baseline controller, the problem becomes more involved. In this case, we typically have constraints on the output of the NN, which reflect the input constraints of the considered system. Moreover, due to the integration of the NN-based controller in a feedback loop with the system to be controlled, deviations from the baseline controller are potentially amplified and may lead to an unstable closed-loop system. Thus, to ensure a safe and reliable operation of a system with an NN as a controller, a detailed analysis of the NN is required. The analysis methods discussed in Section 1.2, based on the approximation error [13, 16], have the problem that they require knowledge of a stabilizing controller. The methods proposed in [38, 40] do not require a stabilizing baseline controller but do require prior knowledge of a control invariant set of the system. In addition, most of the methods, including [38], are only applicable to NN-based controllers used to control linear systems. Thus, the first research question in the realm of analysis aims to address the problems of the existing methods for analyzing NN-based controllers. We aim to answer the research question:
Can we develop a method that allows us to certify crucial properties (stability, recursive feasibility, constraint satisfaction) of NN-based controllers for nonlinear systems without relying on a stabilizing baseline controller?

Investigating this question has the potential to substantially advance existing analysis methods for NN-based controllers, thereby contributing to the development of methods that enable the safe deployment of NNs in real-world systems. Moreover, we address related research questions that involve simplifying and extending the analysis methods based on the approximation error.

1.4 Contributions

The thesis is based on results published in the articles [P1, P2, P3, P4, P5, P6, P7, P8, P9], which aim to provide solutions to the central problems that arise when NNs are used in control. These problems are categorized in Section 1.3 in design and analysis problems. The design of tailored NNs that allow an exact representation of PWQ functions, i.e., the research questions discussed in Section 1.3.1, is addressed in our articles [P1, P2, P9], where we eventually derive tailored NNs that can exactly represent arbitrary continuous PWQ functions with a polyhedral domain partition. We further demonstrate in our articles [P1, P9], through several case studies, that these NNs enhance the learning of PWQ OVFs and Q-functions compared to classical neural networks, which lack the ability to represent PWQ functions exactly. Moreover, in [P5], we show that the structure of the derived tailored NNs allows us to formulate the training of these NNs as a mixed-integer quadratic program (MIQP) for which we can compute the global optimum using standard software [45, 46]. The last contribution under the realm design of NNs is the introduction of tailored polynomial function approximations that allow an efficient encrypted implementation of NNs for a privacy-preserving evaluation. To this end, we present in [P8] a method for increasing the degree of polynomials that can be implemented in an encrypted fashion by one under a constant multiplicative depth.

The articles [P3, P4, P6, P7] are devoted to the development of methods for the analysis of NN-based controllers and thus are guided by the research questions from Section 1.3.2. In [P3], we extend the methods from [13, 16] for certifying stability based on the maximum approximation error of the NN with respect to a baseline controller and the minimum Lipschitz constant of that error. Furthermore, in the article [P4], we present methods for making the computation of the minimum Lipschitz constant more efficient. These methods still require a stabilizing baseline controller and thus do not fully answer the posed research question. In [P6] and [P7], we finally answer the research question by introducing a general framework based on mixed-integer programming (MIP) that allows us to certify stability and constraint satisfaction of PWA systems that are controlled by NN without using a baseline controller. Moreover, the introduced framework even allows the inclusion of additive disturbances in the analysis. This enables the application of the method to nonlinear systems that can be approximated by PWA systems with bounded error, making it applicable to a wide range of systems.

1.5 Outline and Chapter Overview

The remainder of this thesis is structured as follows. In Chapter 2, we provide the necessary background on relevant topics. These include mathematical optimization in Section 2.1, which is relevant for the design as well as for the

analysis of NNs. We separate our introduction to mathematical optimization into two main topics. The first is convex optimization (Section 2.1.1), which is needed for the introduction of MPC in Section 2.2. The second topic is MI optimization (Section 2.1.2), which we need to introduce our MI-based methods from the articles [P3, P6, P7] for the analysis of NN-based controllers, for the consideration of PWA systems [47], and for the design of tailored polynomial function approximations [P8]. In Section 2.2, we introduce MPC for linear and PWA systems. Moreover, we discuss the advantages and disadvantages of MPC and outline how the use of NNs may contribute towards solving some of the disadvantages. Section 2.3 gives a general introduction to the type of NNs considered in this thesis, which are NNs with affine pre- and post-activation and PWA activation. Moreover, we discuss some points on the training and representation capabilities of NNs in Sections 2.3.1 and 2.3.2, which allows us to interpret our results from [P5] and motivates the NNs proposed by us in [P1, P2, P9], respectively. In the final section of Chapter 2, Section 2.4, we summarize relevant literature on learning-based control and discuss known methods for the analysis of NN-based controllers. This discussion is the basis for the derivation of our methods presented in [P3, P4, P6, P7] and summarized in Section 3.2.

In Chapter 3, we summarize and discuss our contributions based on our published articles [P1, P2, P3, P4, P5, P6, P7, P8, P9]. The discussions and summaries of the articles are divided into two main sections, namely Section 3.1 and Section 3.2, each devoted to either the design or analysis of NNs in control. In Section 3.1, we first summarize our results on the exact representation of PWQ functions with polyhedral domain partition by NNs from the articles [P1, P2, P9], with which we aim to address the research questions from Section 1.3.1. Furthermore, the tailored polynomial function approximations for efficient encrypted implementations are presented in [P8]. Based on the topology of the NNs derived in Section 3.1.1, we show in Section 3.1.2 how to formulate the training of the NNs as an MIQP. The results on the MIQP-based training of NNs are published in our article [P5]. The articles addressing the research question from Section 1.3.2 are summarized in Section 3.2. In this section, we deal with the analysis of NN-based controllers and introduce methods for certifying stability, recursive feasibility and constraints satisfaction of closed-loop systems with NN-based controllers. We distinguish between methods that analyze the NN-based controller based on the approximation error with respect to a stabilizing baseline controller and methods based on reachability analysis. The methods of the first type and approaches for making these methods more efficient are summarized in Section 3.2.2, which is based on our articles [P3, P4]. The methods based on reachability analysis from our articles [P6, P7] are summarized and discussed in Section 3.2.3. These methods finally provide a solution to the last problem discussed in Section 1.3.2 since they allow to certify stability, recursive feasibility and constraint satisfaction for PWA systems with bounded disturbances controlled by NN-based controllers.

We give conclusions and an outlook in Chapter 4. These include a short recap of the main contributions. Moreover, we discuss potential improvements of the presented analysis methods from Section 3.2 and further use cases of the tailored NN from Section 3.1 that have not been considered yet.

Part II includes reprinted versions of our articles on which this thesis is based.

Chapter 2

Preliminaries and Background

Before introducing our contributions, we present the background necessary to provide context for the articles in Chapter 3 and to categorize and differentiate the works in relation to existing literature. Topics covered in this chapter include mathematical optimization, MPC, NNs, and learning-based control.

2.1 Mathematical Optimization

A method frequently used in this thesis is mathematical optimization. Mathematical optimization is crucial for many control applications and AI methods. Particularly relevant for this thesis are two classes of optimization problems. The first one is convex optimization, which is often used for optimal control, for example, MPC [48]. The second class is MI optimization, which has attracted much research attention for the analysis of NN [16, 38, P3, P6, P7].

2.1.1 Convex Optimization

In the general form, a convex optimization problem (OP) can be formulated as

$$\min_{z} \; J(z) \quad \text{s.t. } z \in \mathcal{Z}, \tag{2.1}$$

where $J(z) : \mathbb{R}^n \to \mathbb{R}$ is a scalar-valued convex cost function and $\mathcal{Z} \subset \mathbb{R}^n$ is a convex set [49]. An OP without cost function, i.e., $J(z) = 0$, is referred to as a feasibility problem (FP). The set $\mathcal{Z}$ is often represented by a number of equality and inequality constraints $\mathcal{Z} := \{z \in \mathbb{R}^n \mid g(z) \leq \mathbf{0}, h(z) = \mathbf{0}\}$ with $g(z) : \mathbb{R}^n \to \mathbb{R}^{N_i}$ and $h(z) : \mathbb{R}^n \to \mathbb{R}^{N_e}$ [50]. We denote z^* as an optimizer of (2.1) if $J(z^*) \leq J(z)$ for all $z \in \mathcal{Z}$. Moreover, for the problems considered in this thesis, the set $\mathcal{Z}$ is a polyhedral set. In this case the functions $g(z)$ and $h(z)$ are affine and the OP (2.1) becomes

$$\min_{z} \; J(z) \tag{2.2a}$$

$$\text{s.t. } A_{\text{in}} z \leq b_{\text{in}} \tag{2.2b}$$

$$A_{\text{eq}} z = b_{\text{eq}} \tag{2.2c}$$

where the inequalities and equalities with $A_{\text{in}} \in \mathbb{R}^{N_i \times n}$, $b_{\text{in}} \in \mathbb{R}^{N_i}$, $A_{\text{eq}} \in \mathbb{R}^{N_e \times n}$, and $b_{\text{eq}} \in \mathbb{R}^{N_e}$ are understood element-wise.

2.1.1.1 Quadratic programs

For a quadratic cost function $J(z) := 0.5z^\top Hz + c^\top z$ with $H \in \mathbb{R}^{n\times n}$ and $c \in \mathbb{R}^n$, the OP (2.2) is a quadratic program (QP) in the standard form

$$\min_z \frac{1}{2}z^\top Hz + c^\top z \tag{2.3a}$$
$$\text{s.t. } Az \leq b, \tag{2.3b}$$

with $A \in \mathbb{R}^{N_c\times n}$ and $b \in \mathbb{R}^{N_c}$. For H being positive semi-definite, the QP (2.3) is a convex OP [50, Sec. 4.4]. For $H = 0$, the OP (2.3) is a linear program (LP). Note that even though equality constraints are not explicitly considered in (2.3) they can be expressed in terms of inequality constraints by using $\{z \in \mathbb{R}^n \mid A_{\text{eq}}z = b_{\text{eq}}\} = \{z \in \mathbb{R}^n \mid A_{\text{eq}}z \leq b_{\text{eq}}, -A_{\text{eq}}z \leq -b_{\text{eq}}\}$. Then, the constraints (2.2b) and (2.2c) can be compactly written as $Az \leq b$ with

$$A := \begin{pmatrix} A_{\text{in}} \\ A_{\text{eq}} \\ -A_{\text{eq}} \end{pmatrix} \quad \text{and} \quad b := \begin{pmatrix} b_{\text{in}} \\ b_{\text{eq}} \\ -b_{\text{eq}} \end{pmatrix}.$$

There exist several methods for solving QPs of the form (2.3), e.g., interior-point methods [51], active set solvers [52], and gradient-based methods [50]. In the following, we focus on active set methods, as they can be used to show that the explicit control law for linear MPC has a PWA structure and that the associated OVF has a PWQ structure [20]. For the QP (2.3), we can formulate the Lagrange function

$$\mathcal{L}(z, \lambda) = \frac{1}{2}z^\top Hz + c^\top z + \lambda^\top(Az - b). \tag{2.4}$$

The Karush–Kuhn–Tucker (KKT) conditions [50, Sec. 5.5] for the QP (2.3) with Lagrange function (2.4) are

$$Hz^* + c + A^\top\lambda^* = 0 \tag{2.5a}$$
$$Az^* - b \leq 0 \tag{2.5b}$$
$$\lambda^* \geq 0 \tag{2.5c}$$
$$\lambda^{*\top}(Az^* - b) = 0. \tag{2.5d}$$

For feasible QPs, i.e., there exists a $z \in \mathbb{R}^n$ such that $Az \leq b$, the KKT conditions are necessary and sufficient for optimality of (2.3). Thus, given an index set

$$\mathcal{A} := \{i \in \{1, \dots, N_c\} \mid A_i z^* - b_i = 0\}$$

that contains the indices of the constraints that are satisfied with equality at the optimizer, we can solve the linear system of equations

$$Hz^* + c + A_{\mathcal{A}}^\top\lambda_{\mathcal{A}}^* = 0 \tag{2.6a}$$
$$A_{\mathcal{A}}z^* - b_{\mathcal{A}} = 0 \tag{2.6b}$$

derived from (2.5), which results in

$$\begin{pmatrix} \boldsymbol{z}^* \\ \boldsymbol{\lambda}^*_{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H} & \boldsymbol{A}_{\mathcal{A}}^\top \\ \boldsymbol{A}_{\mathcal{A}} & \boldsymbol{0} \end{pmatrix}^{-1} \begin{pmatrix} -\boldsymbol{c} \\ \boldsymbol{b}_{\mathcal{A}} \end{pmatrix}.$$

Using the formula [53, Sec. 3] for inverting 2×2 block diagonal matrices, we can explicitly compute the inverse in the previous equation and obtain

$$\begin{pmatrix} \boldsymbol{z}^* \\ \boldsymbol{\lambda}^*_{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H}^{-1} - \boldsymbol{H}^{-1}\boldsymbol{A}_{\mathcal{A}}^\top \boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \boldsymbol{A}_{\mathcal{A}} \boldsymbol{H}^{-1} & \boldsymbol{H}^{-1}\boldsymbol{A}_{\mathcal{A}}^\top \boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \\ \boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \boldsymbol{A}_{\mathcal{A}} \boldsymbol{H}^{-1} & -\boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \end{pmatrix} \begin{pmatrix} -\boldsymbol{c} \\ \boldsymbol{b}_{\mathcal{A}} \end{pmatrix} \tag{2.7}$$

with $\boldsymbol{S} := \boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^\top$ and $\boldsymbol{S}_{\mathcal{A},\mathcal{A}} := \boldsymbol{A}_{\mathcal{A}}\boldsymbol{H}^{-1}\boldsymbol{A}_{\mathcal{A}}^\top$. Note that for the index set $\mathcal{I} := \{1, \dots, N_c\} \setminus \mathcal{A}$, we have $\boldsymbol{A}_{\mathcal{I}}\boldsymbol{z}^* - \boldsymbol{b}_{\mathcal{I}} < \boldsymbol{0}$ and thus for all the remaining Lagrange multipliers we have, according to (2.5d), $\boldsymbol{\lambda}^*_{\mathcal{I}} = \boldsymbol{0}$. Now, there are different ways of finding the set of active constraints $\mathcal{A}$. The most straightforward is to try all possible combinations for which the inverse $\boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}}$ exists and compute a solution candidate $\boldsymbol{z}^*$ and $\boldsymbol{\lambda}^*_{\mathcal{A}}$ according to (2.7) and terminate if $\boldsymbol{z}^*$ and $\boldsymbol{\lambda}^*_{\mathcal{A}}$ satisfy (2.5b) and (2.5c), respectively. Then, the solution candidate satisfies all KKT conditions and thus is an optimizer of the QP (2.3). More sophisticated methods for solving the QP (2.3) based on active sets are discussed in [52]. However, these methods are not the focus of this thesis. For our purpose, the relation described by (2.7) suffice to derive the desired structural insights for linear MPC.

2.1.1.2 Multi-parametric quadratic programs

In the context of control, we sometimes deal with QPs where $\boldsymbol{c} = \boldsymbol{F}\boldsymbol{x}$ and $\boldsymbol{b} = \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d}$ of the QP (2.3) vary linearly with some parameter vector $\boldsymbol{x} \in \mathbb{R}^{n_p}$. These QPs are denoted as multi-parametric QP (mpQP) [20] and are defined as

$$V(\boldsymbol{x}) := \min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^\top \boldsymbol{H}\boldsymbol{z} + \boldsymbol{x}^\top \boldsymbol{F}^\top \boldsymbol{z} \tag{2.8a}$$

$$\text{s.t. } \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d}. \tag{2.8b}$$

We can use the same method as described in 2.1.1.1 to solve (2.8) for a variable $\boldsymbol{x}$. By substituting $\boldsymbol{c} = \boldsymbol{F}\boldsymbol{x}$, $\boldsymbol{A} = \boldsymbol{G}$, and $\boldsymbol{b} = \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d}$ in (2.7) we can write the solution to (2.8) as

$$\begin{pmatrix} \boldsymbol{z}^* \\ \boldsymbol{\lambda}^*_{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{H}^{-1} - \boldsymbol{H}^{-1}\boldsymbol{G}_{\mathcal{A}}^\top \boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \boldsymbol{G}_{\mathcal{A}} \boldsymbol{H}^{-1} & \boldsymbol{H}^{-1}\boldsymbol{G}_{\mathcal{A}}^\top \boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \\ \boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \boldsymbol{G}_{\mathcal{A}} \boldsymbol{H}^{-1} & -\boldsymbol{S}^{-1}_{\mathcal{A},\mathcal{A}} \end{pmatrix} \begin{pmatrix} -\boldsymbol{F}\boldsymbol{x} \\ \boldsymbol{E}_{\mathcal{A}}\boldsymbol{x} + \boldsymbol{d}_{\mathcal{A}} \end{pmatrix}. \tag{2.9}$$

Now, with a varying parameter $\boldsymbol{x}$, the set of active constraints may vary as well. Thus, the $\boldsymbol{z}^*$ and $\boldsymbol{\lambda}^*$ are functions of the parameter $\boldsymbol{x}$ and take the form

$$\boldsymbol{z}^*(\boldsymbol{x}) = \boldsymbol{K}(\mathcal{A})\boldsymbol{x} + \boldsymbol{l}(\mathcal{A}) \tag{2.10}$$

$$\boldsymbol{\lambda}^*_{\mathcal{A}}(\boldsymbol{x}) = \boldsymbol{L}(\mathcal{A})\boldsymbol{x} + \boldsymbol{p}(\mathcal{A}). \tag{2.11}$$

The parameters $\boldsymbol{K}(\mathcal{A}(\boldsymbol{x}))$, $\boldsymbol{l}(\mathcal{A}(\boldsymbol{x}))$, $\boldsymbol{L}(\mathcal{A}(\boldsymbol{x}))$, and $\boldsymbol{p}(\mathcal{A}(\boldsymbol{x}))$ can be derived by evaluating the multiplication in (2.9). Now, the set containing the indices of the active constraints is defined as

$$\mathcal{A}(\boldsymbol{x}) := \{i \in \{1, \dots, N_c\} \mid \boldsymbol{G}_i \boldsymbol{z}^* - \boldsymbol{E}\boldsymbol{x} - \boldsymbol{d} = 0\}. \tag{2.12}$$

This leads to piecewise constant parameters $\boldsymbol{K}(\mathcal{A})$, $\boldsymbol{l}(\mathcal{A})$, $\boldsymbol{L}(\mathcal{A})$, $\boldsymbol{p}(\mathcal{A})$ and thus (2.10) and (2.11) are continuous PWA functions of $\boldsymbol{x}$ [20, Thm. 4]. We assume that the index set $\mathcal{A}(\boldsymbol{x})$ is such that $\operatorname{rank}(\boldsymbol{G}_{\mathcal{A}(\boldsymbol{x})}) = |\mathcal{A}(\boldsymbol{x})|$, where $|\mathcal{A}(\boldsymbol{x})|$ is the cardinality of the set $\mathcal{A}(\boldsymbol{x})$. This ensures invertibility of $\boldsymbol{S}_{\mathcal{A}(\boldsymbol{x}),\mathcal{A}(\boldsymbol{x})}$ in (2.9). Note, that even for the case $\operatorname{rank}(\boldsymbol{G}_{\mathcal{A}(\boldsymbol{x})}) < |\mathcal{A}(\boldsymbol{x})|$ we can find some subset $\mathcal{S}(\boldsymbol{x}) \subset \mathcal{A}(\boldsymbol{x})$ such that $\operatorname{rank}(\boldsymbol{G}_{\mathcal{S}(\boldsymbol{x})}) = |\mathcal{S}(\boldsymbol{x})|$ and thus ensuring invertibility of $\boldsymbol{S}_{\mathcal{S}(\boldsymbol{x}),\mathcal{S}(\boldsymbol{x})}$ [20, Sec. 4.1.1]. The regions on which the set of active constraints remains constant can be determined by substituting (2.10) and (2.11) in the KKT conditions (2.5b) and (2.5c), which results in the inequalities

$$\underbrace{\begin{pmatrix} \boldsymbol{G}\boldsymbol{K}(\mathcal{A}) - \boldsymbol{E} \\ -\boldsymbol{L}(\mathcal{A}) \end{pmatrix}}_{=:\boldsymbol{H}^{(i)}} \boldsymbol{x} \leq \underbrace{\begin{pmatrix} -\boldsymbol{G}\boldsymbol{l}(\mathcal{A}) + \boldsymbol{d} \\ \boldsymbol{p}(\mathcal{A}) \end{pmatrix}}_{=:\boldsymbol{h}^{(i)}},$$

with $\boldsymbol{H}^{(i)} \in \mathbb{R}^{N_h^{(i)} \times n_p}$ and $\boldsymbol{h}^{(i)} \in \mathbb{R}^{N_h^{(i)}}$. Then the set

$$\mathcal{R}^{(i)} := \{\boldsymbol{x} \in \mathbb{R}^{n_p} \mid \boldsymbol{H}^{(i)}\boldsymbol{x} \leq \boldsymbol{h}^{(i)}\} \tag{2.13}$$

describes a polyhedron. In summary, this means that the optimizer of (2.8) is a PWA function $\boldsymbol{z}^* : \Omega \to \mathbb{R}^m$ of the form

$$\boldsymbol{z}^*(\boldsymbol{x}) = \begin{cases} \boldsymbol{K}_{\mathrm{QP}}^{(1)}\boldsymbol{x} + \boldsymbol{l}_{\mathrm{QP}}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)} \\ \vdots & \\ \boldsymbol{K}_{\mathrm{QP}}^{(s)}\boldsymbol{x} + \boldsymbol{l}_{\mathrm{QP}}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)} \end{cases} \tag{2.14}$$

where $s \in \mathbb{N}$ is the number of regions of the PWA function and $\Omega = \cup_{i=1}^{s} \mathcal{R}^{(i)}$ is the set of parameters for which (2.8) is feasible [20, Thm. 4]. The PWA structure of the optimizer of mpQPs will be crucial for deriving some of the central results of this thesis. Another structural insight of mpQPs we will use is related to the OVF $V(\boldsymbol{x})$ (2.8). By substituting (2.14) in (2.8) we find

$$\begin{aligned} V(\boldsymbol{x}) =& \frac{1}{2}(\boldsymbol{K}_{\mathrm{QP}}^{(i)}\boldsymbol{x} + \boldsymbol{l}_{\mathrm{QP}}^{(i)})^\top \boldsymbol{H}(\boldsymbol{K}_{\mathrm{QP}}^{(i)}\boldsymbol{x} + \boldsymbol{l}_{\mathrm{QP}}^{(i)}) + \boldsymbol{x}^\top \boldsymbol{F}^\top (\boldsymbol{K}_{\mathrm{QP}}^{(i)}\boldsymbol{x} + \boldsymbol{l}_{\mathrm{QP}}^{(i)}) \\ =& \boldsymbol{x}^\top \underbrace{\left(\frac{1}{2}{\boldsymbol{K}_{\mathrm{QP}}^{(i)}}^\top \boldsymbol{H}\boldsymbol{K}_{\mathrm{QP}}^{(i)} + \boldsymbol{F}^\top \boldsymbol{K}_{\mathrm{QP}}^{(i)}\right)}_{=:\boldsymbol{Q}_{\mathrm{QP}}^{(i)}} \boldsymbol{x} + \boldsymbol{x}^\top \underbrace{\left({\boldsymbol{K}_{\mathrm{QP}}^{(i)}}^\top \boldsymbol{H}\boldsymbol{l}_{\mathrm{QP}}^{(i)} + \boldsymbol{F}^\top \boldsymbol{l}_{\mathrm{QP}}^{(i)}\right)}_{=:\boldsymbol{q}_{\mathrm{QP}}^{(i)}} \\ &+ \underbrace{\frac{1}{2}{\boldsymbol{l}_{\mathrm{QP}}^{(i)}}^\top \boldsymbol{H}\boldsymbol{l}_{\mathrm{QP}}^{(i)}}_{=:c_{\mathrm{QP}}^{(i)}}. \end{aligned}$$

Consequently, the OVF is a PWQ function of the form

$$V(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^\top \boldsymbol{Q}_{\mathrm{QP}}^{(1)} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{q}_{\mathrm{QP}}^{(1)} + c_{\mathrm{QP}}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)} \\ \vdots & \\ \boldsymbol{x}^\top \boldsymbol{Q}_{\mathrm{QP}}^{(s)} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{q}_{\mathrm{QP}}^{(s)} + c_{\mathrm{QP}}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)} \end{cases} \tag{2.15}$$

with the same domain partition as the PWA optimizer (2.10). The structural insight provided by (2.15) will be used to derive tailored NNs for learning OVF of mpQPs.

2.1.2 Mixed-Integer Optimization

The methods for analyzing NN-based controllers that we will present in Section 3.2 build on MI optimization. Therefore, in the following, we will provide a short introduction to MI optimization. In MI optimization, a subset of the optimization variables in (2.3) is restricted to the set of integers. For the problems relevant to this thesis, we consider mixed-integer quadratic program (MIQP) of the form

$$\min_{\boldsymbol{z}} \ \frac{1}{2} \begin{pmatrix} \boldsymbol{z}^{(c)\top} & \boldsymbol{z}^{(i)\top} \end{pmatrix} \boldsymbol{H} \begin{pmatrix} \boldsymbol{z}^{(c)} \\ \boldsymbol{z}^{(i)} \end{pmatrix} + \boldsymbol{c}^\top \begin{pmatrix} \boldsymbol{z}^{(c)} \\ \boldsymbol{z}^{(i)} \end{pmatrix} \tag{2.16a}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}, \tag{2.16b}$$

$$\boldsymbol{z}^{(c)} \in \mathbb{R}^{n_c} \tag{2.16c}$$

$$\boldsymbol{z}^{(i)} \in \mathbb{Z}^{n_i} \tag{2.16d}$$

with a positive semi-definite $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ and $n := n_c + n_i$ optimization variables $\boldsymbol{z} := \begin{pmatrix} \boldsymbol{z}^{(c)\top} & \boldsymbol{z}^{(i)\top} \end{pmatrix}^\top$. The optimization variables are separated in n_c continuous optimization variables $\boldsymbol{z}^{(c)} \in \mathbb{R}^{n_c}$ and n_i integer optimization variables $\boldsymbol{z}^{(i)} \in \mathbb{Z}^{n_i}$. For $\boldsymbol{H} = \boldsymbol{0}$, the OP (2.16) is referred to as a mixed-integer linear program (MILP). If we consider OPs of the form (2.16) without specifying the cost function, i.e., quadratic or linear, we refer to (2.16) as a mixed-integer program (MIP). MIPs of the form (2.16) are often solved using branch and bound or cutting plane methods [54, 55]. In these methods, the MIP (2.16) is solved by first solving the LP-relaxation, i.e., a variant of the MIP (2.16) where some of the constraints (2.16d) are replaced by linear constraints with continuous optimization variables until a solution is found where the variables $\boldsymbol{z}^{(i)}$ are integers [54]. However, we will not describe the solution methods for MIPs in detail, since in this thesis, MIPs are mainly used to describe control laws and NNs via MI linear constraints.

2.1.2.1 Modelling Using Mixed-Integer Linear Constraints

In the context of control, MIPs are used in different ways. For example, MIQPs are used to solve optimal control problems (OCP) for PWA systems [47] or MI

linear constraints can be used to model certain types of NNs [P3, 56]. Moreover, MI linear constraints can be used to reformulate mpQPs of the form (2.8) as MI feasibility problems [16].

Mixed-Integer Linear Constraints for Piecewise Affine Functions: For PWA functions $f_{\text{PWA}} : \mathbb{R}^n \to \mathbb{R}^m$ of the type (2.14), we can use MIPs to determine the index i for which we have $\boldsymbol{x} \in \mathcal{R}^{(i)}$ for a given $\boldsymbol{x}$. We use the method presented in [47] and introduce the binary variables $\boldsymbol{\gamma} \in \{0,1\}^s$ that model the relation

$$\gamma_i = 1 \Leftrightarrow \boldsymbol{x} \in \mathcal{R}^{(i)}. \tag{2.17}$$

We consider polyhedral sets that are described by a number of inequalities as in (2.13), where each inequality describes a half-space and the set $\mathcal{R}^{(i)}$ is the intersection of half-spaces. For these sets, (2.17) can be described by

$$\boldsymbol{H}^{(i)}\boldsymbol{x} \leq \boldsymbol{h}^{(i)} + \boldsymbol{M}^{(i)}(1-\gamma_i) \tag{2.18a}$$

$$\mathbf{1}^\top \boldsymbol{\gamma} = 1, \quad \boldsymbol{\gamma} \in \{0,1\}^s, \tag{2.18b}$$

where $\boldsymbol{M}^{(i)}$ is chosen such that

$$\boldsymbol{M}_j^{(i)} := \max_{\boldsymbol{x}\in\Omega} \{\boldsymbol{H}_j^{(i)}\boldsymbol{x} - \boldsymbol{h}_j^{(i)}\}$$

for all $j \in \{1,\dots,N_h^{(i)}\}$ [47, Eq. (20)-(21)]. Due to the typically large constant $\boldsymbol{M}^{(i)}$, this formulation is often referred to as big-M formulation. Note that if the constraints (2.18) are used in an MIP, the smallest possible $\boldsymbol{M}$ for which (2.18) still describes (2.17) is favorable since it leads to better LP-relaxations while solving the MIP. However, for clarity of presentation, we will write in the following for (2.18a) the short version $\boldsymbol{H}^{(i)}\boldsymbol{x} \leq \boldsymbol{h}^{(i)} + \mathbf{1}M(1-\gamma_i)$ although the constant M may vary with i. Now, we have MI linear constraints that select the index i for which we have $\boldsymbol{x} \in \mathcal{R}^{(i)}$. To use these constraints for the evaluation of PWA functions, we still need constraints to select the active affine function $\boldsymbol{K}^{(i)}\boldsymbol{x} + \boldsymbol{l}^{(i)}$ based on the binary variable $\boldsymbol{\gamma}$. According to [47, Sec. 3.1] the MI linear constraints

$$-\mathbf{1}M(1-\gamma_i) \leq \boldsymbol{K}^{(i)}\boldsymbol{x} + \boldsymbol{l}^{(i)} - \tilde{\boldsymbol{f}}^{(i)} \leq \mathbf{1}M(1-\gamma_i) \;\forall\, i \in \{1,\dots,s\}, \tag{2.19a}$$

$$-\mathbf{1}M\gamma_i \leq \tilde{\boldsymbol{f}}^{(i)} \leq \mathbf{1}M\gamma_i \qquad \forall\, i \in \{1,\dots,s\}, \tag{2.19b}$$

$$\boldsymbol{f} = \sum_{j=1}^{s} \tilde{\boldsymbol{f}}^{(j)} \tag{2.19c}$$

are such that we have

$$\gamma_i = 1 \Rightarrow \boldsymbol{f} = \boldsymbol{K}^{(i)}\boldsymbol{x} + \boldsymbol{l}^{(i)}. \tag{2.20}$$

Thus, combining the MI linear constraints (2.18) that model (2.17) and (2.19) for (2.20) results in

$$\boldsymbol{x} \in \mathcal{R}^{(i)} \Leftrightarrow \gamma_i = 1 \Rightarrow \boldsymbol{f} = \boldsymbol{K}^{(i)}\boldsymbol{x} + \boldsymbol{l}^{(i)}.$$

Consequently, the solution to the MIFP

$$\text{find } \boldsymbol{f}, \tilde{\boldsymbol{f}}, \boldsymbol{\gamma} \quad \text{s.t. (2.18) and (2.19)} \tag{2.21}$$

is such that we have $\boldsymbol{f} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x})$. We will use this relation in 3.2 to describe the dynamics of PWA systems by MI linear constraints.

Mixed-Integer Linear Constraints for the ReLU Function: Another important function that can be modeled using MI linear constraints is the ReLU function

$$\text{ReLU}(x) := \max\{x, 0\},$$

which is often used as an activation function for NNs. By using the results from [16, Eq.(18)-(19)], the ReLU function can be described by the MI linear constraints

$$-M(1-\delta) \leq x \leq M\delta + (1-\delta)\epsilon \tag{2.22a}$$
$$-M\delta \leq y \leq M\delta, \tag{2.22b}$$
$$-M(1-\delta) \leq y - x \leq M(1-\delta), \tag{2.22c}$$
$$\delta \in \{0, 1\} \tag{2.22d}$$

where $\epsilon > 0$ is a small constant and M is a large constant as above. Roughly speaking, the first constraints (2.22a) ensure that $\delta = 1$ if and only if $x > 0$ and $\delta = 0$ if and only if $x \leq 0$. Then with the constraints (2.22b) and (2.22c) we have $y = 0$ if and only if $\delta = 0$ and $y = x$ if and only if $\delta = 1$ and consequently with (2.22a) we have

$$y = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases}$$

Thus, the solution of the MIFP

$$\text{find } y, \delta \quad \text{s.t. (2.22)} \tag{2.23}$$

is such that the optimization variable $y = \text{ReLU}(x)$ describes the ReLU function [16, Thm. 6.1]. The formulation (2.22) can be extended to describe the maximum of several affine functions $y = \max\{\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1, \ldots, \boldsymbol{W}_s\boldsymbol{x} + \boldsymbol{b}_s\}$ by MI linear constraints. We will introduce this extension in Sections 3.1.2,3.2, 7, and 9 to describe maxout NN with arbitrary topologies by MI linear constraints, which will allow us to analyze NN-based controllers by MILPs.

Mixed-Integer Linear Constraints for Multi-Parametric Quadratic Programs: MI linear constraints can be used to model complementarity constraints of the form $ab = 0$ with $a, b \in \mathbb{R}$, which is equivalent to $a = 0 \vee b = 0$. These types of constraints are relevant for control since they appear in the KKT-conditions as complementarity condition (2.5d) and thus can be used to compute points that

satisfy all KKT-conditions. Constraints of the form $ab = 0$ can be modeled as described in [57] by the following MI linear constraints

$$-M\delta \leq a \leq M\delta \tag{2.24a}$$
$$-M(1-\delta) \leq b \leq M(1-\delta) \tag{2.24b}$$
$$\delta \in \{0,1\}. \tag{2.24c}$$

where M is a constant that is larger than the largest possible value of a and b. The constraints are such that $\delta = 0$ implies $a = 0$ and $\delta = 1$ implies $b = 0$. This can be used to replace the complementarity condition (2.5d) and formulate the KKT-conditions of the mpQP (2.8) as MI linear constraints

$$\boldsymbol{H}\boldsymbol{z}^*(\boldsymbol{x}) + \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}^\top \boldsymbol{\lambda}^* = \boldsymbol{0}, \tag{2.25a}$$
$$\boldsymbol{r}^* = \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d} - \boldsymbol{G}\boldsymbol{z}^*(\boldsymbol{x}), \tag{2.25b}$$
$$\boldsymbol{0} \leq \boldsymbol{r}^* \leq \operatorname{diag}(\bar{\boldsymbol{r}})(\boldsymbol{1} - \boldsymbol{\delta}^*), \tag{2.25c}$$
$$\boldsymbol{0} \leq \boldsymbol{\lambda}^* \leq \operatorname{diag}(\overline{\boldsymbol{\lambda}})\, \boldsymbol{\delta}^*, \tag{2.25d}$$
$$\boldsymbol{\delta}^* \in \{0,1\}^{N_c}. \tag{2.25e}$$

where $\overline{\boldsymbol{\lambda}} \in \mathbb{R}^{N_c}$ and $\bar{\boldsymbol{r}} \in \mathbb{R}^{N_c}$ are sufficiently large constants that take the role of the big-M constant (see [16] for details). Since for the OP (2.8), the KKT-conditions are sufficient and necessary optimality criterion [50, Sec. 5], a point $\boldsymbol{z}^*$ that satisfies (2.25) is an optimizer of the OP. Thus, we can solve (2.8) by solving the MIFP

$$\text{find}\, \boldsymbol{z}^*, \boldsymbol{r}^*, \boldsymbol{\lambda}^*, \boldsymbol{\delta}^* \quad \text{s.t. (2.25)}. \tag{2.26}$$

The binary optimization variables select the active constraints for a given $\boldsymbol{x}$, i.e., we can write the set (2.12) as

$$\mathcal{A} = \{i \in \{1, \dots, N_c\} \mid \boldsymbol{\delta}_i^* = 1\}.$$

Thus, instead of solving the mpQP (2.8) by solving the linear system of equations given by the KKT-conditions via an active set solver as in (2.9), we can compute a $\boldsymbol{z}^*$ that satisfies the KKT-conditions by solving the MIFP (2.25). The formulation as MIFP offers the possibility of analyzing the optimizer $\boldsymbol{z}^*(\boldsymbol{x})$ in different ways, which we will use in Sections 3.2 and 7 to compute the maximum approximation error of an NN that approximates the optimizer $\boldsymbol{z}^*(\boldsymbol{x})$. There, we will consider OPs of the form

$$\min_{\boldsymbol{x}} J\left(\boldsymbol{z}(\boldsymbol{x})\right) \quad \text{s.t. } \boldsymbol{z}(\boldsymbol{x}) = \boldsymbol{z}^*(\boldsymbol{x}) \tag{2.27}$$

where $J : \mathbb{R}^{n_p} \to \mathbb{R}$ is a linear or quadratic cost function of the optimizer of (2.8). The OP (2.27) is a nested OP since the cost function depends on the solution of the mpQP (2.8). OPs of this type can not be solved directly with the methods introduced so far. However, since the optimizer $\boldsymbol{z}^*(\boldsymbol{x})$ of (2.8) can

be described by MI linear constraints, we can replace the constraint of (2.27) by the MI linear constraints (2.25). This results in the OP

$$\min_{\boldsymbol{x}} J\left(\boldsymbol{z}^*(\boldsymbol{x})\right) \quad \text{s.t. (2.25)} \tag{2.28}$$

which is a MILP or MIQP [16, Thm. 5.3] as in (2.16), that can be solved using standard software, e.g., [45] or [46].

2.2 Model Predicitve Control

MPC has emerged as a popular control strategy in both academia and industry over the past decades. Its popularity stems from its ability to handle complex systems, explicitly incorporate system constraints, and offer inherent robustness to disturbances. This has led to its widespread adoption in various fields, including process control [58], automotive systems [59], and aerospace [60]. In the following, we will provide an overview of the mathematical foundations of MPC and present known results on the structure of the underlying optimal control problem (OCP) that we will use for the design of tailored NNs. We focus on discrete-time systems whose dynamics can be described by

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \tag{2.29}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ and $\boldsymbol{u}(k) \in \mathbb{R}^m$ are, respectively, the system state and the control input. In classical MPC, we consider the optimal control problem (OCP)

$$V_N(\boldsymbol{x}) := \min_{\substack{\hat{\boldsymbol{x}}(0),\dots,\hat{\boldsymbol{x}}(N),\\ \hat{\boldsymbol{u}}(0),\dots,\hat{\boldsymbol{u}}(N-1)}} \|\hat{\boldsymbol{x}}(N)\|_{\boldsymbol{P}}^2 + \sum_{\kappa=0}^{N-1} \|\hat{\boldsymbol{x}}(\kappa)\|_{\boldsymbol{Q}}^2 + \|\hat{\boldsymbol{u}}(\kappa)\|_{\boldsymbol{R}}^2 \tag{2.30a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{x}}(0) = \boldsymbol{x}, \tag{2.30b}$$

$$\hat{\boldsymbol{x}}(\kappa+1) = \boldsymbol{f}(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)), \qquad \forall \kappa \in \{0,\dots,N-1\}, \tag{2.30c}$$

$$(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)) \in \mathcal{X} \times \mathcal{U}, \qquad \forall \kappa \in \{0,\dots,N-1\}, \tag{2.30d}$$

$$\hat{\boldsymbol{x}}(N) \in \mathcal{T} \tag{2.30e}$$

with a prediction horizon of N. The sets $\mathcal{X}$ and $\mathcal{U}$ are state and input constraints, $\mathcal{T}$ is a terminal constraint, $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are positive semi-definite weighting matrices. The terminal set $\mathcal{T}$, in combination with the terminal cost parametrized by $\boldsymbol{P}$, allows to enforce closed-loop stability (see [48] for details). In the deployment of MPC, the OCP (2.30) is solved in every time-step for the current system state $\boldsymbol{x} = \boldsymbol{x}(k)$ to obtain the optimal input sequence

$$\hat{\boldsymbol{U}}_N^* := \begin{pmatrix} \hat{\boldsymbol{u}}^*(0) \\ \vdots \\ \hat{\boldsymbol{u}}^*(N-1) \end{pmatrix} \in \mathbb{R}^{mN}.$$

Typically, only the first element $\hat{\boldsymbol{u}}^*(0)$ of the optimal input sequence is applied to the system before the OCP is solved again. This results in the implicit control law $\boldsymbol{\pi} : \mathcal{F} \to \mathbb{R}^m$ of MPC given by

$$\boldsymbol{\pi}(\boldsymbol{x}) := \hat{\boldsymbol{u}}^*(0), \tag{2.31}$$

where $\mathcal{F} \subset \mathbb{R}^n$ is the set of states for which (2.30) is feasible. Thus, the OCP must be solved repeatedly at each time step within the sampling period of the system. The repeated solving of the OCP (2.30) may be intractable for systems with a short sampling period or OCPs with a large number of constraints.

2.2.1 Model Predictive Control for Linear Systems

A possible way to reduce the computational burden of MPC for linear time-invariant systems of the form

$$\boldsymbol{x}(k+1) = \boldsymbol{f}_{\mathrm{L}}(\boldsymbol{x}(k), \boldsymbol{u}(k)) := \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{2.32}$$

is to reformulate the OCP (2.30). For polyhedral state, input, and terminal constraints of the form $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{\mathcal{X}}\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{X}}\}$, $\mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^m \mid \boldsymbol{H}_{\mathcal{U}}\boldsymbol{u} \leq \boldsymbol{h}_{\mathcal{U}}\}$, and $\mathcal{T} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{\mathcal{T}}\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{T}}\}$, respectively, we can find matrices $\boldsymbol{H}$, $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{E}$, and $\boldsymbol{d}$ that depend on $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{H}_{\mathcal{X}}$, $\boldsymbol{h}_{\mathcal{X}}$, $\boldsymbol{H}_{\mathcal{U}}$, $\boldsymbol{h}_{\mathcal{U}}$, $\boldsymbol{H}_{\mathcal{T}}$, and $\boldsymbol{h}_{\mathcal{T}}$ such that the OCP (2.30) takes the form of a mpQP (2.8) with $\boldsymbol{z} = \hat{\boldsymbol{U}}_N$. Thus, we can use the method described in Section 2.1.2.1 to solve the mpQP by solving the MIFP

$$\begin{aligned}
&\text{find } \hat{\boldsymbol{U}}_N^*, \boldsymbol{r}^*, \boldsymbol{\lambda}^*, \boldsymbol{\delta}^* && \text{(2.33a)}\\
&\text{s.t.} \quad \boldsymbol{0} = \boldsymbol{H}\hat{\boldsymbol{U}}_N^*(\boldsymbol{x}) + \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}^\top\boldsymbol{\lambda}^*, && \text{(2.33b)}\\
&\qquad \boldsymbol{r}^* = \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d} - \boldsymbol{G}\hat{\boldsymbol{U}}_N^*(\boldsymbol{x}), && \text{(2.33c)}\\
&\boldsymbol{0} \leq \boldsymbol{r}^* \leq \operatorname{diag}(\overline{\boldsymbol{r}})(\boldsymbol{1} - \boldsymbol{\delta}^*), && \text{(2.33d)}\\
&\boldsymbol{0} \leq \boldsymbol{\lambda}^* \leq \operatorname{diag}(\overline{\boldsymbol{\lambda}})\,\boldsymbol{\delta}^*, && \text{(2.33e)}\\
&\qquad \boldsymbol{\delta}^* \in \{0,1\}^{N_c}. && \text{(2.33f)}
\end{aligned}$$

Moreover, we can explicitly compute the solution $\hat{\boldsymbol{U}}_N^*(\boldsymbol{x})$ of (2.30) as a function of the system state $\boldsymbol{x}$, which is, as described in Section 2.1.1.2, a PWA function of the form (2.10). Now, since in MPC only the first element of the optimal input sequence is applied, we are interested in the control law

$$\boldsymbol{S}\hat{\boldsymbol{U}}_N^*(\boldsymbol{x}) = \boldsymbol{\pi}(\boldsymbol{x}) = \begin{cases} \boldsymbol{K}^{(1)}\boldsymbol{x} + \boldsymbol{l}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \\ \boldsymbol{K}^{(s)}\boldsymbol{x} + \boldsymbol{l}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)} \end{cases} \tag{2.34}$$

with $\boldsymbol{S} := \begin{pmatrix} \boldsymbol{I}_m & \boldsymbol{0}_{m \times m(N-1)} \end{pmatrix}$, which is a PWA function since $\boldsymbol{\pi}(\boldsymbol{x})$ is a linear transformation of a PWA function [20, Cor. 2]. The sets $\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(s)}$ partition

the feasible set of (2.30) according to $\mathcal{F} = \cup_{i=1}^{s} \mathcal{R}^{(i)}$. We further define the local gain $\boldsymbol{K} : \mathbb{R}^n \to \mathbb{R}^{m \times n}$ as the function

$$\boldsymbol{K}(\boldsymbol{x}) := \begin{cases} \boldsymbol{K}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \\ \boldsymbol{K}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}. \end{cases} \tag{2.35}$$

Moreover, as depicted in Section 2.1.1.2, the OVF of the OCP (2.30) is a continuous PWQ function of the form

$$V_N(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^\top \boldsymbol{Q}^{(1)} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{q}^{(1)} + c^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(s)} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{q}^{(s)} + c^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}. \end{cases} \tag{2.36}$$

Explicitly computing the PWA control law (2.34) offline allows the online application of MPC by evaluating a PWA function in every time step instead of solving the potentially complex QP (2.30) in every time step. However, the offline computation or even the online evaluation of the PWA control law may be too complex for OCPs with a long prediction horizon or a large number of constraints. Both the number of constraints in (2.30) and the prediction horizon N increase the number of constraints in (2.8b). This leads, in the worst case, to an exponential increase in the number of regions of (2.34). Thus, explicitly computing (2.34) is typically only possible for OCPs with a short prediction horizon and a small number of constraints. This drawback is often mitigated by approximating the PWA control law (2.34) by functions that have a small memory footprint and are fast to evaluate, e.g., NNs [6]. We will detail the approximation of control laws by NNs in Section 2.4.

2.2.1.1 Computation of Lipschitz Constants for Model Predictive Control

In general, a Lipschitz constant of a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ on the domain $\mathcal{F} \subset \mathbb{R}^n$ is defined as a constant L_p satisfying

$$\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\tilde{\boldsymbol{x}})\|_p \leq L_p \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_p \tag{2.37}$$

for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{F}$. We here focus on the cases $p \in \{1, \infty\}$. Lipschitz constants have proven beneficial in many applications, as they provide a bound on how sensitive a function is to variations in the input. For MPC, this allows us to analyze to what degree the control law (2.31) provides inherent robustness, i.e., robustness against unmodeled disturbances [61, 62]. The property of inherent robustness can also be used to certify the stability of NN-based approximations of the optimal control law (2.34) by computing Lipschitz constants of the approximation error [16]. To provide less conservative certificates for the inherent robustness or stability of NN-based approximations, small Lipschitz constants

are advantageous. The smallest possible Lipschitz constant of a PWA function of the form (2.34) is, according to [63, Prop. 3.4], given by

$$\mathcal{L}_p(\mathcal{F}) = \max_{\boldsymbol{x}\in\mathcal{F}} \|\boldsymbol{K}(\boldsymbol{x})\|_p = \max_{i\in\{1,\dots,s\}} \left\|\boldsymbol{K}^{(i)}\right\|_p. \tag{2.38}$$

Thus, given the explicit PWA function (2.34), we can compute the minimum Lipschitz constant by finding the maximum p-norm of the matrices $\boldsymbol{K}^{(i)}$, which is defined as

$$\|\boldsymbol{K}\|_p := \sup_{\|\boldsymbol{x}\|_p\leq 1} \|\boldsymbol{K}\boldsymbol{x}\|_p.$$

Computing (2.38) for MPC requires explicitly computing the optimal control law by solving the mpQP (2.8) for all feasible states $\boldsymbol{x} \in \mathcal{F}$, which becomes intractable for OCPs with a long prediction horizon or a large number of constraints (cf. Section 2.2.1). Therefore, in Sections 3.2 and 8, we will present a method for computing the minimum Lischitz constant based on MILP that does not require the explicit computation of the optimal control law. The method builds on reformulating the local gain (2.35) as MIFP based on the MIFP formulation (3.23) of mpQP presented in Section 2.1.2.1. The underlying idea is to introduce n points

$$\boldsymbol{x}^{(j)} := \boldsymbol{x} + \boldsymbol{e}^{(j)} \tag{2.39}$$

for all $j \in \{1,\dots,n\}$, where $\boldsymbol{e}^{(j)}$ are the n orthonormal vectors of the canonical basis. The points (2.39) are used to sample the local function $\boldsymbol{u}^{(j)} := \boldsymbol{K}^{(i)}\boldsymbol{x}^{(j)} + \boldsymbol{l}^{(i)}$ where i is such that $\boldsymbol{x} \in \mathcal{R}^{(i)}$. Then, the local gain for the PWA function can be computed as the difference quotient

$$\boldsymbol{K}(\boldsymbol{x}) = \begin{pmatrix}\boldsymbol{u}^{(1)} - \boldsymbol{u}^*(\boldsymbol{x}) & \dots & \boldsymbol{u}^{(n)} - \boldsymbol{u}^*(\boldsymbol{x})\end{pmatrix} \tag{2.40}$$

where we define $\boldsymbol{u}^*(\boldsymbol{x}) := \boldsymbol{\pi}(\boldsymbol{x})$. For formulating $\boldsymbol{u}^*(\boldsymbol{x})$ as MIFP we can directly use (2.33). Thus, to formulate an MIFP for (2.40), it remains to introduce MI linear constraints for $\boldsymbol{u}^{(1)},\dots,\boldsymbol{u}^{(j)}$. A suitable MIFP can be formulated by expanding the constraints (2.33b)–(2.33f) by the MI linear constraints

$$\boldsymbol{H}\boldsymbol{U}^{(j)} + \boldsymbol{F}(\boldsymbol{x}+\boldsymbol{e}^{(j)}) + \boldsymbol{G}^\top\boldsymbol{\lambda}^{(j)} = \boldsymbol{0} \tag{2.41a}$$

$$\boldsymbol{E}(\boldsymbol{x}+\boldsymbol{e}^{(j)}) + \boldsymbol{d} - \boldsymbol{G}\boldsymbol{U}^{(j)} = \boldsymbol{r}^{(j)} \tag{2.41b}$$

$$-M(\boldsymbol{1} - \boldsymbol{\delta}^*(\boldsymbol{x})) \leq \boldsymbol{r}^{(j)} \leq M(\boldsymbol{1} - \boldsymbol{\delta}^*(\boldsymbol{x})) \tag{2.41c}$$

$$-M\boldsymbol{\delta}^*(\boldsymbol{x}) \leq \boldsymbol{\lambda}^{(j)} \leq M\boldsymbol{\delta}^*(\boldsymbol{x}), \tag{2.41d}$$

presented in [16, Thm. 5.3]. Then, the MIFP

$$\text{find } \hat{\boldsymbol{U}}_N^*, \boldsymbol{U}^{(1)},\dots,\boldsymbol{U}^{(n)}\boldsymbol{r}^*, \boldsymbol{r}^{(1)},\dots,\boldsymbol{r}^{(n)}, \boldsymbol{\lambda}^*, \boldsymbol{\lambda}^{(1)},\dots,\boldsymbol{\lambda}^{(n)}, \boldsymbol{\delta}^* \tag{2.42a}$$

$$\text{s.t.} \quad \text{(2.33b)–(2.33f) and (2.41)} \tag{2.42b}$$

is such that we have $\boldsymbol{S}\boldsymbol{U}^{(j)} = \boldsymbol{u}^{(j)}$ for all $j \in \{1,\dots,n\}$ and thus we can compute the local gain (2.40) by solving the MIFP (2.42). In Sections 3.2 and 8, we will detail how to use the MIFP (2.42) to compute the minimum Lipschitz constant (2.38) of the optimal control law by solving an MILP.

2.2.2 Model Predictive Control for Piecewise Affine Systems

MPC can also be used for the control of systems whose dynamics can be described by a PWA function $\boldsymbol{f}_{\text{PWA}} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ of the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\ &:= \begin{cases} \boldsymbol{A}^{(1)}\boldsymbol{x}(k) + \boldsymbol{B}^{(1)}\boldsymbol{u}(k) + \boldsymbol{p}^{(1)} & \text{if } \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \in \mathcal{P}^{(1)}, \\ \vdots & \\ \boldsymbol{A}^{(s)}\boldsymbol{x}(k) + \boldsymbol{B}^{(s)}\boldsymbol{u}(k) + \boldsymbol{p}^{(s)} & \text{if } \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \in \mathcal{P}^{(s)}, \end{cases} \end{aligned} \tag{2.43}$$

where $\mathcal{P}^{(i)}$ with $i \in \{1, \dots, s\}$ are polyhedral sets

$$\mathcal{P}^{(i)} := \{\boldsymbol{\xi} \in \mathbb{R}^{n+m} \mid \boldsymbol{H}^{(i)}\boldsymbol{\xi} \leq \boldsymbol{h}^{(i)}\} \tag{2.44}$$

with $\text{int}(\mathcal{P}^{(i)}) \cap \text{int}(\mathcal{P}^{(j)}) = \emptyset$ for all $i \neq j$ that partition the state and input space according to $\mathcal{X} \times \mathcal{U} = \cup_{i=1}^{s} \mathcal{P}^{(i)}$. Systems of the form (2.43) have emerged as an important system class since they are highly versatile. They can either be used to describe systems that are inherently PWA, as, e.g., mixed-logical dynamical (MLD) systems [64], or to approximate a large class of nonlinear systems with high accuracy [65]. Thus, formulating the OCP (2.30) for PWA systems increases the applicability of MPC. In principle, the extension to PWA systems is straightforward and results in the nonlinear OP

$$V_N(\boldsymbol{x}) := \min_{\substack{\hat{\boldsymbol{x}}(0),\dots,\hat{\boldsymbol{x}}(N) \\ \hat{\boldsymbol{u}}(0),\dots,\hat{\boldsymbol{u}}(N-1)}} \|\hat{\boldsymbol{x}}(N)\|_{\boldsymbol{P}}^2 + \sum_{\kappa=0}^{N-1} \|\hat{\boldsymbol{x}}(k)\|_{\boldsymbol{Q}}^2 + \|\hat{\boldsymbol{u}}(k)\|_{\boldsymbol{R}}^2 \tag{2.45a}$$

$$\text{s.t.} \quad \text{(2.30b), (2.30d), (2.30e),} \tag{2.45b}$$

$$\hat{\boldsymbol{x}}(\kappa+1) = \boldsymbol{f}_{\text{PWA}}(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)), \qquad \forall \kappa \in \{0, \dots, N-1\}. \tag{2.45c}$$

For reformulating the OCP (2.45) in a form that can be solved using MI solvers, we can use the methods discussed in Section 2.1.2.1, where we showed that constraints of the form (2.45c) can be described by MI linear constraints. Thus, we can replace the nonlinear constraint (2.45c) by MI linear constraints for representing PWA functions, i.e., constraints of the form (2.18) and (2.19). This

allows us to reformulate the OCP (2.45) for PWA systems as follows

$$V_N(\boldsymbol{x}) := \min_{\substack{\hat{\boldsymbol{x}}(0),\dots,\hat{\boldsymbol{x}}(N)\\ \hat{\boldsymbol{u}}(0),\dots,\hat{\boldsymbol{u}}(N-1)\\ \gamma(0),\dots,\gamma(N-1)}} \|\hat{\boldsymbol{x}}(N)\|_{\boldsymbol{P}}^2 + \sum_{\kappa=0}^{N-1} \|\hat{\boldsymbol{x}}(k)\|_{\boldsymbol{Q}}^2 + \|\hat{\boldsymbol{u}}(k)\|_{\boldsymbol{R}}^2 \tag{2.46a}$$

$$\text{s.t.} \quad \text{(2.30b), (2.30d), (2.30e),} \tag{2.46b}$$

$$\boldsymbol{H}^{(i)} \begin{pmatrix} \hat{\boldsymbol{x}}(k) \\ \hat{\boldsymbol{u}}(k) \end{pmatrix} \leq \boldsymbol{h}^{(i)} + \mathbf{1}M(1-\gamma_i(k)) \tag{2.46c}$$

$$\mathbf{1}^\top \gamma(k) = 1, \quad \gamma(k) \in \{0,1\}^s \tag{2.46d}$$

$$\begin{aligned} -\mathbf{1}M(1-\gamma_i(k)) &\leq \boldsymbol{A}^{(i)}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}^{(i)}\hat{\boldsymbol{u}}(k) + \boldsymbol{p}^{(i)} - \tilde{\boldsymbol{x}}^{(i)}(k+1) \\ &\leq \mathbf{1}M(1-\gamma_i(k)) \end{aligned} \tag{2.46e}$$

$$-\mathbf{1}M\gamma_i(k) \leq \tilde{\boldsymbol{x}}^{(i)}(k+1) \leq \mathbf{1}M\gamma_i(k) \tag{2.46f}$$

$$\hat{\boldsymbol{x}}(k+1) = \sum_{j=1}^{s} \tilde{\boldsymbol{x}}^{(j)}(k+1) \tag{2.46g}$$

for all $i \in \{1,\dots,s\}$ and all $k \in \{0,\dots,N-1\}$. As described in Section 2.1.2.1 the constraints (2.46c) and (2.46d) model the relation $\begin{pmatrix}\boldsymbol{x}^\top(k) & \boldsymbol{u}^\top(k)\end{pmatrix}^\top \in \mathcal{P}^{(i)} \Leftrightarrow \gamma_i = 1$ and the constraints (2.46e)–(2.46g) ensure $\gamma_i = 1 \Rightarrow \hat{\boldsymbol{x}}(k+1) = \boldsymbol{A}^{(i)}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}^{(i)}\hat{\boldsymbol{u}}(k)$. In summary we have $\begin{pmatrix}\boldsymbol{x}^\top(k) & \boldsymbol{u}^\top(k)\end{pmatrix}^\top \in \mathcal{P}^{(i)} \Rightarrow \hat{\boldsymbol{x}}(k+1) = \boldsymbol{A}^{(i)}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}^{(i)}\hat{\boldsymbol{u}}(k)$ and consequently $\hat{\boldsymbol{x}}(k+1) = \boldsymbol{f}_{\text{PWA}}(\hat{\boldsymbol{x}}(k), \hat{\boldsymbol{u}}(k))$. Thus, the constraints (2.46c)–(2.46g) can be used as a substitution for (2.45c) and allow for solving the OCP for PWA systems by solving an MIQP.

The complexity of MIQP (2.46) is mainly determined by the number of binary variables [66]. For the problem considered here, we have N binary vectors $\gamma(k)$ of dimension s, which results in a total number of binary variables of $\#_\gamma = sN$. Thus, solving the OCP (2.46) in every time step for the current state becomes even more intricate compared to the linear case described in Section 2.2.1, since for PWA systems, not only the number of constraints increases with the prediction horizon but also the number of binary variables. Meaning that approximating the optimal control law

$$\boldsymbol{\pi}_{\text{PWA}}(\boldsymbol{x}) := \hat{\boldsymbol{u}}(0) \tag{2.47}$$

described by the OCP (2.46) with a function that allows avoiding solving an MIQP promises an even higher speed-up than in the linear case. Therefore, we will consider in Sections 3.2, 10, and 11 NN-based approximations of (2.47) and investigate their closed-loop behavior to certify constraint satisfaction and stability.

2.3 Neural Networks

Neural networks (NN) have been successfully applied in many areas for the data-based approximations of functions as they are universal function approx-

imators [1, 67] that allow for the approximation of a large class of functions with arbitrary accuracy. They are also used in control engineering in several ways to accelerate the evaluation of complex control laws, e.g., MPC [6, 7] by approximating the PWA control law (2.34) or the PWQ OVF (2.36). Since in this thesis we primarily consider these two cases, i.e., approximations of PWA and PWQ functions, we will focus on standard deep feed-forward-NNs [68] defined by

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{f}^{(l+1)} \circ \boldsymbol{g}^{(l)} \circ \boldsymbol{f}^{(l)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{\xi}). \tag{2.48}$$

The NN (2.48) consists of a concatenation of preactivations $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{p_i w_i}$, activation functions $\boldsymbol{g}^{(i)} : \mathbb{R}^{p_i w_i} \to \mathbb{R}^{w_i}$ for all $i \in \{1, \ldots, l\}$, and a postactivation $\boldsymbol{f}^{(l+1)} : \mathbb{R}^{w_l} \to \mathbb{R}^{w_{l+1}}$. The parameters $l \in \mathbb{N}$ and $w_i \in \mathbb{N}$ specify the depth and the number of neurons in each layer, respectively. Finally, $p_i \in \mathbb{N}$ allows us to consider "multi-channel" preactivations as required for maxout (see [21]) and the value of $\max_{1 \leq i \leq l}\{w_i\}$ is referred to as the width of the NN. We consider here an affine pre- and postactivation

$$\boldsymbol{f}^{(i)}(\boldsymbol{y}^{(i-1)}) = \boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{v}^{(i)}, \tag{2.49}$$

for all $i \in \{1, \ldots, l+1\}$, 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 $\boldsymbol{y}^{(0)} := \boldsymbol{\xi}$. The set

$$\boldsymbol{\theta} := \{\boldsymbol{W}^{(1)}, \boldsymbol{v}^{(1)}, \ldots, \boldsymbol{W}^{(l+1)}, \boldsymbol{v}^{(l+1)}\}$$

is defined as the collection of all n_p parameters of the NN. We will 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. For the activation functions, we focus here on PWA activation functions in the form of the ReLU activation function

$$\boldsymbol{g}^{(i)}_{\text{ReLU}}(\boldsymbol{z}^{(i)}) = \max\left\{\boldsymbol{0}, \boldsymbol{z}^{(i)}\right\} := \begin{pmatrix} \max\left\{0, z_1^{(i)}\right\} \\ \vdots \\ \max\left\{0, z_{w_i}^{(i)}\right\} \end{pmatrix} \tag{2.50}$$

and the maxout activation function

$$\boldsymbol{g}^{(i)}_{\max}(\boldsymbol{z}^{(i)}) = \begin{pmatrix} \max\limits_{1 \leq j \leq p_i} \left\{z_j^{(i)}\right\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1 \leq j \leq p_i w_i} \left\{z_j^{(i)}\right\} \end{pmatrix}, \tag{2.51}$$

where we use the shorthand notation

$$\max_{1 \leq j \leq p_i} \left\{z_j^{(i)}\right\} := \max\left\{z_1^{(i)}, \ldots, z_{p_i}^{(i)}\right\}.$$

We will refer to the resulting NNs as ReLU NNs and maxout NNs, respectively. We consider this type of activation since we are interested in approximating

PWA functions, and for ReLU and maxout NNs, it is known that they are continuous PWA functions [25], [21], and thus are well-suited for approximating PWA functions. Moreover, in Section 3.1, and in Chapters 5, 6, 13, we will show how NNs (2.48) with PWA activation can be extended to represent continuous PWQ functions, which will allow an efficient approximation of PWQ functions and thus of the OVF (2.36) of MPC.

2.3.1 Training of Neural Networks

Once the topology of the NN, in terms of depth, width, and activation, has been chosen, the NN can be trained to minimize some sort of loss function for a given data set $\mathcal{D} := \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid y^{(i)} = f(\boldsymbol{x}^{(i)}) \forall i \in \{1, \dots, N_D\}\}$, where $f(\boldsymbol{x})$ is the function to be approximated by the NN. During the training of an NN, typically, an optimization problem (OP) of the form

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{N_D} \ell \left(y^{(k)} - \Phi(\boldsymbol{x}^{(k)}, \boldsymbol{\theta}) \right) \tag{2.52}$$

with a convex loss function $\ell : \mathbb{R} \to \mathbb{R}$ is considered. For regression tasks, i.e., the approximation of a function based on data samples, common choices for the loss function are the mean-squared error (MSE)

$$\ell_{\text{MSE}}(\boldsymbol{\theta}) := \frac{1}{N_D} \sum_{i=1}^{N_D} \left(y^{(i)} - \Phi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \right)^2 \tag{2.53}$$

or the mean-absolute error (MAE)

$$\ell_{\text{MAE}}(\boldsymbol{\theta}) := \frac{1}{N_D} \sum_{i=1}^{N_D} \left| y^{(i)} - \Phi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \right|. \tag{2.54}$$

Due to the nonlinear structure of the NN, the OP (2.52) is nonconvex, making it hard to find a global optimal solution. Thus, for the training of the NN (2.48), the nonconvex OP (2.52) is typically solved approximately using a gradient-based optimization algorithm, e.g., stochastic gradient descent (SGD), RMSProp, or Adam [31]. These methods are highly sensitive to the choice of hyperparameters, such as the learning rate and momentum. Moreover, they may even fail to converge for some choices of the hyperparameters and get stuck in poor local optima. Based on the results on tailored NN from Section 3.1.1, we will present in Section 3.1.2 a method that solves these problems by reformulating the OP 2.52 with (2.53) as MIQP, which can be solved globally.

2.3.2 Exact Function Representation by Neural Networks

The well-known universal approximation theorem, which states that NNs can approximate a large class of functions with arbitrary accuracy, has been stated

in different variants in the late 80s and early 90s [1, 67]. One of the first results was presented in [1, Thm. 3] where the authors showed that for every $\epsilon > 0$ there exists a NN with w_1 neurons in one layer such that

$$|f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| < \epsilon \tag{2.55}$$

holds. The number of neurons w_1 needed for (2.55) to hold depends on ϵ. The smaller ϵ is, the more neurons are needed. The result from [1] applies to NN with so-called sigmoidal activation. An activation $g(x)$ is sigmoidal if $\lim_{x\to\infty} g(x) = 1$ and $\lim_{x\to-\infty} g(x) = 0$. Since we can use the ReLU or maxout activation for constructing a sigmoidal function, e.g., $\max\{0, x\} - \max\{0, x - 1\}$, NN with ReLU and maxout activation are also universal approximators. However, the universal approximation theorems from [67, 69] do not specify the number of neurons needed for (2.55) to hold. Thus, the results can not be used directly to derive an NN topology that leads to a low error approximation. Therefore, the topology of a NN is often chosen by domain experts in a trial-and-error fashion due to the lack of design rules. For providing such rules, we will investigate in Section 3.1 under which conditions we can find NNs that can exactly represent certain functions, i.e., for which we have

$$|f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| = 0. \tag{2.56}$$

We will denote NNs for which we have (2.56) as tailored NNs. For continuous PWA functions $f : \Omega \to \mathbb{R}$ with $\Omega \subset \mathbb{R}^n$ of the form (2.34), NNs for which (2.56) holds can be constructed by decomposing the PWA function into two convex PWA functions $g : \Omega \to \mathbb{R}$ with

$$g(\boldsymbol{x}) := \begin{cases} \boldsymbol{K}_g^{(1)}\boldsymbol{x} + \boldsymbol{l}_g^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}_g^{(1)}, \\ \vdots & \\ \boldsymbol{K}_g^{(s)}\boldsymbol{x} + \boldsymbol{l}_g^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}_g^{(s)}, \end{cases}$$

and $h : \Omega \to \mathbb{R}$ with

$$h(\boldsymbol{x}) := \begin{cases} \boldsymbol{K}_h^{(1)}\boldsymbol{x} + \boldsymbol{l}_h^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}_h^{(1)}, \\ \vdots & \\ \boldsymbol{K}_h^{(s)}\boldsymbol{x} + \boldsymbol{l}_h^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}_h^{(s)} \end{cases} \tag{2.57}$$

such that

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x}) \tag{2.58}$$

holds [70]. Each of the convex PWA functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ can be represented as the maximum of their local functions $\boldsymbol{K}_g^{(i)}\boldsymbol{x} + \boldsymbol{l}_g^{(i)}$ and $\boldsymbol{K}_h^{(i)}\boldsymbol{x} + \boldsymbol{l}_h^{(i)}$, respectively. Thus, the difference (2.58) can be rewritten as

$$f(\boldsymbol{x}) = \max_{1\leq i\leq s} \left\{ \boldsymbol{K}_g^{(i)}\boldsymbol{x} + \boldsymbol{l}_g^{(i)} \right\} - \max_{1\leq i\leq s} \left\{ \boldsymbol{K}_h^{(i)}\boldsymbol{x} + \boldsymbol{l}_h^{(i)} \right\}. \tag{2.59}$$

The representation of PWA functions in the form (2.59) can be interpreted as a maxout NN with two neurons with $p_1 = s$ in one hidden layer. Thus, there exist maxout NNs that exactly represent continuous PWA functions, i.e., maxout NNs for which (2.56) holds for all $\boldsymbol{x} \in \Omega$. Moreover, the existence of ReLU NNs for which (2.56) holds has been shown in, e.g., [24, 25] by decomposing the convex PWA functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ into a concatenation of ReLU functions. Where in [25] the function (2.59) is decomposed by using the fact that the maximum of two values can be rewritten as

$$\begin{aligned}\max\left\{\boldsymbol{K}_1\boldsymbol{x}+l_1, \boldsymbol{K}_2\boldsymbol{x}+l_2\right\} &= \max\left\{0, (\boldsymbol{K}_1-\boldsymbol{K}_2)\boldsymbol{x}+l_1-l_2\right\}+\boldsymbol{K}_2\boldsymbol{x}+l_2 \\ &= \max\left\{0, (\boldsymbol{K}_1-\boldsymbol{K}_2)\boldsymbol{x}+l_1-l_2\right\}+\max\left\{0, \boldsymbol{K}_2\boldsymbol{x}+l_2\right\} \\ &\quad -\max\left\{0, -\boldsymbol{K}_2\boldsymbol{x}-l_2\right\}, \end{aligned} \tag{2.60}$$

which is a ReLU NN with one hidden layer and three neurons. Then, by applying (2.60) $\log_2(s)$ times, the maximum of s values can be computed by a ReLU NN with $\log_2(s)$ layers, and thus (2.59) is a difference of two ReLU NNs and consequently itself a ReLU NN [25]. Now, since both ReLU NNs and maxout NNs can exactly represent continuous PWA functions, they are well-suited for learning-based control where a NN approximates the PWA optimal control law (2.34). Moreover, the connection between PWA functions and NNs can be used to derive bounds on the width and depth a NN should have to represent a PWA function. Thus, the connection allows the derivation of design rules for NNs that approximate PWA functions. As we will detail in the subsequent section, approximations of the optimal control law are not the only use case of NNs in learning-based MPC. It is also possible to use NNs to approximate the continuous and convex PWQ OVF (2.36). For this case, the previously described results on the representation of PWA functions by NNs can not be used to derive tailored NNs. However, we will show in Section 3.1 how the representation (2.58) can be extended to enable an exact representation of continuous PWQ functions of the form (2.36) with polyhedral domain partition. This extension enables the construction of tailored maxout and ReLU NNs for approximating the OVF in linear MPC, which is relevant, e.g., for ADP and reinforcement learning.

2.3.3 Mixed-Integer Linear Constraints for Neural Networks

For NNs with the ReLU activation function, we can use the methods described in Section 2.1.2.1 for modeling the ReLU activation by MI linear constraints for formulating a set of MI linear constraints that model a complete deep feed-forward NN of the form (2.48). According to [16] the MI linear constraints

$$-M(1-\boldsymbol{\beta}_j^{(i)}) \leq \boldsymbol{W}_j^{(i)}\boldsymbol{q}^{(i-1)} + \boldsymbol{v}_j^i \leq M\boldsymbol{\beta}_j^{(i)} + (1-\boldsymbol{\beta}_j^{(i)})\epsilon \tag{2.61a}$$

$$-M\boldsymbol{\beta}_j^{(i)} \leq \boldsymbol{q}_j^{(i)} \leq M\boldsymbol{\beta}_j^{(i)}, \tag{2.61b}$$

$$-M(1-\boldsymbol{\beta}_j^{(i)}) \leq \boldsymbol{q}_j^{(i)} - \boldsymbol{W}_j^{(i)}\boldsymbol{q}^{(i-1)} - \boldsymbol{v}_j^i \leq M(1-\boldsymbol{\beta}_j^{(i)}), \tag{2.61c}$$

$$\boldsymbol{q}^{(0)} = \boldsymbol{x}, \tag{2.61d}$$

$$\boldsymbol{\beta}_j^{(i)} \in \{0,1\} \tag{2.61e}$$

for all $i \in \{1,\dots,l\}$ and for all $j \in \{1,\dots,w_i\}$ can be used to formulate the MIFP

$$\text{find}\, \boldsymbol{q}^{(0)},\dots,\boldsymbol{q}^{(l)},\boldsymbol{\beta}^{(1)},\dots,\boldsymbol{\beta}^{(l)} \tag{2.62a}$$

$$\text{s.t.} \quad (2.61) \tag{2.62b}$$

which is such that $\boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} = \boldsymbol{\Phi}(\boldsymbol{q}^{(0)}) = \boldsymbol{\Phi}(\boldsymbol{x})$ holds. Furthermore, with the binary variables of the MIFP arranged in the diagonal matrix

$$\Delta^{(i)} := \begin{pmatrix} \boldsymbol{\beta}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & & \vdots \\ \vdots & & & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{\beta}_{w_i} \end{pmatrix},$$

we can compute the local gain of an NN as

$$\boldsymbol{K}_\Phi(\boldsymbol{x}) = \boldsymbol{W}^{(l+1)} \prod_{i=1}^{l} \Delta^{(i)} \boldsymbol{W}^{(i)}. \tag{2.63}$$

The MI linear constraints (2.61) describing a ReLU and the local gain (2.63) can be used in combination with the MI linear constraints (2.42) for the optimal control law of MPC to analyze the closed-loop behavior of NN-based controllers using MILP and MIQP. We will discuss this in more detail in Section 2.4.

2.3.4 Privacy-Preserving Evaluation of Neural Networks

For some applications, it is desirable to protect the input data of the NN from the entity that evaluates the NN. For example, in cases where sensitive data, such as healthcare data or measurements of safety-critical systems, is processed by a third party, e.g., a cloud computing provider. In these cases, HE may be used to evaluate an NN on encrypted input data without intermediate decryption. However, HE-shemes typically only allow for encrypted additions [43] and a limited number of encrypted multiplications [42]. In the following, we denote the encryption of an input z using an unspecified HE-scheme as $\text{Enc}(z)$.

We will not focus on the cryptographic details. An overview of suitable HE-schemes can be found in [71]. For our case, it suffices to assume that the HE-scheme supports encrypted additions [43] via an operation $\oplus$ according to

$$\text{Enc}(z_1 + z_2) = \text{Enc}(z_1) \oplus \text{Enc}(z_2) \tag{2.64}$$

and a limited number of multiplications [42] via an operation $\otimes$ according to

$$\text{Enc}(z_1 z_2) = \text{Enc}(z_1) \otimes \text{Enc}(z_2). \tag{2.65}$$

Thus, additions and multiplications can be performed on encrypted data without prior decryption, allowing for privacy-preserving computations. However, due to the limitation to additions and multiplications, HE can not directly be used to evaluate NNs with non-linear activation functions, such as ReLU or maxout. A workaround is to first approximate the non-linear ReLU activation by a polynomial, which can be evaluated using only additions and multiplications, and then replace the ReLU activation in the NN with the polynomial approximation [72] of the form

$$p(x) := c_0 + c_1 x + c_2 x^2 + \cdots + x^r c_r. \tag{2.66}$$

Typically, the number of consecutive encrypted multiplications, or multiplicative depth d, supported by an HE scheme is limited. Consequently, the degree of polynomials that can be evaluated in an encrypted fashion is limited and given by $r = 2^d - 1$ [44]. The limitation arises since $d - 1$ multiplications are required to evaluate the monomials in (2.66) up to the order of $2^d - 1$. The last remaining multiplication is needed to multiply the monomial x^i with the corresponding coefficients c_i. For example, a monomial of degree 7 can be evaluated using 3 consecutive multiplications by $c_7 x^7 = (((xx)(xx))((xx)(c_7 x)))$, where the brackets indicate the multiplicative depth.

Since the accuracy of polynomial approximations increases with the degree, the limited multiplicative depth also restricts the accuracy of the polynomials that are used to approximate the ReLU activation. We will present in Section 12 and in our paper [P8] a modification of standard methods that allows us to increase the degree of the polynomials by one to $r = 2^d$ without increasing the multiplicative depth. Thus, we enable a more accurate polynomial approximation of non-linear activation functions for privacy-preserving NNs. These more accurate privacy-preserving NNs can also be used in control to approximate the control law (2.31) and subsequently evaluate it in an encrypted fashion.

2.4 Learning-Based Control

As this thesis focuses on the design and analysis of NNs, the following will mainly address learning-based control methods using NNs. NNs are most commonly used in control to either approximate a given computationally demanding control law, e.g., the solution of the OCP (2.30), or to approximate

OVFs, as the OVF (2.36) of linear MPC. The approaches are also referred to as an approximation in the policy space and in the value space, respectively [3].

2.4.1 Approximations of Control Laws

To avoid the need to solve the OCP (2.30) at every time step in order to compute the optimal control input $\hat{\boldsymbol{u}}^*$, NNs $\boldsymbol{\Phi}(\boldsymbol{x})$ can be used to approximate the optimal control law $\boldsymbol{\pi}(\boldsymbol{x})$, as demonstrated in, e.g., [6, 16]. In this case, determining $\hat{\boldsymbol{u}}^*$ merely requires the evaluation of an NN rather than the repeated solution of an OP, resulting in a significant computational speed-up [7]. This, in turn, enables the implementation of MPC on hardware platforms with limited computational resources. Furthermore, by replacing the activation functions in the NN with polynomial approximations as defined in (2.66), the NN can be utilized for privacy-preserving evaluation of the control law, as discussed in Section 2.3.4. Hence, the use of NNs for approximating control laws offers multiple advantages, combining computational efficiency with the potential for privacy-preserving evaluation.

NNs for an approximation in the policy space are typically trained on samples of control laws as described in Section 2.3.1 by using a gradient-based optimization algorithm to choose parameters $\boldsymbol{\theta}$ that minimize

$$\ell_{\text{MSE}}(\boldsymbol{\theta}) = \frac{1}{N_D} \sum_{i=1}^{N_D} \left(\boldsymbol{\pi}(\boldsymbol{x}^{(i)}) - \Phi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \right)^2$$

for the samples $\left((\boldsymbol{x}^{(i)}, \boldsymbol{\pi}(\boldsymbol{x}^{(i)})), \dots, (\boldsymbol{x}^{(N_D)}, \boldsymbol{\pi}(\boldsymbol{x}^{(N_D)})) \right)$. Since the approximation error

$$\boldsymbol{e}(\boldsymbol{x}) := \boldsymbol{\pi}(\boldsymbol{x}) - \Phi(\boldsymbol{x})$$

is only evaluated for the points in the data set and the commonly used optimization algorithm, e.g., SGD, RMSProp, and Adam [31] typically do not find a global optimal solution, i.e., we do not have $\boldsymbol{e}(\boldsymbol{x}) = \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathcal{F}$. Thus, there will be a deviation between the optimal control law $\boldsymbol{\pi}(\boldsymbol{x})$ and the output of the NN $\boldsymbol{\Phi}(\boldsymbol{x})$. Due to this deviation, the guarantees of the original control law of MPC, such as stability, constraint satisfaction, and recursive feasibility, are typically lost for the NN-based approximation of the control law.

2.4.1.1 Analysis of NN-based Controllers

To recover the guarantees of the original controller, different methods have been proposed to analyze NN-based controllers for linear systems. We will discuss methods based on the approximation error with respect to the optimal control law and methods based on reachability analysis.

Approximation Error Analysis: As discussed in Sections 2.2.1 and 2.3, the optimal control law of linear MPC and NNs with ReLU and maxout activations

are continuous PWA functions. Because of the common PWA structure, ReLU and maxout NNs are the natural choice for approximating the optimal control law of linear MPC. Indeed, in most of the recent articles, as in, e.g., [14, 73], precisely these NNs are used. Moreover, the link between the considered NNs and the optimal control law, given by the common PWA structure, enables the use of MIP to analyze the approximation error of the NNs and the closed-loop behavior of linear systems controlled by NNs. Using the MI linear constraints (2.33b)–(2.33f) for the OCP (2.30) and the MI linear constraints (2.61) for ReLU NNs from Section 2.2.2 and Section 2.3.3 we can formulate the MIP

$$\max_{\substack{\hat{\boldsymbol{U}}_N^*,\boldsymbol{r}^*,\boldsymbol{\lambda}^*,\boldsymbol{\delta}^* \\ \boldsymbol{q}^{(0)},\dots,\boldsymbol{q}^{(l)},\boldsymbol{\beta}^{(1)},\dots,\boldsymbol{\beta}^{(l)}}} \quad \left\| \boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} - \boldsymbol{S}\hat{\boldsymbol{U}}_N^* \right\|_p \tag{2.67a}$$

$$\text{s.t.} \quad \text{(2.33b)–(2.33f), (2.61), and } \boldsymbol{x} \in \mathcal{F}. \tag{2.67b}$$

For $p \in \{1, \infty\}$, the cost function of the MIP (2.67) can be reformulated using MI linear constraints. This turns the MIP into an MILP [16, Thm. 6.1]. Furthermore, as described in the Sections 2.2.1 and 2.3.3 the MI linear constraints in the MIP (2.67) ensure $\boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} = \boldsymbol{\Phi}(\boldsymbol{x})$ and $\boldsymbol{S}\hat{\boldsymbol{U}}_N^* = \boldsymbol{\pi}(\boldsymbol{x})$. We thus have $\boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} - \boldsymbol{S}\hat{\boldsymbol{U}}_N^* = \boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{\pi}(\boldsymbol{x}) = \boldsymbol{e}(\boldsymbol{x})$ and can reformulate the MILP (2.67) as

$$\max_{\boldsymbol{x}} \left\| \boldsymbol{e}(\boldsymbol{x}) \right\|_p \tag{2.68a}$$

$$\text{s.t. } \boldsymbol{x} \in \mathcal{F}. \tag{2.68b}$$

Hence, the maximum approximation error $e_{\max} := \max_{\boldsymbol{x}\in\mathcal{F}} \left\| \boldsymbol{e}(\boldsymbol{x}) \right\|_p$ (cf. (2.68)) can be computed by solving the MILP (2.67). If the maximum approximation error e_p and the minimum Lipschitz constant of the error

$$\mathcal{L}_p(\mathcal{T}) := \max_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{T},\boldsymbol{x}\neq\boldsymbol{y}} \frac{\left\| \boldsymbol{e}(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{y}) \right\|_p}{\left\| \boldsymbol{x} - \boldsymbol{y} \right\|_p}$$

on a terminal set $\mathcal{T}$ are below certain threshold values, then stability of the closed-loop system $\boldsymbol{x}(k+1) = \boldsymbol{f}_{\mathrm{L}}(\boldsymbol{x}(k), \boldsymbol{\Phi}(\boldsymbol{x}(k)))$ can be certified [16, Thm. 3.4]. Since both the optimal control law $\boldsymbol{\pi}(\boldsymbol{x})$ and the NN $\Phi(\boldsymbol{x})$ are PWA functions, the approximation error $\boldsymbol{e}(\boldsymbol{x})$ is a PWA function as well, and thus the minimum Lipschitz constant is

$$\mathcal{L}_p(\mathcal{T}) = \max_{\boldsymbol{x}\in\mathcal{T}} \left\| \boldsymbol{K}(\boldsymbol{x}) - \boldsymbol{K}_\Phi(\boldsymbol{x}) \right\|_p \tag{2.69}$$

according to [63, Prop. 3.4]. Now, with the MI formulation (2.42) for the local gain $\boldsymbol{K}(\boldsymbol{x})$ of the optimal control law and the MI formulation for local gain (2.63) of the NN, the minimum Lipschitz constant (2.69) can also be computed be solving an MILP [16, Thm. 6.1].

Reachability Analysis: The drawback of the previously described method for certifying stability is that it fails to certify stability if the approximation error is too large, even if the NN provides a stabilizing controller. This may happen in cases where the parameters of the OCP are not known to the party that uses and analyzes the NN-based controller. Thus, methods based on the approximation error can only be used if the original controller that is approximated, in our case, the optimal control law of linear MPC $\boldsymbol{\pi}(\boldsymbol{x})$, is known and can be represented by MI linear constraints. In cases where this is not possible, methods based on reachability analysis are a promising alternative. These methods do not require knowledge of the control law that is approximated by the NN. In fact, as long as maxout or ReLU NNs are used to control linear systems, the NNs can be trained using any method on data from arbitrary controllers. The main idea behind these methods is to describe the closed-loop system consisting of a linear system and an NN-based controller as an MIFP [38]. For linear systems controlled by ReLU NNs, we can combine the MI linear constraints (2.61) with the linear equation (2.32) describing the linear system dynamics to formulate the MILP

$$\begin{aligned}
\max_{\substack{\boldsymbol{x}(0),\dots,\boldsymbol{x}(K),\boldsymbol{\eta}\\ \boldsymbol{q}^{(0)}(0),\dots,\boldsymbol{q}^{(l)}(0),\dots,\boldsymbol{q}^{(0)}(K-1),\dots,\boldsymbol{q}^{(l)}(K-1)\\ \boldsymbol{\beta}^{(1)}(0),\dots,\boldsymbol{\beta}^{(l)}(0),\dots,\boldsymbol{\beta}^{(1)}(K-1),\dots,\boldsymbol{\beta}^{(l)}(K-1)}} \quad & \boldsymbol{\eta}^\top \boldsymbol{x}(K) && \text{(2.70a)}\\
\text{s.t.} \quad & (2.61), && \text{(2.70b)}\\
& \boldsymbol{q}^{(0)}(k) = \boldsymbol{x}(k), && \text{(2.70c)}\\
& \boldsymbol{x}(0) \in \mathcal{F}, && \text{(2.70d)}\\
& \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}(\boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)}(k) + \boldsymbol{v}^{(l+1)}). && \text{(2.70e)}
\end{aligned}$$

for all $k \in \{0,\dots,K-1\}$. As described in Section 2.3.3 the constraints (2.61) ensure $\boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)}(k) + \boldsymbol{v}^{(l+1)} = \boldsymbol{\Phi}(\boldsymbol{q}^{(0)}(k))$ for all $k \in \{0,\dots,K-1\}$ and thus with (2.70c) and (2.70e) the MILP (2.70) is equivalent to

$$\begin{aligned}
c_K^*(\boldsymbol{\eta}) := \max_{\boldsymbol{x}(0),\dots,\boldsymbol{x}(K),\boldsymbol{\eta}} \quad & \boldsymbol{\eta}^\top \boldsymbol{x}(K) && \text{(2.71a)}\\
\text{s.t. } & \boldsymbol{x}(0) \in \mathcal{F}, && \text{(2.71b)}\\
& \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{\Phi}(\boldsymbol{x}(k)). && \text{(2.71c)}
\end{aligned}$$

The optimal value $c_K^*(\boldsymbol{\eta})$ can be identified as the value of the support function of the reachable sets

$$\begin{aligned}
\mathcal{R}_0(\mathcal{F}) &:= \mathcal{F}\\
\mathcal{R}_{k+1}(\mathcal{F}) &:= \{\boldsymbol{x} \mid \boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{\Phi}(\boldsymbol{x}(k)), \boldsymbol{x}(k) \in \mathcal{R}_k(\mathcal{F})\}
\end{aligned}$$

of the closed-loop system $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{\Phi}(\boldsymbol{x}(k))$ (see, e.g., [P6, Lem. 2]), where the support function $h_{\mathcal{R}_k(\mathcal{F})}(\boldsymbol{\eta}) : \mathbb{R}^n \to \mathbb{R}$ of the set $\mathcal{R}_k(\mathcal{F})$ is defined as

$$h_{\mathcal{R}_k(\mathcal{F})}(\boldsymbol{\eta}) := \max_{\boldsymbol{x} \in \mathcal{R}_k(\mathcal{F})} \boldsymbol{\eta}^\top \boldsymbol{x} = c_K^*(\boldsymbol{\eta}). \tag{2.72}$$

Intuitively, the support function $h_{\mathcal{R}_k(\mathcal{F})}(\boldsymbol{\eta})$ of the set $\mathcal{R}_k(\mathcal{F})$ evaluated for a vector $\boldsymbol{\eta}$ can be interpreted as the size of the set in the direction specified by the normal vector $\boldsymbol{\eta}$. Based on the reachable sets, we can analyze the closed-loop behavior and certify boundedness and constraint satisfaction [74]. If we have

$$\mathcal{R}_1(\mathcal{F}) \subseteq \mathcal{F} \tag{2.73}$$

for a set $\mathcal{F} \subseteq \mathcal{X}$ then the set $\mathcal{F}$ is positively invariant (PI) [75] and thus $\mathcal{R}_k(\mathcal{F}) \subseteq \mathcal{X}$ holds for all $k \in \mathbb{N}$, i.e., guaranteed satisfaction of the state constraints. Moreover, if in addition there exists a $k^* \in \mathbb{N}$ with

$$\mathcal{R}_k^*(\mathcal{F}) \subseteq \mathcal{T} \tag{2.74}$$

for some small terminal set $\mathcal{T}$, then $\mathcal{R}_k(\mathcal{F}) \subseteq \mathcal{T}$ for all $k \geq k^*$ and thus trajectories of the closed-loop system stay in the set $\mathcal{T}$ once they enter this set. For a polyhedral $\mathcal{F} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_\mathcal{F}\boldsymbol{x} \leq \boldsymbol{h}_\mathcal{F}\}$ and $\mathcal{T} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_\mathcal{T}\boldsymbol{x} \leq \boldsymbol{h}_\mathcal{T}\}$, the relations (2.73) and (2.74) can be verified based on the support function of the reachable set using

$$\boldsymbol{h}_{\mathcal{R}_1(\mathcal{F})}(\boldsymbol{H}_\mathcal{F}) \leq \boldsymbol{h}_\mathcal{F}$$

and

$$\boldsymbol{h}_{\mathcal{R}_{k^*}(\mathcal{F})}(\boldsymbol{H}_\mathcal{T}) \leq \boldsymbol{h}_\mathcal{T},$$

respectively. Thus, we can check whether (2.73) and (2.74) hold by solving the MILP (2.70) for the support function of the reachable set. Building on this foundation, we will present our results on reachability analysis for NN-based controllers in Sections 3.2.3, 10, and 11, which enable the verification of stability and constraint satisfaction. Moreover, we will present methods for computing suitable sets $\mathcal{F}$ and $\mathcal{T}$.

2.4.2 Approximations of Optimal Value Functions

Besides the possibility described in Section 2.4.1 of using NNs for approximating the optimal control law of linear MPC, NNs can also be used for approximating the OVF (2.36) or variants of it. Examples are model-based RL or ADP [3, 27, 76, 77]. In these methods, the approximated OVF $\tilde{V}_{N-1}(\boldsymbol{x}) \approx V_{N-1}(\boldsymbol{x})$ can be used to reduce the prediction horizon N in (2.30) to one and thus to reduce the computational complexity without significantly decreasing the control performance [8, 27]. The control input can be derived based on the approximated OVF by solving

$$\begin{aligned} \boldsymbol{u}_V^*(0) := \arg\min_{\boldsymbol{u}} \ & \|\boldsymbol{x}\|_\boldsymbol{Q}^2 + \|\boldsymbol{u}\|_\boldsymbol{R}^2 + \tilde{V}_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{x}(0) \\ & \boldsymbol{u} \in \mathcal{U} \\ & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \in \mathcal{S} \end{aligned} \tag{2.75}$$

where $\mathcal{S}$ is a control invariant set that can be computed using methods from [78]. The OVF is approximated in ADP based on samples $(\boldsymbol{x}, V_{N-1}(\boldsymbol{x}))$ for which the OCP (2.30) has been solved or in model-based RL iteratively by, e.g., temporal difference learning [11, 79]. However, in both methods, model knowledge in terms of the system parameters $\boldsymbol{A}$ and $\boldsymbol{B}$ is needed to compute the control input $\boldsymbol{u}_V^*(0)$ (2.75). Alternatively, in Q-learning, the so-called Q-function

$$Q_N(\boldsymbol{x}, \boldsymbol{u}) := \|\boldsymbol{x}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}\|_{\boldsymbol{R}}^2 + V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \tag{2.76}$$

is approximated by a NN $\tilde{Q}_N(\boldsymbol{x}, \boldsymbol{u}) \approx Q_N(\boldsymbol{x}, \boldsymbol{u})$ and can be used to derive a control input model-free by solving

$$\begin{aligned} \boldsymbol{u}_Q^*(0) := \arg\min_{\boldsymbol{u}} \ & \tilde{Q}_N(\boldsymbol{x}, \boldsymbol{u}) \\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{x}(0) \\ & \boldsymbol{u} \in \mathcal{U} \\ & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \in \mathcal{S}. \end{aligned} \tag{2.77}$$

The Q-function is typically learned model-free with temporal difference learning [79] based on state measurements.

If these methods are applied to MPC for linear systems with quadratic cost and polyhedral constraints, then the function to be approximated is no longer the continuous PWA optimal control law as in Section 2.4.1 but the OVF (2.36) and the Q-function (2.76), which are both continuous PWQ functions with polyhedral domain partitions (cf. (2.36)). Thus, using the PWA NNs, as for the approximation of the optimal control law in Section 2.4.1, is no longer the ideal choice. Ideally, the structure of the NN should harmonize with the functions to be approximated, i.e., the NN should be a continuous PWQ function. Nevertheless, PWQ NNs have rarely been considered in this context. There are some approaches that use PWQ NNs to approximate OVFs [27]. However, these approaches do not include an analysis of the representation capabilities of the NN. Thus, there may exist PWQ functions that can not be represented by these NNs. In Section 3.1, we will derive NNs that can exactly represent arbitrary continuous PWQ functions. This allows us to connect NNs and the OVF of linear MPC, similarly to the connection between PWA functions and NN used in Section 2.4.1 for the approximation of the optimal control law.

Chapter 3

Article Summaries and Discussion

In the previous two chapters, we have seen that the application of NNs in control faces challenges unique to specific use cases, and for this reason, they are often insufficiently addressed in the existing literature. As outlined in Chapter 1, these challenges can broadly be grouped into two categories, namely the design of tailored NN topologies and the subsequent analysis of these NNs as a controller in a closed-loop system. The two categories differ primarily in the stage of implementation at which they arise. The design problem is typically considered before the first training of a NN, as it concerns the question of which topologies are suitable for the intended task. The analysis, by contrast, becomes relevant after the training procedure has been completed, when the performance of the network must be evaluated with respect to specific measures that are important for its use in control.

In this chapter, both challenges are investigated in more detail, and the results of our articles included in Part II are presented. The first part of the discussion, given in Section 3.1, is devoted to design problems. There, we develop NNs that allow for the exact representation of continuous PWQ functions defined on polyhedral domain partitions. Such functions play an essential role in RL for control. Building on the developed NNs, we further introduce in Section 3.1.2 a tailored training method that enables NNs to be trained to global optimality. A further aspect of the design problem is addressed in Section 2.3.4, where we consider the construction of tailored polynomial approximations of NNs for efficient privacy-preserving evaluations, as introduced in Section 2.3.4.

The second part of the chapter, presented in Section 3.2, shifts the focus to the analysis problem. Here, we present results from our articles [P3, P4, P6, P7], which demonstrate how linear and PWA systems controlled by NNs can be rigorously analyzed to certify properties such as constraint satisfaction, boundedness, convergence, and stability.

3.1 Tailored Neural Networks

This section is devoted to the design of several methods tailored to specific use cases relevant to control. More specifically, we will present results on the exact representation of continuous PWQ functions with polyhedral domain parti-

tions by NNs, on training NNs to global optimality, and finally on polynomial approximations of NNs for efficient privacy-preserving evaluation. All results are based on our published articles [P1, P2, P5, P8, P9] which are included in Part II.

3.1.1 Representation of Piecewise Quadratic Functions by NNs

Motivated by the success of tailored NNs that enable the exact representation of PWA functions in the approximation of PWA control laws for linear MPC [7, 26], we now investigate the problem of exactly representing PWQ functions. It is well known that PWA functions can be represented exactly by NNs, and this connection has provided valuable insights into the design of topologies well suited for the approximation of control laws. As discussed in Section 2.4.2, however, when neural networks are used to approximate OVFs or Q-functions, the underlying functions are PWQ. For such functions, no general connection to NNs has been established so far that could systematically guide their design. The following section addresses this gap and presents our results on the exact representation of PWQ functions by NNs.

3.1.1.1 Problem Overview

Topologies of NNs that allow an exact representation of continuous PWA functions $f(\boldsymbol{x})$ often build on the decomposition of these functions into a difference of two convex PWA functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$ [70]. Each of these functions can be represented as the maximum of their local functions, as specified by (2.59) and shown in Figure 3.1. In Figure 3.1, an exemplary continuous PWA function is decomposed into continuous and convex PWA functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$, shown in Figure 3.1.(b) and Figure 3.1.(c), respectively. The blue vertical line in Figure 3.1.(b) illustrates the evaluation of all local functions $g^{(i)}(x)$ of $g(x)$ for $x = -0.15$. Since the function is convex and PWA we have $\max\{g^{(1)}(-0.15), g^{(2)}(-0.15), g^{(3)}(-0.15)\} = \max\{-0.5 \times (-0.15) + 0.5, 1, 0.5 \times (-0.15) + 0.5\} = \max\{0.575, 1, 0.425\} = 1 = g(-0.15)$. Thus $g(x)$ and similarly $h(x)$ can be evaluated by evaluating all local functions and subsequently taking the maximum. The decomposition of the PWA function $f(x)$ in the form

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x}) = \max_{1 \leq i \leq s} \left\{ g^{(i)}(\boldsymbol{x}) \right\} - \max_{1 \leq i \leq s} \left\{ h^{(i)}(\boldsymbol{x}) \right\} \tag{3.1}$$

can be directly represented by a maxout NN [21]. Consequently, the optimal control law of linear MPC (2.31) can also be represented by a maxout NN. Moreover, results on the connection of maxout and ReLU NNs [24, 25] can be used to derive ReLU NNs for an exact representation of PWA functions and thus of the optimal control law of linear MPC.

For continuous PWQ functions, representations by NNs have rarely been considered, although such representations are crucial for RL methods in con-

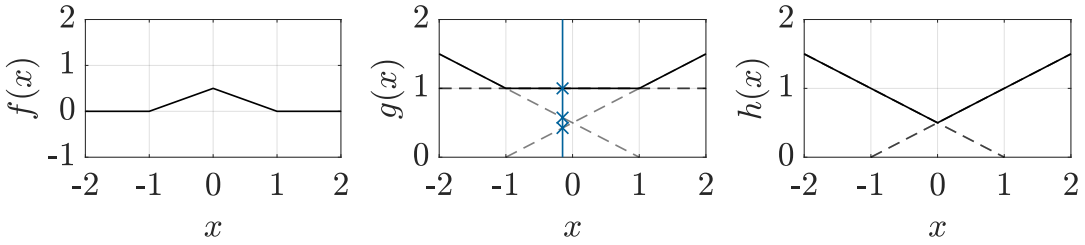


Figure 3.1. PWA function $f(\boldsymbol{x})$ with 4 regions in (a) and the decomposition of $f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x})$ into the difference of two convex PWA functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$. The extension of the local functions outside their regions is indicated by the dashed gray lines in (b) and (c).

trol where the OVF (2.36) or the Q-function (2.76) is approximated, as discussed in Section 2.4.2. In principle, continuous PWQ functions can also be decomposed into a difference of two convex PWQ functions [80]. Figure 3.2 shows such a decomposition of the continuous PWQ function $f(x)$ into the convex PWQ functions $g(x)$ and $h(x)$. However, unlike in the PWA case shown in Figure 3.1, the convex PWQ functions can not be represented as the maximum of their local functions.

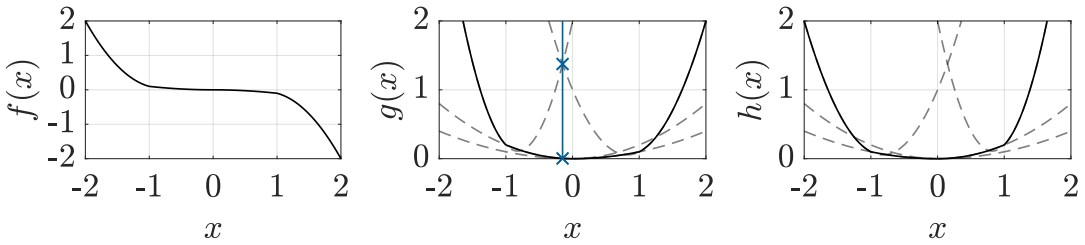


Figure 3.2. PWQ function $f(\boldsymbol{x})$ with 4 regions in (a) and the decomposition of $f(\boldsymbol{x}) = g(\boldsymbol{x}) - h(\boldsymbol{x})$ into the difference of two convex PWQ functions $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$. The extension of the local functions outside their regions is indicated by the dashed gray lines in (b) and (c).

In Figure 3.2, the local functions of $g(x)$ are evaluated for $x = -0.15$ which results in

$$\begin{aligned}
&\max\{g^{(1)}(-0.15), g^{(2)}(-0.15), g^{(3)}(-0.15)\} \\
&= \max\{2.8 \times (-0.15)^2 + 4.6 \times (-0.15) + 2, 0.2 \times (-0.15)^2, \\
&\qquad 0.1 \times (-0.15)^2, 1.4 \times (-0.15)^2 - 2.3 \times (-0.15) + 1\} \\
&= \max\{1.373, 4.5 \times 10^{-3}, 2.25 \times 10^{-3}, 1.3765\} = 1.3765.
\end{aligned}$$

Furthermore for $g(x)$ we have $g(-0.15) = 4.5 \times 10^{-3}$ and thus

$$\max\{g^{(1)}(-0.15), g^{(2)}(-0.15), g^{(3)}(-0.15)\} \neq g(-0.15),$$

i.e., the convex PWQ function can not be represented as the maximum of its local quadratic functions. Consequently, we can not use the results on the representation of PWA functions for deriving NNs for the representation of PWQ functions. Therefore, our Articles [P1, P2, P9] explore different methods for representing continuous PWQ functions with polyhedral domain partitions by NNs. The main question that is investigated in these articles is which NNs can exactly represent continuous PWQ functions $\varphi(\boldsymbol{x}) : \Omega \to \mathbb{R}$ of the form

$$\varphi(\boldsymbol{x}) := \begin{cases} \varphi^{(1)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \varphi^{(s)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{3.2}$$

with local quadratic functions

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

and 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

$$\mathcal{R}^{(i)} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}^{(i)} \boldsymbol{x} \leq \boldsymbol{h}^{(i)}\} \tag{3.3}$$

with pairwise disjoint interiors. We further define the set of indices of neighboring regions as

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

and denote two regions $\mathcal{R}^{(i)}$ and $\mathcal{R}^{(j)}$ as neighboring if $(i,j) \in \mathcal{N}$. The hyperplane separating two neighboring regions $(i,j) \in \mathcal{N}$ is defined as

$$\mathcal{B}^{(i,j)} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}^{(i,j)} \boldsymbol{x} = h^{(i,j)}\}, \tag{3.4}$$

where $\boldsymbol{H}^{(i,j)}$ and $h^{(i,j)}$ are rows of $\boldsymbol{H}^{(i)}$ and $\boldsymbol{h}^{(i)}$, respectively for which we have $\boldsymbol{H}^{(i)} \boldsymbol{x} = \boldsymbol{h}^{(i)}$ if $\boldsymbol{x} \in \mathcal{R}^{(i)} \cap \mathcal{R}^{(j)}$. Note that OVFs (2.36) and Q-functions (2.76) for linear MPC are of this type, and thus the NNs we will derive can be used for representing these functions.

3.1.1.2 Summary of Articles [P1, P2, P9]

We will now summarize our results from [P1, P2, P9] on the representation of PWQ functions by NNs. Starting with some special cases and eventually turning to the general case of representing arbitrary continuous PWQ with polyhedral domain partitions by NNs.

Representation of one-dimensional PWQ Functions by ReLU NNs: Our first attempts at representing continuous PWQ functions by NNs are described in our article [P1]. In this article, we considered the problem for the simplified

case of representing a one-dimensional continuous PWQ function, i.e., a function with a domain $\Omega \subset \mathbb{R}$, by a ReLU NN. In this case the regions (3.3) are closed and bounded intervals of the form

$$\mathcal{R}^{(i)} := \{x \in \mathbb{R} \mid \underline{x}^{(i)} \leq x \leq \overline{x}^{(i)}\}. \tag{3.5}$$

with $\underline{x}^{(i)} < \overline{x}^{(i)} = \underline{x}^{(i+1)}$ for all $i \in \{1, \dots, s-1\}$. Moreover, the local quadratic functions simplify to

$$\varphi^{(i)}(x) := Q^{(i)}x^2 + l^{(i)}x + c^{(i)}.$$

One-dimensional functions have several useful features, which make them a good starting point. The features we will be using include, among others, that the intersection of two neighboring regions is a zero-dimensional point $\mathcal{R}^{(i)} \cap \mathcal{R}^{(j)} = \{x \in \mathbb{R} \mid H^{(i,j)}x = h^{(i,j)}\}$ and that every region has either 1 or 2 neighboring regions. These features allow the construction of ReLU neurons that are active, i.e., have a non-zero output, in only one region. In the following theorem, we will make use of such ReLU neurons for representing PWQ functions.

Theorem 1 ([P1, Thm. 1])**.** *Let $n = 1$ and assume that the regions $\mathcal{R}^{(i)}$ are as in* (3.5) *and bounded. Further assume an NN of the form* (2.48) *with $\boldsymbol{\xi} := \begin{pmatrix} x & x^2 \end{pmatrix}^\top$ and weights*

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \underline{x}^{(1)} + \overline{x}^{(1)} & -1 \\ \vdots & \vdots \\ \underline{x}^{(s)} + \overline{x}^{(s)} & -1 \end{pmatrix}, \qquad \boldsymbol{v}^{(1)} := \begin{pmatrix} -\underline{x}^{(1)}\overline{x}^{(1)} \\ \vdots \\ -\underline{x}^{(s)}\overline{x}^{(s)} \end{pmatrix},$$
$$\boldsymbol{W}^{(2)} := \begin{pmatrix} -Q^{(1)} & \dots & -Q^{(s)} \end{pmatrix}, \qquad \boldsymbol{v}^{(2)} := 0.$$

Then, $h(x) := \varphi(x) - \Phi(x)$ is a continuous and PWA function of x.

We will only give an intuitive description of the result stated in Theorem 1. For a formal proof of the theorem, we refer to our article [P1, Thm. 1]. The main idea is to use ReLU neurons, which are active only in one region and have a quadratic term that is equal to that of the function $\varphi(x)$. Substituting the weights and biases from Theorem 1 in the definition of an NN from Section 2.3 results in

$$\Phi(x) = \sum_{i=1}^{s} -Q^{(i)} \max((\underline{x}^{(i)} + \overline{x}^{(i)})x - x^2 - \underline{x}^{(i)}\overline{x}^{(i)}, 0). \tag{3.6}$$

Each ReLU neuron $-Q^{(i)} \max((\underline{x}^{(i)} + \overline{x}^{(i)})x - x^2 - \underline{x}^{(i)}\overline{x}^{(i)}, 0)$ in this sum is active, i.e, has a non-zero output, if $(\underline{x}^{(i)} + \overline{x}^{(i)})x - x^2 - \underline{x}^{(i)}\overline{x}^{(i)} \geq 0$ and is inactive otherwise. The quadratic inequality is equivalent to $x \leq \overline{x}^{(i)} \wedge x \geq \underline{x}^{(i)}$ with (3.5) we thus have

$$\max((\underline{x}^{(i)}+\overline{x}^{(i)})x-x^2-\underline{x}^{(i)}\overline{x}^{(i)},0) = \begin{cases} -x^2+(\underline{x}^{(i)}+\overline{x}^{(i)})x-\underline{x}^{(i)}\overline{x}^{(i)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(i)}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

Now, combining (3.6) with (3.7) results in

$$\Phi(x) = \begin{cases} Q^{(1)}x^2 - Q^{(1)}(\underline{x}^{(1)} + \overline{x}^{(1)})x + Q^{(1)}\underline{x}^{(1)}\overline{x}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ Q^{(s)}x^2 - Q^{(s)}(\underline{x}^{(s)} + \overline{x}^{(s)})x + Q^{(s)}\underline{x}^{(s)}\overline{x}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}. \end{cases} \tag{3.8}$$

We consequently have

$$\varphi(x) - \Phi(x) = \begin{cases} \big(l^{(1)} + Q^{(1)}(\underline{x}^{(1)} + \overline{x}^{(1)})\big)x + c^{(1)} - Q^{(1)}\underline{x}^{(1)}\overline{x}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \big(l^{(s)} + Q^{(s)}(\underline{x}^{(s)} + \overline{x}^{(s)})\big)x + c^{(s)} - Q^{(s)}\underline{x}^{(s)}\overline{x}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases}$$

which is a PWA function that we define as $h(x) := \varphi(x) - \Phi(x)$. Moreover, since $\varphi(x)$ and $\Phi(x)$ are both continuous functions, the difference is continuous as well. Consequently, the difference $h(x)$ is a continuous PWA function, for which we can use known methods to represent it by a ReLU NN [26]. In summary, we decomposed the PWQ function (3.2) into two parts

$$\varphi(x) = \Phi(x) + \varphi(x) - \Phi(x) = \underbrace{\Phi(x)}_{\text{1. ReLU NN}} + \underbrace{h(x)}_{\text{2. ReLU NN}}, \tag{3.9}$$

where each part can be represented by a ReLU NN. Thus, the sum (3.9) representing the whole PWQ function (3.2) can be represented by a ReLU NN [25, Lem. D.2]. With this result, we partially solved the problem of representing PWQ functions for the special case of $n = 1$. However, it turned out that extending this solution to the general case $n > 1$ and beyond ReLU NNs is hardly possible. The main reason for this is that the construction of neurons, which are active in exactly one polyhedral region as in (3.7), is only possible in a one-dimensional domain, where the regions are closed and bounded intervals.

Representation of One-Dimensional Convex PWQ Functions by Maxout NNs: For extending the results on the representation of PWQ functions by NNs, we considered in [P2] a slightly different approach than in [P1]. Nevertheless, the approach still builds on the idea of adding a continuous PWA function of the form

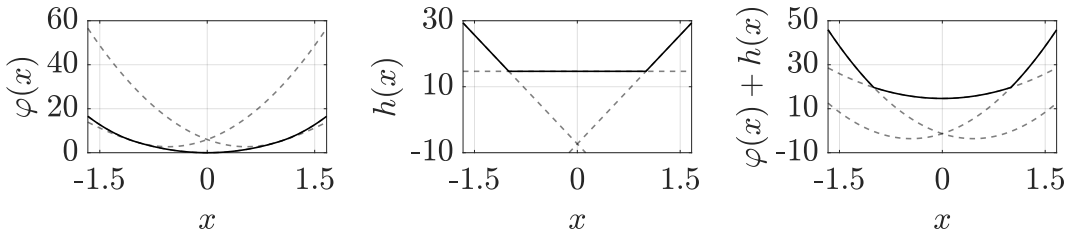
$$h(\boldsymbol{x}) := \begin{cases} h^{(1)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ h^{(s)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{3.10}$$

with $h^{(i)}(\boldsymbol{x}) := \boldsymbol{K}_h^{(i)}\boldsymbol{x} + l_h^{(i)}$ for all $i \in \{1, \dots, s\}$ to the original PWQ function according to

$$\tilde{\varphi}(x) := \varphi(x) + h(x) \tag{3.11}$$

such that $\tilde{\varphi}(x)$ is a function that can be represented by a NN. In the following, we will mainly concentrate on the representation of $\tilde{\varphi}(x)$ and $h(x)$ as the maximum of their local functions $\tilde{\varphi}^{(i)}(x)$ and $h^{(i)}(x)$, respectively, since such a

representation can be interpreted as a maxout neuron of the form (2.51). The idea of a *lifting* via a convex PWA function according to (3.11) is shown in Figure 3.3.



(a) **(b)** **(c)**

Figure 3.3. PWQ function $\varphi(x)$ with 3 regions in (a) and a convex PWA function $h(x)$ in (b) which is such that $\tilde{\varphi}(x) = \varphi(x) + h(x)$ can be represented by a maxout NN. The extension of the local functions outside their regions is indicated by the dashed gray lines.

The PWQ function $\varphi(x)$ from Figure 3.3.(a) can not directly be represented by a maxout NN since we have $\varphi(x) \neq \max\{\varphi^{(1)}(x), \varphi^{(2)}(x), \varphi^{(3)}(x)\}$. However, we can construct a convex PWA function $h(x)$ such that $\tilde{\varphi}(x)$ can be represented as the maximum of its local functions, i.e., we have $\tilde{\varphi}(x) = \varphi(x) + h(x) = \max\{\tilde{\varphi}^{(1)}(x), \tilde{\varphi}^{(2)}(x), \tilde{\varphi}^{(3)}(x)\}$ as shown in Figure 3.3.(c). Moreover, for a convex PWA $h(\boldsymbol{x})$ we always have $h(x) = \max\{h^{(1)}(x), h^{(2)}(x), h^{(3)}(x)\}$ [70]. This leads to a representation of the form

$$\varphi(\boldsymbol{x}) = \underbrace{\tilde{\varphi}(\boldsymbol{x})}_{\text{1. maxout neuron}} - \underbrace{h(\boldsymbol{x})}_{\text{2. maxout neuron}} \tag{3.12}$$

with

$$\tilde{\varphi}(\boldsymbol{x}) = \max\{\tilde{\varphi}^{(1)}(\boldsymbol{x}), \ldots, \tilde{\varphi}^{(s)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) \text{ and} \tag{3.13a}$$

$$h(\boldsymbol{x}) = \max\{h^{(1)}(\boldsymbol{x}), \ldots, h^{(s)}(\boldsymbol{x})\}, \tag{3.13b}$$

where we define $\tilde{\varphi}^{(i)}(\boldsymbol{x}) := \varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ for all $i \in \{1, \ldots, s\}$. As can be seen from (2.49), the input of an activation in an NN is a linear combination of the NN input $\boldsymbol{x}$. Thus, to interpret the representation (3.12) as a maxout NN, we have to consider an extended input vector

$$\boldsymbol{\xi} := \begin{pmatrix} \boldsymbol{x}^\top & x_1^2 & x_1 x_2 & \ldots & x_1 x_n & \ldots & x_n^2 \end{pmatrix}^\top \tag{3.14}$$

that includes all $d := {}^{n^2+3n}/_2$ quadratic monomials of the input $\boldsymbol{x}$. Note that in Theorem 1 we already used (3.14) with $n = 1$. The extended input vector allows us to formulate the local quadratic functions $\tilde{\varphi}^{(i)}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{Q}^{(i)} \boldsymbol{x} + \boldsymbol{l}^{(i)} \boldsymbol{x} + c^{(i)}$ of (3.13a) as a linear combination of the elements of the extended input vector $\boldsymbol{\xi}$. Thus, the representation in the form (3.12) is a maxout NN

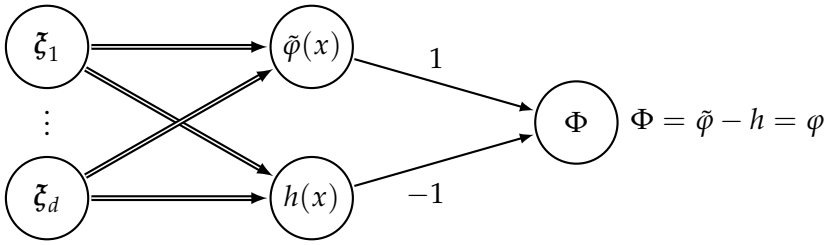


Figure 3.4. Maxout NN with two neurons and extended input vector (3.14) of dimension $d := {}^{n^2+3n}/_2$ for the exact representation of PWQ functions. The double arrows indicate the multi-channel preactivation required for the maxout activation.

with an extended input vector and two neurons representing $\tilde{\varphi}(x)$ and $h(\boldsymbol{x})$, respectively. The topology of the resulting maxout NN is shown in Figure 3.4.

Based on the exemplary representation of a one-dimensional function by a maxout NN, we formulate a conjecture that guided the research presented in our articles [P2] and [P9].

Conjecture 2. *For every continuous PWQ function $\varphi : \mathbb{R}^n \to \mathbb{R}$ there exists a continuous and convex PWA function $h : \mathbb{R}^n \to \mathbb{R}$ such that* (3.13a) *holds.*

Proving this conjecture would enable the representation of PWQ functions by maxout NNs with a topology as shown in Figure 3.4. In the conjecture, we only state that (3.13a) should hold, and (3.13b) is not explicitly mentioned since it always holds for a continuous and convex $h(\boldsymbol{x})$. Furthermore, based on the maxout NNs, we could use methods [24, 25] for representing functions of the form (3.13) by ReLU NNs to derive topologies for ReLU NNs that allow an exact representation of PWQ functions.

As with Theorem 1, we will consider a special case as a starting point before turning to the general result. Therefore, we will present the results from our article [P2] where we showed the existence of a suitable $h(\boldsymbol{x})$ that is in line with Conjecture 2 for a convex and continuous PWQ $\varphi(x)$ with $n = 1$. For this special case, we showed the following.

Theorem 3 ([P2, Cor. 3])**.** *Every convex PWQ of the form* (3.2) *with $n = 1$ can be exactly represented by a maxout-NN as defined in* (2.48) *with one hidden layer of width $w_1 = 2$, $p_1 = s$, and $\boldsymbol{\xi}$ as in* (3.14)*.*

The existence of a maxout NN $\Phi(x)$ with $\Phi(x) = \varphi(x)$ as proposed by Theorem 3 is proven in [P2, Cor. 3] by showing that Conjecture 2 holds for convex PWQ functions with $n = 1$. The formal proof is here omitted so that we can proceed directly to the general and more relevant case of continuous PWQ functions with arbitrary $n \in \mathbb{N}$ that are not necessarily convex.

Representation of n-dimensional PWQ Functions by Maxout NNs: The general case is considered in our article [P9] where we showed the following.

Theorem 4 ([P9, Thm. 5]). *For every continuous PWQ function $\varphi(\boldsymbol{x})$ of the form (3.2) with a polyhedral domain partition as in (3.3), there exists a convex PWA function $h(\boldsymbol{x})$ as in (3.10) such that*

$$\varphi(\boldsymbol{x}) = \max_{1\leq i\leq s}\{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\} - \max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\} \tag{3.15}$$

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

From Theorem 4 we can deduce that (3.13a) holds, since we have

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

where the second equality holds since $h(\boldsymbol{x})$ is convex and PWA. Thus, Theorem 4 shows that Conjecture 2 is true. Furthermore, with the extended input vector (3.14), we can find weights and biases for a maxout NN such that

$$\begin{aligned}\max_{1\leq i\leq s}\{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\} &= \max_{1\leq i\leq s}\{\boldsymbol{x}^\top \boldsymbol{Q}^{(i)}\boldsymbol{x} + (\boldsymbol{l}^{(i)} + \boldsymbol{K}_h^{(i)})\boldsymbol{x} + c^{(i)} + l_h^{(i)}\} \\ &= \max_{1\leq i\leq s}\{\boldsymbol{W}_i^{(2)}\boldsymbol{\xi} + \boldsymbol{v}_i^{(2)}\}\end{aligned}$$

holds, i.e., we can write the local functions $\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ as affine combinations of the vector $\boldsymbol{\xi}$ and thus the first term of (3.15) as a maxout neuron. Moreover, since the local functions $h^{(i)}(\boldsymbol{x})$ are already affine combinations of $\boldsymbol{x}$ and $\boldsymbol{x}$ is included in $\boldsymbol{\xi}$, the second term of (3.15) is a maxout neuron as well. This leads to the following summarizing statement about NNs for the exact representation of continuous PWQ functions with polyhedral domain partitions.

Theorem 5 ([P9, Thm. 6–8]). *Every continuous PWQ function of the form (3.2) with a polyhedral domain partition as in (3.3) can be exactly represented by the following NNs:*

1. *Maxout NN with $l = 1$ hidden layer, $w_1 = 2$ neurons with $p_1 = s$, input vector $\boldsymbol{\xi}$ as in (3.14), postactivation weights $\boldsymbol{W}_1^{(2)} = 1$, $\boldsymbol{W}_2^{(2)} = -1$, and postactivtion bias $v^{(2)} = 0$.*

2. *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 $i \in \{1, \ldots, l\}$, and input vector $\boldsymbol{\xi}$ as in (3.14).*

3. *ReLU NN with $l = 2s$ hidden layers, $w_i = \frac{n^2+3n}{2} + 3$ neurons in each layer $i \in \{1, \ldots, l\}$, and input vector $\boldsymbol{\xi}$ as in (3.14).*

As discussed in detail in [P9], the basis for the first NN, i.e., the maxout NN of Theorem 5, is the representation of PWQ functions by (3.15) from Theorem 4. The second and third NNs are derived by using different methods from [25] and [24], respectively, to represent the functions (3.13) which appear in (3.15) by ReLU NNs.

3.1.1.3 Discussion

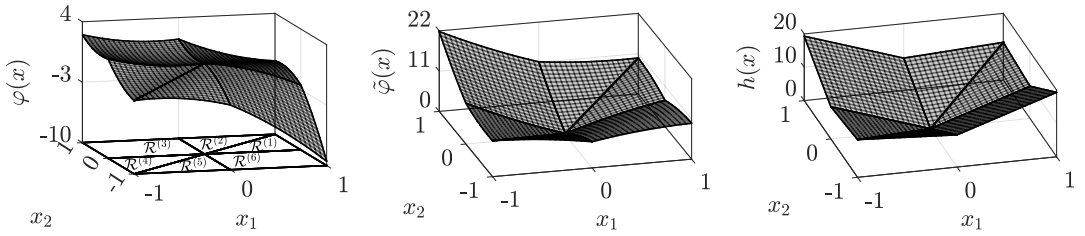
In Section 1.3.1, we motivated and stated the first central research question, which is:
Can every continuous PWQ function with a polyhedral domain partition be exactly represented by an NN?

The articles [P1, P2, P9] summarized in the previous section positively answer this question by presenting different NNs with maxout and ReLU activation that can exactly represent arbitrary continuous PWQ functions with polyhedral domain partitions. The resulting NNs are summarized in Theorem 5. These NNs are the first that can provably represent every continuous PWQ function with a polyhedral domain partition.

Representation of PWQ functions: Results from the literature on the representation of PWQ functions mainly focus on one-dimensional functions [81] or require assumptions that limit the applicability of the results (see, e.g., [29, 82]). In [82], for example, the authors showed that PWQ functions that possess the so-called consistent variation property (CVP) [83, Def. 5] can be represented by a sum of ReLU-like functions, where ReLU-like means functions of the form $\max\{f^{(i)}(\boldsymbol{x}), 0\}$ with a quadratic $f^{(i)} : \mathbb{R}^n \to \mathbb{R}$. For our case, the CVP means roughly speaking that the difference of any two local functions $\varphi^{(\imath)}(\boldsymbol{x})$ and $\varphi^{(\jmath)}(\boldsymbol{x})$ separated by the same hyperplane $\mathcal{B}^{(i,j)}$, i.e., $\mathcal{R}^{(\imath)} \cap \mathcal{R}^{(\jmath)} \subset \mathcal{B}^{(i,j)}$ can be described by a function with constant parameters along the hyperplane $\mathcal{B}^{(i,j)}$. A detailed explanation of the CVP can be found in [P9, Sec. II.B]. The CVP does generally not hold for arbitrary piecewise quadratic (PWQ) functions. As a result, the CVP prevents the use of the result from [82] for a more general setup. Another result on the representation of PWQ functions is presented in [29], where the authors showed that PWQ functions with a non-degenerate partition can be represented by functions that are interpretable as NNs. A non-degenerate partition is, according to the definition in [83], a partition where the intersection of any k hyperplanes that partition the domain Ω of the PWQ function $\varphi(\boldsymbol{x})$ is of dimension less than or equal to $n - k$. This again restricts the results to a subset of all PWQ functions with polyhedral domain partitions. Finally, in [27], an NN that represents a PWQ function is presented. However, contrary to our results, there are no theoretical guarantees that the NNs from [81] can represent any PWQ function.

In Figure 3.5, a PWQ function for which the CVP does not hold is shown. In addition, the partition is degenerate since the intersection of the three boundaries separating the regions at the origin is of dimension $0 \geq 2 - 3 = n - k$. For the partition to be non-degenerate, the intersection should be non-existent, i.e., an empty set. Thus, the function in Figure 3.5.(a) can not be represented using known methods. However, since the representation derived in Theorem 4, which is the basis for the NNs from Theorem 5 requires neither the CVP nor a non-degenerate partition, we can use the proposed NNs to represent this

function. In Figure 3.5, the representation of a PWQ by a NN with the topology from Figure 3.4 is shown. The functions represented by the two maxout neurons are shown in Figure 3.5.(b) and Figure 3.5.(c).



(a) **(b)** **(c)**

Figure 3.5. PWQ function $\varphi(\boldsymbol{x})$ with 6 regions in (a) and the representation of $\varphi(\boldsymbol{x}) = \tilde{\varphi}(\boldsymbol{x}) - h(\boldsymbol{x})$ by a maxout NN with the first topology from Theorem 5.

Experimental Results: Since OVFs (2.36) and Q-functions (2.76) for linear MPC are PWQ functions, we can use the proposed NNs to represent them. However, the exact representation of a given function is not the typical use case of a NN. Typically, NNs are used to approximate functions based on data samples. To investigate whether the proposed NNs provide benefits in these use cases, we performed several numerical studies in our article [P9, Sec. V]. These studies showed that the proposed NNs lead to approximations with significantly lower approximation error when they are used to approximate OVFs or Q-functions of linear MPC. The results are summarized in Table 13.2 of [P9, Tbl. 2]. In the experiments, we used the NNs from Theorem 5 for an approximation of the OVF of the OCP (2.30) for different systems and compared the results to an approximation by NNs that have the same topology as described in Theorem 5, except for the input, where we chose $\boldsymbol{\xi} = \boldsymbol{x}$. Thus, we compared our NNs to standard NNs with a PWA structure that are not able to represent the OVF exactly. In all test cases, we observed a relative improvement in the approximation error, which was at least 10.14 in the worst case and 136.36 in the best case.

Connection to Representation of PWA Functions: Also worth noting is that our results on the representation of PWQ functions stated in Theorem 4 fit seamlessly into results on the representation of PWA functions [70, 84] and can be seen as an extension of those. For PWA functions, the local functions in both terms of (3.15) from Theorem 4 are affine, and thus (3.15) is a difference of two terms, which are the maximum of affine functions. Consequently, for PWA functions (3.15) is the difference of two convex PWA functions, i.e, for the PWA case, Theorem 4 states that $\varphi(\boldsymbol{x})$ can be decomposed into two convex PWA functions. This is exactly the result known from the literature [70]. Thus, our theorem extends the known results and includes them as a special case.

3.1.2 Training Neural Networks to Global Optimality

Building on the results presented in Section 3.1.1, we now introduce a method that enables the training of NNs to global optimality by solving the OP (2.52). The central idea is to exploit the representation of piecewise-defined functions as the difference of two max-expressions by (3.15) presented in Theorem 4. This representation allows us to reformulate the training OP (2.52) as an MIQP. By casting the problem into this well-studied optimization framework, we obtain an approach that guarantees global optimality, thereby overcoming the limitations of conventional gradient-based training methods, which often only find local minima.

3.1.2.1 Problem Overview

NNs are often used in regression problems to approximate functions $f(\boldsymbol{x})$ based on samples from a given data set $\mathcal{D} := \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid y^{(i)} = f(\boldsymbol{x}^{(i)}) \forall i \in \{1, \dots, N_D\}\}$, which involves solving the OP

$$\min_{\boldsymbol{\theta}} \frac{1}{N_D} \sum_{k=1}^{N_D} \left\| y^{(k)} - \Phi(\boldsymbol{x}^{(k)}, \boldsymbol{\theta}) \right\|_2^2. \tag{3.16}$$

Due to the nonlinear structure of the NN $\Phi(\boldsymbol{x}, \boldsymbol{\theta})$, arising from the ReLU or maxout activation, (3.16) is a nonlinear OP. This OP is typically solved approximately using gradient-based training methods, e.g., stochastic gradient descent (SGD), RMSProp, or Adam [31]. However, the training success of these methods in terms of approximation error, training speed, and convergence depends on the choice of several hyperparameters. Moreover, the popular RMSProp and Adam algorithms are lacking general convergence guarantees. We addressed this problem in [P5] by showing that the OP (3.16) can be solved globally for a special type of NN.

3.1.2.2 Summary of Article [P5]

The starting point for the results of [P5] is a representation of the form

$$\Phi(\boldsymbol{x}) := \max\{\boldsymbol{W}^{(g)}\boldsymbol{g}(\boldsymbol{x})\} - \max\{\boldsymbol{W}^{(h)}\boldsymbol{h}(\boldsymbol{x})\} \tag{3.17}$$

with $\boldsymbol{W}^{(g)} \in \mathbb{R}^{p_1 \times r_1}$, $\boldsymbol{W}^{(h)} \in \mathbb{R}^{p_2 \times r_2}$, $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^{r_1}$, and $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^{r_2}$, where $\max\{\boldsymbol{W}^{(g)}\boldsymbol{g}(\boldsymbol{x})\}$ and $\max\{\boldsymbol{W}^{(h)}\boldsymbol{h}(\boldsymbol{x})\}$ are shorthand for $\max_{1 \leq i \leq p_1}\{\boldsymbol{W}_i^{(g)}\boldsymbol{g}(\boldsymbol{x})\}$ and $\max_{1 \leq i \leq p_2}\{\boldsymbol{W}_i^{(h)}\boldsymbol{h}(\boldsymbol{x})\}$, respectively. This representation is a generalization of (3.15), which we derived in Section 3.1.1 for PWQ functions. In fact, if we choose

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{pmatrix} 1 \\ \boldsymbol{\xi} \end{pmatrix} \text{ and } \boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix} \tag{3.18}$$

with ξ as in (3.14) then we can find parameters $W^{(g)}$ and $W^{(h)}$ such that (3.15) and (3.17) are equivalent since the max-expressions in (3.15) and (3.17) both contain affine combinations of ξ and x, respectively. Moreover, for

$$g(x) = h(x) = \begin{pmatrix} 1 \\ x \end{pmatrix} \tag{3.19}$$

the function (3.17) becomes

$$\Phi(x) := \max\{W^{(g)}_{:,2:n+1}x + W^{(g)}_{:,1}\} - \max\{W^{(h)}_{:,2:n+1}x + W^{(h)}_{:,1}\}, \tag{3.20}$$

which is a maxout NN with one hidden layer and two neurons. Thus, depending on the choice of $g(x)$ and $h(x)$, the function (3.17) is a classical maxout NN that can represent every continuous PWA function [21], or for (3.18), the function (3.17) is a NN with a topology as proposed in Theorem 5 that can represent every continuous PWQ function with a polyhedral partition (cf. Theorem 4). Meaning that (3.17) is a universal representation that captures a wide range of functions, including but not limited to maxout NNs and PWQ functions. When (3.17) is used as NN in the OP (3.16) then we have to solve

$$\min_{W^{(g)},W^{(h)}} \frac{1}{N_D} \sum_{k=1}^{N_D} \left\| y^{(k)} - \max\{W^{(g)}g(x^{(k)})\} + \max\{W^{(h)}h(x^{(k)})\} \right\|_2^2, \tag{3.21}$$

for the training. This is still a nonlinear OP. However, we can now use techniques introduced in Section 2.1.2.1 to describe the max-expressions in (3.21) by MI linear constraints. We first focus on representing $\max\{W^{(g)}g(x^{(k)})\}$ by MI linear constraints. Therefore, we consider the constraints

$$W^{(g)}g(x) \leq \mathbf{1}\alpha, \tag{3.22a}$$

$$M(\delta - \mathbf{1}) - W^{(g)}g(x) \leq -\mathbf{1}\alpha, \tag{3.22b}$$

$$\sum_{k=1}^{p_1} \delta_k = 1, \tag{3.22c}$$

$$\delta \in \{0,1\}^{p_1} \tag{3.22d}$$

which are such that for any solution to the MIFP

$$\text{find}\, \alpha, \delta \quad \text{s.t. (3.22)} \tag{3.23}$$

we have $\alpha = \max\{W^{(g)}g(x)\}$. This holds, since the constraints are such that there exists exactly one k^* with $\delta_{k^*} = 1$ and $\delta_i = 0$ for all $i \in \{1, \dots, p_1\} \setminus k^*$. Thus, the constraints (3.22) are equivalent to

$$\begin{aligned} W^{(g)}_{k^*}g(x) &= \alpha, \\ W^{(g)}_{i}g(x) &\leq W^{(g)}_{k^*}g(x), \\ M + W^{(g)}_{i}g(x) &\geq W^{(g)}_{k^*}g(x), \end{aligned}$$

for all $i \in \{1, \dots, p_1\} \setminus k^*$. For a sufficiently large M, as specified in [P5, Thm. 1] this is equivalent to $\alpha = \boldsymbol{W}_{k^*}^{(g)} \boldsymbol{g}(\boldsymbol{x}) = \max\{\boldsymbol{W}^{(g)} \boldsymbol{g}(\boldsymbol{x})\}$. This eventually allows us to replace both max-expressions in the OP (3.21) by MI linear constraints, where we need for each data point $(\boldsymbol{x}^{(i)}, y^{(i)})$ one set of MI linear constraints of the form (3.22). Resulting in a reformulation of (3.21) by

$$\min_{\boldsymbol{W}^{(g)}, \boldsymbol{W}^{(h)}, \alpha^{(i)}, \boldsymbol{\gamma}^{(i)}, \boldsymbol{\delta}^{(i)}} \frac{1}{N_D} \sum_{k=1}^{N_D} \left\| y^{(k)} - \alpha^{(k)} + \beta^{(k)} \right\|_2^2 \tag{3.25a}$$

$$\text{s.t. } \boldsymbol{W}^{(g)} \boldsymbol{g}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\alpha^{(i)}, \quad \boldsymbol{W}^{(g)} \boldsymbol{g}(\boldsymbol{x}^{(i)}) + M(\mathbf{1} - \boldsymbol{\delta}^{(i)}) \geq \mathbf{1}\alpha^{(i)}, \tag{3.25b}$$

$$\sum_{k=1}^{p_1} \delta_k^{(i)} = 1, \quad \boldsymbol{\delta}^{(i)} \in \{0,1\}^{p_1}, \tag{3.25c}$$

$$\boldsymbol{W}^{(h)} \boldsymbol{h}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\beta^{(i)}, \quad \boldsymbol{W}^{(h)} \boldsymbol{h}(\boldsymbol{x}^{(i)}) + M(\mathbf{1} - \boldsymbol{\gamma}^{(i)}) \geq \mathbf{1}\beta^{(i)}, \tag{3.25d}$$

$$\sum_{k=1}^{p_2} \gamma_k^{(i)} = 1, \quad \boldsymbol{\gamma}^{(i)} \in \{0,1\}^{p_2}, \tag{3.25e}$$

for $i \in \{1, \dots, N_D\}$. As previously described, we have for each data point i and for each max-expression a distinct set of constraints of the form (3.22). Consequently, the MI linear constraints (3.25b)-(3.25c) and (3.25d)-(3.25e) lead, similar to (3.22), to the relation $\alpha^{(i)} = \max\{\boldsymbol{W}^{(g)} \boldsymbol{g}(\boldsymbol{x}^{(i)})\}$ and $\beta^{(i)} = \max\{\boldsymbol{W}^{(h)} \boldsymbol{g}(\boldsymbol{x}^{(i)})\}$, respectively for all $i \in \{1, \dots, N_D\}$. The main idea behind the reformulation is that we introduce the binary variables $\boldsymbol{\delta}^{(i)} \in \{0,1\}^{p_1}$ and $\boldsymbol{\gamma}^{(i)} \in \{0,1\}^{p_2}$ for each of the N_D data points. Then the remaining constraints are constructed in a way such that the binary variables assign each data point to a local function $\boldsymbol{W}_j^{(g)} \boldsymbol{g}(\boldsymbol{x}^{(i)})$ and $\boldsymbol{W}_j^{(h)} \boldsymbol{h}(\boldsymbol{x}^{(i)})$ of the max-expressions $\max\{\boldsymbol{W}^{(g)} \boldsymbol{g}(\boldsymbol{x}^{(k)})\}$ and $\max\{\boldsymbol{W}^{(h)} \boldsymbol{h}(\boldsymbol{x}^{(k)})\}$, respectively. We proved in our article [P5] the equivalence of the MIQP (3.25) and (3.21) with the following theorem.

Theorem 6 ([P5, Thm. 1])**.** *There exists an $M \in \mathbb{R}_{>0}$ for which the MIQP* (3.25) *is equivalent to the nonlinear OP* (3.21)*.*

Theorem 6 states that the problem of fitting a function of the form (3.17) to data points can be solved globally by solving the MIQP (3.25). The effect of M is shown in Figure 3.6, where we used the MIQP (3.25) to solve (3.21) with $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (3.19), i.e., we fitted a PWA function to the data points marked by the gray crosses in the figure. The dotted gray line shows $\max\{\boldsymbol{W}^{(g)} \boldsymbol{g}(\boldsymbol{x})\}$ and the dashed gray line $\max\{\boldsymbol{W}^{(h)} \boldsymbol{h}(\boldsymbol{x})\}$, the difference of both is $\Phi(\boldsymbol{x})$, which is illustrated by the black line. We can see in Figure 3.6.(a) that for a small $M = 5$ we have a deviation between the data points and the function $\Phi(\boldsymbol{x})$. Thus, in this case, the OPs (3.21) and (3.25) are not equivalent because for (3.21) there exists a solution with a cost of 0 that is not found by (3.25). For the cases shown in Figure 3.6.(b) and Figure 3.6.(c), the constant M is large enough such that the solution computed with (3.25) is the global optimum with a cost of 0.

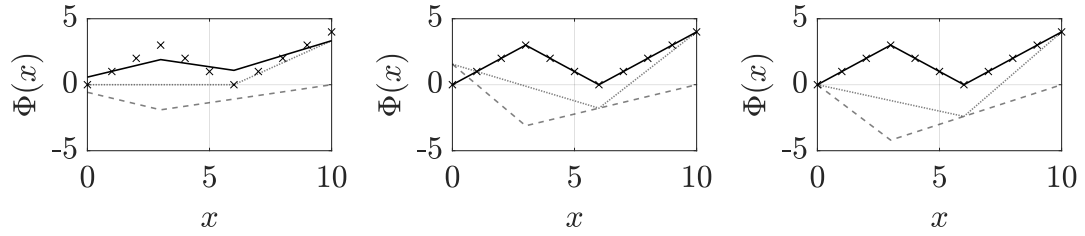


Figure 3.6. Optimal solution of the OP (3.25) for $p_1 = p_2 = 2$ with $M = 5$ in (a), $M = 20$ in (b), and $M = 100$ in (c).

When approximating a function by the OP (3.21), the design parameters are p_1 and p_2, which determine the number of terms in the max-expressions of (3.17) and thus the number of regions in the final function. Additional design parameters are the r_1 and r_2 functions included in $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$. Depending on the choice of $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$, we can fit different functions to the data. For example, for the previously discussed cases with (3.18), we fit a PWQ function, with which we can represent, according to Theorem 4, every continuous PWQ function with a polyhedral domain partition. For (3.19), we have a PWA model that can represent every continuous PWA function with a polyhedral domain partition [21]. Thus, the connection of the results from [P9] and [P5] allows us to globally solve the problem of fitting PWA and PWQ functions to data.

3.1.2.3 Discussion

The MIQP (3.25) introduced in our article [P5, Eq. (6)] is the first method that allows for globally solving the problem of fitting continuous PWA and PWQ functions to data points. Known methods for fitting piecewise-defined functions [85, 86] typically alternate between clustering the data points and computing a regression for each cluster individually until a certain termination criterion is met. However, due to the switching between clustering and regression, the solution will not be globally optimal. To compute a global optimal solution to (2.52), clustering and regression must be performed simultaneously. This is achieved in our approach by using binary variables that assign points to regions while at the same time minimizing the approximation error.

Approximations of PWA and PWQ Functions: To illustrate, for concrete examples, what it means to globally solve the OP (2.52), we will discuss the application of our method to cases where the data are sampled from a PWA and PWQ function, respectively. If the data is sampled from a continuous PWA function then we can use the convex decomposition of PWA functions (2.59) to reformulate the data points as $\big(\boldsymbol{x}^{(i)}, y^{(i)} = \max_{1\leq j\leq s}\{\boldsymbol{K}_g^{(j)}\boldsymbol{x}^{(i)} + \boldsymbol{l}_g^{(j)}\} -$

$\max_{1\leq j\leq s}\{\boldsymbol{K}_h^{(j)}\boldsymbol{x}^{(i)}+\boldsymbol{l}_h^{(j)}\}\big)$ for all $i\in\{1,\dots,N_D\}$. Thus the OP (3.21) becomes

$$\min_{\boldsymbol{W}^{(g)},\boldsymbol{W}^{(h)}}\frac{1}{N_D}\sum_{k=1}^{N_D}\Big|\Big|\max_{1\leq j\leq s}\{\boldsymbol{K}_g^{(j)}\boldsymbol{x}^{(k)}+\boldsymbol{l}_g^{(j)}\}-\max_{1\leq j\leq s}\{\boldsymbol{K}_h^{(j)}\boldsymbol{x}^{(k)}+\boldsymbol{l}_h^{(j)}\}\\ -\max_{1\leq j\leq p_1}\{\boldsymbol{W}_j^{(g)}\boldsymbol{g}(\boldsymbol{x}^{(k)})\}+\max_{1\leq j\leq p_2}\{\boldsymbol{W}_j^{(h)}\boldsymbol{h}(\boldsymbol{x}^{(k)})\}\Big|\Big|_2^2. \quad (3.26)$$

For $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (3.19) and $p_1=p_2=s$, the optimal solution to (3.26) is $\boldsymbol{W}_{j,1}^{(g)}=\boldsymbol{l}_g^{(j)}$, $\boldsymbol{W}_{j,2:n+1}^{(g)}=\boldsymbol{K}_g^{(j)}$, $\boldsymbol{W}_{j,1}^{(h)}=\boldsymbol{l}_h^{(j)}$, and $\boldsymbol{W}_{j,2:n+1}^{(h)}=\boldsymbol{K}_h^{(j)}$ for all $j\in\{1,\dots,s\}$. By substituting these parameters in (3.26), we obtain a cost of zero, and since the cost is lower bounded by zero, this solution must be the global optimum.

In case the data is sampled from a continuous PWQ function with a polyhedral domain partition, then according to Theorem 5, there exists a maxout NN that represents this function exactly. Thus, we can represent $y^{(i)}$ from the data set by $y^{(i)}=\max_{1\leq j\leq s}\{\boldsymbol{W}_j^{(1)}\boldsymbol{\xi}^{(i)}+\boldsymbol{v}_j^{(1)}\}-\max_{1\leq j\leq s}\{\boldsymbol{W}_{j+s}^{(1)}\boldsymbol{\xi}^{(i)}+\boldsymbol{v}_{j+s}^{(1)}\}$ for all $i\in\{1,\dots,N_D\}$ with $\boldsymbol{\xi}$ as in (3.14). Formulating the OP (3.21) for this data set results in

$$\min_{\boldsymbol{W}^{(g)},\boldsymbol{W}^{(h)}}\frac{1}{N_D}\sum_{k=1}^{N_D}\Big|\Big|\max_{1\leq j\leq s}\{\boldsymbol{W}_j^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_j^{(1)}\}-\max_{1\leq j\leq s}\{\boldsymbol{W}_{j+s}^{(1)}\boldsymbol{\xi}+\boldsymbol{v}_{j+s}^{(1)}\}\\ -\max_{1\leq j\leq p_1}\{\boldsymbol{W}_j^{(g)}\boldsymbol{g}(\boldsymbol{x}^{(k)})\}+\max_{1\leq j\leq p_2}\{\boldsymbol{W}_j^{(h)}\boldsymbol{h}(\boldsymbol{x}^{(k)})\}\Big|\Big|_2^2, \quad (3.27)$$

with $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (3.18). Now, we can use the same arguments as before to see that for $p_1=p_2=s$ the solution with $\boldsymbol{W}_{j,1}^{(g)}=\boldsymbol{v}_j^{(1)}$, $\boldsymbol{W}_{j,2:d+1}^{(g)}=\boldsymbol{W}_j^{(1)}$, $\boldsymbol{W}_{j,1}^{(h)}=\boldsymbol{v}_{j+s}^{(1)}$, and $\boldsymbol{W}_{j,2:d+1}^{(h)}=\boldsymbol{v}_{j+s}^{(1)}$ for all $j\in\{1,\dots,s\}$ with $d:=(n^2+3n)/2$ has a cost of zero and is thus globally optimal for (2.52).

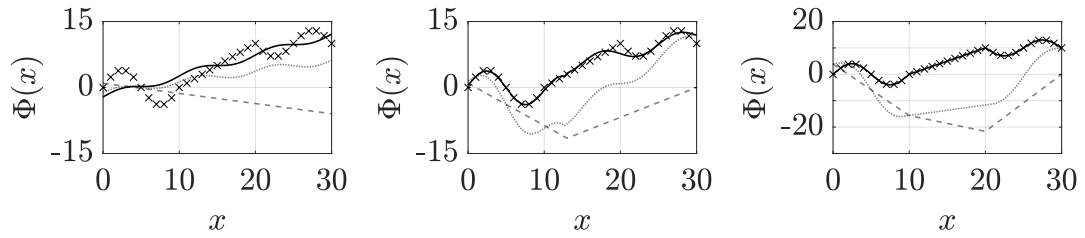
For both discussed cases, i.e., PWA and PWQ, we can use Theorem 6 to solve the OP (3.26) and (3.27) by the MIQP (3.25), which will result in the previously discussed optimal solution with a cost of zero. Our method for solving these regression problems is the first that can fit globally optimal PWA and PWQ functions to data points. Known methods either solve the problem approximately [85, 86, 87] or have a limited applicability. For example, the methods from [85] can only be applied to fit convex PWA functions, and [33, 88, 89] can only be used to fit PWA functions with $n=1$.

Approximations of Piecewise-Defined Functions: Our method extends the state-of-the-art not only because it allows for globally optimal fits, but also because it is not limited to a specific type of piecewise-defined functions. As described in Section 3.1.2.2, the functions $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ determine the function that is fitted to the data. Interestingly, the choice of these functions is not limited to the presented options (3.19) and (3.18). Every continuous function

can be included in $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$. For example for the functions shown in Figure 3.7 we solved (3.25) with

$$\boldsymbol{g}(x) = \begin{pmatrix} 1 \\ x \\ \sin(0.2x) \end{pmatrix} \text{ and } \boldsymbol{h}(x) = \begin{pmatrix} 1 \\ x \end{pmatrix}.$$

for different values of p_1 and p_2. For $p_1 = p_2 = 1$, we fit the function $\Phi(\boldsymbol{x}) = \boldsymbol{W}_1^{(g)} - \boldsymbol{W}_1^{(h)} + (\boldsymbol{W}_2^{(g)} - \boldsymbol{W}_2^{(h)})x + \boldsymbol{W}_3^{(g)} \sin(0.2x)$ and thus the MIQP returns the same solution as an ordinary least squares fit. For $p_1 = p_2 = 2$ the solution is a piecewise-defined function with 2 regions (see Figure 3.7.(b)), thereby resulting in an approximation with a lower error. Finally, for $p_1 = p_2 = 3$, we can compute an optimal solution with a cost of zero using (3.25). Thus, the black line in Figure 3.7.(c) representing $\Phi(x)$ passes exactly the data points.



(a) **(b)** **(c)**

Figure 3.7. Optimal solution of the OP (3.21) for $M = 70$ with $p_1 = p_2 = 1$ in (a), $p_1 = p_2 = 2$ in (b), and $p_1 = p_2 = 3$ in (c).

This makes our method the first to offer such flexibility, as it not only guarantees global optimal solutions but also allows fitting arbitrary piecewise-defined functions.

Complexity of the MIQP: The complexity in terms of runtime of the MIQP (3.25) for solving (2.52) increases in the worst case exponentially with the number of binary variables. In our case, the number of binary variables grows linearly with the number of data points and is given by $N_D(p_1 + p_2)$. Thus, the MIQP (3.25) that can be used to train maxout NNs to global optimality may only be solvable in a reasonable time for small data sets. To counteract this effect, we presented in [P5, Sec. 4] different methods to reduce the number of binary variables needed and thus the complexity. The first idea for a simplification is to add symmetry breaking constraints [90] to the MIQP (3.25) that exclude solutions with the same cost. These solutions arise from the fact that the maximum used in (3.21) is invariant under a permutation of the segments, i.e., we have $\max\{x_1, x_2\} = \max\{x_2, x_1\}$. If we consider an example with two regions, $p_1 = p_2 = 2$, and two data points, $N_D = 2$, then it only matters whether the data points are assigned to the same region or to different regions. The specific region they are assigned to, i.e., the first or second, does not alter

the solution. Formally, this means that in case the data points are assigned to different regions, we can either have the binary variables

$$\delta^{(1)} = \begin{pmatrix}1\\0\end{pmatrix},\ \delta^{(2)} = \begin{pmatrix}0\\1\end{pmatrix},\ \gamma^{(1)} = \begin{pmatrix}1\\0\end{pmatrix},\ \gamma^{(2)} = \begin{pmatrix}0\\1\end{pmatrix},$$

which results in

$$\begin{aligned}\max\{W_1^{(g)}x^{(1)}, W_2^{(g)}x^{(1)}\} - \max\{W_1^{(h)}x^{(1)}, W_2^{(h)}x^{(1)}\} &= W_1^{(g)}x^{(1)} - W_1^{(h)}x^{(1)},\\ \max\{W_1^{(g)}x^{(2)}, W_2^{(g)}x^{(2)}\} - \max\{W_1^{(h)}x^{(2)}, W_2^{(h)}x^{(2)}\} &= W_2^{(g)}x^{(2)} - W_2^{(h)}x^{(2)}\end{aligned}$$

or we can have

$$\delta^{(1)} = \begin{pmatrix}0\\1\end{pmatrix},\ \delta^{(2)} = \begin{pmatrix}1\\0\end{pmatrix},\ \gamma^{(1)} = \begin{pmatrix}0\\1\end{pmatrix},\ \gamma^{(2)} = \begin{pmatrix}1\\0\end{pmatrix},$$

which results in

$$\begin{aligned}\max\{W_1^{(g)}x^{(1)}, W_2^{(g)}x^{(1)}\} - \max\{W_1^{(h)}x^{(1)}, W_2^{(h)}x^{(1)}\} &= W_2^{(g)}x^{(1)} - W_2^{(h)}x^{(1)},\\ \max\{W_1^{(g)}x^{(2)}, W_2^{(g)}x^{(2)}\} - \max\{W_1^{(h)}x^{(2)}, W_2^{(h)}x^{(2)}\} &= W_1^{(g)}x^{(2)} - W_1^{(h)}x^{(2)}.\end{aligned}$$

However, both choices for the binary variables lead to the same solution for the MIQP (3.25) in terms of cost, as they refer to a perturbation of the segments in the max-expressions. Excluding all equivalent solutions that result from the perturbation invariance of the maximum is still an ongoing area of research. However, some of these equivalent solutions can be excluded by including the symmetry breaking constraints (9.11) from [P5, Eq. (11)]. The advantage of this type of symmetry breaking constraints is that they do not alter the solution of the MIQP (3.25) and thus Theorem 6 still applies, and we can still solve (3.21) globally by the MIQP.

Another method to reduce the complexity of the MIQP (3.25) is the preclustering presented in Section 9.4.1 ([P5, Sec. 4.2]). In the preclustering method the data points are clustered in K disjoint clusters of the form $\mathcal{C}_j := \{(x^{(i)}, y^{(i)}) \in \mathcal{D} \mid i \in \mathcal{I}_j\}$ with $\mathcal{I}_j \subseteq \{1, \dots, N_D\}$, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i \neq j$, and $\cup_{j=1}^{K} \mathcal{C}_j = \mathcal{D}$ using an arbitrary clustering method, e.g., K-means++ [91]. Then the constraints in the MIQP (3.25) are modified in a way that not just single data points are assigned to regions, but whole clusters. The modified constraints (9.12) can be found in our article [P5, Eq. (12)]. This method allows for reducing the number of binary variables by a factor of K/N_D, from $N_D(p_1 + p_2)$ to $K(p_1 + p_2)$. However, the preclustering reduces the flexibility of the MIQP (3.25), which means that the (3.21) is no longer solved globally optimally by the MIQP (3.25). Thus, we can trade off flexibility and accuracy against complexity.

3.1.3 Polynomial Approximations of Activation Functions

We will now consider the problem of computing tailored polynomial approximations of activation functions. The motivation for this arises from the fact that

standard nonlinear activations cannot be evaluated directly within the framework of HE, which is increasingly used to guarantee data privacy in machine learning applications. To address this limitation, we investigate how activation functions can be approximated in such a way that they allow for an efficient encrypted evaluation.

3.1.3.1 Problem overview

HE allows, in principle, computations on encrypted data. However, as detailed in Section 2.3.4 the supported computations typically include only additions and a limited number of consecutive multiplications. Since the focus of this thesis is the tailored use of NNs and not HE itself, we will not detail specific HE-schemes. We assume a HE-scheme with a multiplicative depth of d that supports encrypted additions and up to d consecutive encrypted multiplications, as defined in (2.64) and (2.65), respectively. The limitations to additions and multiplications restrict encrypted evaluations to functions that can be evaluated using only these operations, which are polynomial functions of the form

$$p(x) := c_0 + c_1 x + c_2 x^2 + \cdots + c_r x^r \tag{3.28}$$

with $c_i \in \mathbb{R}$ for all $i \in \{0, \ldots, r\}$. By using (2.64) and (2.65) the encrypted function value of $p(x)$ can be computed by

$$\begin{aligned} &\mathrm{Enc}\left(p(x)\right) = \mathrm{Enc}\left(c_0 + c_1 x + c_2 x^2 + \cdots + x^r c_r\right) \\ &= \mathrm{Enc}(c_0) \oplus \mathrm{Enc}(c_1 x) \oplus \cdots \oplus \mathrm{Enc}(x^r c_r) \\ &= \mathrm{Enc}(c_0) \oplus (\mathrm{Enc}(c_1) \otimes \mathrm{Enc}(x)) \oplus \cdots \oplus (\mathrm{Enc}(x^r) \otimes \mathrm{Enc}(c_r)) \end{aligned} \tag{3.29}$$

purely on encrypted input data $\mathrm{Enc}(x)$. The multiplicative depth limits the maximum degree of polynomials that can be evaluated. In fact, for a multiplicative depth of d, the maximum degree is $r = 2^d - 1$ [44]. In Figure 3.8, the evaluation of $c_3 x_3$ with a generic coefficient $c_3 \in \mathbb{R}$ is shown.

The dotted lines at each level indicate the needed number of consecutive multiplications up to the respective level. At the depth 0, we only have the variable x and the constant c_3. At the first level at depth 1, x^2 and $c_3 x$ are computed in the nodes. Finally, x^2 and $c_3 x$ are multiplied at depth $d = 2$ to compute $c_3 x^3$. This shows exemplarily how a polynomial of degree $r = 3 = 2^2 - 1 = 2^d - 1$ can be evaluated with a multiplicative depth of $d = 2$.

When aiming for an encrypted and thus privacy-preserving evaluation of NNs, the nonlinear activation functions need to be approximated by polynomials. To keep the approximation error low, polynomials with a high degree are beneficial. Therefore, the current state-of-the-art is to use for these approximations the maximum possible degree for a given depth d, i.e., $r = 2^d - 1$ [72, 92]. We will present in the following section our results from [P8], which allow us to increase the maximum degree of polynomials implementable with a multiplicative depth of d to $r = 2^d$. This enables approximations with a higher accuracy without increasing the multiplicative depth.

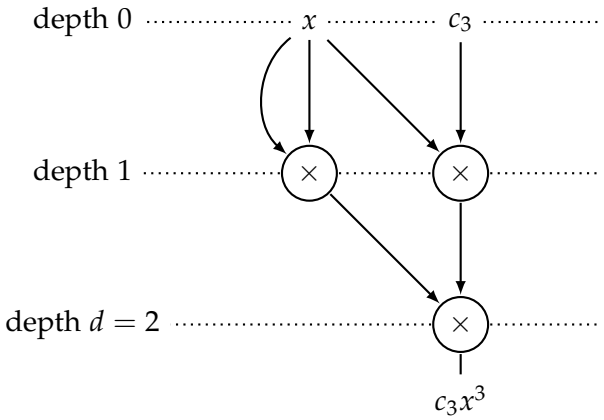


Figure 3.8. Computational circuit for the evaluation of c_3x^3. The dotted lines indicate the number of consecutive multiplications needed at each level.

3.1.3.2 Summary of Article [P8]

A drawback of the evaluation of polynomials shown in Figure 3.8 is that it requires a multiplication to compute $c_{2^d-1}x^{2^d-1}$ from c_{2^d-1} and x^{2^d-1}. If it is possible to avoid this multiplication, then the degree of implementable polynomials could be increased. The main idea to achieve this is to restrict the coefficient of the monomial with the highest degree in (3.28) to the set of integers. This small modification allows the evaluation of the leading monomial c_rx^r by computing

$$c_rx^r = \sum_{i=1}^{c_r} x^r$$

with $c_r \in \mathbb{Z}$ by using only encrypted additions, i.e., without increasing the multiplicative depth. The method is illustrated in Figure 3.9.

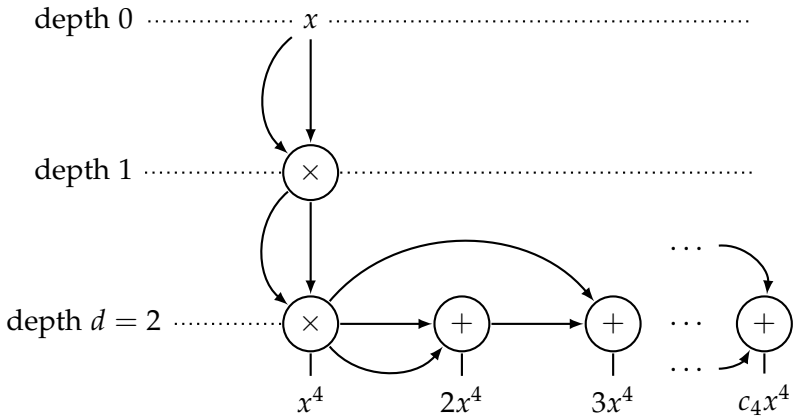


Figure 3.9. Computational circuit for the evaluation of c_4x^4 with $c_4 \in \mathbb{Z}$. The dotted lines indicate the number of consecutive multiplications needed at each level.

In contrast to the computational circuit shown in Figure 3.8, here the avail-

able multiplicative depth d is used to compute x^r with $r = 2^d$. Then, $c_r x^r$ is computed based on x_r by solely using additions. Thus, computing $c_r x^r$ with $c_r \in \mathbb{Z}$ does not consume one level of the available depth. This allows us to increase the degree from $r = 2^d - 1$ to $r = 2^d$ under a constant multiplicative depth.

Using polynomials to approximate non-linear functions $f(\boldsymbol{x})$ based on sampling points from a data set $\mathcal{D} := \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid y^{(i)} = f(\boldsymbol{x}^{(i)}) \forall i \in \{1, \dots, N_D\}\}$ typically involves solving a regression problem of the form

$$\min_{c_0,\dots,c_r} J(c_0, \dots, c_r, x_1, \dots, x_{N_D}), \tag{3.30}$$

which is an unconstrained OP. Now, if the proposed polynomials with leading integer coefficients are used in a regression problem, then the OP is initially an MIP of the form

$$\min_{\hat{c}_0,\dots,\hat{c}_r} J(\hat{c}_0, \dots, \hat{c}_r, x_1, \dots, x_{N_D}) \quad \text{s.t.} \quad \hat{c}_n \in \mathbb{Z}. \tag{3.31}$$

For standard performance measures such as the mean absolute error (MAE)

$$J_{\text{MAE}}(c_0, \dots, c_r, x_1, \dots, x_{N_D}) := \frac{1}{N} \sum_{i=1}^{N} |f(x_i) - p(x_i)| \tag{3.32}$$

or the mean squared error (MSE)

$$J_{\text{MSE}}(c_0, \dots, c_r, x_1, \dots, x_{N_D}) := \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - p(x_i))^2 \tag{3.33}$$

the resulting problem is an MILP and an MIQP, respectively. These problems can be solved using MIP solvers (e.g., [45, 46]). However, by exploiting the fact that only one optimization variable in the MIP (3.31) is restricted to the set of integers, we can reformulate the MIP and compute an optimum by solving at most three LPs if the MAE (3.32) is used or three QPs if the MSE (3.33) is used. We formally showed in [P8] the following.

Theorem 7 ([P8, Thm. 1–2])**.** *Assume the cost function J in* (3.30) *is convex in the coefficients c_i and assume $\boldsymbol{c}^* := (c_0^*, \dots, c_r^*)$ is an optimizer of the unconstrained OP* (3.30)*. Then, there exists an optimizer $\hat{\boldsymbol{c}}^* := (\hat{c}_0^*, \dots, \hat{c}_r^*)$ for the constrained problem* (3.31) *with $\hat{c}_r^* = \lfloor c_r^* \rfloor$ or $\hat{c}_r^* = \lceil c_r^* \rceil$. If the cost function J in* (3.30) *in addition possess the point symmetry $J(\boldsymbol{c}^* + \boldsymbol{c}) = J(\boldsymbol{c}^* - \boldsymbol{c})$ for every $\boldsymbol{c} \in \mathbb{R}^{r+1}$. Then, there exists an optimizer $\hat{\boldsymbol{c}}^*$ for* (3.31) *with $\hat{c}_r^* = \lfloor c_r^* \rceil$.*

The theorem is a summary of [P8, Thm 1] and [P8, Thm 2]. The assumptions of Theorem 7 are satisfied for most standard performance measures. In fact, the MAE and MSE from (3.32) and (3.33), respectively, are convex and point symmetric according to $J(\boldsymbol{c}^* + \boldsymbol{c}) = J(\boldsymbol{c}^* - \boldsymbol{c})$. Thus, for these cost functions, a

solution to the MIP (3.31) can be computed by first solving the unconstrained OP (3.30) to compute $\boldsymbol{c}^* = \left(c_0^*, \ldots, c_r^*\right)$ and subsequently solving

$$\min_{\hat{c}_0,\ldots,\hat{c}_r} J(\hat{c}_0, \ldots, \hat{c}_r, x_1, \ldots, x_{N_D}) \quad \text{s.t.} \quad \hat{c}_r = \lfloor c_r^* \rceil. \tag{3.34}$$

This gives us the optimal solution $\hat{\boldsymbol{c}}^* := \left(\hat{c}_0^*, \ldots, \hat{c}_{r-1}^*, \lfloor c_r^* \rceil\right)$ for (3.31). Note that since we typically have $\hat{c}_i^* \neq c_i^*$ for all $i \in \{1, \ldots, r-1\}$ we have to solve the second OP (3.34) and can not simply reuse the coefficients $\left(c_0^*, \ldots, c_{r-1}^*\right)$.

For non-symmetric cost functions, the procedure for computing optimal coefficients differs slightly. In this case, the optimal leading coefficient may be $\hat{c}_r^* = \lfloor c_r^* \rfloor$ or $\hat{c}_r^* = \lceil c_r^* \rceil$. Thus, in addition to the unconstrained problem (3.30), we have to solve the two OPs

$$\min_{\tilde{c}_0,\ldots,\tilde{c}_r} J(\tilde{c}_0, \ldots, \tilde{c}_r, x_1, \ldots, x_{N_D}) \quad \text{s.t.} \quad \tilde{c}_r = \lfloor c_r^* \rfloor. \tag{3.35}$$

with the optimizer $\tilde{\boldsymbol{c}}^* = \left(\tilde{c}_0^*, \ldots, \tilde{c}_r^*\right)$ and

$$\min_{\bar{c}_0,\ldots,\bar{c}_r} J(\bar{c}_0, \ldots, \bar{c}_r, x_1, \ldots, x_{N_D}) \quad \text{s.t.} \quad \bar{c}_r = \lceil c_r^* \rceil \tag{3.36}$$

with the optimizer $\bar{\boldsymbol{c}}^* = \left(\bar{c}_0^*, \ldots, \bar{c}_r^*\right)$. Then, the solution with the lowest cost provides an optimizer for (3.31). Or more precisely, for a convex and non-symmetric J, an optimizer of (3.31) is

$$\hat{\boldsymbol{c}}^* = \begin{cases} \left(\tilde{c}_0^*, \ldots, \tilde{c}_{r-1}^*, \lfloor c_r^* \rfloor\right) & \text{if } J(\tilde{c}_0^*, \ldots, \tilde{c}_{r-1}^*, \lfloor c_r^* \rfloor) \leq J(\bar{c}_0^*, \ldots, \bar{c}_{r-1}^*, \lceil c_r^* \rceil), \\ \left(\bar{c}_0^*, \ldots, \bar{c}_{r-1}^*, \lceil c_r^* \rceil\right) & \text{otherwise.} \end{cases}$$

In summary, Theorem 7 allows us to solve the MIP (3.31) by solving only QPs or LPs, depending on the chosen cost function, without relying on MI solvers.

We have seen that given a multiplicative depth of d we can either choose to implement polynomials of degree $r = 2^d - 1$ with $c_i \in \mathbb{R}$ for all $i \in \{1, \ldots, r\}$ or with degree $\hat{r} = 2^d$ with $c_i \in \mathbb{R}$ for all $i \in \{1, \ldots, \hat{r} - 1\}$ and $c_{\hat{r}} \in \mathbb{Z}$. It seems natural to always choose the latter since this variant does not increase the complexity of the HE-scheme in terms of the multiplicative depth but allows for more accurate approximations due to the higher degree. This can also be formalized as follows.

Corollary 8 ([P8, Cor. 1]). *Consider any $\hat{r} \in \mathbb{N}$ and let $\hat{\boldsymbol{c}}^*$ be an optimizer for* (3.31) *and $r := \hat{r}$. Furthermore, let $\boldsymbol{c}^*$ be an optimizer for* (3.30) *and an degree $r \in \mathbb{N}$ smaller than $\hat{r}$. Then, $J(\hat{\boldsymbol{c}}^*) \leq J(\boldsymbol{c}^*)$, where J refers to the same cost function instantiated for the two different degrees (but same number of sampling points N_D).*

The intuitive interpretation of Corollary 8 is that considering the additional degree provided by the leading integer coefficient will never worsen the approximation error with respect to the chosen cost function in comparison to existing methods, where all coefficients are real numbers.

Chebyshev Polynomials: In principle, the proposed polynomials with leading integer coefficients can directly be used in regression problems of the form (3.31). However, when using polynomials in the canonical power basis in regression problems, we may have to deal with numerical problems that arise when high-order polynomials are used. More specifically, the problems are that the coefficients of the high-order monomials become either very large or very small, depending on the problem. To avoid these problems, Chebyshev polynomials are a common approach. Therefore, we present in the article [P8] an extension of our method to Chebyshev polynomials. When Chebyshev polynomials of the first kind are used as basis functions. Then the polynomial $p(x)$ (3.28) in the canonical power basis is replaced by

$$p(x) := \hat{c}_0 + \hat{c}_1 T_1(x) + \hat{c}_2 T_2(x) + \cdots + \hat{c}_r T_r(x) \tag{3.37}$$

where the basis functions are Chebyshev polynomials defined by the recursion

$$T_0(x) := 1, \quad T_1(x) := x, \quad T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x).$$

Ensuring a leading integer coeffieicent as in (3.31) when Chebyshev polynomials are used requires to replace the constraint $\hat{c}_n \in \mathbb{Z}$ by

$$s\hat{c}_n \in \mathbb{Z} \tag{3.38}$$

where $s \in \mathbb{R}$ is some scaling factor. We introduced the polynomial approximations here only for the standard constraint $\hat{c}_n \in \mathbb{Z}$. However, the method can be extended to include constraints of the form (3.38) with a scaling factor. Since the results are structurally identical, we refer for the details of this extension to our article [P8, Sec. III].

3.1.3.3 Discussion

The main contributions of the article [P8], summarized in Section 3.1.3.2, are to increase the degree of polynomials implementable under a constant multiplicative depth by one and to provide a tailored solution method for solving regression problems with the proposed polynomials. This allows for increasing the accuracy of polynomial approximations without increasing the complexity in terms of the multiplicative depth. We demonstrated in [P8, Sec. IV] that, particularly for low-order polynomials, i.e., up to a degree of 32, the additional degree often results in a significant improvement in the approximation accuracy. Application scenarios for the presented method primarily involve encrypted function evaluation, where the number of consecutive multiplications is restrictive. However, the results are not limited to this scenario but are of general interest in applications where multiplications require more resources than additions, and it is therefore desirable to keep the number of consecutive multiplications low. This is, for example, often the case on embedded systems [93]. Regarding the use of the proposed polynomials in regression tasks,

we focused on approximations of non-linear activation functions to enable a privacy-preserving evaluation of NNs. But again, the results are not limited to this application. The polynomials may also be used to directly approximate nonlinear control laws [94], which would allow a privacy-preserving evaluation of control laws, i.e, an evaluation directly on encrypted state or output measurements.

Beneficial Leading Integer Coefficient: The theoretical results allow us to specify some cases where including the additional monomial of order $r = 2^d$ with integer coefficient can not result in an improvement. From Theorem 7 we can deduce that for a convex symmetric J the leading integer coefficient $\hat{c}^*_{2^d}$ of all polynomials for which the optimizer of the OP (3.30) includes a leading coefficient with $c^*_{2^d} \in]-0.5, 0.5[$, will be rounded to 0. Thus, in this case, the integer coefficient is not used, and we have a polynomial of degree $r = 2^d - 1$ as in existing methods. Thus, there may be cases where our method does not result in an improvement. However, with Corollary 8 we can lower bound the improvement, since the corollary states that including the leading integer coefficient will never worsen the approximation accuracy. In fact, as we will describe next, we often observe a significant improvement over existing methods.

Experimental Results: In the article [P8], we described the results of several experiments where we used polynomials of degree $r = 2^d - 1$ with $c_i \in \mathbb{R}$ for all $i \in \{1, \dots, r\}$, representing the state-of-art, as well as polynomials of degree $r = 2d$ with $c_i \in \mathbb{R}$ for all $i \in \{1, \dots, r-1\}$ and $c_r \in \mathbb{Z}$ as proposed by us to approximate different nonlinear functions. The outcomes of the experiments are summarized [P8, Tab. 1–6] and confirm the theoretical results in many aspects. First, the experiments show that none of the polynomials with $r = 2^d - 1$ outperforms our method, as predicted by Corollary 8. For testing our method, we chose to approximate the ReLU function $f_1(x) = \mathrm{ReLU}(x)$ (2.50), the (scaled) hyperbolic tangent $f_2(x) = \tanh(5x)$, and a sigmoid function $f_3(x) = 1/1+e^{-10x}$ on different symmetric (i.e., $a = b$) and non-symmetric intervals of the form $\mathcal{I} := [a, b]$. We noted that odd functions on symmetric intervals tend to be approximated by odd polynomials. This leads to the fact that the leading integer coefficient of the polynomials proposed by us is typically $\hat{c}^*_r = 0$ for this case, since $\hat{c}^*_r \neq 0$ for $r = 2^d$ would lead to an even polynomial. Thus, for odd functions on symmetric intervals, our method is as good as existing methods. Whereas for most other cases, we observed a significant improvement. In summary, the experiments show improvements in 86 of the 210 test cases considered.

3.2 Analysis of Neural Network-based Controller

While Section 3.1 was devoted to the design and training of NNs tailored to the specific requirements of control systems, we now shift the focus to the second central problem from the introduction, namely the analysis of NN–based controllers. The importance of addressing this problem lies in the fact that, even if an NN has been carefully designed and trained, its practical use ultimately depends on whether one can certify properties such as convergence, stability, and constraint satisfaction. Without such guarantees, the deployment of NN-based controllers in safety-critical applications would remain highly limited.

In this part of the chapter, we therefore present two analysis methods. The first is based on computing the approximation error of an NN with respect to a stabilizing base-line controller. This method, described in Section 3.2.2, extends a framework developed in [16] for analyzing ReLU NN-based approximations of control laws of linear MPC. The second method does not need a stabilizing base-line controller and instead builds on the computation of reachable sets of the closed-loop system controlled by an NN, thereby providing a more general and flexible analysis framework. The results presented in this section are based on the published articles [P3, P4, P6, P7], which are reprinted in Part II of this thesis.

3.2.1 Problem Overview

MPC is a powerful control method that is used successfully in many applications (see, e.g., [58] for an overview). However, since the evaluation of MPC requires solving an OP at every time step, implementing MPC on hardware with limited computational resources remains challenging. A common approach to mitigating this challenge is to use a lightweight approximation of MPC for implementation, where lightweight means an approximation that is fast to evaluate and has a low memory footprint. NNs are capable of offering approximations with the desired features. Especially, NNs that share the PWA input-to-output structure of the optimal control law (2.34) of MPC for linear systems seem to be promising. As discussed in Section 2.3.2, ReLU and maxout NNs are continuous PWA functions and therefore well-suited for approximating linear MPC. In fact, there are several approaches for the approximation of linear MPC using ReLU NNs [6, 7, 13, 16, 26]. Some of them [7, 26] even use the fact that the control law and ReLU NNs are PWA functions for the design of these NNs.

Now, when an NN $\Phi(\boldsymbol{x})$ is trained, using a gradient-based optimization method, to approximate linear MPC $\boldsymbol{\pi}(\boldsymbol{x})$, then there will be a deviation between the optimal control law and the NN. Meaning we have

$$\boldsymbol{e}(\boldsymbol{x}) := \boldsymbol{\pi}(\boldsymbol{x}) - \Phi(\boldsymbol{x}) \neq \mathbf{0} \tag{3.39}$$

for some $\boldsymbol{x} \in \mathcal{F}$. The error may lead to a loss of the guarantees of linear MPC, such as stability and constraint satisfaction, for the NN-based approximation.

There are various methods for recovering these guarantees. These methods include projecting the output of the NN onto a control invariant set [6, 95], or using the output of the NN as an initial guess to warm start a solver for solving the OCP (2.30). Both methods require additional computations, which may jeopardize the benefit in evaluation speed gained by the NN. Ideally, we want to certify the stability of a system that is directly controlled by an NN without relying on additional computations. Therefore, we focus on methods that enable direct analysis of the closed-loop behavior of systems controlled by NNs. The most widely used approaches in this context are based on MIP [13, 38], semi-definite programming [39], or satisfiability modulo theory [37]. Among these methods, the MIP-based ones provide the highest flexibility, as they allow us to exactly describe the error function (3.39) and they are applicable to commonly used PWA activations. Moreover, as we will detail in Section 3.2.3, the MIP-based methods even allow extending the analysis to nonlinear systems controlled by NNs.

3.2.2 Methods Based on Approximation Error

The first MIP-based method we consider enables the computation of the maximum approximation error of a maxout NN with respect to the optimal control law of linear MPC, as well as the computation of the minimum Lipschitz constant associated with this error. The method was first presented in [16] for ReLU approximations of linear MPC and later extended in [13] to include all kinds of control laws that can be represented by MI linear constraints. As already detailed in Section 2.4.1.1, the method introduced in [16] combines the MI linear constraints (2.33b)–(2.33f) that model the OCP (2.30) with the MI linear constraints (2.61) for the ReLU NN-based approximation to reformulate

$$e_{\max} := \max_{\boldsymbol{x}\in\mathcal{F}} \|\boldsymbol{e}(\boldsymbol{x})\|_p = \max_{\boldsymbol{x}\in\mathcal{F}} \|\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{\pi}(\boldsymbol{x})\|_p \tag{3.40}$$

as the MILP

$$\begin{aligned} e_{\max} = \max_{\substack{\hat{\boldsymbol{u}}_N^*,\boldsymbol{r}^*,\boldsymbol{\lambda}^*,\boldsymbol{\delta}^* \\ \boldsymbol{q}^{(0)},\dots,\boldsymbol{q}^{(l)},\boldsymbol{\beta}^{(1)},\dots,\boldsymbol{\beta}^{(l)}}} & \left\| \boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} - \boldsymbol{S}\hat{\boldsymbol{u}}_N^* \right\|_p && \text{(3.41a)} \\ \text{s.t.} \quad & (2.33\text{b})\text{–}(2.33\text{f}) && \text{(3.41b)} \\ & (2.61) && \text{(3.41c)} \end{aligned}$$

with $p \in \{1, \infty\}$, whose optimal solution is the maximum approximation error of a ReLU NN with respect to the optimal control law of linear MPC. In a similar way, it is possible to compute the minimum Lipschitz constant

$$\mathcal{L}_p(\mathcal{T}) := \max_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{T},\boldsymbol{x}\neq\boldsymbol{y}} \frac{\|\boldsymbol{e}(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{y})\|_p}{\|\boldsymbol{x} - \boldsymbol{y}\|_p} \tag{3.42}$$

of (3.39) by solving an MILP of the form

$$c^* = \max_{\boldsymbol{\xi},\boldsymbol{\beta},\boldsymbol{\zeta},\boldsymbol{\delta}} \|\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} - \boldsymbol{G}\boldsymbol{\zeta}\|_p \tag{3.43a}$$

$$\text{s.t.} \quad \boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\delta}) \leq \boldsymbol{0},\ \boldsymbol{\delta} \in \{0,1\}^{N_1}, \tag{3.43b}$$

$$\boldsymbol{h}_\Phi(\boldsymbol{\xi},\boldsymbol{\beta}) \leq \boldsymbol{0},\ \boldsymbol{\beta} \in \{0,1\}^{N_2}, \tag{3.43c}$$

where $\boldsymbol{h}_\Phi(\boldsymbol{\xi},\boldsymbol{\delta})$ and $\boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\beta})$ are affine functions of the continuous optimization variables $\boldsymbol{\xi} \in \mathbb{R}^{N_\xi}$, $\boldsymbol{\zeta} \in \mathbb{R}^{N_\zeta}$ and the binary optimization variables $\boldsymbol{\delta}$, $\boldsymbol{\beta}$. The constraints are such that the local gain (2.35) of the optimal control law can be computed by the MIFP

$$\text{find}\, \boldsymbol{\zeta},\boldsymbol{\delta} \quad \text{s.t.}\ \boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\delta}) \leq \boldsymbol{0},\ \boldsymbol{\delta} \in \{0,1\}^{N_1} \tag{3.44}$$

as $\boldsymbol{G}\boldsymbol{\zeta} = \boldsymbol{K}(\boldsymbol{x})$ and the local gain (2.63) of the NN by the MIFP

$$\text{find}\, \boldsymbol{\xi},\boldsymbol{\beta} \quad \text{s.t.}\ \boldsymbol{h}_\Phi(\boldsymbol{\xi},\boldsymbol{\beta}) \leq \boldsymbol{0},\ \boldsymbol{\beta} \in \{0,1\}^{N_2}, \tag{3.45}$$

as $\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} = \boldsymbol{K}_\Phi(\boldsymbol{x})$. In [16], it is shown that for ReLU NNs, such constraints and matrices $\boldsymbol{T}$, $\boldsymbol{C}$, $\boldsymbol{G}$ indeed exist. The constraints $\boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\delta}) \leq \boldsymbol{0}$ that model the local gain $\boldsymbol{K}(\boldsymbol{x})$ of the control law $\boldsymbol{\pi}(\boldsymbol{x})$ are in this case given by (2.33b)–(2.33f) and (2.41) ([16, Eq. (13), Eq. (15)]) as in the MIFP (2.42). The constraints for the local gain $\boldsymbol{K}_\Phi(\boldsymbol{x})$ of the ReLU NN are given by [16, Eq. (18), Eq. (20)]. This allows us to reformulate (3.43) as follows

$$c^* = \max_{\boldsymbol{x}\in\mathcal{T}} \|\boldsymbol{K}_\Phi(\boldsymbol{x}) - \boldsymbol{K}(\boldsymbol{x})\|_p$$

we thus have $c^* = \mathcal{L}_p(\mathcal{T})$ [63, Prop. 3.4], which can be computed by an MILP of the form (3.43) [16, Thm. 6.1].

Note that the MILP (3.41) for computing the maximum approximation error can also be cast in the form (3.43) by choosing the constraints (2.33b)–(2.33f) as $\boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\delta}) \leq \boldsymbol{0}$, (2.61) as $\boldsymbol{h}_\Phi(\boldsymbol{\xi},\boldsymbol{\beta}) \leq \boldsymbol{0}$, and the optimization variables as

$$\boldsymbol{\xi} := \begin{pmatrix} \boldsymbol{q}^{(0)} \\ \vdots \\ \boldsymbol{q}^{(l)} \end{pmatrix}, \boldsymbol{\beta} := \begin{pmatrix} \boldsymbol{\beta}^{(1)} \\ \vdots \\ \boldsymbol{\beta}^{(l)} \end{pmatrix}, \boldsymbol{\zeta} := \begin{pmatrix} \hat{\boldsymbol{U}}_N^* \\ \boldsymbol{r}^* \\ \boldsymbol{\lambda}^* \\ \boldsymbol{\delta}^* \end{pmatrix}, \text{ and } \boldsymbol{\delta} := \boldsymbol{\delta}^*.$$

Then with

$$\boldsymbol{T} := \begin{pmatrix} \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{W}^{(l+1)} \end{pmatrix}, \boldsymbol{C} := \boldsymbol{v}^{(l+1)}, \text{ and } \boldsymbol{G} := \begin{pmatrix} \boldsymbol{I}_m & \boldsymbol{0} & \dots & \boldsymbol{0} \end{pmatrix}$$

we have $\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} = \boldsymbol{\Phi}(\boldsymbol{x})$ and $\boldsymbol{G}\boldsymbol{\zeta} = \boldsymbol{\pi}(\boldsymbol{x})$ and thus the optimal value of MILP (3.43) is the maximum approximation error $e_{\max} = c^*$.

In [16, Thm. 3.4], it is shown that if the values of the maximum approximation error $e_{\max}$ and the minimum Lipschitz constant $\mathcal{L}_p(\mathcal{T})$ are below certain thresholds, which depend on values of the OCP, then the closed-loop system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\Phi(\boldsymbol{x}) \tag{3.46}$$

converges exponentially to the origin. In summary, depending on two values that can be computed by solving an MILP, it is possible to certify the stability of a linear system controlled by a ReLU NN.

Our aim is to extend this method to make it applicable to a wider range of NNs and to increase the efficiency of the MILP (3.43) by reducing the number of binary variables.

3.2.2.1 Summary of Articles [P3, P4]

We will now summarize our extension from [P3] of the method from [16] to maxout NN approximations of linear MPC. Furthermore, we will introduce our refinement from [P4], which simplifies the computation of the minimum Lipschitz constant (3.42).

Extension to Maxout NNs [P3]: To extend the known method for analyzing NN-based controllers with ReLU activation to maxout NNs, we need to construct constraints $\boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta}) \leq \boldsymbol{0}$, and matrices $\boldsymbol{T}$, $\boldsymbol{C}$ such that we have $\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} = \boldsymbol{\Phi}(\boldsymbol{x})$ and $\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} = \boldsymbol{K}_\Phi(\boldsymbol{x})$, respectively, for a maxout NN $\boldsymbol{\Phi}(\boldsymbol{x})$. Then, by combining these constraints with the known constraints for linear MPC, the MILP (3.43) can be used to compute the maximum approximation error (3.40) and the minimum Lipschitz constant (3.42) for maxout NN approximations. The output of a maxout NN can be modeled by the MI linear constraints already used in our article [P5] and introduced in Section 3.1.2.2 for a slightly different setup. For a maxout NN $\Phi(\boldsymbol{x})$ as defined in Section 2.3 we can formulate the MI linear constraints [P3, Eq. (18)]

$$\boldsymbol{q}_s^{(i)} \leq \boldsymbol{W}_j^{(i)} \boldsymbol{q}^{(i-1)} + \boldsymbol{v}_j^{(i)} + M(1 - \boldsymbol{\beta}_j^{(i)}), \tag{3.47a}$$

$$-\boldsymbol{q}_s^{(i)} \leq -\boldsymbol{W}_j^{(i)} \boldsymbol{q}^{(i-1)} - \boldsymbol{v}_j^{(i)} - \varepsilon(1 - \boldsymbol{\beta}_j^{(i)}), \tag{3.47b}$$

$$\boldsymbol{q}^{(0)} = \boldsymbol{x}, \tag{3.47c}$$

$$\sum_{k \in \mathcal{A}_s^{(i)}} \boldsymbol{\beta}_k^{(i)} = 1 \tag{3.47d}$$

for all $j \in \mathcal{A}_s^{(i)} := \{p_i(s-1)+1, \ldots, p_i s\}$, for all $s \in \{1, \ldots, w_i\}$, and for all $i \in \{1, \ldots, l\}$. As already briefly discussed in Section 3.1.2.2 these constraints ensure that for every maxout neuron, i.e., $j \in \mathcal{A}_s^{(i)}$ there exists exactly one j^* with $\boldsymbol{\beta}_{j^*}^{(i)} = 1$ and $\boldsymbol{\beta}_j^{(i)} = 0$ for all $j \in \mathcal{A}_s^{(i)} \setminus \{j^*\}$. Consequently, we have

$$\begin{aligned} \boldsymbol{q}_s^{(i)} &= \boldsymbol{W}_{j^*}^{(i)} \boldsymbol{q}^{(i-1)} + \boldsymbol{v}_{j^*}^{(i)} \\ \boldsymbol{W}_{j^*}^{(i)} \boldsymbol{q}^{(i-1)} + \boldsymbol{v}_{j^*}^{(i)} &\geq \boldsymbol{W}_j^{(i)} \boldsymbol{q}^{(i-1)} + \boldsymbol{v}_j^{(i)} + \varepsilon \end{aligned}$$

for all $j \in \mathcal{A}_s^{(i)} \setminus \{j^*\}$ and thus

$$\max_{p_i(s-1)+1 \leq j \leq p_i} \{\boldsymbol{W}_j^{(i)} \boldsymbol{q}^{(i-1)} + \boldsymbol{v}_j^{(i)}\} = \boldsymbol{W}_{j^*}^{(i)} \boldsymbol{q}^{(i-1)} + \boldsymbol{v}_{j^*}^{(i)}. \tag{3.48}$$

In a similar way, we can show that (3.48) holds for all maxout neurons in layer $s \in \{1, \dots, w_i\}$ and for all layers $i \in \{1, \dots, l\}$. Thus, a solution to the MIFP

$$\text{find } \boldsymbol{q}^{(0)}, \dots, \boldsymbol{q}^{(l)}, \boldsymbol{\beta}^{(1)}, \dots, \boldsymbol{\beta}^{(l)} \quad \text{s.t. (3.47)}, \ \boldsymbol{\beta} \in \{0,1\}^{N_2}, \tag{3.49}$$

is such that

$$\boldsymbol{W}^{(l+1)}\boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} = \Phi(\boldsymbol{x}) \tag{3.50}$$

holds. Moreover, we can cast the MIFP (3.49) into the form (3.45) to identify the needed matrices $\boldsymbol{T}$ and $\boldsymbol{C}$ as well as the constraints $\boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta}) \leq \boldsymbol{0}$. This leads to the following formal result, which is proved in [P3].

Lemma 9 ([P3, Lem. 2])**.** *Let $\Phi(\boldsymbol{x})$ be a maxout NN, consider the optimization variables and matrices*

$$\boldsymbol{\xi} := \begin{pmatrix} \boldsymbol{q}^{(0)} \\ \vdots \\ \boldsymbol{q}^{(l)} \end{pmatrix}, \boldsymbol{\beta} := \begin{pmatrix} \boldsymbol{\beta}^{(1)} \\ \vdots \\ \boldsymbol{\beta}^{(l)} \end{pmatrix}, \boldsymbol{T} := \begin{pmatrix} \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{W}^{(l+1)} \end{pmatrix}, \boldsymbol{C} = \boldsymbol{v}^{(l+1)}, \tag{3.51}$$

and choose the constraint $\boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta}) \leq \boldsymbol{0}$ as (3.47) with $\varepsilon = 0$. Then, the solution to the MIFP (3.45) is such that $\Phi(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C}$ holds.

Thus, with (3.47) and (3.51), we have the constraints and matrices necessary to compute the maximum error of a maxout NN approximation of linear MPC by solving the MILP (3.43). It remains to derive the constraints and matrices for computing the local gain of a maxout NN by the MIFP (3.45). Such constraints can be used in combination with (2.33b)–(2.33f) and (2.41) to compute the minimum Lipschitz constant of the error by solving the MILP (3.43). For computing the local gain of a maxout NN, we used in [P3] the observation that, based on the MI linear constraints (3.47), the output of a maxout NN can be described by the recursion

$$\begin{aligned} \boldsymbol{y}^{(0)} &= \boldsymbol{x}, \\ \boldsymbol{y}^{(i)} &= \Delta^{(i)}(\boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{v}^{(i)}) \text{ for } 1 \leq i \leq l, \\ \boldsymbol{\Phi}(\boldsymbol{x}) &= \boldsymbol{W}^{(l+1)}\boldsymbol{y}^{(l)} + \boldsymbol{v}^{(l+1)} \end{aligned} \tag{3.52a}$$

with

$$\Delta^{(i)} := \begin{pmatrix} \boldsymbol{\beta}_{1:p_i}^{(i)\top} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & & \vdots \\ \vdots & & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{\beta}_{p_i(w_i-1)+1:p_i w_i}^{(i)\top} \end{pmatrix}, \tag{3.53}$$

where the binary variables $\boldsymbol{\beta}^{(i)}$ are subject to the constraints (3.47). Applying the chain rule for the derivative to the recursion (3.52) results in a local gain of

$$\boldsymbol{K}_\Phi(\boldsymbol{x}) = \boldsymbol{W}^{(\ell+1)} \prod_{i=1}^{\ell} \Delta^{(i)} \boldsymbol{W}^{(i)} \tag{3.54}$$

Thus, using the binary variables from the MI linear constraints (3.47), we can compute the local gain of a maxout NN by (3.54). However, this formulation can not be cast into the MILP formulation (3.43) for computing the minimum Lipschitz constant, since the local gain is not expressed as a linear combination of the optimization variables. In (3.54), the local gain is a polynomial of order l of the binary optimization variables $\boldsymbol{\beta}^{(i)}$. To compute the local gain as a linear combination of the optimization variables, we introduced in [P3, Eq. (23)] the additional constraints

$$-M\boldsymbol{\beta}_h^{(i)} \leq \tilde{\boldsymbol{\xi}}_{h,r}^{(i)} \leq M\boldsymbol{\beta}_h^{(i)}, \tag{3.55a}$$

$$-M(1-\boldsymbol{\beta}_h^{(i)}) \leq \tilde{\boldsymbol{\xi}}_{h,r}^{(i)} - \tilde{\boldsymbol{W}}_{h,r}^{(i)} \leq M(1-\boldsymbol{\beta}_h^{(i)}), \tag{3.55b}$$

$$\boldsymbol{\xi}^{(0)} = \boldsymbol{I} \tag{3.55c}$$

$$\boldsymbol{\xi}_{1,r}^{(i)} = \sum_{\tilde{h}=1}^{p_i} \tilde{\boldsymbol{\xi}}_{\tilde{h},r}^{(i)}, \tag{3.55d}$$

$$\vdots \tag{3.55e}$$

$$\boldsymbol{\xi}_{w_i,r}^{(i)} = \sum_{\tilde{h}=p_i(w_i-1)+1}^{p_i w_i} \tilde{\boldsymbol{\xi}}_{\tilde{h},r}^{(i)}, \tag{3.55f}$$

$$\tilde{\boldsymbol{W}}^{(i)} = \boldsymbol{W}^{(i)}\boldsymbol{\xi}^{(i-1)}, \tag{3.55g}$$

for all $h \in \{1,\dots,p_i w_i\}$, for all $r \in \{1,\dots,n\}$, and for all $i \in \{1,\dots,l\}$. We proved in [P3, Lem. 3] that these constraints are such that the solution of

$$\text{find } \boldsymbol{q}^{(0)},\dots,\boldsymbol{q}^{(l)},\boldsymbol{\beta}^{(1)},\dots,\boldsymbol{\beta}^{(l)},\boldsymbol{\xi}^{(0)},\dots,\boldsymbol{\xi}^{(l)},\tilde{\boldsymbol{W}}^{(1)},\dots,\tilde{\boldsymbol{W}}^{(l)} \tag{3.56a}$$

$$\text{s.t. (3.47), (3.55), } \boldsymbol{\beta} \in \{0,1\}^{N_2}, \tag{3.56b}$$

can be used to compute the local gain of maxout NN (3.54) as a linear combination of the optimization variables from the MIFP (3.56).

Lemma 10 ([P3, Lem. 3])**.** *Let $\boldsymbol{\Phi}(\boldsymbol{x})$ be a maxout NN and consider the MIFP* (3.56)*. Then, the local gain of $\boldsymbol{\Phi}(\boldsymbol{x})$* (3.54) *is given by*

$$\boldsymbol{K}_{\Phi}(\boldsymbol{x}) = \boldsymbol{W}^{(\ell+1)}\boldsymbol{\xi}^{(\ell)}. \tag{3.57}$$

From Lemma 10 we can directly deduce that there exist matrices $\boldsymbol{T}$, $\boldsymbol{C}$, and MI linear constraints $\boldsymbol{h}_{\Phi}(\boldsymbol{\xi},\boldsymbol{\beta})$ in the form of (3.47) and (3.55) such that the MIFP (2.62) can be used to compute the local gain of a maxout NN by $\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} = \boldsymbol{K}_{\Phi}(\boldsymbol{x})$. Consequently, the minimum Lipschitz constant of the error can be computed by solving an MILP of the form (3.43) (cf. [P3, Thm. 4]).

Ultimately, the presented extension, summarized in Lemma 9 and 10, allows us to compute the maximum approximation error (3.40) and the minimum Lipschitz constant of the error (3.42) for a maxout NN approximation of linear MPC. However, the MILP (3.43) that needs to be solved may scales badly with

the size of the NN and the size, i.e., number of constraints, of the underlying OCP (2.30). An effective way to simplify the computation is to reduce the number of binary variables needed in the MILP. Since the MILP (3.43) includes two sets of constraints $\boldsymbol{h}_\pi(\boldsymbol{\zeta}, \boldsymbol{\delta})$ and $\boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta})$ that are independent of each other, one for the OCP and one for the maxout NN, we can either simplify the computation by reducing the number of binary variables needed to model the OCP or the NN. In the following, we will focus on the MI linear constraints for the OCP. More specifically, we will present approaches for simplifying the computation of the minimum Lipschitz constant for linear MPC, since this value is of interest not just in the context of learning-based control but also in general for MPC [61, 62, 96].

MI Linear Constraints for an Efficient Norm Computation [P4]: We now present the simplification for computing the minimum Lipschitz constant developed in [P4]. The simplification builds on two main concepts. The first is a more efficient implementation of the p-norm computation, with $p \in \{1, \infty\}$, occurring in the cost function of (3.43). The second exploits saturation effects and symmetries often encountered in the optimal control law of linear MPC.

For computing the minimum Lipschitz constant $\mathcal{L}_p(\mathcal{T})$ via the MILP (3.43), we need to reformulate the p-norm in the cost function by MI linear constraints such that the p-norm can be computed by solving an MIFP. For a matrix $\boldsymbol{K} \in \mathbb{R}^{m \times n}$, known methods for reformulating $\|\boldsymbol{K}\|_p$ require $n(1+2m)$ and $m(1+2n)$ binary variables for $p = 1$ and $p = \infty$, respectively [16]. In [P4, Eq. (13)] we derived the constraints

$$\boldsymbol{K} = \boldsymbol{K}^+ - \boldsymbol{K}^-, \tag{3.58a}$$

$$\boldsymbol{0} \leq \boldsymbol{K}^+ \leq M \begin{pmatrix} \boldsymbol{\psi}^{(1)} & \dots \boldsymbol{\psi}^{(m)} \end{pmatrix}^\top, \tag{3.58b}$$

$$\boldsymbol{0} \leq \boldsymbol{K}^- \leq M \left(\boldsymbol{1} - \begin{pmatrix} \boldsymbol{\psi}^{(1)} & \dots & \boldsymbol{\psi}^{(m)} \end{pmatrix}^\top \right), \tag{3.58c}$$

$$\boldsymbol{c}^\top = \boldsymbol{1}^\top (\boldsymbol{K}^+ + \boldsymbol{K}^-), \tag{3.58d}$$

$$\boldsymbol{c} \leq \boldsymbol{1}k \leq \boldsymbol{c} + M(\boldsymbol{1} - \boldsymbol{\psi}^{(m+1)}), \tag{3.58e}$$

$$\boldsymbol{1}^\top \boldsymbol{\psi}^{(m+1)} = 1, \tag{3.58f}$$

$$\boldsymbol{\psi}^{(1)}, \dots, \boldsymbol{\psi}^{(m+1)} \in \{0,1\}^n \tag{3.58g}$$

where M is a constant that is larger than the largest absolute value of the entries in $\boldsymbol{K}$, and proved in [P4, Lem. 1] that any solution to

$$\text{find } k, \boldsymbol{K}^+, \boldsymbol{K}^-, \boldsymbol{c}, \boldsymbol{\psi}^{(1)}, \dots, \boldsymbol{\psi}^{(m+1)} \quad \text{s.t. (3.58)} \tag{3.59}$$

is such that $k = \|\boldsymbol{K}\|_1$. Using the relation $\|\boldsymbol{K}\|_\infty = \|\boldsymbol{K}^\top\|_1$ we can find similar constraints for computing the ∞-norm. Thus, the MI linear constraints (3.58) allow to reformulate the MIP (3.43), which is used for computing the maximum

error and the minimum Lipschitz constant of the error function, as the MILP

$$
\begin{aligned}
c^* = \max_{\xi,\beta,\zeta,\delta} \; & k && \text{(3.60a)}\\
\text{s.t.} \quad & \boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\delta}) \leq \mathbf{0},\ \boldsymbol{\delta} \in \{0,1\}^{N_1}, && \text{(3.60b)}\\
& \boldsymbol{h}_\Phi(\boldsymbol{\xi},\boldsymbol{\beta}) \leq \mathbf{0},\ \boldsymbol{\beta} \in \{0,1\}^{N_2}, && \text{(3.60c)}\\
& \boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} - \boldsymbol{G}\boldsymbol{\zeta} = \boldsymbol{K}^+ - \boldsymbol{K}^-, && \text{(3.60d)}\\
& \text{(3.58b)–(3.58g)} && \text{(3.60e)}
\end{aligned}
$$

which has $n(m+1)$ additional binary variables instead of $n(1+2m)$, which would be needed when known methods are used for the reformulation. The reformulation from (3.43) to (3.60) is possible since according to [P4, Lem. 1] we can replace the last two constraints (3.60d) and (3.60e) by

$$k = \|\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C} - \boldsymbol{G}\boldsymbol{\zeta}\|_1 .$$

The constraints (3.58) for computing the p-norm are the first simplification presented in our article [P4] and allow us to reduce the number of binary variables needed in the MILP (3.60) by mn.

Simplification of the Minimum Lipschitz Constant Computation [P4]: The second simplification aims to reduce the complexity of the MILP for computing the minimum Lipschitz constant of the optimal control law, i.e., we want to reduce the number of binary variables needed in the MIFP (2.33) to compute the solution of the OCP (2.30). For convenience, we recap here the MIFP

$$
\begin{aligned}
& \text{find } \hat{\boldsymbol{U}}_N^*, \boldsymbol{r}^*, \boldsymbol{\lambda}^*, \boldsymbol{\delta}^* && \text{(3.61a)}\\
\text{s.t.} \quad & \mathbf{0} = \boldsymbol{H}\hat{\boldsymbol{U}}_N^*(\boldsymbol{x}) + \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}^\top \boldsymbol{\lambda}^*, && \text{(3.61b)}\\
& \boldsymbol{r}^* = \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d} - \boldsymbol{G}\hat{\boldsymbol{U}}_N^*(\boldsymbol{x}), && \text{(3.61c)}\\
& \mathbf{0} \leq \boldsymbol{r}^* \leq \operatorname{diag}(\overline{\boldsymbol{r}})(\mathbf{1} - \boldsymbol{\delta}^*), && \text{(3.61d)}\\
& \mathbf{0} \leq \boldsymbol{\lambda}^* \leq \operatorname{diag}(\overline{\boldsymbol{\lambda}})\,\boldsymbol{\delta}^*, && \text{(3.61e)}\\
& \boldsymbol{\delta}^* \in \{0,1\}^{N_c}. && \text{(3.61f)}
\end{aligned}
$$

introduced in Section 2.2.1 that can be used to compute the solution of the OCP (2.30). Now, since the constraints (3.61b)–(3.61f) are part of $\boldsymbol{h}_\pi(\boldsymbol{\zeta},\boldsymbol{\delta})$ in MILP (3.43) for computing the minimum Lipschitz constant of the error, reducing the number of binary variables in (3.61) would allow for a more efficient computation of the minimum Lipschitz constant. In [P4], we proved that the number of binary variables in (3.61) can be reduced based on the following theorem.

Theorem 11 ([P4, Thm. 3, Cor. 4]). *Let $i \in \{1,\dots,N_c\}$ and let $\mathcal{F}$ be full-dimensional.*

Consider the MILP

$$\max_{\substack{\boldsymbol{U}^*,\boldsymbol{U}^{(1)},\dots,\boldsymbol{U}^{(m)},\boldsymbol{x},\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(m)},\\ \boldsymbol{\lambda}^*,\boldsymbol{\lambda}^{(1)},\dots,\boldsymbol{\lambda}^{(m)},\boldsymbol{r}^*,\boldsymbol{r}^{(1)},\dots,\boldsymbol{r}^{(m)},\boldsymbol{\delta}^*}} \sum_{j=1}^{m} \boldsymbol{U}_j^{(j)} - \boldsymbol{U}_j^* \tag{3.62a}$$

$$\text{s.t. (3.61b)–(3.61f)}, \boldsymbol{\delta}_i^* = 1, \tag{3.62b}$$

$$\boldsymbol{H}\boldsymbol{U}^{(j)} + \boldsymbol{F}\boldsymbol{x}^{(j)} + \boldsymbol{G}^\top \boldsymbol{\lambda}^{(j)} = \boldsymbol{0}, \tag{3.62c}$$

$$\boldsymbol{E}\boldsymbol{x}^{(j)} + \boldsymbol{d} - \boldsymbol{G}\boldsymbol{U}^{(j)} = \boldsymbol{r}^{(j)}, \tag{3.62d}$$

$$\boldsymbol{0} \leq \boldsymbol{r}^{(j)} \leq M(\boldsymbol{1} - \boldsymbol{\delta}^*), \tag{3.62e}$$

$$\boldsymbol{0} \leq \boldsymbol{\lambda}^{(j)} \leq M\,\boldsymbol{\delta}^* \tag{3.62f}$$

with $\boldsymbol{x}^{(j)} := \boldsymbol{x} + \boldsymbol{e}^{(j)}$ *for every* $j \in \{1, \dots, m\}$*, where* $\boldsymbol{e}^{(j)}$ *is the j-th orthonormal vector of the canonical basis. Then the i-th constraint can be omitted for the computation of the Lipschitz constant* $\max_{\boldsymbol{x}\in\mathcal{F}} \|\boldsymbol{K}(\boldsymbol{x})\|_p$ *if*

1. *the MILP* (3.62) *is infeasible*

or

2. *the MILP* (3.62) *is feasible and returns* 0 *as the optimal objective function value.*

To interpret the result stated in Theorem 11, we first note that the MIFP (3.61) has one binary variable for each of the N_c constraints in the OCP (2.30). The binary variables are such that we have $\boldsymbol{\delta}_i^* = 1$ if the associated constraint is active, i.e., for $\boldsymbol{r}_i^* = 0$. Meaning that if (3.62) is infeasible for $\boldsymbol{\delta}_i^* = 1$, as stated in the theorem, then the i-th constraint is inactive for all $\boldsymbol{x} \in \mathcal{F}$. Thus, we can omit the i-th constraint or equivalently set $\boldsymbol{\delta}_i^* = 0$ without altering the solution of the MIFP (3.61). The second case of Theorem 11 refers to the case of a saturating control law with $\boldsymbol{K}(\boldsymbol{x}) = \boldsymbol{0}$, as shown in Figure 3.10. We proved in [P4, Thm. 3] that if the MILP (3.62) is feasible and returns 0 as the optimal objective function value, then the control law saturates, i.e., $\boldsymbol{K}(\boldsymbol{x}) = \boldsymbol{0}$ for $\boldsymbol{\delta}_i^* = 1$. Since the minimum Lipschitz constant is $\mathcal{L} := \max_{\boldsymbol{x}\in\mathcal{F}} \|\boldsymbol{K}(\boldsymbol{x})\|_p > 0$, we can omit the i-th constraint or equivalently set $\boldsymbol{\delta}_i^* = 0$ for the computation of the minimum Lipschitz constant if the 2. case of Theorem 11 applies.

In our article [P4], we also showed that the computation of the minimum Lipschitz constant can be further simplified if the optimal control law exhibits symmetries of the form [97, Def. 1]

$$\boldsymbol{\Omega}^{(i)}\boldsymbol{\pi}(\boldsymbol{x}) = \boldsymbol{\pi}(\boldsymbol{\Theta}^{(i)}\boldsymbol{x}) \tag{3.63}$$

for every $\boldsymbol{x} \in \mathcal{F}$ where $\boldsymbol{\Theta}^{(i)}$ and $\boldsymbol{\Omega}^{(i)}$ are invertible matrices for all $i \in \{1, \dots, \sigma\}$. Figure 3.10 shows the optimal control law for a double integrator with rotational symmetry. In this case, we have $\sigma = 1$ symmetry with

$$\boldsymbol{\Theta}^{(1)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \text{ and } \boldsymbol{\Omega}^{(1)} = -1. \tag{3.64}$$

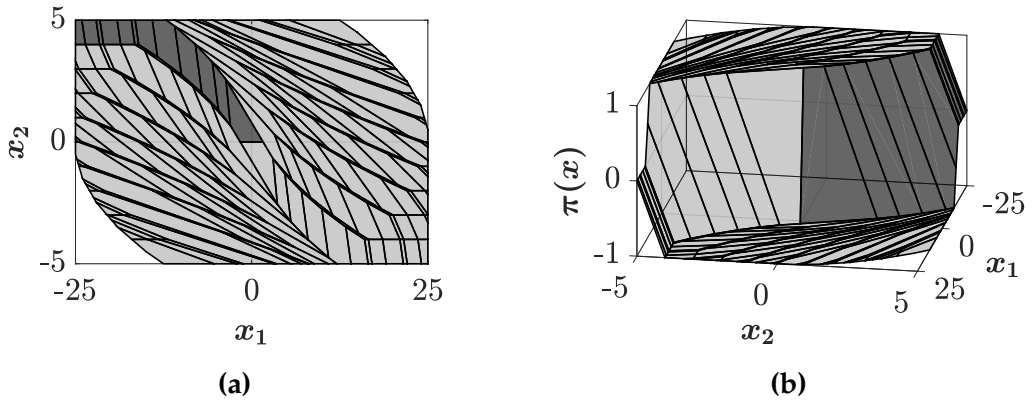


Figure 3.10. Optimal control law of linear MPC for a double integrator. Due to saturation and symmetries, only the regions highlighted in dark gray are relevant for computing the minimum Lipschitz constant of $\boldsymbol{\pi}(\boldsymbol{x})$.

Thus, only the regions with $\boldsymbol{x}_2 \geq 0$ in Figure 3.10 are relevant for computing the minimum Lipschitz constant. The local gain of all regions with $\boldsymbol{x}_2 < 0$ can be recovered from the control law of the upper state space, i.e., $\boldsymbol{x}_2 \geq 0$, by using the symmetry (3.63). For symmetries with $\|\boldsymbol{\Theta}^{(i)}\|_p = \|\boldsymbol{\Omega}^{(i)}\|_p = 1$ for all $i \in \{1, \ldots, \sigma\}$, we showed in [P4, Lem. 5] that for computing the minimum Lipschitz constant, only regions in the so-called fundamental domain $\mathcal{X}_{\text{fun}}$ [97] with

$$\mathcal{F} \subseteq \bigcup_{i=1}^{\sigma} \boldsymbol{\Theta}^{(i)} \mathcal{X}_{\text{fun}} \tag{3.65}$$

are relevant, since the local gain of all other regions can be computed based on the local gain of regions inside the fundamental domain using (3.63). Thus, by adding the constraint $\boldsymbol{x} \in \mathcal{X}_{\text{fun}}$ to the MILP (3.62), we can identify constraints that are always inactive for $\boldsymbol{x} \in \mathcal{X}_{\text{fun}}$ and thus can be omitted for symmetric control laws. As validated experimentally in [P4, Sec. IV] for several examples, this typically leads to a significant simplification since only a fraction of the regions need to be considered. For the example from Figure 3.10, a fundamental domain is $\mathcal{X}_{\text{fun}} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid |\boldsymbol{x}_1| \leq 25, 0 \leq \boldsymbol{x}_2 \leq 5\}$, this halves the number of relevant regions. If, in addition, the 2. case of Theorem 11 is used to detect saturation, then all regions with $\boldsymbol{K}(\boldsymbol{x}) = \boldsymbol{0}$ are omitted and only the dark gray regions are considered in the MILP for the computation of the minimum Lipschitz constant.

3.2.2.2 Discussion

Our articles [P3] and [P4] extend and simplify MI-based methods for computing the approximation error and the minimum Lipschitz constant of NN approximation of linear MPC. The extension presented in [P3] and summarized in Section 3.2.2.1 can be seen as a generalization of the results for ReLU approximations [13, 16], since it includes the ReLU approximations as a special

case. A ReLU function with parameters $\boldsymbol{W}^{(i)}$ and $\boldsymbol{v}^{(i)}$ of the form (2.50) can always be written as a maxout activation with $p_i = 2$ by using

$$\begin{pmatrix} \max\{0, \boldsymbol{W}_1^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{v}_1^{(i)}\} \\ \vdots \\ \max\{0, \boldsymbol{W}_{w_i}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{v}_{w_i}^{(i)}\} \end{pmatrix} = \begin{pmatrix} \max\limits_{1 \le j \le p_i} \{\tilde{\boldsymbol{W}}_j^{(i)}\boldsymbol{y}^{(i-1)} + \tilde{\boldsymbol{v}}_j^{(i)}\} \\ \vdots \\ \max\limits_{p_i(w_i-1) \le j \le p_i w_i} \{\tilde{\boldsymbol{W}}_j^{(i)}\boldsymbol{y}^{(i-1)} + \tilde{\boldsymbol{v}}_j^{(i)}\} \end{pmatrix} \tag{3.66}$$

where the parameters of the maxout activation are

$$\tilde{\boldsymbol{W}}^{(i)} = \begin{pmatrix} \boldsymbol{W}_1^{(i)} \\ \boldsymbol{0}^\top \\ \vdots \\ \boldsymbol{W}_{w_i}^{(i)} \\ \boldsymbol{0}^\top \end{pmatrix} \text{ and } \tilde{\boldsymbol{v}}^{(i)} = \begin{pmatrix} \boldsymbol{v}_1^{(i)} \\ 0 \\ \vdots \\ \boldsymbol{v}_{w_i}^{(i)} \\ 0 \end{pmatrix}.$$

Thus, a given ReLU NN can always be transformed to a maxout NN with the same topology as the original ReLU NN, by replacing all ReLU activations in the NN by maxout activations using the relation (3.66). The transformation from maxout NN to ReLU NN is not possible without changing the topology of the NN, since, e.g., a maxout NN with one layer and $p_i \geq 3$ can only be represented by a ReLU NN with at least two layers [25]. Consequently, methods developed for maxout NN can be applied directly to the analysis of ReLU NN, but the converse is not possible.

Regarding the simplifications presented in Section 3.2.2.1, we showed in several experiments, summarized in [P4, Tbl. II], that the number of binary variables in the MIFP (3.61) can be significantly reduced. This reduction in the number of binary variables decreased the computation time for the minimum Lipschitz constant of the control law on average by a factor of 58 and at most by a factor 598 compared to the computation without simplifications.

Implications Beyond NN-based Controllers: The presented results are primarily used to analyze NN-Based controllers by computing the maximum error (3.40) and the minimum Lipschitz constant of the error function (3.42). The MI linear constraints $\boldsymbol{h}_\pi(\boldsymbol{\zeta}, \boldsymbol{\delta}) \leq \boldsymbol{0}$ for the optimal control law and $\boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta}) \leq \boldsymbol{0}$ for the NN are used in combination in the MILP (3.60) to perform this analysis. However, the constraints $\boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta}) \leq \boldsymbol{0}$ can also be used individually in an MILP of the form

$$c_\Phi^* = \max_{\boldsymbol{\xi}, \boldsymbol{\beta}} \pm \|\boldsymbol{T}\boldsymbol{\xi} + \boldsymbol{C}\|_p \tag{3.67a}$$

$$\text{s.t.} \quad \boldsymbol{h}_\Phi(\boldsymbol{\xi}, \boldsymbol{\beta}) \leq \boldsymbol{0}, \ \boldsymbol{\beta} \in \{0,1\}^{N_2}, \tag{3.67b}$$

to analyze NNs in general and not just in the context of control. The $\pm$ in front of the cost function indicates that the MILP can also be used for minimization.

In a similar way, we can use $\boldsymbol{h}_\pi(\boldsymbol{\zeta}, \boldsymbol{\delta}) \leq \mathbf{0}$ individually in an MILP of the form

$$c_\pi^* = \max_{\boldsymbol{\zeta}, \boldsymbol{\delta}} \pm \|\boldsymbol{G}\boldsymbol{\zeta}\|_p \tag{3.68a}$$

$$\text{s.t.} \quad \boldsymbol{h}_\pi(\boldsymbol{\zeta}, \boldsymbol{\delta}) \leq \mathbf{0}, \ \boldsymbol{\delta} \in \{0,1\}^{N_1} \tag{3.68b}$$

to analyze the optimal control law. The MILPs allow, e.g., to compute the maximum and minimum of $\boldsymbol{\pi}(\boldsymbol{x})$ and $\boldsymbol{\Phi}(\boldsymbol{x})$, respectively. Moreover, we can use (3.67) and (3.68) to compute Lipschitz constants for the optimal control law and the NN, respectively, which are useful for analyzing robustness of the functions against perturbations in the input [36, 61, 62, 98].

The method presented in Theorem 11 can also be used beyond its presented application for simplifying the computation of the minimum Lipschitz constant. Especially the first case of the theorem, i.e., when the MILP (3.62) is infeasible, is of interest for an efficient implementation of MPC. By identifying all binary variables that lead to infeasibility, we can determine all constraints of the OCP (2.30) that are inactive for all $\boldsymbol{x} \in \mathcal{F}$ [P4, Lem. 2]. Consequently, these constraints can be removed from the original OCP to create a new OCP with fewer constraints that has the same optimal solution. This reduction of constraints enables a more efficient implementation of MPC, as only relevant constraints are included in the OCP.

3.2.3 Methods Based on Reachability Analysis

The methods discussed in Section 3.2.2 rely on analyzing the approximation error (3.39) of an NN with respect to a stabilizing base-line controller, such as linear MPC. These methods, therefore, require knowledge of the controller that the NN is intended to approximate. This is problematic if the NN is not trained by the user of the controller. A further limitation is that large approximation errors may prevent certification of stability, even if the closed-loop system remains stable in practice. For example, consider two stabilizing control laws, $\boldsymbol{\pi}_1(\boldsymbol{x})$ and $\boldsymbol{\pi}_2(\boldsymbol{x})$, obtained from solving the OCP (2.30) with different parameters $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\mathcal{T}$, and N. If an NN exactly represents $\boldsymbol{\pi}_2(\boldsymbol{x})$, i.e., $\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{\pi}_2(\boldsymbol{x})$, then the resulting NN-based controller is stabilizing. However, evaluating its error relative to $\boldsymbol{\pi}_1(\boldsymbol{x})$ produces a large maximum error, which may lead to a false conclusion of instability. Consequently, precise knowledge of the parameters used in generating the training data is crucial for error-based methods.

Due to the aforementioned drawbacks of the error-based methods, we investigated in our articles [P6, P7] methods where reachable sets of the closed-loop system $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k))$ with NN-based controller $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ are used for the analysis [38, 40]. Reachable sets describe the set of all states that can be reached by the closed-loop system starting in some given set $\mathcal{F}$ in the

k-th time step. Formally, they are defined by

$$\mathcal{R}_0(\mathcal{F}) := \mathcal{F} \tag{3.69a}$$
$$\mathcal{R}_{k+1}(\mathcal{F}) := \{\boldsymbol{x} \mid \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{\Phi}(\boldsymbol{x}(k))), \boldsymbol{x}(k) \in \mathcal{R}_k(\mathcal{F})\}. \tag{3.69b}$$

As discussed in Section 2.4.1.1, reachable sets possess several useful features that we will now utilize to certify convergence, stability, and constraint satisfaction of the closed-loop system.

3.2.3.1 Summary of Articles [P6, P7]

We consider a PWA system of the form

$$\boldsymbol{x}(k+1) = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{u}(k)) + \boldsymbol{d}(k) \tag{3.70}$$

where $\boldsymbol{f}_{\text{PWA}}$ is defined as in (2.43) and $\boldsymbol{d}(k)$ is bounded by a state- and input-dependent polyhedral set $\mathcal{D}^{(i)} \subset \mathbb{R}^n$, i.e.,

$$\boldsymbol{d}(k) \in \mathcal{D}^{(i)} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{\mathcal{D}^{(i)}} \boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{D}^{(i)}}\} \text{ if } \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \in \mathcal{P}^{(i)} \tag{3.71}$$

for all $i \in \{1, \ldots, s\}$ with $\mathcal{P}^{(i)}$ as in (2.44). The analysis of NN-based controllers presented in our articles [P6, P7] mainly builds on reachable sets defined by

$$\mathcal{R}_0^{\mathcal{D}}(\mathcal{A}) := \mathcal{A}, \tag{3.72a}$$
$$\mathcal{R}_{k+1}^{\mathcal{D}}(\mathcal{A}) := \{\boldsymbol{x}^+ \mid \boldsymbol{x}^+ = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x})) + \boldsymbol{d}, \boldsymbol{x} \in \mathcal{R}_k(\mathcal{F}), \boldsymbol{d} \in \mathcal{D}^{(i)}\} \tag{3.72b}$$

and robustly positively invariant (RPI) sets $\mathcal{F}$ of the controlled PWA system with disturbances. Based on the reachable sets, it is possible to verify whether a set is RPI [75]. A set $\mathcal{F}$ is RPI if

$$\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{F} \tag{3.73}$$

holds and positively invariant (PI) if (3.73) with $\mathcal{D}^{(i)} = \{\boldsymbol{0}\}$ holds for all $i \in \{1, \ldots, s\}$. Moreover, if an RPI set $\mathcal{F} \subseteq \mathcal{X}$ is in addition such that there exists a finite $k^* \in \mathbb{N}$ and a small RPI set $\mathcal{T}$ that contains the origin with

$$\mathcal{R}_{k^*}^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{T} \tag{3.74}$$

then we know that all closed-loop trajectories that start in $\mathcal{F}$ will remain in $\mathcal{F} \subseteq \mathcal{X}$ and thus in $\mathcal{X}$ for all time steps. Consequently, the state constraints hold for all time steps. Moreover, from (3.74) we can deduce that for $\boldsymbol{x}(0) \in \mathcal{F}$ we have $\boldsymbol{x}(k) \in \mathcal{T}$ for all $k \geq k^*$. Thus, all closed-loop trajectories starting in $\mathcal{F}$ will enter $\mathcal{T}$ after at most k^* time steps and will remain in $\mathcal{T}$ for all subsequent time steps.

Results for Disturbed PWA Systems Controlled by NNs: In the remainder of this section, we will present several illustrative examples that demonstrate the main results. For clarity, the specific parameters of the underlying systems and NNs are omitted, as they are not essential for illustrating the core concepts. For the details, we refer to the case studies in articles [P6, Sec. IV] and [P7, Sec. V]. The first example is given in Figure 3.11, which shows the RPI sets $\mathcal{F} \subseteq \mathcal{X}$ with (3.73) and $\mathcal{T} \subseteq \mathcal{F}$ with (3.74) for a nominal PWA system, i.e., $\boldsymbol{d}(k) = \mathbf{0}$ for all $k \in \mathbb{N}_0$, with $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\boldsymbol{x}\|_\infty \leq 10\}$ as well as three trajectories with $\boldsymbol{x}(0) \in \mathcal{F}$. All these trajectories stay in the PRI set $\mathcal{F}$ and enter $\mathcal{T}$ after a finite number of time steps without violating any constraints. Formally this means that for all $\boldsymbol{x}(0) \in \mathcal{F}$ we have $\boldsymbol{x}(k) \in \mathcal{X}$ for all $k \in \mathbb{N}_0$ and $\boldsymbol{x}(\tilde{k}) \in \mathcal{T}$ for $\tilde{k} \geq k^*$ for the example trajectories in Figure 3.11. In fact, not just the example trajectories but all trajectories with $\boldsymbol{x}(0) \in \mathcal{F}$ show the same behavior. In [P6], we formalized this observation by the following theorem.

Theorem 12 ([P6, Thm. 8])**.** *Let $\mathcal{F} \in \mathcal{X}$ be a set for which* (3.73) *holds and $\mathcal{T}$ an RPI set for which* (3.74) *holds. Then, the system* (3.70) *with NN-based controller $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ is ultimately bounded in $\mathcal{T}$, uniformly in $\mathcal{F}$. Moreover, $\boldsymbol{x}(k) \in \mathcal{X}$ and $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ holds for all $k \in \mathbb{N}_0$.*

We define uniform ultimate boundedness of a system with respect to a C-set (i.e., a convex and compact set containing the origin in its interior) according to [99, Def. 2.4].

Definition 13. *A system is denoted as ultimately bounded in a C-set $\mathcal{T}$, uniformly in $\mathcal{F}$, if for every initial condition $\boldsymbol{x}(0) \in \mathcal{F}$, there exits a $k^*(\boldsymbol{x}(0))$ such that $\boldsymbol{x}(k) \in \mathcal{T}$ holds for all $k \geq k^*(\boldsymbol{x}(0))$.*

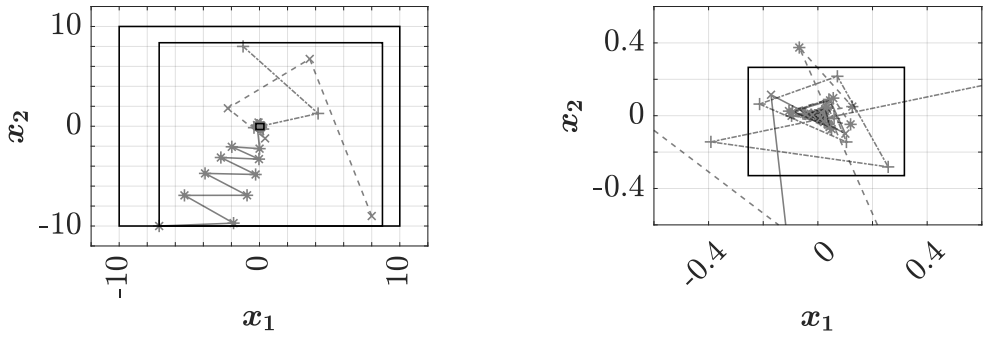


Figure 3.11. The black boxes in (a) and (b) are the state constraints $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\boldsymbol{x}\|_\infty \leq 10\}$ and the RPI sets $\mathcal{F}$ and $\mathcal{T}$ with $\mathcal{T} \subseteq \mathcal{F} \subseteq \mathcal{X}$. The gray lines show example trajectories with $\boldsymbol{x}(0) \in \mathcal{F}$ of the nominal closed-loop system. All trajectories that start in $\mathcal{F}$ stay in that set and enter $\mathcal{T}$ after a finite number of time step without violating any constraints.

Theorem 12 states that if we have sets $\mathcal{F} \subseteq \mathcal{X}$ and $\mathcal{T}$ for which (3.73) and (3.74) hold, then the PWA system (3.70) controlled by an NN can be operated safely in the sense that all constraints are satisfied for all time steps and that all trajectories starting in $\mathcal{F}$ will eventually enter and remain in the set $\mathcal{T}$. However, computing suitable sets for PWA systems controlled by NNs is not trivial, as the nonlinear structure of the system and the NN typically results in non-convex or even non-connected sets [74]. Therefore, we introduced in [P6] over-approximations

$$\overline{\mathcal{R}}_0^{\mathcal{D}}(\mathcal{A}) := \mathcal{A} \tag{3.75a}$$

$$\overline{\mathcal{R}}_{k+1}^{\mathcal{D}}(\mathcal{A}) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}\left(\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A})\right)}(\boldsymbol{C})\} \tag{3.75b}$$

of the k-step reachable set (3.72) with $\mathcal{R}_k^{\mathcal{D}}(\mathcal{A}) \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A})$ for all $k \in \mathbb{N}_0$, which are computed based on the support function

$$\boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(\boldsymbol{C}) := \begin{pmatrix} \boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(\boldsymbol{C}_1) \\ \vdots \\ \boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(\boldsymbol{C}_r) \end{pmatrix} := \begin{pmatrix} \max\limits_{\boldsymbol{x} \in \mathcal{R}_1^{\mathcal{D}}(\mathcal{A})} \boldsymbol{C}_1^\top \boldsymbol{x} \\ \vdots \\ \max\limits_{\boldsymbol{x} \in \mathcal{R}_1^{\mathcal{D}}(\mathcal{A})} \boldsymbol{C}_r^\top \boldsymbol{x} \end{pmatrix} \tag{3.76}$$

of the 1-step reachable set $\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})$. The matrix $\boldsymbol{C} \in \mathbb{R}^{r \times n}$ is a user-chosen parameter that determines the shape of the over-approximation. These over-approximations can be used to compute sets that satisfy (3.73) and (3.74). We showed in our article [P7] that the support function of the 1-step (and k-step) reachable set of disturbed PWA systems controlled by maxout NNs can be evaluated exactly by solving MILPs. For the evaluation, we introduced the MI linear constraints

$$\boldsymbol{H}^{(i)} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{pmatrix} \leq \boldsymbol{h}^{(i)} + \boldsymbol{1}M(1 - \gamma_i), \tag{3.77a}$$

$$\boldsymbol{1}^\top \gamma = 1, \tag{3.77b}$$

$$-\boldsymbol{1}M(1 - \gamma_i) \leq \boldsymbol{A}^{(i)}\boldsymbol{x} + \boldsymbol{B}^{(i)}\boldsymbol{u} + \boldsymbol{p}^{(i)} + \boldsymbol{d} - \boldsymbol{x}_+^{(i)} \leq \boldsymbol{1}M(1 - \gamma_i), \tag{3.77c}$$

$$-\boldsymbol{1}M\gamma_i \leq \boldsymbol{x}_+^{(i)} \leq \boldsymbol{1}M\gamma_i, \tag{3.77d}$$

$$\boldsymbol{H}_{\mathcal{D}^{(i)}}\boldsymbol{d} \leq \boldsymbol{h}_{\mathcal{D}^{(i)}} + \boldsymbol{1}M(1 - \gamma_i), \tag{3.77e}$$

$$\boldsymbol{x}_+ = \sum_{j=1}^{s} \boldsymbol{x}_+^{(i)}, \tag{3.77f}$$

$$\gamma \in \{0, 1\}^s. \tag{3.77g}$$

for all $i \in \{1, \ldots, s\}$. The constraints are derived based on the known MI linear constraints for nominal PWA systems, i.e., systems with $\boldsymbol{d} = \boldsymbol{0}$, by including the constraints (3.77e) and the variable $\boldsymbol{d}$. The additional constraints deactivate irrelevant constraints on the disturbance by adding a large constant on the

right-hand side of (3.77e) for all i with $(\boldsymbol{x}\ \boldsymbol{u})^\top \notin \mathcal{P}^{(i)}$ and thus ensure (3.71). We showed in [P7, Lem. 1] that the MIFP

$$\text{find } \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}_+, \boldsymbol{\gamma} \tag{3.78a}$$
$$\text{s.t. (3.77)}, \ \boldsymbol{x} \in \mathcal{A}, \ \boldsymbol{\gamma} \in \{0,1\}^s. \tag{3.78b}$$

with some polyhedral set $\mathcal{A}$, is feasible if and only if

$$\boldsymbol{x}_+ = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{d}, \ \boldsymbol{x} \in \mathcal{A}, \ \boldsymbol{d} \in \mathcal{D}^{(i)} \tag{3.79}$$

holds. Thus, the system dynamics of a disturbed PWA system can be described by the MI linear constraints (3.77). This allows us to combine the MI linear constraints (3.77) with the MI linear constraints (3.47) for a maxout NN to formulate an MILP that can be used to evaluate the support function (3.76), which results in the MILP

$$c^*(\boldsymbol{C}_1, \mathcal{A}) := \max_{\substack{\boldsymbol{q}^{(0)},\dots,\boldsymbol{q}^{(l)},\boldsymbol{\beta}^{(1)},\dots,\boldsymbol{\beta}^{(l)},\\ \boldsymbol{x},\boldsymbol{u},\boldsymbol{x}_+,\boldsymbol{\gamma}}} \boldsymbol{C}_1 \boldsymbol{x}_+ \tag{3.80a}$$
$$\text{s.t. (3.47), (3.77)}, \tag{3.80b}$$
$$\boldsymbol{u} = \boldsymbol{W}^{(l+1)} \boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)}, \tag{3.80c}$$
$$\boldsymbol{q}^{(0)} = \boldsymbol{x}, \tag{3.80d}$$
$$\boldsymbol{x} \in \mathcal{A}, \tag{3.80e}$$
$$\boldsymbol{\beta} \in \{0,1\}^{N_2}, \boldsymbol{\gamma} \in \{0,1\}^s. \tag{3.80f}$$

As discussed in Section 3.2.2.1, we demonstrated in [P3, Lem. 2] that the constraints (3.47), together with (3.80d), ensure that $\boldsymbol{W}^{(l+1)} \boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)} = \boldsymbol{\Phi}(\boldsymbol{x})$ holds. Moreover, with (3.77) and (3.80c) we have $\boldsymbol{x}_+ = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{d} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{W}^{(l+1)} \boldsymbol{q}^{(l)} + \boldsymbol{v}^{(l+1)}) + \boldsymbol{d} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x})) + \boldsymbol{d}$. Thus we can rewrite the MILP (3.80) as

$$c^*(\boldsymbol{C}_1, \mathcal{A}) = \max_{\boldsymbol{x}, \boldsymbol{x}_+} \boldsymbol{C}_1 \boldsymbol{x}_+ \tag{3.81a}$$
$$\text{s.t. } \boldsymbol{x}_+ = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x})) + \boldsymbol{d}, \boldsymbol{x} \in \mathcal{A}, \boldsymbol{d} \in \mathcal{D}^{(i)}. \tag{3.81b}$$

The constraints of this OP can be summarized by $\boldsymbol{x}_+ \in \mathcal{R}_1^{\mathcal{D}}(\mathcal{A})$ (cf. (3.72)). Consequently, we have $c^*(\boldsymbol{C}_1, \mathcal{A}) = h_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(\boldsymbol{C}_1)$, which allows us to compute the value of the support function of the 1-step reachable set for a polyhedral $\mathcal{A}$ by solving the MILP (3.80). All the previously described reformulation steps for deriving the MILP for the support function of the reachable set are proved in [P7] and formalized in the following lemma.

Lemma 14 ([P7, Lem. 2])**.** *Let the k-step reachable set be defined as in* (3.76)*. Then the MILP* (3.80) *with* $\boldsymbol{C}_1 \in \mathbb{R}^{1 \times n}$*, is such that* $c^*(\boldsymbol{C}_1, \mathcal{A}) = \boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(\boldsymbol{C}_1)$ *holds.*

If we define

$$c^*(C, \mathcal{A}) := \begin{pmatrix} c^*(C_1, \mathcal{A}) \\ \dots \\ c^*(C_r, \mathcal{A}) \end{pmatrix}$$

then we have according to Lemma 14, $c^*(C, \mathcal{A}) = h_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(C)$. Moreover, since the support function $h_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(C_1)$ can be evaluated for arbitrary polyhedral sets $\mathcal{A}$ by solving the MILP (3.80) and the over-approximations $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A})$ (3.75) are for every $k \in \mathbb{N}_0$ polyhedral sets, we can compute $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A})$ iteratively by (3.80). For $k = 1$, this results in

$$\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{A}) = \{x \in \mathbb{R}^n \mid Cx \leq h_{\mathcal{R}_1^{\mathcal{D}}\left(\overline{\mathcal{R}}_0^{\mathcal{D}}(\mathcal{A})\right)}(C)\} = \{x \in \mathbb{R}^n \mid Cx \leq h_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{A})}(C)\}$$

and thus we have to solve the MILP as in (3.80). For $k = 2$ we have

$$\overline{\mathcal{R}}_2^{\mathcal{D}}(\mathcal{A}) = \{x \in \mathbb{R}^n \mid Cx \leq h_{\mathcal{R}_1^{\mathcal{D}}(\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{A}))}(C)\}$$

which can be computed based on the polyhedral set $\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{A})$ by solving the MILP (3.80) with $x \in \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{A})$ instead of $x \in \mathcal{A}$. We can continue in a similar way to compute $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A})$ for $k \in \mathbb{N}_0$ iteratively by solving the MILP (3.80).

Now, we can use the over-approximations $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A})$ to compute suitable sets $\mathcal{F}$ and $\mathcal{T}$ as in (3.73) and (3.74), respectively such that all trajectories starting in $\mathcal{F}$ converge to $\mathcal{T}$. Ideally, $\mathcal{F}$ should be as large as possible to provide a large set of states in which the controller can be operated safely, and $\mathcal{T}$ should be as small as possible. The smallest possible set to which all trajectories starting in $\mathcal{F}$ converge is $\mathcal{R}_\infty^{\mathcal{D}}(\mathcal{F})$ [100]. Unfortunately, this set is typically not finitely determined and thus can not be computed exactly. However, we presented in [P6] and [P7] methods for computing approximations of $\mathcal{R}_\infty^{\mathcal{D}}(\mathcal{F})$ and large sets $\mathcal{F}$.

A suitable set $\mathcal{F}$ is given by

$$\mathcal{F} = \mathcal{F}_{\bar{k}} \tag{3.82}$$

computed by iterating

$$\mathcal{F}_{k+1} = \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_k) \cap \mathcal{X} \text{ with } \mathcal{F}_0 := \mathcal{X}. \tag{3.83}$$

[P7, Eq. (21)] until $\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}_k) \subseteq \mathcal{F}_k$ for some $k = \bar{k}$, which can be verified by solving the MILP (3.80). We showed in [P7, Prop. 3] that if $\overline{\mathcal{R}}_{k^*+1}^{\mathcal{D}}(\mathcal{X}) \subseteq \mathcal{X}$ holds then we have $\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}_k) \subseteq \mathcal{F}_k$ after at most $k = k^*$ iterations. Consequently, the iteration (3.83) is guaranteed to terminate.

For the set $\mathcal{T}$ we showed in [P6, Prop. 7] that if $\mathcal{F}$ is chosen as previously described and k^* is such that

$$\frac{1}{1+\epsilon}\overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{F}) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}\left(\frac{1}{1+\epsilon}\overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{F})\right) \tag{3.84}$$

holds for a user-chosen $\epsilon > 0$, then

$$\mathcal{T} = \overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{F}) \tag{3.85}$$

is RPI and a tight over-approximation of $\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F})$ in the sense

$$\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{T} \subseteq (1+\epsilon)\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F}). \tag{3.86}$$

In summary if $\mathcal{F}$ and $\mathcal{T}$ are chosen as in (3.82) and (3.85), respectively then both sets are RPI and we have $\mathcal{F} \subseteq \mathcal{X}$ and $\mathcal{R}_{k^*}^{\mathcal{D}}(\mathcal{F}) \subseteq \overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{F}) = \mathcal{T}$, i.e., (3.73) and (3.74) hold. Consequently, Theorem 12 with $\mathcal{F}$ and $\mathcal{T}$ as in (3.82) and (3.85), respectively, applies to disturbed PWA systems controlled by maxout NNs and can be used to certify boundedness.

The computation of the set $\mathcal{T}$ according to (3.85) is shown in Figure 3.12 for an exemplary PWA system with $\mathcal{D}^{(i)} = \{\boldsymbol{d} \in \mathbb{R}^n \mid \|\boldsymbol{d}\|_\infty \leq 0.15\}$ controlled by a maxout NN. Figure 3.12.(a) shows the sequence of shrinking sets $\overline{\mathcal{R}}_i^{\mathcal{D}}(\mathcal{F})$ with $i \in \{1, \ldots, k^* - 1\}$ during the computation of the set $\mathcal{T} = \overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{F})$, which is shown in Figure 3.12.(b). The set $\mathcal{F} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid -7.3 \leq \boldsymbol{x}_1 \leq 8.91, -10 \leq \boldsymbol{x}_2 \leq 8.52\}$ is computed according to (3.82). The boundaries of the sets $(1+\epsilon)\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F})$, $\mathcal{T}$, and $\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F})$ are shown in Figure 3.12.(c) in detail. Since k^* is chosen such that (3.84) holds, we have the inclusion (3.86).

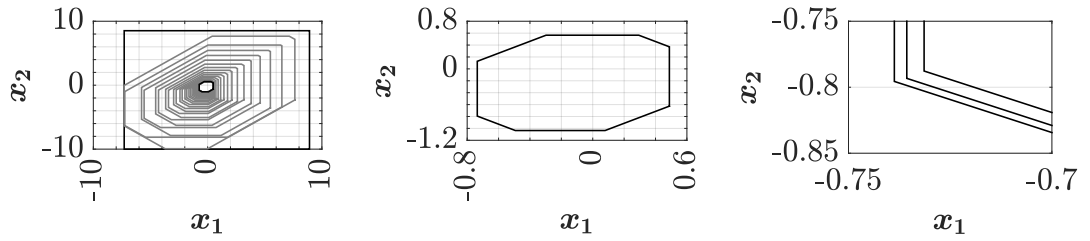
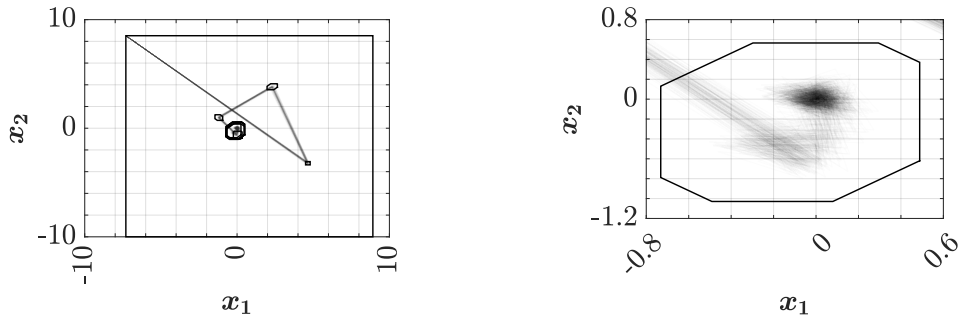


Figure 3.12. Illustration of the computation of $\mathcal{T} = \overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{F})$ starting from the set $\mathcal{F}$. The gray sets in (a) show the sequence of shrinking sets $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{F})$ during the computation. The final set $\mathcal{T}$, shown in (b), is such that the inclusion (3.86) holds. The detailed view in (c) shows from left to right the boundaries of the sets $(1+\epsilon)\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F})$, $\mathcal{T}$, and $\overline{\mathcal{R}}_{\infty}^{\mathcal{D}}(\mathcal{F})$.

Now, according to Theorem 12, all closed loop trajectories starting in $\mathcal{F}$ converge to $\mathcal{T}$. We illustrated this behavior in Figure 3.13. For the simulations, we chose the upper left vertex of the set $\mathcal{F}$ as the initial state for the trajectories. For this initial state, we computed 100 trajectories with a random additive disturbance of $\|\boldsymbol{d}(k)\|_\infty \leq 0.15$ in every time step. We can see in Figure 3.13.(a) that all trajectories remain in the set $\mathcal{F} \subseteq \mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\boldsymbol{x}\|_\infty \leq 10\}$ and thereby satisfy the state constraints $\mathcal{X}$ for all time steps. The small sets within $\mathcal{F}$ are over-approximations $\overline{\mathcal{R}}_k^{\mathcal{D}}(\{\begin{pmatrix}-7.3 & 8.52\end{pmatrix}^\top\})$ of the

reachable sets computed according to (3.75) by solving the MILP (3.80). Since we have $\boldsymbol{x}(k) \in \mathcal{R}_k^{\mathcal{D}}(\{\begin{pmatrix}-7.3 & 8.52\end{pmatrix}^\top\}) \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}(\{\begin{pmatrix}-7.3 & 8.52\end{pmatrix}^\top\})$ for $k \in \mathbb{N}_0$ with $\boldsymbol{x}(0) = \begin{pmatrix}-7.3 & 8.52\end{pmatrix}^\top$, the states $\boldsymbol{x}(k)$ of the closed-loop trajectories are guaranteed to be in the sets $\overline{\mathcal{R}}_k^{\mathcal{D}}(\{\begin{pmatrix}-7.3 & 8.52\end{pmatrix}^\top\})$ for all $k \in \mathbb{N}_0$. We additionally showed in Figure 3.13.(b) that, as predicted by Theorem 12, even in the presence of the bounded disturbance, all trajectories eventually enter the set $\mathcal{T}$ after a finite number of time steps k^* and remain in that set $\boldsymbol{x}(k)$ for all subsequent $k \geq k^*$.



(a) **(b)**

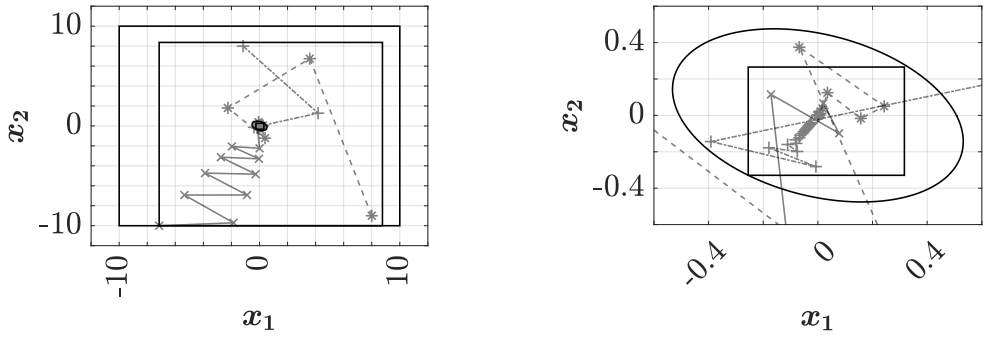
Figure 3.13. Disturbed PWA system with $\|\boldsymbol{d}(k)\|_\infty \leq 0.15$ for all $k \in \mathbb{N}_0$, controlled by a maxout NN and the sets $\mathcal{T} \subseteq \mathcal{F} \subseteq \mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \|\boldsymbol{x}\|_\infty \leq 10\}$. The sets $\mathcal{F}$ and $\mathcal{T}$ are computed according to (3.82) and (3.85), respectively. The light gray lines represent 100 trajectories of the closed-loop system with an additive disturbance of $\|\boldsymbol{d}(k)\|_\infty \leq 0.15$ in every time step. The starting point $\boldsymbol{x}(0)$ is the upper left vertex of $\mathcal{F}$.

Results for Nominal PWA Systems Controlled by NNs: For the nominal case, i.e, $\boldsymbol{d}(k) = \boldsymbol{0}$ for all $k \in \mathbb{N}$, we showed in [P6, Thm. 9] that the PWA system can be asymptotically stabilized by applying a dual-mode controller of the form

$$\boldsymbol{\pi}(\boldsymbol{x}) := \begin{cases} \boldsymbol{\kappa}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in s\mathcal{F}_0, \\ \boldsymbol{\Phi}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{F} \setminus s\mathcal{F}_0, \end{cases} \tag{3.87}$$

where $s\mathcal{F}_0$ with $\mathcal{F} \subseteq s\mathcal{F}_0$ is a PI set for the PWA system controlled by $\boldsymbol{\kappa}(\boldsymbol{x})$. The controller $\boldsymbol{\kappa}(\boldsymbol{x})$ is a stabilizing controller that can be computed by using known methods for the LMI-based controller synthesis for PWA systems, e.g., [101, Eq. (9)-(10)].

Figure 3.14 shows the behavior of a nominal PWA system controlled by the dual-mode controller (3.87). Since the sets $\mathcal{F}$ and $\mathcal{T}$ are chosen according to (3.82) and (3.85), respectively, and in addition $\mathcal{T} \subseteq s\mathcal{F}_0$ holds, all trajectories starting in $\mathcal{F}$ are guaranteed to enter $\mathcal{F}$ and thus $s\mathcal{F}_0$ after at most k^* steps. Once the trajectories enter $s\mathcal{F}_0$, the controller $\boldsymbol{\kappa}(\boldsymbol{x})$ is applied and asymptotically stabilizes the system.



(a) **(b)**

Figure 3.14. Trajectories of a nominal PWA system controlled by the dual-mode controller (3.87) and the sets $\mathcal{T} \subseteq s\mathcal{F}_0 \subseteq \mathcal{F} \subseteq \mathcal{X}$. The sets $\mathcal{F}$ and $\mathcal{T}$ are poyhedral and computed according to (3.82) and (3.85), respectively. The set $s\mathcal{F}_0$ is an ellipsoidal PI set of the PWA systems controlled by $\boldsymbol{\kappa}(\boldsymbol{x})$.

Extension to Nonlinear Systems: With the methods presented for disturbed PWA systems of the form (3.70), we laid the foundation for the analysis of nonlinear[1] systems of the form

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \tag{3.88}$$

that can be approximated by PWA systems with bounded disturbances, i.e., we assume

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{D}^{(i)} \tag{3.89}$$

for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$. We proved in our article [P7] that for such systems the following theorem holds.

Theorem 15 ([P7, Thm. 6]). *Let $\mathcal{F}$ and $\mathcal{T}$ be as in* (3.82) *and* (3.85)*, respectively and assume that* (3.89) *holds for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$. Then, the system* (3.70) *with NN-based controller $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ is ultimately bounded in $\mathcal{T}$, uniformly in $\mathcal{F}$. Moreover, $\boldsymbol{x}(k) \in \mathcal{X}$ and $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ holds for all $k \in \mathbb{N}_0$.*

Note that the condition of Theorem 15 that (3.89) holds for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$ is satisfied if we have an approximation of $\boldsymbol{f}$ with $\mathcal{Y}(\boldsymbol{x}, \boldsymbol{u}) \subseteq \mathcal{Y}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{u})$ for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$ [P7, Lem. 7], where we define

$$\mathcal{Y}(\boldsymbol{x}, \boldsymbol{u}) := \{\boldsymbol{x}^+ \in \mathbb{R}^n \mid \boldsymbol{x}^+ = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k))\} \text{ and} \tag{3.90}$$

$$\mathcal{Y}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{u}) := \{\boldsymbol{x}^+ \in \mathbb{R}^n \mid \boldsymbol{x}^+ = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{d}, \boldsymbol{d} \in \mathcal{D}^{(i)}\}. \tag{3.91}$$

This kind of approximation leads to reachable sets $\mathcal{R}_k(\mathcal{F})$ (3.69) of the nonlinear system (3.88) that are a subset of the reachable sets $\mathcal{R}_k^{\mathcal{D}}(\mathcal{F})$ of the disturbed PWA system. Formally, we then have $\mathcal{R}_k(\mathcal{F}) \subseteq \mathcal{R}_k^{\mathcal{D}}(\mathcal{F}) \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{F})$ for

[1] We are referring here and in the following to nonlinear systems that are in addition non-PWA.

all $k \in \mathbb{N}_0$ and for all RPI sets $\mathcal{F} \subseteq \mathcal{X}$ [P7, Lem. 5]. Thus, if a set is RPI for a PWA approximation with (3.89) of a nonlinear system, then the set is PI for the original nonlinear system. This allows for the adaptation of Theorem 12 to nonlinear systems, thereby leading to Theorem 15.

Figure 3.16 shows an artificial nonlinear system of the form $x(k+1) = f(x(k)) = 0.02x^3(k) - 2x(k)$ and an PWA approximation with the $s = 3$ regions $\mathcal{P}^{(1)} = [-10, -5.77]$, $\mathcal{P}^{(2)} = [-5.77, 5.77]$, and $\mathcal{P}^{(2)} = [5.77, 10]$. The gray area indicates the set $\mathcal{Y}_{\text{PWA}}(x)$ which is such that $\mathcal{Y}(x) \subseteq \mathcal{Y}_{\text{PWA}}(x)$ with $\mathcal{D}^{(1)} = [-1.48, 0.88]$, $\mathcal{D}^{(1)} = [-1.48, 1.48]$, and $\mathcal{D}^{(1)} = [-0.88, 1.48]$ holds for all $x \in \mathcal{X} = [-10, 10]$. Consequently, (3.89) holds for all $x \in \mathcal{X}$. Note that, as shown in Figure 3.15 the sets $\mathcal{D}^{(i)}$ can differ for $i \in \{1, 2, 3\}$. Moreover, for Theorem 15 to apply, the PWA approximation does not need to be continuous at the boundaries between neighboring regions. This allows for more flexible PWA approximations with lower error. In Figure 3.16, we used such a PWA approximation to analyze the behavior of the nonlinear double integrator with

$$\boldsymbol{f}(\boldsymbol{x}, u) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} u + \begin{pmatrix} 0.025 \\ 0.025 \end{pmatrix} \left(\boldsymbol{x}^\top \boldsymbol{x} \right)$$

from [102, Sec. 6] controlled by a maxout NN. For the example shown in Figure 3.16, Theorem 15 applies, since the PWA approximation is chosen such that (3.89) holds for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$. Consequently, we can observe that the trajectory, starting in $\mathcal{F}$, of the nonlinear system $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \Phi(\boldsymbol{x}(k)))$ enters the small set $\mathcal{T}$ around the origin after a finite number of time steps. The small sets around the states $\boldsymbol{x}(k)$ are the over-approximation $\overline{\mathcal{R}}_k^{\mathcal{D}}(\{\boldsymbol{x}(0)\})$ of the reachable sets of the disturbed PWA system. These sets can be computed by solving the MILP (3.80). Moreover, since (3.89) holds, the true trajectory of the controlled nonlinear system is guaranteed to be in these sets.

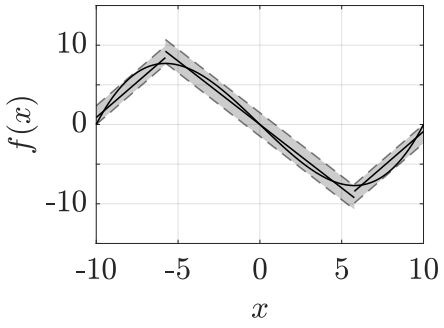


Figure 3.15. Artificial nonlinear system of the form $x(k+1) = f(x(k))$ and PWA approximation with three regions which is such that $\mathcal{Y}(x) \subseteq \mathcal{Y}_{\text{PWA}}(x)$ for all $x \in \mathcal{X} = [-10, 10]$.

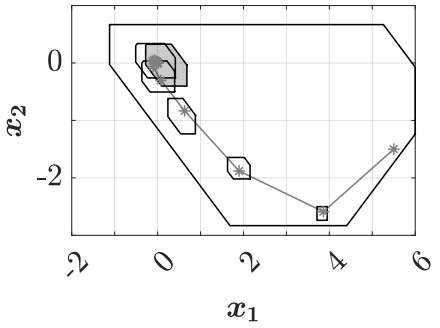


Figure 3.16. Trajectory of a nonlinear double integrator controlled by a maxout NN. The small sets are the reachable sets $\mathcal{R}_k^{\mathcal{D}}(\{\begin{pmatrix} 5.5 & -1.5 \end{pmatrix}^\top\})$ of the PWA approximation.

3.2.3.2 Discussion

The summarized methods from our articles [P6] and [P7] provide a framework for analyzing NN–based controllers for disturbed PWA systems. These methods are the first capable of addressing PWA systems subject to bounded additive disturbances. In contrast, existing methods [13, 38, 39, 40] are typically restricted to linear systems or lack the ability to account for disturbances, thereby limiting their applicability. The main tool we used for our analysis are reachable sets. Building on these reachable sets, we computed an RPI set $\mathcal{F}$ in which the system with an NN-based controller can be operated safely and an RPI set $\mathcal{T}$ that all trajectories of the closed-loop system reach after a finite time. Moreover, we presented results on the finite determinedness of the sets $\mathcal{F}$ and $\mathcal{T}$ and a result that quantifies the quality of the set $\mathcal{F}$ in terms of the inclusion (3.86). These results represent a significant improvement. Methods from the literature [38] often assume a given RPI set $\mathcal{F}$ without providing a method for computing this set for the considered system type, and do not quantify the deviation between $\mathcal{T}$ and $\mathcal{R}_\infty^{\mathcal{D}}(\mathcal{T})$. In addition, even though the used NN-based controller may be trained with data from a stabilizing classical, i.e., non-NN, controller, we do not use any knowledge of a stabilizing controller for the analysis. This is in contrast to the methods discussed in Section 3.2.2, where the controller that is used to generate training samples for the NN needs to be known. Thus, the methods discussed in this section provide a versatile framework for analyzing NN-based controllers, regardless of the specific training method employed. This flexibility enables the derivation of formal guarantees for stability and constraint satisfaction even in cases where the training method itself, e.g., RL [6, 11], imitation learning [7], or the method presented in Section 3.1.2, does not inherently provide such guarantees. Consequently, the proposed framework complements a broad range of existing NN training methods for control.

The consideration of bounded additive disturbances in (3.70) allows us to extend the framework to the analysis of NN-based controllers for nonlinear systems that can be approximated by PWA systems with a bounded error, which is, to the best of our knowledge, unique. Suitable PWA approximations of nonlinear functions may be computed using our method from [P5], which is summarized in Section 3.1.2. By including constraints of the form

$$-\bar{e} \leq y^{(i)} - \alpha^{(i)} + \beta^{(i)} \leq \bar{e}$$

for all $i \in \{1, \dots, N_D\}$ in the MIQP (3.25) that is used for training, we can compute a PWA approximation that has a maximum error of $\bar{e}$ at the sample points. Together with constraints that limit the maximum Lipschitz constant of the PWA approximation, we can ensure that the error is bounded as required by (3.89). Thus, by combining our approaches for training from [P5] and for analyzing the NN-based controller [P6, P7], we may develop a method that seemingly integrates training and analysis.

Chapter 4

Conclusions and Outlook

4.1 Conclusions

At the beginning of the thesis, we have identified two central challenges for the integration of ML methods, and in particular of NNs, into control systems. The first challenge is the design of tailored NN topologies harmonizing with the structure of functions relevant to control, thereby providing a lightweight and fast to evaluate approximation. The second challenge involves analyzing NN-based controllers in a closed control loop to certify boundedness, stability, and constraint satisfaction after training. Together, solutions to these challenges form a theoretical foundation for ensuring that NN-based controllers meet the requirements of modern control applications.

After providing a relevant background in the Chapters 1 and 2, we first tackled in Section 3.1.1 the design problems by deriving NNs capable of exactly representing continuous PWQ functions with polyhedral domain partitions. These functions play a central role in RL and ADP for approximating PWQ Q-functions and OVFs. Furthermore, by exploiting the connection between NNs and piecewise-defined functions, we developed in Section 3.1.2 a training method that reformulates the underlying optimization problem as an MIQP. This reformulation allows us to train the NN to global optimality. This is, to our knowledge, the first method with sufficient flexibility to both attain globally optimal solutions and fit arbitrary continuous piecewise-defined functions. Beyond tailored NN topologies and training methods, we also investigated in Section 3.1.3 the design of polynomial approximations of activation functions to enable a privacy-preserving evaluation of NNs using HE. The presented method enables us to increase the degree of the polynomials used for approximation by one compared to known methods, thereby increasing the accuracy of the approximations.

In the second part of Chapter 3, we addressed problems regarding the analysis of NN-based controllers. More precisely, we considered the challenge of certifying convergence, stability, and constraint satisfaction of linear and PWA systems with bounded additive disturbances. We discussed two classes of analysis methods in Section 3.2. The first class builds on quantifying the approximation error of an NN relative to a known stabilizing baseline controller, such as linear MPC. We extended these known error-based methods to max-

out NNs and presented simplifications for computing the maximum error and the minimum Lipschitz constant of the error in Section 3.2.2. We discussed the limitations of these methods, which are that they require exact knowledge of a stabilizing baseline controller and its parameters, and may fail to certify stability if the approximation error is too large. To overcome these limitations, we introduced in Section 3.2.3 a second class of methods that does not require a baseline controller and instead directly analyzes the closed-loop behavior by computing reachable sets. The reachable sets are used to compute suitable RPI sets to verify convergence, stability, and constraint satisfaction. These methods are the first to enable the analysis of NN-based controllers for PWA systems with bounded additive disturbances. Furthermore, the consideration of bounded additive disturbances significantly extends the applicability of the introduced analysis method, as it allows for considering nonlinear systems that can be approximated by PWA systems with a suitably small error. This is a significant step beyond existing approaches, which are typically limited to linear systems or disturbance-free settings.

Together, these contributions establish a flexible and comprehensive framework for both the design and analysis of NNs for control. The presented reachability-based analysis method is compatible with arbitrary training methods, e.g., RL or imitation learning, providing formal guarantees on stability and constraint satisfaction where such guarantees are not inherent to the training process. In this way, our work combines the benefits of fast NN-based controllers with guarantees of classical controllers, contributing towards a reliable and interpretable deployment of NNs in safety-critical control applications.

4.2 Outlook

For future research, several promising directions emerge. One promising direction concerns scalability. The MIPs on which several of the proposed methods are based could be enhanced through different techniques to improve computational tractability. Initial steps have already been taken in [P4] and [P5] by developing methods that leverage symmetries and saturation effects of the underlying problems. However, these methods still leave room for future research. Moreover, the methods for analyzing NNs presented in Section 3.2.3 are based on over-approximations of reachable sets, which are computed by solving MIPs. However, the results on stability and boundedness apply independently of the particular method used for computing the reachable sets. Thus, it may be possible to replace the MIP-based computation with another, computationally more efficient method, e.g., a method based on semi-definite programming, without affecting the validity of the results from Section 3.2.3.

Another exciting research direction is the combination of privacy-preserving evaluation of NNs with the methods for analyzing systems controlled by NNs. To enable a privacy-preserving evaluation of NN-based controllers, the NN

needs to be approximated by a polynomial function. If the maximum error of this polynomial approximation with respect to the NN is known, then it can be treated as a bounded additive disturbance in PWA systems of the form (3.70), which enables us to consider the approximation error in the MILP (3.80) that is used to compute the reachable sets. Thus, with such an MILP, we could analyze the closed-loop behavior of a system controlled by an NN that is evaluated using HE, and at the same time account for the approximation error arising from the polynomial approximation. In summary, such an extension would enable not only a provably safe but also a privacy-preserving control of the system by NNs.

Lastly, until now, we have considered the design and analysis of NN-based controllers separately. Meaning, that first the NN is designed and trained, and afterward its closed-loop behavior is analyzed to certify stability and constraint satisfaction. For future research, it would be interesting to investigate whether one can use the proposed methods for developing an integrated design and analysis framework that enables the design of NNs that inherently comply with the requirements of control systems, thereby eliminating the need for a subsequent analysis.

Part II:

Articles

In the following, we collect reprinted versions of the articles [P1, P2, P3, P4, P5, P6, P7, P8, P9] that are summarized and discussed in Chapter 3. Their layout has been revised, but the notation is as in the original articles and may differ from the notation introduced in Part I. The articles [P10, P11, P12, P13] of the author are not reprinted and only listed in the bibliography as they are thematically not relevant for the thesis.

Chapter 5

Tailored neural networks for learning optimal value functions in MPC*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. Learning-based predictive control is a promising alternative to optimization-based MPC. However, efficiently learning the optimal control policy, the optimal value function, or the Q-function requires suitable function approximators. Often, artificial neural networks (ANN) are considered but choosing a suitable topology is also non-trivial. Against this background, it has recently been shown that tailored ANN allow, in principle, to exactly describe the optimal control policy in linear MPC by exploiting its piecewise affine structure. In this paper, we provide a similar result for representing the optimal value function and the Q-function that are both known to be piecewise quadratic for linear MPC.

Keywords. Neural networks, rectified linear units (ReLU), MPC, machine learning, piecewise quadratic functions

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

5.1 Introduction

Learning-based predictive control becomes more and more prominent in research and practice. Various approaches have been presented that underline the capabilities of (deep) learning in the context of control (see, e.g., [8], [103], [104], [10] [6], [105], or [7]). Most of these approaches either aim for approximations of the optimal control law in the policy space (such as [7, 104]) or the optimal value function (or Q-function) in the value space (such as [10, 105]). In both cases, suitable parametrizations for the functions to be learned have to be chosen. Common candidates are, for example, artificial neural networks (ANN) or Bayesian networks.

Choosing the type of parametrization and underlying specifications (such as layer width and depth or activation functions in ANN) is often done in a trial-and-error fashion. However, some setups support more "educated guesses". In fact, it has recently been shown that certain types of ANN allow, in principle, to exactly describe model predictive control (MPC) laws for linear systems [7, 26]. The key observation here is that both the MPC law and certain ANN reflect piecewise affine (PWA) functions on polyhedral partitions. More specifically, feed-forward ANN with rectified linear units (ReLU) or max-out activations [21] offers this feature [23].

Now, the optimal value function (OVF) in linear MPC is well-known to be piecewise quadratic (PWQ) [20]. The main contribution of this paper is to show that suitable ANN can also be found for such functions. Interestingly, these ANN will also build on ReLU activations. However, successful realizations require to extend the input layer by quadratic terms of the system states (such as x_1^2, x_2^2, and $x_1 x_2$). Extended inputs of this form have also been considered elsewhere. For instance, [106] and [107] exploit their ability to create nonlinear decision boundaries. Clearly, this feature is not helpful here since we are dealing with polyhedral partitions. In contrast, we will use the quadratic terms to eliminate the quadratic expressions of the OVF by suitably choosing the affine preactivations. For the remaining PWA structure, we can exploit the known results mentioned above. In combination, we obtain an approach for exactly describing the OVF via ANN with ReLU activations. In addition, we present a similar approach for representing Q-functions.

The paper is organized as follows. In Section 5.2, we collect preliminaries on ANN, MPC, and learning-based predictive control. Further, we briefly summarize the known representation of PWA functions using ANN. We present our main result, i.e., tailored ANN for exactly describing PWQ functions, in Section 5.3. More specifically, we construct ANN that exactly describe the OVF in linear MPC for the special case of one-dimensional states (i.e., $n = 1$). Moreover, we show that ANN for describing the Q-function can be derived whenever a construction for the OVF is known (for any n). Finally, we illustrate our results with various numerical examples in Section 5.4 and state conclusions in Section 5.5.

5.2 Preliminaries and background

5.2.1 Neural networks with rectifier activation

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

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{f}^{(l+1)} \circ \boldsymbol{g}^{(l)} \circ \boldsymbol{f}^{(l)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{\xi}), \tag{5.1}$$

where the functions $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i}$ refer to preactivations (for $i \leq l$) and a postactivation (for $i = l+1$) and where $\boldsymbol{g}^{(i)} : \mathbb{R}^{w_i} \to \mathbb{R}^{w_i}$ are activation functions. The pre- and postactivations are typically affine, i.e.,

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

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{w_i \times w_{i-1}}$ is a weighting matrix, $\boldsymbol{a}^{(i)} \in \mathbb{R}^{w_i}$ is a bias vector, and $\boldsymbol{y}^{(i-1)}$ denotes the output of the previous layer with $\boldsymbol{y}^{(0)} := \boldsymbol{\xi}$. Now, while various activation functions are established, we here focus on activations via ReLU that are characterized by

$$\boldsymbol{g}^{(i)}(\boldsymbol{z}^{(i)}) = \max\left\{\boldsymbol{0}, \boldsymbol{z}^{(i)}\right\} := \begin{pmatrix} \max\left\{0, z_1^{(i)}\right\} \\ \vdots \\ \max\left\{0, z_{w_i}^{(i)}\right\} \end{pmatrix}.$$

We will refer to the resulting networks as ReLU-ANN.

5.2.2 Model predictive control and structural insights

Classical MPC builds on solving an optimal control problem (OCP) of the form

$$\begin{aligned} V_N(\boldsymbol{x}) := \min_{\substack{\hat{\boldsymbol{x}}(0),\dots,\hat{\boldsymbol{x}}(N) \\ \hat{\boldsymbol{u}}(0),\dots,\hat{\boldsymbol{u}}(N-1)}} \; & \varphi(\hat{\boldsymbol{x}}(N)) + \sum_{\kappa=0}^{N-1} \ell(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)) \\ \text{s.t.} \quad & \hat{\boldsymbol{x}}(0) = \boldsymbol{x}, \\ & \hat{\boldsymbol{x}}(\kappa+1) = \boldsymbol{A}\,\hat{\boldsymbol{x}}(\kappa) + \boldsymbol{B}\hat{\boldsymbol{u}}(\kappa), \quad \forall \kappa \in \{0,\dots,N-1\}, \\ & (\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)) \in \mathcal{X} \times \mathcal{U}, \qquad \forall \kappa \in \{0,\dots,N-1\}, \\ & \hat{\boldsymbol{x}}(N) \in \mathcal{T} \end{aligned} \tag{5.2}$$

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

$$\varphi(\boldsymbol{x}) := \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \quad \text{and} \quad \ell(\boldsymbol{x}, \boldsymbol{u}) := \boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u} \tag{5.3}$$

denote the terminal and stage cost, respectively, where the weighting matrices $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are positive (semi-) definite. The dynamics of the linear prediction model are described by $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times m}$. State and input constraints

can be incorporated via the polyhedral sets $\mathcal{X}$ and $\mathcal{U}$. Finally, the terminal set $\mathcal{T}$ allows to enforce closed-loop stability (see [48] for details). The resulting control policy $\pi : \mathcal{F}_N \to \mathcal{U}$ is defined as

$$\pi(\boldsymbol{x}) := \hat{\boldsymbol{u}}^*(0), \tag{5.4}$$

where $\mathcal{F}_N$ denotes the feasible set of (5.2) and where $\hat{\boldsymbol{u}}^*(0)$ refers to the first element of the optimal input sequence.

Now, it is well known that $\pi(\boldsymbol{x})$ is a (continuous) PWA function of the form

$$\pi(\boldsymbol{x}) = \begin{cases} \boldsymbol{K}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \boldsymbol{K}^{(s)}\boldsymbol{x} + \boldsymbol{b}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{5.5}$$

where the regions $\mathcal{R}^{(i)}$ represent polyhedral sets with pairwise disjoint interiors [20, Thm. 4]. A related observation is that the optimal value function (OVF) is a (continuous and convex) piecewise quadratic (PWQ) function of the form

$$V_N(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^\top \boldsymbol{S}^{(1)}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{l}^{(1)} + c^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \boldsymbol{x}^\top \boldsymbol{S}^{(s)}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{l}^{(s)} + c^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}. \end{cases} \tag{5.6}$$

5.2.3 Learning-based predictive control

In principle, the structure (5.5) is quite beneficial for efficiently implementing MPC. In fact, for moderate system dimensions and prediction horizons, (5.5) can be computed explicitly and stored on the controller device. However, for larger dimensions or horizons, this procedure is intractable. If, in addition, an online solution of the OCP is computationally too demanding, machine learning (ML) is a promising alternative. Most intuitively, ML can be used to approximate (5.5), which refers to an approximation in the policy space. This approach in combination with ANN of the form (5.1) has, e.g., been considered in [6, 7]. Alternatively, one can also aim for approximating (5.6), i.e., an approximation in the value space [10], [3]. In fact, an (almost) optimal control input can also be inferred from (an approximation of) the OVF [103]. Remarkably, approximating (5.5) or (5.6) is typically realized offline using supervised learning. Hence, it usually involves the solution of (5.2) for sampled states $\boldsymbol{x}$ and thus requires the knowledge of a model in terms of $\boldsymbol{A}$ and $\boldsymbol{B}$. Nevertheless, learning-based predictive control can also be realized without a model. To this end, one aims for approximating the so-called Q-function (see, e.g., [3, P. 13])

$$Q_N(\boldsymbol{x}, \boldsymbol{u}) := \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \tag{5.7}$$

that reflects the costs for a first step with an undetermined $\boldsymbol{u}$ followed by an optimal control sequence of length $N-1$. We note, in this context, that the

successor state $\boldsymbol{x}^+ := \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ can either be computed (using a model) or measured (without a model). Based on the Q-function, one can then easily derive an optimal input by evaluating $\arg\min_{\boldsymbol{u}} Q_N(\boldsymbol{x}, \boldsymbol{u})$. Approximating (5.7) in the context of MPC has, e.g., been considered in [105].

5.2.4 Tailored neural networks for representing PWA functions

In this paper, we aim for tailored ANN that are capable of exactly describing the PWQ OVF (5.6) or the Q-function (5.7). This goal is motivated by the observation that ReLU-ANN can, in principle, exactly describe any (continuous) PWA function of the form (5.5) (see, e.g., [25], [24]). However, the corresponding network topologies can quickly become intractable. For instance, the approach in [24, Thm. 2] requires at least s hidden layers, which typically results in an ANN with unfavorable depth. Useful topologies can, however, be found for special cases. In fact, for $n = 1$ (and arbitrary m), (5.5) can exactly be described by a ReLU-ANN with one hidden layer of width $w_1 = s$, i.e., by a function of the form

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{W}^{(2)} \max\left\{\boldsymbol{0}, \boldsymbol{W}^{(1)}\boldsymbol{\xi} + \boldsymbol{a}^{(1)}\right\} + \boldsymbol{a}^{(2)} \tag{5.8}$$

with $\boldsymbol{W}^{(1)}, \boldsymbol{a}^{(1)} \in \mathbb{R}^s$, $\boldsymbol{W}^{(2)} \in \mathbb{R}^{m\times s}$, and $\boldsymbol{a}^{(2)} \in \mathbb{R}^m$. Suitable weights and biases are given in [26]. Their construction builds on the trivial observation that, for one-dimensional $\boldsymbol{x}$, the regions $\mathcal{R}^{(i)}$ reflect proper intervals of the form

$$\mathcal{R}^{(i)} = [\underline{x}^{(i)}, \overline{x}^{(i)}] \subset \mathbb{R} \tag{5.9}$$

that can be sorted such that $\overline{x}^{(i)} = \underline{x}^{(i+1)}$ for every $i \in \{1, \dots, s-1\}$. Using these conditions, equality of (5.5) and (5.8) holds for the parametrization

$$\begin{aligned}
\boldsymbol{W}^{(1)} &:= \begin{pmatrix} -1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \qquad \boldsymbol{a}^{(1)} := \begin{pmatrix} \overline{x}^{(1)} \\ -\overline{x}^{(1)} \\ \vdots \\ -\overline{x}^{(s-1)} \end{pmatrix}, \\
\boldsymbol{W}^{(2)} &:= \begin{pmatrix} -\boldsymbol{K}^{(1)} & \boldsymbol{K}^{(2)} & \boldsymbol{K}^{(3)} - \boldsymbol{K}^{(2)} & \dots \boldsymbol{K}^{(s)} - \boldsymbol{K}^{(s-1)} \end{pmatrix}, \\
\boldsymbol{a}^{(2)} &:= \boldsymbol{K}^{(1)}\overline{x}^{(1)} + \boldsymbol{b}^{(1)}
\end{aligned} \tag{5.10}$$

and the choice $\boldsymbol{\xi} := \boldsymbol{x}$ [26, Thm. 1]. The paper now intends to derive analogue parametrizations for describing (5.6) and (5.7).

5.3 Tailored neural networks for PWQ functions

5.3.1 Problem specification

As mentioned in the introduction, our goal is to design ReLU-ANN that allow to eliminate the quadratic terms in (5.6) or (5.7). In fact, the remaining functions will then be PWA and, hence, we can reuse the results from Section 5.2.4

for describing the remainders using further ReLU-ANN. A formalization of this concept leads to the two following problem statements.

Problem 5.1. *Identify an ANN of the form* (5.8) *and a mapping* $\boldsymbol{h}_v : \mathbb{R}^n \to \mathbb{R}^{w_0}$ *such that*

$$\Delta V_N(\boldsymbol{x}) := V_N(\boldsymbol{x}) - \boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x})) \tag{5.11}$$

is PWA in $\boldsymbol{x}$*.*

Problem 5.2. *Identify an ANN of the form* (5.8) *and a mapping* $\boldsymbol{h}_q : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{w_0}$ *such that*

$$\Delta Q_N(\boldsymbol{x}, \boldsymbol{u}) := Q_N(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{\Phi}(\boldsymbol{h}_q(\boldsymbol{x}, \boldsymbol{u})) \tag{5.12}$$

is PWA in $\boldsymbol{x}$ *and* $\boldsymbol{u}$*.*

Here, it is import to note that choosing suitable network inputs via $\boldsymbol{h}_v$ and $\boldsymbol{h}_q$ is part of the problems. In fact, the naive choices $\boldsymbol{\xi} := \boldsymbol{x}$ as in Section 5.2.4 or $\boldsymbol{\xi}^\top := (\, \boldsymbol{x}^\top \;\; \boldsymbol{u}^\top \,)^\top$ will, in general, not solve the problems. We further note that both V_N and Q_N are scalar-valued functions. Hence, also the desired $\boldsymbol{\Phi}$ is scalar-valued in both cases and the corresponding parameters have the dimensions $\boldsymbol{W}^{(1)} \in \mathbb{R}^{w_1 \times w_0}$, $\boldsymbol{a}^{(1)} \in \mathbb{R}^{w_1}$, $\boldsymbol{W}^{(2)} \in \mathbb{R}^{1 \times w_1}$, and $\boldsymbol{a}^{(2)} \in \mathbb{R}$. We still use bold characters for $\boldsymbol{\Phi}$ and $\boldsymbol{a}^{(2)}$ for consistency.

In the following, we present (partial) solutions to Problems 5.1 and 5.2. More precisely, we solve Problem 5.1 for the special case $n = 1$ that has also been addressed in [26] for the PWA analogue. Further, we show that a solution to Problem 5.2 can be derived from a given solution to Problem 5.1 for arbitrary $n \geq 1$.

5.3.2 Solution to Problem 1 for $n = 1$

The following theorem provides a solution to Problem 5.1 for one-dimensional system states. The underlying construction is based on the observation that ReLU can be designed to be active only inside one of the regions (5.9).

Theorem 5.3. *Let* $n = 1$ *and assume that the regions* $\mathcal{R}^{(i)}$ *are as in* (5.9) *(with* $\underline{x}^{(i)} < \overline{x}^{(i)} = \underline{x}^{(i+1)}$*) and bounded. Then, a solution to Problem 5.1 is given in terms of*

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \underline{x}^{(1)} + \overline{x}^{(1)} & -1 \\ \vdots & \vdots \\ \underline{x}^{(s)} + \overline{x}^{(s)} & -1 \end{pmatrix}, \qquad \boldsymbol{a}^{(1)} := \begin{pmatrix} -\underline{x}^{(1)}\overline{x}^{(1)} \\ \vdots \\ -\underline{x}^{(s)}\overline{x}^{(s)} \end{pmatrix},$$
$$\boldsymbol{W}^{(2)} := \begin{pmatrix} -\boldsymbol{S}^{(1)} & \dots & -\boldsymbol{S}^{(s)} \end{pmatrix}, \qquad \boldsymbol{a}^{(2)} := 0$$

and the mapping $\boldsymbol{h}_v(\boldsymbol{x}) := \begin{pmatrix} \boldsymbol{x} & \boldsymbol{x}^2 \end{pmatrix}^\top$.

Proof. We obviously have

$$(\underline{x}^{(i)} + \overline{x}^{(i)})\boldsymbol{x} - \boldsymbol{x}^2 - \underline{x}^{(i)}\overline{x}^{(i)} = (\boldsymbol{x} - \underline{x}^{(i)})(\overline{x}^{(i)} - \boldsymbol{x}).$$

Hence, the proposed parametrization yields

$$\boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x})) = \sum_{i=1}^{s} -\boldsymbol{S}^{(i)} \max\left\{0, (\boldsymbol{x} - \underline{x}^{(i)})(\overline{x}^{(i)} - \boldsymbol{x})\right\}. \tag{5.13}$$

Clearly, $(\boldsymbol{x} - \underline{x}^{(i)})(\overline{x}^{(i)} - \boldsymbol{x}) > 0$ if and only if the conditions

$$\boldsymbol{x} - \underline{x}^{(i)} > 0 \quad \text{and} \quad \overline{x}^{(i)} - \boldsymbol{x} > 0 \tag{5.14}$$

or

$$\boldsymbol{x} - \underline{x}^{(i)} < 0 \quad \text{and} \quad \overline{x}^{(i)} - \boldsymbol{x} < 0 \tag{5.15}$$

hold. Now, conditions (5.14) are equivalent to $\boldsymbol{x} \in \operatorname{int}\big(\mathcal{R}^{(i)}\big)$ and conditions (5.15) are infeasible for proper intervals with $\underline{x}^{(i)} < \overline{x}^{(i)}$. Since we further have $\operatorname{int}\big(\mathcal{R}^{(i)}\big) \cap \operatorname{int}\big(\mathcal{R}^{(j)}\big) = \emptyset$ whenever $i \neq j$, we obtain

$$\boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x})) = \begin{cases} -\boldsymbol{S}^{(1)}(\boldsymbol{x} - \underline{x}^{(i)})(\overline{x}^{(i)} - \boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \quad \vdots & \qquad \vdots \\ -\boldsymbol{S}^{(s)}(\boldsymbol{x} - \underline{x}^{(i)})(\overline{x}^{(i)} - \boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases}$$

where $\boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x})) = 0$ results whenever $\boldsymbol{x}$ is located on the boundary of any $\mathcal{R}^{(i)}$. Evaluating (5.11) finally results in the PWA function

$$\Delta V_N(\boldsymbol{x}) = \begin{cases} \kappa^{(1)}\boldsymbol{x} + \beta^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \quad \vdots & \qquad \vdots \\ \kappa^{(s)}\boldsymbol{x} + \beta^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)} \end{cases} \tag{5.16}$$

with the parameters $\kappa^{(i)} := \boldsymbol{l}^{(i)} + \boldsymbol{S}^{(i)}(\underline{x}^{(i)} + \overline{x}^{(i)})$ and $\beta^{(i)} := c^{(i)} - \boldsymbol{S}^{(i)}\underline{x}^{(i)}\overline{x}^{(i)}$, which completes the proof. ■

Roughly speaking, Theorem 5.3 provides a ReLU-ANN that reflects the quadratic terms of $V_N(\boldsymbol{x})$. In fact, we proved that the difference (5.11) is PWA. Now, as summarized in Section 5.2.4, it has been known before that PWA functions can likewise be described by ReLU-ANN with one hidden layer. Thus, by combining both results, we can construct a ReLU-ANN of the form (5.8) that completely reflects $V_N(\boldsymbol{x})$. However, when applying the results from Section 5.2.4, one has to take into account that representing the quadratic terms requires $\boldsymbol{\xi} = \boldsymbol{h}_v(\boldsymbol{x})$. Fortunately, $\boldsymbol{h}_v(\boldsymbol{x})$ contains the state $\boldsymbol{x}$. Hence, we can easily adapt the parametrization from (5.10) according to the following Corollary.

Corollary 5.4. *Let $\boldsymbol{h}_v$ and ΔV_N be as in Theorem 5.3 and* (5.16)*, respectively. Then, $\Delta V_N(\boldsymbol{x}) = \boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x}))$ holds for an ANN as in* (5.8) *with the parameters*

$$\begin{aligned}
\boldsymbol{W}^{(1)} &:= \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{a}^{(1)} := \begin{pmatrix} \overline{x}^{(1)} \\ -\overline{x}^{(1)} \\ \vdots \\ -\overline{x}^{(s-1)} \end{pmatrix}, \\
\boldsymbol{W}^{(2)} &:= \begin{pmatrix} -\kappa^{(1)} & \kappa^{(2)} & \kappa^{(3)} - \kappa^{(2)} & \dots & \kappa^{(s)} - \kappa^{(s-1)} \end{pmatrix}, \\
\boldsymbol{a}^{(2)} &:= \kappa^{(1)} \overline{x}^{(1)} + \beta^{(1)}.
\end{aligned}$$

The combination of Theorem 5.3 and Corollary 5.4 leads to the following summarizing statement. An illustration with an example will follow in Section 5.4.1.

Corollary 5.5. *Let $n = 1$. Then, V_N as in* (5.6) *can be exactly described by a ReLU-ANN with one hidden layer of width $w_1 = 2s$ and the inputs $\boldsymbol{\xi} = \begin{pmatrix} \boldsymbol{x} & \boldsymbol{x}^2 \end{pmatrix}^\top$.*

5.3.3 Conditional solution to Problem 2 for arbitrary n

Interestingly, the Problems 5.1 and 5.2 are closely related. In fact, the following theorem shows that a solution to the former problem implies a solution to the latter. In other words, an ANN representing the OVF can be easily adapted for describing the Q-function.

Theorem 5.6. *Assume Problem 5.1 has been solved for $N' = N - 1$ and let $\boldsymbol{W}_v^{(1)}$, $\boldsymbol{W}_v^{(2)}$, $\boldsymbol{a}_v^{(1)}$, and $\boldsymbol{a}_v^{(2)}$ denote the corresponding weights and biases in* (5.8)*. Then, a solution to Problem 5.2 is given in terms of*

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \boldsymbol{W}_v^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{pmatrix}, \qquad \boldsymbol{a}^{(1)} := \begin{pmatrix} \boldsymbol{a}_v^{(1)} \\ 0 \end{pmatrix}, \tag{5.17a}$$

$$\boldsymbol{W}^{(2)} := \begin{pmatrix} \boldsymbol{W}_v^{(2)} & 1 \end{pmatrix}, \qquad \boldsymbol{a}^{(2)} := \boldsymbol{a}_v^{(2)} \tag{5.17b}$$

and the mapping

$$\boldsymbol{h}_q(\boldsymbol{x}, \boldsymbol{u}) := \begin{pmatrix} \boldsymbol{h}_v(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \\ \ell(\boldsymbol{x}, \boldsymbol{u}) \end{pmatrix}. \tag{5.18}$$

Proof. The assumed solution to Problem 5.1 implies

$$V_{N-1}(\boldsymbol{x}) = \boldsymbol{W}_v^{(2)} \max\left\{\boldsymbol{0}, \boldsymbol{W}_v^{(1)} \boldsymbol{h}_v(\boldsymbol{x}) + \boldsymbol{a}_v^{(1)}\right\} + \boldsymbol{a}_v^{(2)} + \Delta V_{N-1}(\boldsymbol{x})$$

with $\Delta V_{N-1}(\boldsymbol{x})$ being PWA in $\boldsymbol{x}$. Hence, the Q-function (5.7) can be written as

$$\begin{aligned}
Q_N(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{W}_v^{(2)} \max\left\{\boldsymbol{0}, \boldsymbol{W}_v^{(1)} \boldsymbol{h}_v(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) + \boldsymbol{a}_v^{(1)}\right\} \\
+ \boldsymbol{a}_v^{(2)} + \Delta V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) + \ell(\boldsymbol{x}, \boldsymbol{u}).
\end{aligned}$$

Now, the proposed solution to 5.2 involves the ANN

$$\boldsymbol{\Phi}(\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u})) = \boldsymbol{W}^{(2)} \max\left\{\boldsymbol{0}, \boldsymbol{W}^{(1)}\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u}) + \boldsymbol{a}^{(1)}\right\} + \boldsymbol{a}^{(2)}.$$

Hence, substituting the proposed weights and biases and evaluating (5.12) results in

$$\Delta Q_N(\boldsymbol{x},\boldsymbol{u}) = \Delta V_{N-1}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u}) + \ell(\boldsymbol{x},\boldsymbol{u}) - \max\{0, \ell(\boldsymbol{x},\boldsymbol{u})\}. \tag{5.19}$$

Clearly, $\Delta V_{N-1}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u})$ is PWA in $\boldsymbol{x}$ and $\boldsymbol{u}$. This completes the proof since $\ell(\boldsymbol{x},\boldsymbol{u}) - \max\{0, \ell(\boldsymbol{x},\boldsymbol{u})\} = 0$ due to positive semi-definiteness of ℓ. ■

From a theoretical point of view, Theorem 5.6 is quite useful since it closely relates Problems 5.1 and 5.2. Unfortunately, from a practical point of view, the proposed mapping (5.18) is not convenient. In fact, Q-learning allows model-free learning but evaluating (5.18) (for data preparation) requires a model in terms of $\boldsymbol{A}$ and $\boldsymbol{B}$. Fortunately, this issue can be easily solved. To this end, we assume that the solution to Problem 5.1 builds on the mapping

$$\boldsymbol{h}_v(\boldsymbol{x}) := \begin{pmatrix}\boldsymbol{x}^\top & \boldsymbol{x}_1^2 & \boldsymbol{x}_1\boldsymbol{x}_2 & \dots & \boldsymbol{x}_1\boldsymbol{x}_n & \dots & \boldsymbol{x}_n^2\end{pmatrix}^\top, \tag{5.20}$$

i.e., an extension of the state $\boldsymbol{x}$ by all possible products of two individual states. We note that this assumption is reasonable for various reasons. First, (5.20) includes our solution for the one-dimensional case in Theorem 5.3. Second, similar approaches have been considered for (loosely) related problems [8, 107]. Third, we show in Section 5.4.4 that (5.20) also applies to exemplary extensions with $n > 1$. Now, the structure of (5.20) can be easily extended to the $\boldsymbol{x}$-$\boldsymbol{u}$-domain yielding the mapping

$$\boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u}) := \begin{pmatrix}\boldsymbol{h}_v(\boldsymbol{x}) \\ \boldsymbol{u} \\ \boldsymbol{u}_1\boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_m^2 \\ \boldsymbol{x}_1\boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{x}_n\boldsymbol{u}_m\end{pmatrix} \tag{5.21}$$

that includes, in addtion to $\boldsymbol{x}$ and $\boldsymbol{u}$, all possible products of two entries in $\begin{pmatrix}\boldsymbol{x}^\top & \boldsymbol{u}^\top\end{pmatrix}$. As shown next, there exists a linear relation between $\boldsymbol{h}_q$ from Theorem 5.6 and $\boldsymbol{h}'_q$ as in (5.21).

Lemma 5.7. *There exists a matrix $\boldsymbol{L} \in \mathbb{R}^{w_0 \times w'_0}$ such that*

$$\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{L}\boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u}), \tag{5.22}$$

where $w_0 := n(n+3)/2 + 1$ and $w'_0 := (n+m)(n+m+3)/2$.

Proof. Clearly, each entry in $\boldsymbol{h}_v(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u})$ can be written as a linear combination of the entries in $\boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u})$. The same observation holds for $\ell(\boldsymbol{x},\boldsymbol{u})$. The corresponding coefficients form the entries of $\boldsymbol{L}$. The dimensions w_0 and w'_0 simply reflect the output dimensions of $\boldsymbol{h}_q$ and $\boldsymbol{h}'_q$. ■

Relation (5.22) allows to reformulate the statement in Theorem 5.6 using the more convenient network inputs $\boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u})$. In fact, as specified in the following corollary, the matrix $\boldsymbol{L}$ can simply be including in the construction of $\boldsymbol{W}^{(1)}$.

Corollary 5.8. *Consider the same solution to Problem 5.1 as in Theorem 5.6. Further, let $\boldsymbol{L}$ satisfy* (5.22)*. Then, a solution to Problem 5.2 can be obtained analogously to Theorem 5.6 except that $\boldsymbol{W}^{(1)}$ and $\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u})$ are replaced by*

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \boldsymbol{W}_v^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{pmatrix} \boldsymbol{L}$$

and $\boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u})$, respectively.

Inspired by Corollaries 5.4 and 5.5, we could now aim for statements about suitable ReLU-ANN for describing ΔQ_N and Q_N, respectively. However, Theorem 5.6 assumes a solution to Problem 5.1 and it is unclear whether such a solution exists for $n > 1$ and, if it exists, how it is structured. Thus, we only provide the following statement for $n = 1$.

Corollary 5.9. *Let $n = 1$ and assume V_{N-1} is defined on s' regions. Then, Q_N as in* (5.7) *can be exactly described by a ReLU-ANN with one hidden layer of width $w_1 = 2s' + 1$ and the inputs $\boldsymbol{\xi} = \boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u})$.*

Proof. We know from Theorem 5.3 that Problem 5.1 can be solved for $N' = N - 1$ based on a ReLU-ANN with one hidden layer of width s'. Now, Theorem 5.6 tells us that adding one neuron allows us to solve Problem 5.2. From (5.19), we further infer that the resulting PWA $\Delta Q_N(\boldsymbol{x},\boldsymbol{u})$ is equivalent to $\Delta V_{N-1}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u})$. Corollary 5.4 shows that another s' neurons are required to represent ΔV_{N-1}, which already leads to the proposed width of $2s' + 1$. It remains to comment on suitable inputs for the ANN. In this context, we first note that $\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u})$ from Theorem 5.6 contains $\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u}$ for $\boldsymbol{h}_v$ as in (5.20). Hence, $\Delta V_{N-1}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u})$ can be described based on the ReLU-ANN for ΔV_{N-1} with the inputs $\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u})$. This completes the proof since Lemma 5.7 implies that the inputs $\boldsymbol{h}_q(\boldsymbol{x},\boldsymbol{u})$ can easily be replaced by $\boldsymbol{\xi} = \boldsymbol{h}'_q(\boldsymbol{x},\boldsymbol{u})$ without changing the width w_1. ■

5.4 Numerical examples

We illustrate (and extend) our results with four numerical examples. The two first examples demonstrate the application of Theorems 5.3 and 5.6. Moreover, we point out the benefits of the proposed ANN for learning the Q-function. The last example shows a possible direction for an open extension to systems with $n > 1$.

5.4.1 Exact representation of the optimal value function

We consider the system from [108, Ex. 2] with the dynamics

$$x(k+1) = \tfrac{6}{5}x(k) + u(k) \tag{5.23}$$

and the constraints $x \in [-10, 10] \subset \mathbb{R}$ and $u \in [-1, 1] \subset \mathbb{R}$. As in [108], we choose $Q = 19/5$, $R = 1$, $P = 5$, and $\mathcal{T} = [-1, 1]$. Finally, we select $N = 1$ for illustration purposes here. Explicitly solving (5.2) then leads to

$$V_1(x) = \begin{cases} 11x^2 + 12x + 6 & \text{if} \quad x \in \mathcal{R}^{(1)}, \\ 5x^2 & \text{if} \quad x \in \mathcal{R}^{(2)}, \\ 11x^2 - 12x + 6 & \text{if} \quad x \in \mathcal{R}^{(3)}, \end{cases} \tag{5.24}$$

where the three regions refer to the intervals

$$\mathcal{R}^{(1)} = \left[-\tfrac{5}{3}, -1\right], \quad \mathcal{R}^{(2)} = [-1, 1], \quad \mathcal{R}^{(3)} = \left[1, \tfrac{5}{3}\right].$$

Now, according to Theorem 5.3, a solution to Problem 5.1 is given in terms of the parameters

$$W_v^{(1)} := \begin{pmatrix} -\frac{8}{3} & -1 \\ 0 & -1 \\ \frac{8}{3} & -1 \end{pmatrix}, \qquad a_v^{(1)} := \begin{pmatrix} -\frac{5}{3} \\ 1 \\ -\frac{5}{3} \end{pmatrix}, \tag{5.25a}$$

$$W_v^{(2)} := \begin{pmatrix} -11 & -5 & -11 \end{pmatrix}, \qquad a_v^{(2)} := 0 \tag{5.25b}$$

and the mapping $h_v(x) := (\, x \;\; x^2 \,)^\top$. Figure 5.1 confirms this result by visualizing the PWA ΔV_1. We can further exploit the structure of ΔV_1 based on Corollary 5.4. In fact, with $\kappa^{(i)}$ and $\beta^{(i)}$ as in the proof of Theorem 5.3, we find that ΔV_1 can be described as an ANN of the form (5.8) with the parameters

$$W_\Delta^{(1)} := \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}, \qquad a_\Delta^{(1)} := \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix}, \tag{5.26a}$$

$$W_\Delta^{(2)} := \begin{pmatrix} \frac{52}{3} & 0 & \frac{52}{3} \end{pmatrix}, \qquad a_\Delta^{(2)} := 5. \tag{5.26b}$$

By combining both results, we can construct a ReLU-ANN that exactly describes V_1. In fact, the parametrization

$$W^{(1)} := \begin{pmatrix} W_v^{(1)} \\ W_\Delta^{(1)} \end{pmatrix}, \qquad a^{(1)} := \begin{pmatrix} a_v^{(1)} \\ a_\Delta^{(1)} \end{pmatrix},$$

$$W^{(2)} := \begin{pmatrix} W_v^{(2)} & W_\Delta^{(2)} \end{pmatrix}, \qquad a^{(2)} := a_\Delta^{(2)}$$

leads to $V_1(x) = \Phi(h_v(x))$ and thus confirms Corollary 5.5.

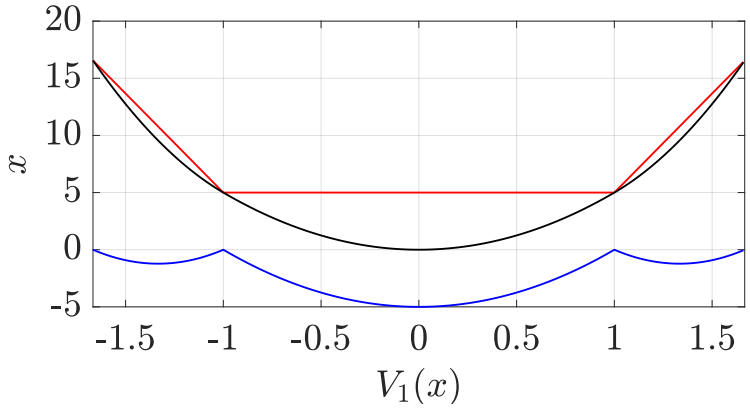


Figure 5.1. Illustration of the PWQ function V_1 (black), the proposed ANN for the quadratic terms (blue), and the PWA difference ΔV_1 (red) for the example in Section 5.4.1.

5.4.2 Adaptation for Q-function

We next illustrate the application of Theorem 5.6. In order to reuse the results from the previous example, we consider $N = 2$ here and consequently the Q-function

$$Q_2(\boldsymbol{x}, \boldsymbol{u}) = \ell(\boldsymbol{x}, \boldsymbol{u}) + V_1\left(\tfrac{6}{5}\boldsymbol{x} + \boldsymbol{u}\right) \tag{5.27}$$

with V_1 as in (5.24). Now, according to Theorem 5.6, a solution to Problem 5.2 can be obtained by substituting (5.25) in (5.17) and by considering the mapping

$$\boldsymbol{h}_q(\boldsymbol{x}, \boldsymbol{u}) := \begin{pmatrix} \boldsymbol{h}_v\left(\frac{6}{5}\boldsymbol{x} + \boldsymbol{u}\right) \\ \ell(\boldsymbol{x}, \boldsymbol{u}) \end{pmatrix} = \begin{pmatrix} \frac{6}{5}\boldsymbol{x} + \boldsymbol{u} \\ \frac{36}{25}\boldsymbol{x}^2 + \frac{12}{5}\boldsymbol{x}\boldsymbol{u} + \boldsymbol{u}^2 \\ \frac{19}{5}\boldsymbol{x}^2 + \boldsymbol{u}^2 \end{pmatrix}.$$

Clearly, evaluating this mapping requires the model (5.23). Fortunately, Lemma 5.7 and Corollary 5.8 tell us that we can easily switch to the more convenient network inputs

$$\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u}) := \begin{pmatrix} \boldsymbol{x} & \boldsymbol{x}^2 & \boldsymbol{u} & \boldsymbol{u}^2 & \boldsymbol{x}\boldsymbol{u} \end{pmatrix}^\top. \tag{5.28}$$

To this end, we quickly verify that (5.22) is satisfied by

$$\boldsymbol{L} = \begin{pmatrix} \frac{6}{5} & 0 & 1 & 0 & 0 \\ 0 & \frac{36}{25} & 0 & 1 & \frac{12}{5} \\ 0 & \frac{19}{5} & 0 & 1 & 0 \end{pmatrix}.$$

Hence, Theorem 5.6 in combination with the parameters (5.25) and Corollary 5.8 implies that Problem 5.2 is solved by

$$\boldsymbol{W}_q^{(1)} := \begin{pmatrix} -\frac{16}{5} & -\frac{36}{25} & -\frac{8}{3} & -1 & -\frac{12}{5} \\ 0 & -\frac{36}{25} & 0 & -1 & -\frac{12}{5} \\ \frac{16}{5} & -\frac{36}{25} & \frac{8}{3} & -1 & -\frac{12}{5} \\ 0 & \frac{19}{5} & 0 & 1 & 0 \end{pmatrix}, \qquad \boldsymbol{a}_q^{(1)} := \begin{pmatrix} -\frac{5}{3} \\ 1 \\ -\frac{5}{3} \\ 0 \end{pmatrix},$$
$$\boldsymbol{W}_q^{(2)} := \begin{pmatrix} -11 & -5 & -11 & 1 \end{pmatrix}, \qquad \boldsymbol{a}_q^{(2)} := 0.$$

and the mapping (5.28). In order to describe the entire Q-function, it remains to construct an ANN for $\Delta Q_2(\boldsymbol{x}, \boldsymbol{u}) = \Delta V_1(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u})$. Clearly, we can reuse the parametrization for ΔV_1 from (5.26). However, we have to include a matrix that selects $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ from $\boldsymbol{h}_q$ and we have to take into account the switching to $\boldsymbol{h}'_q$ via $\boldsymbol{L}$. Doing so, we can easily confirm that $\Delta V_1(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) = \boldsymbol{\Phi}(\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u}))$ holds for

$$\boldsymbol{W}_\delta^{(1)} := \boldsymbol{W}_\Delta^{(1)} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \boldsymbol{L} = \begin{pmatrix} -\frac{6}{5} & 0 & -1 & 0 & 0 \\ \frac{6}{5} & 0 & 1 & 0 & 0 \\ \frac{6}{5} & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$\boldsymbol{a}_\delta^{(1)} := \boldsymbol{a}_\Delta^{(1)},\ \boldsymbol{W}_\delta^{(2)} := \boldsymbol{W}_\Delta^{(2)}, \text{ and } \boldsymbol{a}_\delta^{(2)} := \boldsymbol{a}_\Delta^{(2)}.$$

Combining the solution to Problem 5.2 and the description of ΔQ_2 from above leads to a ReLU-ANN for Q_2 with the parameters

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \boldsymbol{W}_q^{(1)} \\ \boldsymbol{W}_\delta^{(1)} \end{pmatrix}, \qquad \boldsymbol{a}^{(1)} := \begin{pmatrix} \boldsymbol{a}_q^{(1)} \\ \boldsymbol{a}_\delta^{(1)} \end{pmatrix},$$
$$\boldsymbol{W}^{(2)} := \begin{pmatrix} \boldsymbol{W}_q^{(2)} & \boldsymbol{W}_\delta^{(2)} \end{pmatrix}, \qquad \boldsymbol{a}^{(2)} := \boldsymbol{a}_\delta^{(2)}$$

and the inputs $\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u})$. As predicted by Corollary 5.9, this ANN has one hidden layer of width $w_1 = 2s' + 1 = 7$.

5.4.3 Learning of the Q-function

The presented work is motivated by the observation that, in the context of learning, ANN topologies are often chosen based on trial-and-error. Our results show that, for MPC-related learning, more educated choices exist that even allow to exactly describe the functions of interest. Interestingly, the identified topologies (summarized in the Corollaries 5.5 and 5.9) seem to be useful beyond the proposed exact descriptions. To substantiate this observation, we investigated different learning-based approximations of Q_2 from the previous example. More precisely, we trained various ReLU-ANN with different inputs, widths, and depths (using supervised learning and Matlab's Deep Learning Toolbox [109] for simplicity) over 1000 epochs. We repeated the training 100

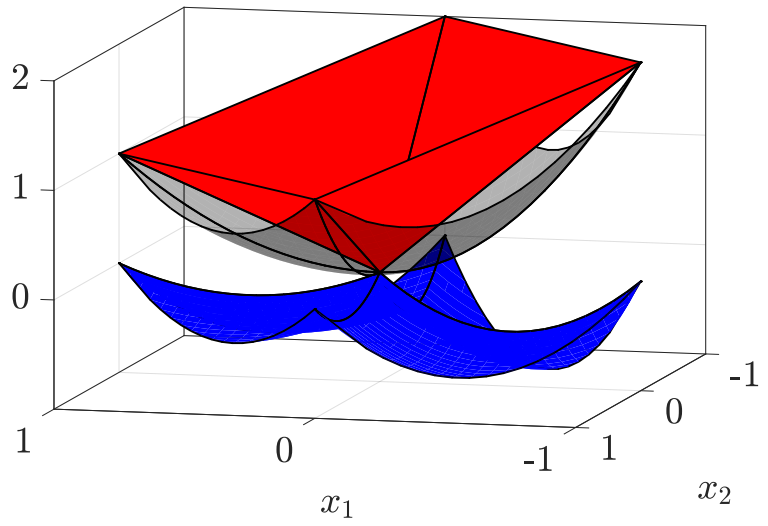


Figure 5.2. Illustration of the PWQ function V_N (grey), the proposed ANN for the quadratic terms (blue), and the PWA difference ΔV_N (red) for the example in Section 5.4.4.

times for each ReLU-ANN, where we used the same randomly sampled points from the Q-function as training data for every trial. At the end of every trial, we computed the root-mean-square-error (RMSE). The mean values of the resulting RMSE are listed in Table 5.1. As apparent from the data, the proposed topology with one hidden layer ($l = 1$) of width $w_1 = 7$ and the inputs $\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u})$ performed best even though some ANN incorporated more parameters.

Table 5.1. RMSE for different ReLU-ANN (where w_i refers to $w_1, \ldots, w_l$ and $\#_p$ reflects the number of parameters).

ξ	l	w_i	$\#_p$	RMSE
$\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u})$	1	5	36	0.4084
$\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u})$	1	6	43	0.2589
$\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u})$	1	7	50	0.1854
$\boldsymbol{h}'_q(\boldsymbol{x}, \boldsymbol{u})$	1	8	57	0.1999
$(\boldsymbol{x} \ \boldsymbol{u})^\top$	1	12	49	0.2065
$(\boldsymbol{x} \ \boldsymbol{u})^\top$	2	5	51	0.3092
$(\boldsymbol{x} \ \boldsymbol{u})^\top$	3	4	57	0.2735

5.4.4 Extension to two-dimensional case

We next show that solutions to Problem 5.1 may also exist for $n > 1$ and that they may be derived using strategies similar to those underlying our solution for $n = 1$. To this end, we consider the fictive function

$$V_N(\boldsymbol{x}) := \begin{cases} x_1^2 + x_2^2 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ 2x_1^2 + x_2^2 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(2)}, \\ 2x_1^2 + 2x_2^2 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(3)}, \\ x_1^2 + 2x_2^2 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(4)} \end{cases} \tag{5.29}$$

(without specifying N) defined on the regions

$$\begin{aligned} \mathcal{R}^{(1)} &:= \{\boldsymbol{x} \in \mathbb{R}^2 \,|\, x_1 \geq 0,\, x_2 \geq 0, \quad x_1 + x_2 \leq 1\}, \\ \mathcal{R}^{(2)} &:= \{\boldsymbol{x} \in \mathbb{R}^2 \,|\, x_1 \leq 0,\, x_2 \geq 0,\, -x_1 + x_2 \leq 1\}, \\ \mathcal{R}^{(3)} &:= \{\boldsymbol{x} \in \mathbb{R}^2 \,|\, x_1 \leq 0,\, x_2 \leq 0,\, -x_1 - x_2 \leq 1\}, \\ \mathcal{R}^{(4)} &:= \{\boldsymbol{x} \in \mathbb{R}^2 \,|\, x_1 \geq 0,\, x_2 \leq 0, \quad x_1 - x_2 \leq 1\}. \end{aligned}$$

We next show that

$$\begin{aligned} \boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x})) = &- \max\{0, x_1(-x_1 + x_2 - 1)\} \\ &- \max\{0, x_2(-x_1 - x_2 - 1)\} \\ &- \max\{0, x_1(-x_1 - x_2 - 1)\} \\ &- \max\{0, x_2(x_1 - x_2 - 1)\} \\ &- 0.5\max\{0, -x_2(x_1 + x_2 - 1)\} \\ &- 0.5\max\{0, -x_1(x_1 + x_2 - 1)\} \\ &- 0.5\max\{0, -x_2(-x_1 + x_2 - 1)\} \\ &- 0.5\max\{0, -x_1(x_1 - x_2 - 1)\} \end{aligned} \tag{5.30}$$

with $\boldsymbol{h}_v$ as in (5.20) represents a solution to Problem 5.1. In this context, we first note that $V_N(\boldsymbol{x}) - \boldsymbol{\Phi}(\boldsymbol{h}_v(\boldsymbol{x}))$ is indeed PWA as visualized in Figure 5.2. Furthermore, $\boldsymbol{\Phi}$ equals (5.8) with the parameters

$$\boldsymbol{W}^{(1)} = \begin{pmatrix} -1 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 & -1 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & 1 & -1 \\ 1 & 0 & -1 & 1 & 0 \end{pmatrix}, \; \boldsymbol{W}^{(2)} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \end{pmatrix}^\top,$$

$\boldsymbol{a}^{(1)} = \boldsymbol{0}$, and $\boldsymbol{a}^{(2)} = 0$ and the inputs $\boldsymbol{\xi} := \boldsymbol{h}_v(\boldsymbol{x})$. Hence, (5.30) is indeed a solution to Problem 5.1.

We briefly comment on the derivation of this solution. First, it is easy to see that every ReLU in (5.30) is constructed in such a way that it is either active or inactive on the individual regions $\mathcal{R}^{(i)}$. For instance, $\max\{0, x_1(-x_1 + x_2 - 1)\}$ is active on $\mathcal{R}^{(2)}, \mathcal{R}^{(3)}$ and inactive on $\mathcal{R}^{(1)}, \mathcal{R}^{(4)}$. In total, four ReLU are active on every region. Since we can freely choose the eight weighting factors of the ReLU, we can use these degrees of freedom to eliminate the eight quadratic terms in (5.29). The strategy is closely related to the proof of Theorem 5.3 and can, in principle, be generalized. However, formulating and solving the underlying system of equations is highly non-trivial.

5.5 Conclusions and Outlook

In this paper, we presented methods for exactly describing PWQ functions using tailored ANN with ReLU activations. In particular, we showed that the OVF in linear MPC can always be described by a ReLU-ANN with one hidden layer for the special case $n = 1$ (see Thm. 5.3 and Cor. 5.5). Moreover, according Theorem 5.6, a given description for the OVF can always be modified for exactly representing the associated Q-function (for any $n \geq 1$).

We illustrated our results with various numerical examples. Two observations are of particular interest. First, the example in Section 5.4.3 indicates an advantage of the proposed ANN topologies for learning the Q-function. Second, the example in Section 5.4.4 shows that the approach for representing the OVF might be extendable to $n > 1$. Both observations offer interesting directions for further research.

Chapter 6

Tailored max-out networks for learning convex PWQ functions*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. Convex piecewise quadratic (PWQ) functions frequently appear in control and elsewhere. For instance, it is well-known that the optimal value function (OVF) as well as Q-functions for linear MPC are convex PWQ functions. Now, in learning-based control, these functions are often represented with the help of artificial neural networks (NN). In this context, a recurring question is how to choose the topology of the NN in terms of depth, width, and activations in order to enable efficient learning. An elegant answer to that question could be a topology that, in principle, allows to exactly describe the function to be learned. Such solutions are already available for related problems. In fact, suitable topologies are known for piecewise affine (PWA) functions that can, for example, reflect the optimal control law in linear MPC. Following this direction, we show in this paper that convex PWQ functions can be exactly described by max-out-NN with only one hidden layer and two neurons.

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

6.1 Introduction

Learning-based and data-driven methods are heavily used in all kind of sectors including health, finance, transportation, industry, or energy. While the applications are as diverse as the sectors, using supervised or reinforcement learning in order to approximate an unknown function is still a recurring task. Popular choices for the models to be learned then are, for example, artificial neural networks (NN) or Bayesian networks. In both cases, choosing a suitable network topology is crucial for efficient learning. As well-founded design rules are rare, choices are often made by trial-and-error. However, some tasks allow for more educated guesses. For instance, if the function to be approximated is known to be piecewise affine (PWA), NN with max-out or rectified linear unit (ReLU) activations are ideal as their input-output relation is likewise PWA. Moreover, if also the number of affine segments is available, then it is even possible to specify NN that, in principle, allow to exactly describe the function of interest (see, e.g., [23, 25, 73]). This observation has led to tailored NN for (linear) model predictive control (MPC) [6, 7, 26], where the optimal control law is well-known to be PWA [20].

Another interesting property of linear MPC is that the optimal value function (OVF) is piecewise quadratic (PWQ) and convex. In this context, the OVF or related Q-functions play a central role for data-driven predictive control based on reinforcement learning [10, 15]. Nevertheless, tailored NN for representing convex PWQ have only been rarely considered. One exception can be found in [P1], where we recently showed that one-dimensional PWQ functions with s segments can be exactly described by a ReLU-NN with one hidden layer of width $2s$ (and an augmented input). While this observation was interesting, the underlying method is restricted in that an extension to the multi-dimensional inputs (or states in the context of MPC) seems hard if not impossible. As a consequence, we reconsider the problem of identifying tailored NN for convex PWQ functions here. However, in contrast to [P1], we will focus on max-out-NN as they seem more suitable for exploiting convexity and for addressing multi-dimensional inputs. In fact, it is well-known that convex PWA functions (with one or more inputs) can be trivially represented by one max-out neuron (per output). While this feature does, in general, not apply to convex PWQ functions φ, we will show that a systematic modification of φ allows to derive a representation via two max-out neurons.

We organize the presentation of the novel approach as follows. In Section 6.2 we provide some preliminaries on PWQ functions and background on NN. Further, we briefly summarize the results from [P1]. We present our main result, i.e., tailored max-out-NN for exactly describing PWQ functions, in Section 6.3. More specifically, we construct max-out-NN that exactly describes a convex PWQ function for the special case of one-dimensional inputs (i.e., $n = 1$) and show a possible direction for an extension to functions with higher-dimensional inputs (i.e. $n > 1$) on an exemplary function in the Section 6.4.

Finally, in the aforementioned section we also illustrate our main results and state conclusions in Section 6.5.

6.2 Preliminaries and background

Throughout the paper, we deal with convex PWQ functions of the form

$$\varphi(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^\top \boldsymbol{Q}^{(1)} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{l}^{(1)} + c^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \boldsymbol{x}^\top \boldsymbol{Q}^{(s)} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{l}^{(s)} + c^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{6.1}$$

for scalar or multi-dimensional inputs $\boldsymbol{x} \in \mathbb{R}^n$. The parameters $\boldsymbol{Q}^{(i)} \in \mathbb{R}^{n \times n}$, $\boldsymbol{l}^{(i)} \in \mathbb{R}^{(n \times 1)}$, and $c^{(i)} \in \mathbb{R}$ reflect the quadratic, the linear, and the constant parts of the various segments. The regions $\mathcal{R}^{(i)} \subset \mathbb{R}^n$ are convex polytopes with pairwise disjoint interiors.

6.2.1 Neural networks with rectifier and max-out activations

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

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{f}^{(\ell+1)} \circ \boldsymbol{g}^{(\ell)} \circ \boldsymbol{f}^{(\ell)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{\xi}). \tag{6.2}$$

Here, the functions $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{p_i w_i}$ for $i \in \{1, \ldots, \ell\}$ refer to preactivations, where the parameter $p_i \in \mathbb{N}$ allows to consider "multi-channel" preactivations as required for max-out. Moreover, $\boldsymbol{g}^{(i)} : \mathbb{R}^{p_i w_i} \to \mathbb{R}^{w_i}$ stand for activation functions and $\boldsymbol{f}^{(\ell+1)} : \mathbb{R}^{w_\ell} \to \mathbb{R}^{w_{\ell+1}}$ reflects postactivation. The functions $\boldsymbol{f}^{(i)}$ are typically affine, i.e.,

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

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i w_i \times w_{i-1}}$ is a weighting matrix, $\boldsymbol{b}^{(i)} \in \mathbb{R}^{p_i w_i}$ is a bias vector, and $\boldsymbol{y}^{(i-1)}$ denotes the output of the previous layer with $\boldsymbol{y}^{(0)} := \boldsymbol{\xi}$.

Now, various activation functions have been proposed. As already stated in the introduction, we here focus on ReLU and max-out neurons [21]. The corresponding activation functions are specified as

$$\boldsymbol{g}^{(i)}(\boldsymbol{z}^{(i)}) = \max\left\{\boldsymbol{0}, \boldsymbol{z}^{(i)}\right\} := \begin{pmatrix} \max\left\{0, z_1^{(i)}\right\} \\ \vdots \\ \max\left\{0, z_{w_i}^{(i)}\right\} \end{pmatrix}$$

for ReLU and

$$\boldsymbol{g}^{(i)}(\boldsymbol{z}^{(i)}) = \begin{pmatrix} \max\limits_{1 \le j \le p_i} \left\{\boldsymbol{z}_j^{(i)}\right\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1 \le j \le p_i w_i} \left\{\boldsymbol{z}_j^{(i)}\right\} \end{pmatrix}$$

for max-out, where $z_j^{(i)}$ denotes the j-th component of $z^{(i)} \in \mathbb{R}^{p_i w_i}$ and where we use the shorthand notation

$$\max_{1 \le j \le p_i} \{z_j^{(i)}\} := \max\{z_1^{(i)}, \dots, z_{p_i}^{(i)}\}.$$

We will refer to the resulting networks as ReLU-NN and max-out-NN, respectively.

6.2.2 Tailored ReLU-NN for representing PWQ functions

We recently showed in [P1] that PWQ functions can be represented by a ReLU-NN for the special case of scalar variables x (i.e., $n = 1$). Since this case will also be in the focus of this paper, we specify the notation of (6.1) for ease of presentation. Hence, we consider

$$\varphi(x) = \begin{cases} q_1 x^2 + l_1 x + c_1 & \text{if } x \in [\underline{x}_1, \overline{x}_1], \\ \vdots & \vdots \\ q_s x^2 + l_s x + c_s & \text{if } x \in [\underline{x}_s, \overline{x}_s] \end{cases} \tag{6.3}$$

instead of (6.1) for the majority of the paper, where we assume that the relation $\underline{x}_i < \overline{x}_i = \underline{x}_{i+1} < \overline{x}_{i+1}$ holds for each $i \in \{1, \dots, s-1\}$ and that the interval $[\underline{x}_1, \overline{x}_s]$ is bounded. We further define

$$\varphi_i(x) := q_i x^2 + l_i x + c_i.$$

The specification further allows to particularize (conditions for) continuity and convexity. In fact, these properties require

$$q_i \overline{x}_i^2 + l_i \overline{x}_i + c_i = q_{i+1} \overline{x}_i^2 + l_{i+1} \overline{x}_i + c_{i+1}, \tag{6.4a}$$

$$2q_i \overline{x}_i + l_i \le 2q_{i+1} \overline{x}_i + l_{i+1} \tag{6.4b}$$

for every $i \in \{1, \dots, s-1\}$ as well as $q_i \ge 0$ for every $i \in \{1, \dots, s\}$. Now, according to [P1, Cor. 3], (6.3) can be represented by a ReLU-NN with one hidden layer of width $w_1 = 2s$. In fact, [P1, Thm. 1] combined with [26, Thm. 1] leads to the ReLU-NN

$$\mathbf{\Phi}(\boldsymbol{\xi}) = \boldsymbol{W}^{(2)} \max\left\{\mathbf{0}, \boldsymbol{W}^{(1)} \boldsymbol{\xi} + \boldsymbol{b}^{(1)}\right\} + \boldsymbol{b}^{(2)} \tag{6.5}$$

with the parameters

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \underline{x}_1 + \overline{x}_1 & -1 \\ \vdots & \vdots \\ \underline{x}_s + \overline{x}_s & -1 \\ -1 & 0 \\ 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{b}^{(1)} := \begin{pmatrix} -\underline{x}_1 \overline{x}_1 \\ \vdots \\ -\underline{x}_s \overline{x}_s \\ \overline{x}_1 \\ -\overline{x}_1 \\ \vdots \\ -\overline{x}_{s-1} \end{pmatrix},$$

$$\boldsymbol{W}^{(2)} := \begin{pmatrix} -q_1 & \dots & -q_s & \kappa_1 & \kappa_2 & \kappa_3 - \kappa_2 & \dots & \kappa_s - \kappa_{s-1} \end{pmatrix},$$

$$\boldsymbol{b}^{(2)} := q_1 \overline{x}_1^2 + l_1 \overline{x}_1 + c_1 = \varphi_1(\overline{x}_1),$$

and the augmented input $\boldsymbol{\xi}(x) := \begin{pmatrix} x & x^2 \end{pmatrix}^\top$, which is such that $\varphi(x) = \boldsymbol{\Phi}(\boldsymbol{\xi}(x))$ for every $x \in [\underline{x}_1, \overline{x}_s]$, where

$$\kappa_i := l_i + q_i(\underline{x}_i + \overline{x}_i)$$

for every $i \in \{1, \ldots, s\}$.

6.3 Tailored max-out-NN for PWQ functions

While the approach summarized in Section 6.2.2 works well for finding tailored ReLU-NN for PWQ functions with $n = 1$, the example in [P1, Sect. IV-D] shows that an extension to higher dimensional problems (with $n > 1$) is not straightforward. In fact, the central proof of [P1, Thm. 1] makes use of ReLU-neurons that are "active" only for $\boldsymbol{x}$ inside a certain region $\mathcal{R}^{(i)}$. While such a construction can easily be derived for $n = 1$, an extension to $n > 1$ is hard if not impossible. This observation motivates the work at hand, where we aim for an exact representation of convex PWQ functions that applies for $n \geq 1$. However, similar to the approach in [P1], we will initially focus on the special case $n = 1$. Nevertheless, we will illustrate that an extension to $n > 1$ is within reach with a numerical example in Section 6.4.2.

The central idea here is to exploit the convexity of the PWQ function more explicitly than in [P1, Thm. 1]. In this context, it is well-known that a convex PWA function $h(\boldsymbol{x})$ can be evaluated by computing the maximum of all affine segments $h_i(\boldsymbol{x})$ (independent of their various domains), i.e.,

$$h(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{h_i(\boldsymbol{x})\}. \tag{6.6}$$

It is easy to see that this operation is equivalent to the evaluation of a max-out neuron. Unfortunately, the relation

$$\varphi(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{\varphi_i(\boldsymbol{x})\}, \tag{6.7}$$

which is inspired by (6.6), does not hold in general. In fact, Fig. 6.3 shows a convex PWQ function for which (6.7) is violated since some quadratic segments "dominate" outside their domains. The leading idea now is to compensate this defect by adding a suitable convex PWA function $h(\boldsymbol{x})$ to $\varphi(\boldsymbol{x})$ such that

$$\varphi(\boldsymbol{x}) + h(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{\varphi_i(\boldsymbol{x}) + h_i(\boldsymbol{x})\} \tag{6.8}$$

applies for all $\boldsymbol{x}$ in the domain of φ. Clearly, this would immediately allow to represent φ in terms of a max-out network with one hidden layer and two neurons. In fact, this follows from the trivial observation that

$$\varphi(\boldsymbol{x}) = \underbrace{\varphi(\boldsymbol{x}) + h(\boldsymbol{x})}_{\text{first neuron}} - \underbrace{h(\boldsymbol{x})}_{\text{second neuron}},$$

where the two neurons reflect the right-hand sides in (6.6) and (6.8). Based on the previous discussion, we formulate the following conjecture that will guide us through the remaining paper.

Conjecture 6.1. *Every convex PWQ function can be exactly represented by a max-out-NN with one hidden layer and two neurons.*

Before deriving a proof of the conjecture for the special case $n = 1$, we briefly note that it can be considered as a tailored extension of the central observation in [21]. In fact, [21, Thm. 4.3] states that that a max-out-NN with one hidden layer and two neurons allows to approximate any continous function arbitrarily well (but not necessarily exactly).

Now, specifying the ideas from above for $n = 1$ leads to the consideration of convex PWQ functions of the form (6.3) and convex PWA functions

$$h(x) := \begin{cases} \alpha_1 x + \beta_1 & \text{if } x \in [\underline{x}_1, \overline{x}_1], \\ \quad \vdots & \quad \vdots \\ \alpha_s x + \beta_s & \text{if } x \in [\underline{x}_s, \overline{x}_s]. \end{cases} \tag{6.9}$$

Analogously to (6.4), continuity and convexity of h can now easily be specified by the conditions

$$\alpha_i \overline{x}_i + \beta_i = \alpha_{i+1} \overline{x}_i + \beta_{i+1} \quad \text{and} \tag{6.10a}$$

$$\alpha_i \leq \alpha_{i+1}, \tag{6.10b}$$

for every $i \in \{1, \ldots, s-1\}$, respectively. Now, aiming for the relation (6.8), we pose the additional constraints

$$\varphi_j(\underline{x}_i) + \alpha_j \underline{x}_i + \beta_j \leq \varphi_i(\underline{x}_i) + \alpha_i \underline{x}_i + \beta_i \tag{6.11a}$$

$$\varphi_j(\overline{x}_i) + \alpha_j \overline{x}_i + \beta_j \leq \varphi_i(\underline{x}_i) + \alpha_i \underline{x}_i + \beta_i + (\overline{x}_i - \underline{x}_i)(2q_i \underline{x}_i + l_i + \alpha_i) \tag{6.11b}$$

for every $i \in \{2, \ldots, s\}$ and $j \in \{1, \ldots, i-1\}$ as well as

$$\varphi_j(\overline{x}_i) + \alpha_j \overline{x}_i + \beta_j \leq \varphi_i(\overline{x}_i) + \alpha_i \overline{x}_i + \beta_i \tag{6.12a}$$

$$\varphi_j(\underline{x}_i) + \alpha_j \underline{x}_i + \beta_j \leq \varphi_i(\overline{x}_i) + \alpha_i \overline{x}_i + \beta_i - (\overline{x}_i - \underline{x}_i)(2q_i \overline{x}_i + l_i + \alpha_i) \tag{6.12b}$$

for every $i \in \{1, \ldots, s-1\}$ and $j \in \{i+1, \ldots, s\}$. The underlying concepts will be clarified in the proof of the following theorem.

Theorem 6.2. *Assume there exist $\alpha_1, \ldots, \alpha_s \in \mathbb{R}$ and $\beta_1, \ldots, \beta_s \in \mathbb{R}$ satisfying the constraints* (6.10)–(6.12)*. Then,*

$$\varphi(x) + h(x) = \max_{1 \leq i \leq s} \{\varphi_i(x) + \alpha_i x + \beta_i\}. \tag{6.13}$$

Proof. To prove (6.13), we consider any $i \in \{1, \ldots, s\}$ and $\hat{x} \in [\underline{x}_i, \overline{x}_i]$. By definition of φ and h, we then obtain

$$\varphi(\hat{x}) + h(\hat{x}) = \varphi_i(\hat{x}) + \alpha_i \hat{x} + \beta_i.$$

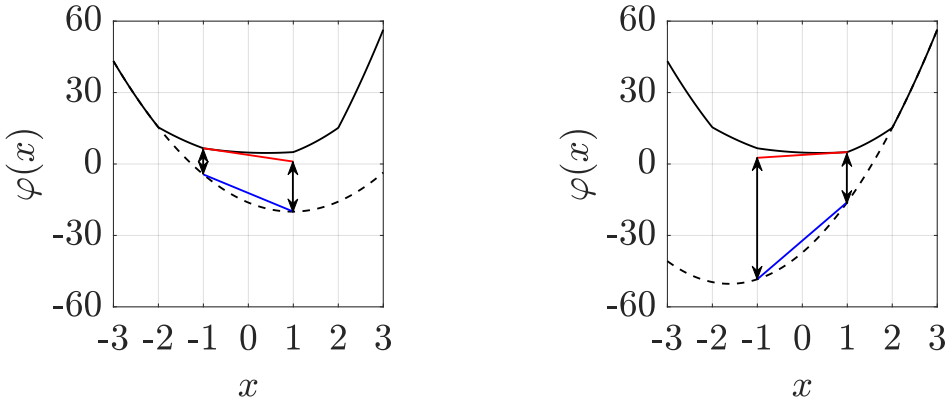


Figure 6.1. Exemplary illustration of the conditions (6.11) and (6.12) for $i = 3$ and $j \in \{1, 5\}$. In (a), (6.11) is shown for $j = 1 < i$ with the tangent on $\varphi(\underline{x}_i) + h(\underline{x}_i)$ in red evaluated at the points $\underline{x}_i$ and $\overline{x}_i$. In (b), (6.12) is illustrated for $j = 5 > i$, where the red line is the tangent on $\varphi(\overline{x}_i) + h(\overline{x}_i)$ evaluated at the point $\underline{x}_i$ and $\overline{x}_i$. In both plots, the blue line illustrates the interpolation of $\varphi_j(\hat{x}) + \alpha_j \hat{x} + \beta_j$ between the points $\underline{x}_i$ and $\overline{x}_i$.

Since the segment i also appears on the right-hand side of (6.13), we find

$$\varphi(\hat{x}) + h(\hat{x}) \leq \max_{1 \leq i \leq s} \{\varphi_i(\hat{x}) + \alpha_i \hat{x} + \beta_i\}$$

by construction. Hence, (6.13) can only be violated if there exists a $j \in \{1, \dots, s\} \setminus \{i\}$ such that

$$\varphi_i(\hat{x}) + \alpha_i \hat{x} + \beta_i < \varphi_j(\hat{x}) + \alpha_j \hat{x} + \beta_j, \tag{6.14}$$

which, however, leads to a contradiction. To see this, we distinguish the two cases (I) $j < i$ and (II) $i < j$. Regarding the first case, we initially note that (6.14) would require $\hat{x} > \underline{x}_i$ to comply with (6.11a). Hence,

$$\eta := \frac{\hat{x} - \underline{x}_i}{\overline{x}_i - \underline{x}_i} \in (0, 1].$$

Further, due to convexity of segments i and j, we require

$$\varphi_i(\hat{x}) + \alpha_i \hat{x} + \beta_i \geq \varphi_i(\underline{x}_i) + \alpha_i \underline{x}_i + \beta_i + (\hat{x} - \underline{x}_i)(2q_i \underline{x}_i + l_i + \alpha_i), \tag{6.15a}$$

$$\varphi_j(\hat{x}) + \alpha_j \hat{x} + \beta_j \leq (1 - \eta) \left(\varphi_j(\underline{x}_i) + \alpha_j \underline{x}_i + \beta_j\right) + \eta \left(\varphi_j(\overline{x}_i) + \alpha_j \overline{x}_i + \beta_j\right). \tag{6.15b}$$

In fact, $\varphi_i(\hat{x}) + \alpha_i \hat{x} + \beta_i$ is restricted to lie above (or at) any tangent of segment i and the right-hand side of (6.15a) reflects the tangent at $\underline{x}_i$ (see the red line in Fig. 6.1). Analogously $\varphi_j(\hat{x}) + \alpha_j \hat{x} + \beta_j$ has to lie below (or at) any interpolation between two points on segment j and the right-hand side of (6.15b) reflects the interpolation between the supporting points $\underline{x}_i$ and $\overline{x}_i$ (which is illustrated by the blue line in Fig. 6.1). Now, multiplying (6.11b) with η results in

$$\eta \left(\varphi_j(\overline{x}_i) + \alpha_j \overline{x}_i + \beta_j\right) \leq \eta \left(\varphi_i(\underline{x}_i) + \alpha_i \underline{x}_i + \beta_i\right) + (\hat{x} - \underline{x}_i)(2q_i \underline{x}_i + l_i + \alpha_i)$$

Next, we use (6.15a) to overestimate the right-hand side of the former relation and substitute the result on the right-hand side of (6.15b) in order to obtain

$$\begin{aligned}\varphi_j(\hat{x}) + \alpha_j\hat{x} + \beta_j \leq\ & (1-\eta)\left(\varphi_j(\underline{x}_i) + \alpha_j\underline{x}_i + \beta_j\right)\\ & - (1-\eta)\left(\varphi_i(\underline{x}_i) + \alpha_i\underline{x}_i + \beta_i\right)\\ & + \varphi_i(\hat{x}) + \alpha_i\hat{x} + \beta_i.\end{aligned}$$

Finally, taking (6.11a) and $1-\eta \in [0,1)$ into account, we find

$$\varphi_j(\hat{x}) + \alpha_j\hat{x} + \beta_j \leq \varphi_i(\hat{x}) + \alpha_i\hat{x} + \beta_i,$$

which indeed contradicts (6.14). Since the second case can be handled analogously, the proof is complete. ■

Remark 6.3. *Since the conditions* (6.10)–(6.12) *required to guarantee* (6.13) *are affine in α_i and β_i, we might be able to compute a suitable PWA function $h(x)$ by solving an optimization problem (OP) of the form*

$$\begin{aligned}\min_{\alpha_1,\beta_1,\ldots,\alpha_s,\beta_s}\ & J\left(\alpha_1,\beta_1,\ldots,\alpha_s,\beta_s\right) \qquad (6.16)\\ \text{s.t.}\quad & \text{(6.10)–(6.12)}\ \textit{(for i and j as above).}\end{aligned}$$

Here, the cost function J can be chosen to enforce certain shapes of h. For instance, if the impact of h should be as small as possible, $\sum_{i=1}^{s}\alpha_i^2+\beta_i^2$ is a suitable choice for J, which results in a quadratic program (QP).

It remains to comment on the feasibility of the constraints (6.10)–(6.12) and, hence, the feasibility of (6.16). In this context, Theorem 6.5 further below states the constraints are always feasible since a feasible solution can always be computed according to Algorithm 6.4.

Algorithm 6.4. *Feasible solution to* (6.10)–(6.12).

initialize $\alpha_i \leftarrow 0$ *and* $\beta_i \leftarrow 0$ *for every* $i \in \{1, \dots, s\}$
for $i = 1, \dots, s$ ***do***
  set the auxiliary quantities $\gamma_1 \leftarrow \varphi_i(\underline{x}_i) + \alpha_i \underline{x}_i + \beta_i$, $\gamma_2 \leftarrow \gamma_1 + (\overline{x}_i - \underline{x}_i)(2q_i \underline{x}_i + l_i + \alpha_i)$, $\gamma_3 \leftarrow \varphi_i(\overline{x}_i) + \alpha_i \overline{x}_i + \beta_i$, *and* $\gamma_4 \leftarrow \gamma_3 - (\overline{x}_i - \underline{x}_i)(2q_i \overline{x}_i + l_i + \alpha_i)$
  for $j = 1, \dots, i-1$ ***do***
    set $\Delta\alpha \leftarrow 0$
    if (6.11a) *is violated* ***then***
      Set $\Delta\alpha \leftarrow \dfrac{\gamma_1 - \varphi_j(\underline{x}_i) - \alpha_j \underline{x}_i - \beta_j}{\underline{x}_i - \overline{x}_j}$
    if (6.11b) *is violated* ***then***
      $\Delta\alpha \leftarrow \min\left\{\Delta\alpha, \dfrac{\gamma_2 - \varphi_j(\overline{x}_i) - \alpha_j \overline{x}_i - \beta_j}{\overline{x}_i - \overline{x}_j}\right\}$
    if $\Delta\alpha < 0$ ***then***
      perform the updates $\alpha_k \leftarrow \alpha_k + \Delta\alpha$ *and* $\beta_k \leftarrow \beta_k - \Delta\alpha \overline{x}_j$ *for every* $k \in \{1, \dots, j\}$
  for $j = i+1, \dots, s$ ***do***
    set $\Delta\alpha \leftarrow 0$
    if (6.12a) *is violated* ***then***
      Set $\Delta\alpha \leftarrow \dfrac{\varphi_j(\overline{x}_i) + \alpha_j \overline{x}_i + \beta_j - \gamma_3}{\underline{x}_j - \overline{x}_i}$
    if (6.12b) *is violated* ***then***
      $\Delta\alpha \leftarrow \max\left\{\Delta\alpha, \dfrac{\varphi_j(\underline{x}_i) + \alpha_j \underline{x}_i + \beta_j - \gamma_4}{\underline{x}_j - \underline{x}_i}\right\}$
    if $\Delta\alpha > 0$ ***then***
      perform the updates $\alpha_k \leftarrow \alpha_k + \Delta\alpha$ *and* $\beta_k \leftarrow \beta_k - \Delta\alpha \underline{x}_j$ *for every* $k \in \{j, \dots, s\}$
return $\alpha_1, \dots, \alpha_s$ *and* $\beta_1, \dots, \beta_s$

Theorem 6.5. *Algorithm 6.4 provides a feasible solution to the constraints* (6.10)–(6.12).

Proof. For every i and $j \neq i$, we have to satisfy a pair of constraints in (6.11) and (6.12). Obviously, Alg. 1 runs through these constraints iteratively. Let us take a snapshot of the algorithms for some i and $j \neq i$. Clearly, after evaluating the corresponding body of the for-loops, the corresponding pair of constraints holds. If we can prove that all previously addressed constraints still hold as well, we are done.

Assume the constraints have already been satisfied for the pair $(\hat{\imath}, \hat{\jmath})$ and show that the modifications associated with (i, j) will not alter this. Since we have $\hat{\imath} \neq \hat{\jmath}$ and $i \neq j$, we can distinguish the four cases (1) $\hat{\imath} < \hat{\jmath}$ and $i < j$, (2)

$\hat{\imath} < \hat{\jmath}$ and $j < i$, (3) $\hat{\jmath} < \hat{\imath}$ and $i < j$, and (4) $\hat{\jmath} < \hat{\imath}$ and $j < i$. We next prove the second case as it nicely illustrates the involved steps. Since this case offers $\hat{\imath} < \hat{\jmath}$, the conditions

$$\varphi_{\hat{\jmath}}(\overline{x}_{\hat{\imath}}) + \alpha_{\hat{\jmath}}\overline{x}_{\hat{\imath}} + \beta_{\hat{\jmath}} \leq \varphi_{\hat{\imath}}(\overline{x}_{\hat{\imath}}) + \alpha_{\hat{\imath}}\overline{x}_{\hat{\imath}} + \beta_{\hat{\imath}} \tag{6.17a}$$

$$\varphi_{\hat{\jmath}}(\underline{x}_{\hat{\imath}}) + \alpha_{\hat{\jmath}}\underline{x}_{\hat{\imath}} + \beta_{\hat{\jmath}} \leq \varphi_{\hat{\imath}}(\overline{x}_{\hat{\imath}}) + \alpha_{\hat{\imath}}\overline{x}_{\hat{\imath}} + \beta_{\hat{\imath}} - (\overline{x}_{\hat{\imath}} - \underline{x}_{\hat{\imath}})(2q_{\hat{\imath}}\overline{x}_{\hat{\imath}} + l_{\hat{\imath}} + \alpha_{\hat{\imath}}) \tag{6.17b}$$

hold before the modifications. Due to $j < i$, the modifications will affect the parameters $\alpha_1, \ldots, \alpha_j$ and $\beta_1, \ldots, \beta_j$. Clearly, the modified indices may or may not involve $\hat{\imath}$ and $\hat{\jmath}$. Hence, it is reasonable to distinguish the subcases (2.1) $j < \hat{\imath}$, (2.2) $\hat{\imath} \leq j < \hat{\jmath}$, and (2.3) $\hat{\jmath} \leq j$. Taking $\hat{\imath} < \hat{\jmath}$ into account, the first subcase implies that modifications do not affect the conditions (6.17). In the second subcase, we obtain

$$\begin{aligned}\varphi_{\hat{\jmath}}(\overline{x}_{\hat{\imath}}) + \alpha_{\hat{\jmath}}\overline{x}_{\hat{\imath}} + \beta_{\hat{\jmath}} &\leq \varphi_{\hat{\imath}}(\overline{x}_{\hat{\imath}}) + (\alpha_{\hat{\imath}} + \Delta\alpha)\overline{x}_{\hat{\imath}} + \beta_{\hat{\imath}} - \Delta\alpha\overline{x}_j \\ \varphi_{\hat{\jmath}}(\underline{x}_{\hat{\imath}}) + \alpha_{\hat{\jmath}}\underline{x}_{\hat{\imath}} + \beta_{\hat{\jmath}} &\leq \varphi_{\hat{\imath}}(\overline{x}_{\hat{\imath}}) + (\alpha_{\hat{\imath}} + \Delta\alpha)\overline{x}_{\hat{\imath}} + \beta_{\hat{\imath}} - \Delta\alpha\overline{x}_j \\ &\quad (\overline{x}_{\hat{\imath}} - \underline{x}_{\hat{\imath}})(2q_{\hat{\imath}}\overline{x}_{\hat{\imath}} + l_{\hat{\imath}} + \alpha_{\hat{\imath}} + \Delta\alpha).\end{aligned}$$

The conditions remain valid since the modifications only increase the right-hand sides of (6.17). In fact, the novel terms

$$\Delta\alpha\overline{x}_{\hat{\imath}} - \Delta\alpha\overline{x}_j = -\Delta\alpha(\overline{x}_j - \overline{x}_{\hat{\imath}}) \quad \text{and} \tag{6.18a}$$

$$-\Delta\alpha(\overline{x}_{\hat{\imath}} - \underline{x}_{\hat{\imath}}) \tag{6.18b}$$

are non-negative due to the relations $\Delta\alpha < 0$, $\overline{x}_j - \overline{x}_{\hat{\imath}} \geq 0$, and $\overline{x}_{\hat{\imath}} - \underline{x}_{\hat{\imath}} > 0$. Now, in the third subcase, the term (6.18a) also appears on the left-hand sides and, hence, the corresponding terms cancel out. The remaining term (6.18b) is non-negative as before and, consequently, the conditions remain valid.

We will leave the remaining cases for the interested reader. However, we briefly note that, taking the order of iterations in Alg. into account, we find the additional condition $\hat{\imath} \leq i$. As a consequence, case (3) can be specified to $\hat{\jmath} < \hat{\imath} \leq i < j$. Under this specification, it quickly turns out that any modifications associated with this case does not alter the conditions obtained for $(\hat{\imath}, \hat{\jmath})$. Hence, only the first and fourth case remain open (but can be easily handled analogously to the second case). ■

The combination of Theorems 6.2 and 6.5 leads to the following summarizing statement about the structure of a tailored max-out-NN for (6.3) that is in line with Conjecture 6.1.

Corollary 6.6. *Every convex PWQ of the form* (6.3) *can be exactly represented by a max-out-NN with one hidden layer of width* $w_1 = 2$, $p_1 = s$, *and* $\boldsymbol{\xi} := (x \;\; x^2)^\top$.

Proof. A max-out-NN with the specified structure is given by

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{W}^{(2)} \begin{pmatrix} \max\limits_{1 \leq i \leq s} \{\boldsymbol{W}_i^{(1)}\boldsymbol{\xi} + \boldsymbol{b}_i^{(1)}\} \\ \max\limits_{s+1 \leq i \leq 2s} \{\boldsymbol{W}_i^{(1)}\boldsymbol{\xi} + \boldsymbol{b}_i^{(1)}\} \end{pmatrix} + \boldsymbol{b}^{(2)},$$

where $\boldsymbol{W}_i^{(1)}$ denotes the i-th row of $\boldsymbol{W}^{(1)}$. Now, we assume that α_i and β_i have been chosen such (6.10)-(6.12) hold, which is always possible according to Theorem 6.5. Then, specifying the weights and biases as

$$\boldsymbol{W}^{(1)} := \begin{pmatrix} \alpha_1 + l_1 & q_1 \\ \vdots & \vdots \\ \alpha_s + l_s & q_s \\ \alpha_1 & 0 \\ \vdots & \vdots \\ \alpha_1 & 0 \end{pmatrix}, \qquad \boldsymbol{b}^{(1)} := \begin{pmatrix} \beta_1 + c_1 \\ \vdots \\ \beta_s + c_s \\ \beta_1 \\ \vdots \\ \beta_s \end{pmatrix}, \tag{6.19a}$$

$$\boldsymbol{W}^{(2)} := \begin{pmatrix} 1 & -1 \end{pmatrix} \qquad \boldsymbol{b}^{(2)} := 0 \tag{6.19b}$$

results in

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \max_{1 \le i \le s} \{\varphi_i(x) + \alpha_i x + \beta_i\} - \max_{1 \le i \le s} \{\alpha_i x + \beta_i\}. \tag{6.20}$$

This is equivalent to $\varphi(x)$ since the right-hand side in (6.20) evaluates to $(\varphi(x) + h(x)) - h(x) = \varphi(x)$ according to Theorem 6.2 and due to convexity of $h(x)$. ■

Fig. 6.2 shows the structure of the resulting max-out-NN with one neuron for the computation of the PWQ function $\varphi(x) + h(x)$ and one neuron for the computation of the PWA function $h(x)$.

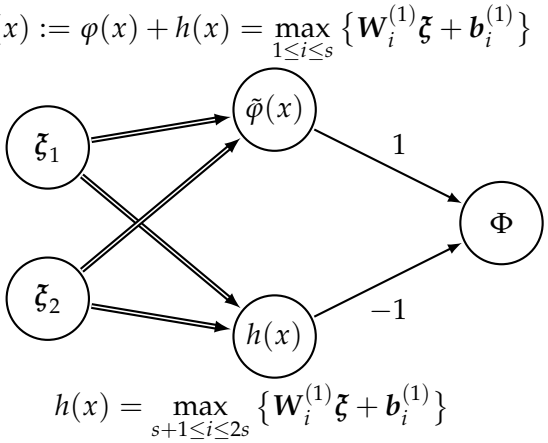


Figure 6.2. Illustration of the max-out-NN according to Corollary 6.6 for an exact representation of the convex PWQ function φ. Double arrows highlight multi-channel preactivations with $p_1 = s$.

6.4 Numerical examples

We illustrate our results with two numerical examples. The first examples demonstrate the application of Theorem 6.2 and Corollary 6.6. The second example shows that the methodology can, in principle, also be applied for $n > 1$.

6.4.1 1D PWQ function

Explicitly solving the optimal control problem (OCP)

$$\begin{aligned} \varphi(x) := \min_{x_0, x^+, u} \; & 5(x^+)^2 + \tfrac{19}{5}x_0^2 + u^2 \\ \text{s.t.} \quad & x_0 = x \\ & x^+ = \tfrac{6}{5}x_0 + u \\ & x_0 \in [-10, 10],\ x^+ \in [-1, 1],\ u \in [-1, 1], \end{aligned}$$

where x is the current system state results in the OVF

$$\varphi(x) = \begin{cases} 11x^2 + 12x + 6 & \text{if} \quad x \in \left[-\frac{5}{3}, -1\right], \\ 5x^2 & \text{if} \quad x \in [-1, 1], \\ 11x^2 - 12x + 6 & \text{if} \quad x \in \left[1, \frac{5}{3}\right], \end{cases} \tag{6.21}$$

where we refer to [108, Ex. 2] for details on the underlying MPC problem and system. Now, Fig. 6.3 shows that we have

$$\varphi(x) \neq \max_{1 \leq i \leq 3} \{\varphi_i(x)\},$$

since the segment $\varphi_2(x)$ is smaller than $\varphi_1(x)$ and $\varphi_3(x)$ for all $x \in (-1, 1)$. Hence, φ does not offer a trivial representation by one max-out neuron.

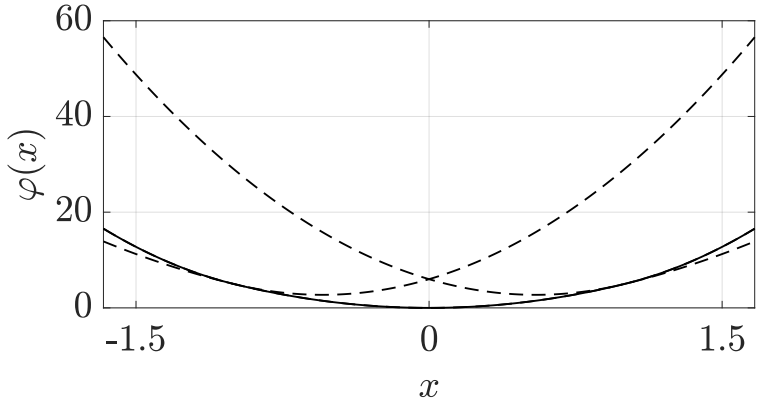


Figure 6.3. Illustration of the PWQ function $\varphi(x)$ in black and the extension of the quadratic segments outside their regions in black dashed.

However, an exact representation in terms of a max-out-NN can be computed according to the proposed procedure in Section 6.3. To this end, we initially apply Algorithm 6.4 in order to compute the parameters

$$\tilde{\alpha}_1 = -\frac{56}{3}, \quad \tilde{\alpha}_2 = \frac{10}{3}, \quad \tilde{\alpha}_3 = \frac{76}{3}, \tag{6.22a}$$

$$\tilde{\beta}_1 = -\frac{56}{3}, \quad \tilde{\beta}_2 = \frac{10}{3}, \quad \tilde{\beta}_3 = -\frac{56}{3} \tag{6.22b}$$

that satisfy the conditions (6.10)–(6.12). Thus we have

$$\begin{aligned}\varphi(x) = &\max\left\{11x^2 - \tfrac{20}{3}x - \tfrac{38}{3}, 5x^2 + \tfrac{10}{3}x + \tfrac{10}{3}, 11x^2 + \tfrac{40}{3}x - \tfrac{38}{3}\right\} \\ &- \max\left\{-\tfrac{56}{3}x - \tfrac{56}{3}, \tfrac{10}{3}x + \tfrac{10}{3}, \tfrac{76}{3}x - \tfrac{56}{3}\right\}. \end{aligned} \tag{6.23}$$

Fig. 6.4(a) shows that in contrast to the function $\varphi(x)$ (Fig. 6.3), the quadratic segments of the function $\varphi(x) + h(x)$ are such that they are greater than all other segments inside their region which allows the representation of (6.23) in the form (6.13) for $\alpha_i = \tilde{\alpha}_i$ and $\beta_i = \tilde{\beta}$.

As mentioned in Remark 6.3 the function h can also be computed as a solution of the OP (6.16). Therefore we consider in this example the cost function

$$J\left(\hat{\alpha}_1, \hat{\beta}_1, \ldots, \hat{\alpha}_s, \hat{\beta}_s\right) = \sum_{i=1}^{s} \hat{\alpha}_i^2 + \hat{\beta}_i^2. \tag{6.24}$$

The parameters

$$\hat{\alpha}_1 = -22, \quad \hat{\alpha}_2 = 0, \quad \hat{\alpha}_3 = 22, \tag{6.25a}$$

$$\hat{\beta}_1 = -\frac{22}{3}, \quad \hat{\beta}_2 = \frac{44}{3}, \quad \hat{\beta}_3 = -\frac{22}{3} \tag{6.25b}$$

solve the OP (6.16). According to Theorem 6.2 the constraints of the OP guarantee (6.13) for $\alpha_i = \hat{\alpha}_i$ and $\beta_i = \hat{\beta}$. Fig. 6.4(b) illustrates that in this case the modification of $\varphi(x)$ by the PWA function results in a symmetric function $\varphi(x) + h(x)$.

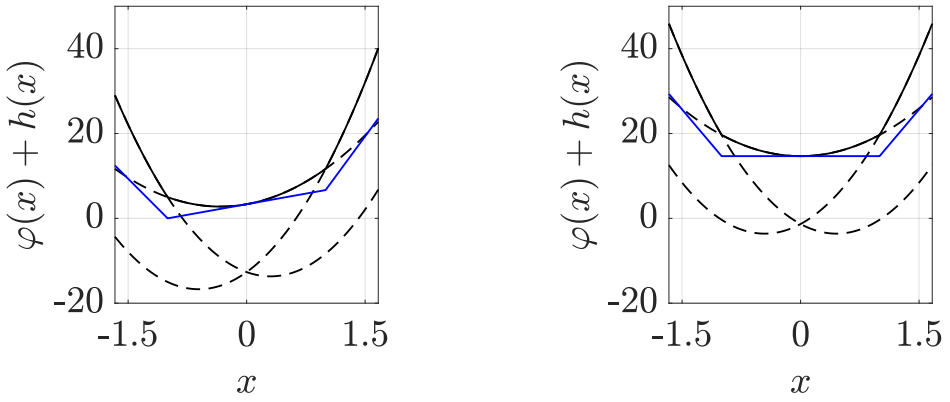


Figure 6.4. Illustration of the PWQ function $\varphi(x) + h(x)$ in black, the extension of the quadratic segments outside their regions in black dashed, and the PWA function $h(x)$ in blue. The function $h(x)$ is illustrated for $\alpha_i = \tilde{\alpha}_i$, $\beta_i = \tilde{\beta}_i$ in (a) and for $\alpha_i = \hat{\alpha}_i$, $\beta_i = \hat{\beta}_i$ in (b) with $1 \leq i \leq 3$.

According to Corollary 6.6 a max-out-NN with one hidden layer of width $w_1 = 2$, $p_1 = s = 3$ and weights and biases as per (6.19) exactly describes the function (6.21). Since both solutions (6.22) and (6.25) are such that (6.11) and (6.12) hold, we can either use $\alpha_i = \tilde{\alpha}_i$ and $\beta_i = \tilde{\beta}_i$ or $\alpha_i = \hat{\alpha}_i$ and $\beta_i = \hat{\beta}_i$ for the weights and biases.

6.4.2 2D PWQ function

We consider the OCP

$$\begin{aligned} \varphi(\boldsymbol{x}) := \min_{x_0, x^+, u} \ & \left\|x^+\right\|_{\boldsymbol{P}}^2 + \|x_0\|_2^2 + u^2 \qquad (6.26) \\ \text{s.t.} \quad & x_0 = \boldsymbol{x}, \\ & x^+ = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_0 + \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} u, \\ & x_0 \in \{\boldsymbol{x} \in \mathbb{R}^2 \mid -10 \leq x_1 \leq 5, \ -2 \leq x_1 \leq 4\}, \\ & x^+ \in \{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{e}\}, \\ & u \in \{u \in \mathbb{R} \mid |u| \leq 1\}, \end{aligned}$$

with

$$\boldsymbol{G} := \begin{pmatrix} 0 & -1 \\ -0.43 & -1.03 \\ 0.43 & 1.03 \\ 0.43 & 0.03 \\ -0.11 & 0.18 \end{pmatrix}, \quad \boldsymbol{e} := \begin{pmatrix} 2 \\ 1 \\ 1 \\ 2 \\ 1 \end{pmatrix},$$

which arises in the context of MPC for a system with double-integrator dynamics, where $\boldsymbol{x}$ is the current system state, the prediction horizon is chosen as $N = 1$ and where a stabilizing terminal set is considered. Explicitly solving (6.26) leads to the OVF shown in Fig. 6.5, which is defined over six polyhedral regions as detailed in Fig. 6.6.

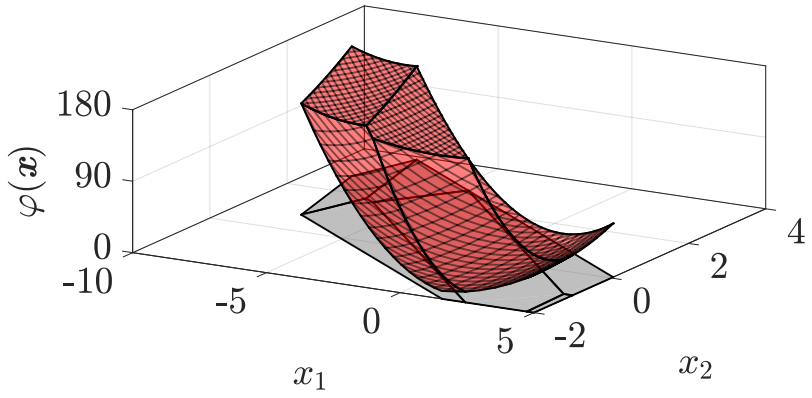


Figure 6.5. Illustration of the convex PWQ function $\varphi(\boldsymbol{x})$ in red and the state space partition with 6 regions in gray.

Analogously to the first example, we also have

$$\varphi(\boldsymbol{x}) \neq \max_{1 \leq i \leq 6} \{\varphi_i(\boldsymbol{x})\}$$

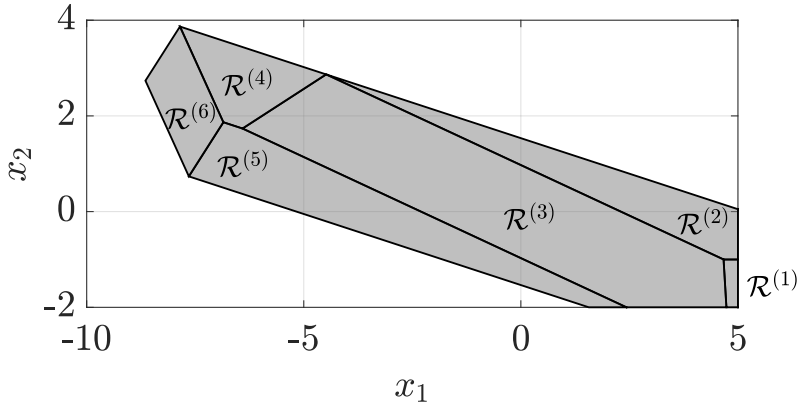


Figure 6.6. Illustration of the partition of the functions $\varphi(\boldsymbol{x})$ and $h(\boldsymbol{x})$.

here. Nevertheless, it is possible to adapt the procedure used for the one-dimensional case by adding the convex PWA function

$$h(\boldsymbol{x}) = \begin{cases} 99.2x_1 - 24.34x_2 - 292.77 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ 99.2x_1 + 245.83x_2 - 22.61 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(2)}, \\ -18.18x_1 - 32.03x_2 + 247.56 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(3)}, \\ -24.21x_1 - 21.76x_2 + 191.07 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(4)}, \\ -48.75x_1 - 104.4x_2 + 177.19 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(5)}, \\ -107.26x_1 - 63.29x_2 - 300.45 & \text{if } \boldsymbol{x} \in \mathcal{R}^{(6)}, \end{cases}$$

defined on the same partition (see Fig 6.6) as the function $\varphi(\boldsymbol{x})$ which is such that the sum $\varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ can be represented in the form

$$\varphi(\boldsymbol{x}) + h(\boldsymbol{x}) = \max_{1 \leq i \leq 6} \{\varphi_i(\boldsymbol{x}) + h_i(\boldsymbol{x})\}. \tag{6.27}$$

With the relation (6.27) we can represent $\varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ in terms of a single max-out neuron. Which immediately allow to represent $\varphi(\boldsymbol{x})$ by a max-out-NN with one hidden layer and two neurons, as $h(\boldsymbol{x})$ itself can be represented by a second neuron (see (6.6)). In contrast to the case of $n = 1$, we need here an augmented input of the form

$$\boldsymbol{\xi} = \begin{pmatrix} x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{pmatrix}^\top,$$

because the max-out neuron for the representation $\varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ involves the evaluation of quadratic segments. Note that the presented exemplary solution for $n = 2$ is in line with Conjecture 6.1.

We briefly comment on the derivation of this solution. For the computation of the function $h(\boldsymbol{x})$ we first compute a convex lifting $H(\boldsymbol{x})$ of the state space partition of the function $\varphi(\boldsymbol{x})$ according to [110, Alg. 3]. Then we choose $h(\boldsymbol{x}) = \gamma H(\boldsymbol{x})$ with $\gamma > 0$. The scaling factor γ is chosen such that we have (6.13). Although this approach seems to work in general the proof that it is always possible to find a γ such that (6.13) holds and extending the conditions (6.11) and (6.12) to $n > 1$ is not straightforward.

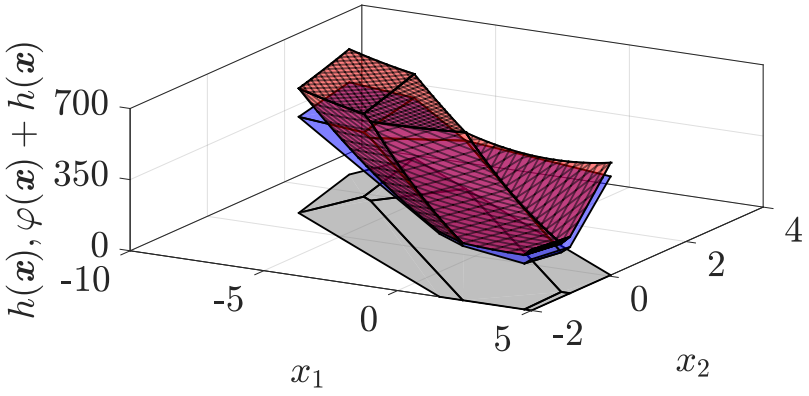


Figure 6.7. Illustration of the lifted function $\varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ in red, the convex PWA function $h(\boldsymbol{x})$ in blue and the state space partition in gray. The sum $\varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ and the function $h(\boldsymbol{x})$ are represented by the first and second neuron of the max-out-NN, respectively.

6.5 Conclusions and outlook

In this paper, we proved that convex PWQ functions with scalar variables, i.e., $x \in \mathbb{R}$ can always be computed as the sum of two terms of the form (6.6) and (6.8), which are the maximum of s affine and s quadratic segments, respectively (see Thm. 6.2 and 6.5). We further showed in Corollary 6.6 how to use this insight to derive a max-out-NN with one hidden layer of width $w_1 = 2$, which exactly represents the PWQ function. This result can be useful for deriving design guidelines for the choice of the topology when learning PWQ functions, as required in Q-learning for linear MPC.

Example 6.4.2 shows that the central idea of adding a convex PWA function $h(\boldsymbol{x})$ to the original PWQ function such that (6.8) holds is extendable to convex PWQ functions with $n > 1$. The formal proof of this approach should involve the extension of the presented Theorems 6.2 and 6.5 which is an interesting direction for future research.

Chapter 7

Error bounds for maxout neural network approximations of model predictive control*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. Neural network (NN) approximations of model predictive control (MPC) are a versatile approach if the online solution of the underlying optimal control problem (OCP) is too demanding and if an exact computation of the explicit MPC law is intractable. The drawback of such approximations is that they typically do not preserve stability and performance guarantees of the original MPC. However, such guarantees can be recovered if the maximum error with respect to the optimal control law and the Lipschitz constant of that error are known. We show in this work how to compute both values exactly when the control law is approximated by a maxout NN. We build upon related results for ReLU NN approximations and derive mixed-integer (MI) linear constraints that allow a computation of the output and the local gain of a maxout NN by solving an MI feasibility problem. Furthermore, we show theoretically and experimentally that maxout NN exist for which the maximum error is zero.

Keywords. Learning for control, Machine Learning, Predictive control, Neural networks.

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

7.1 Introduction

Model predictive control (MPC) (see, e.g, [17]) has become a standard tool for the control of dynamical systems with state and input constraints and has been successfully applied in different industrial fields (see, e.g., [58] for an overview). In the classical setup, MPC requires to solve an optimization problem (OP) in every time step. For systems with a short sampling period as, e.g., in power electronics [111], this can be challenging because the OP may be too complex to be solved within the sampling period. If we consider a linear discrete-time prediction model in combination with a quadratic cost function, the resulting OP is a quadratic program (QP). In principle, we can compute the solution of the parametric QP offline for all feasible states. This results in an explicit control law with a piecewise affine (PWA) input-output relation defined on a polyhedral partition of the state space [20]. Given the explicit control law, the online computational effort reduces to the evaluation of the PWA function. However, the number of polyhedral regions may grow exponentially with the state dimension and the number of constraints in the OP. Hence, exactly computing the explicit MPC law becomes untractable for complex systems. As a consequence, various techniques have been developed to approximate the control law in MPC [112, 113]. In this context, neural networks (NN) are very popular [6, 7, 114] since they can approximate a large class of functions, including PWA functions, with arbitrary accuracy [67]. In addition, besides their computational demanding offline training, NN are typically fast to evaluate online, which is essential if they are used as controllers. Moreover, some types of NN share the PWA structure of the control law [24, 25, 26], making them the perfect choice for approximating MPC. Unfortunately, in general, stability cannot be guaranteed for the approximated controllers. One possibility to recover stability and recursive feasibility is to project the output of the NN onto a suitable set as, e.g., in [6, 95]. Alternatively, the output of the NN can be used as an initial guess for a solver and not directly for control [114]. The drawback of both approaches is that they require an additional optimization-based computation step online. Ideally, the NN can be used as a controller without additional online computation, while still providing stability guarantees. In [16], it is proven that this is possible if the maximum error with respect to the optimal control law and the Lipschitz constant of the corresponding error function are known. Both values can indeed be computed exactly presupposed the output and the local gain of the used NN can be computed by solving a mixed-integer (MI) feasibility problem. This is known to be possible for NN using rectified linear units (ReLU) as activation functions [16, Thm. 6.1].

In the work at hand, we will extend the result of [16] by showing that the maximum error and the Lipschitz constant of the error can also be computed exactly for a controller approximation based on a maxout NN. Since maxout NN include other NN with PWA input-output relation such as, e.g., ReLU and leaky ReLU, as a special case, the results provide a generalization to a

broader class of PWA NN. Furthermore, we use the PWA structure of maxout NN to compute NN that exactly describe MPC control laws and validate experimentally that these exact maxout NN indeed lead to a maximum error and Lipschitz constant of zero.

The paper is organized as follows. In the remainder of this section, we introduce relevant notation. In Section 7.2, we summarize some basics on MPC as well as PWA NN and describe concepts for approximating MPC in more depth. Section 7.3 is devoted to our main result, i.e., the computation of the maximum error and the Lipschitz constant of the error function related to maxout NN approximating MPC. The obtained method is applied to various maxout NN approximations of MPC laws in Section 7.4. Finally, conclusions and an outlook are given in Section 7.5.

7.1.1 Notation

We will denote the index set containing $p_i \in \mathbb{N}$ integers starting at $p_i(l-1)+1$, $l \in \mathbb{N}$ by

$$\mathcal{A}_l^{(i)} := \{p_i(l-1)+1, \dots, p_i l\}.$$

For vectors $\boldsymbol{x} \in \mathbb{R}^n$ we denote the i-th element by $\boldsymbol{x}_i$ and the elements between the indices a_1 and $a_2 > a_1$ by $\boldsymbol{x}_{a_1:a_2}$. For matrices $\boldsymbol{K} \in \mathbb{R}^{m \times n}$ we denote the element in the i-th row and j-th column by $\boldsymbol{K}_{i,j}$, the i-row and j-th column by $\boldsymbol{K}_{i,:}$ and $\boldsymbol{K}_{:,j}$, respectively. If we only write $\boldsymbol{K}_i$ then we refer to the i-th row. A block diagonal matrix is defined as

$$\operatorname{diag}(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_N) := \begin{pmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & & \vdots \\ \vdots & & & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{\alpha}_{w_i} \end{pmatrix},$$

with $\boldsymbol{\alpha} \in \mathbb{R}^{w_i \times p_i}$. A continuous function $\boldsymbol{F}(\boldsymbol{x}) : \mathcal{P} \subset \mathbb{R}^n \to \mathbb{R}^m$ of the form

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{cases} \boldsymbol{G}^{(1)}\boldsymbol{x} + \boldsymbol{g}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{P}^{(1)}, \\ \vdots & \vdots \\ \boldsymbol{G}^{(s)}\boldsymbol{x} + \boldsymbol{g}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{P}^{(r)}, \end{cases} \tag{7.1}$$

with a polyhedral partition $\mathcal{P} = \cup_{i=1}^r \mathcal{P}^{(i)}$ and $\operatorname{int}(\mathcal{P}^{(i)}) \cap \operatorname{int}(\mathcal{P}^{(j)}) = \emptyset \; \forall i \neq j$ is denoted as piecewise affine (PWA) function. We further define the local gain $\boldsymbol{K}(\boldsymbol{x}) : \cup_{i=1}^r \operatorname{int}(\mathcal{P}^{(i)}) \to \mathbb{R}^m$ of a PWA function as

$$\boldsymbol{K}(\boldsymbol{x}) := \begin{cases} \boldsymbol{G}^{(1)} & \text{if } \boldsymbol{x} \in \operatorname{int}(\mathcal{P}^{(1)}), \\ \vdots & \vdots \\ \boldsymbol{G}^{(s)} & \text{if } \boldsymbol{x} \in \operatorname{int}(\mathcal{P}^{(r)}). \end{cases} \tag{7.2}$$

7.2 Fundamentals of MPC and NN

7.2.1 Model predicitive control

Model predictive control (MPC) for linear discrete-time systems builds on solving an optimal control problem (OCP) of the form

$$
\begin{aligned}
V_N(\boldsymbol{x}) := \min_{\substack{\hat{\boldsymbol{x}}(0),\dots,\hat{\boldsymbol{x}}(N)\\ \hat{\boldsymbol{u}}(0),\dots,\hat{\boldsymbol{u}}(N-1)}} \; & \varphi(\hat{\boldsymbol{x}}(N)) + \sum_{\kappa=0}^{N-1} \ell(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)) \\
\text{s.t.} \quad & \hat{\boldsymbol{x}}(0) = \boldsymbol{x}, \\
& \hat{\boldsymbol{x}}(\kappa+1) = \boldsymbol{A}\,\hat{\boldsymbol{x}}(\kappa) + \boldsymbol{B}\hat{\boldsymbol{u}}(\kappa), \quad \forall \kappa \in \{0,\dots,N-1\}, \\
& (\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)) \in \mathcal{X} \times \mathcal{U}, \qquad \forall \kappa \in \{0,\dots,N-1\}, \\
& \hat{\boldsymbol{x}}(N) \in \mathcal{T}
\end{aligned}
\tag{7.3}
$$

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

$$
\varphi(\boldsymbol{x}) := \boldsymbol{x}^\top \boldsymbol{P} \boldsymbol{x} \quad \text{and} \quad \ell(\boldsymbol{x}, \boldsymbol{u}) := \boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u} \tag{7.4}
$$

denote the terminal and stage cost, respectively, where the weighting matrices $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are positive (semi-) definite. The dynamics of the linear prediction model are described by $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times m}$. State and input constraints can be incorporated via the polyhedral sets $\mathcal{X}$ and $\mathcal{U}$. Finally, the terminal set $\mathcal{T}$ allows to enforce closed-loop stability (see [48] for details). The resulting control law $\boldsymbol{\pi} : \mathcal{F}_N \to \mathcal{U}$ is defined as

$$
\boldsymbol{\pi}(\boldsymbol{x}) := \hat{\boldsymbol{u}}^*(0), \tag{7.5}
$$

where $\mathcal{F}_N$ denotes the feasible set of (7.3) and where $\hat{\boldsymbol{u}}^*(0)$ refers to the first element of the optimal input sequence. For the considered setup it is well known that $\boldsymbol{\pi}(\boldsymbol{x})$ is a PWA function [20, Thm. 4] of the form (7.1) with $\boldsymbol{G}^{(i)} = \boldsymbol{K}^{(i)}$, $\boldsymbol{g}^{(i)} = \boldsymbol{b}^{(i)}$, $r = r_{\text{MPC}}$, polyhedral sets $\mathcal{P}^{(i)} = \mathcal{R}^{(i)}$ $\forall i \in \{1,\dots,r_{\text{MPC}}\}$ and local gain $\boldsymbol{K}_{\text{MPC}}(\boldsymbol{x})$.

7.2.2 Neural networks with piecewise affine activations

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

$$
\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{f}^{(\ell+1)} \circ \boldsymbol{g}^{(\ell)} \circ \boldsymbol{f}^{(\ell)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{x}). \tag{7.6}
$$

Here, the functions $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{p_i w_i}$ for $i \in \{1,\dots,\ell\}$ refer to preactivations, where the parameter $p_i \in \mathbb{N}$ allows to consider "multi-channel" preactivations as required for maxout (see [21]). Moreover, $\boldsymbol{g}^{(i)} : \mathbb{R}^{p_i w_i} \to \mathbb{R}^{w_i}$ stand

for activation functions and $\boldsymbol{f}^{(\ell+1)} : \mathbb{R}^{w_\ell} \to \mathbb{R}^{w_{\ell+1}}$ reflects postactivation. The functions $\boldsymbol{f}^{(i)}$ are typically affine, i.e.,

$$\boldsymbol{f}^{(i)}(\boldsymbol{y}^{(i-1)}) = \boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}, \tag{7.7}$$

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i w_i \times w_{i-1}}$ is a weighting matrix, $\boldsymbol{b}^{(i)} \in \mathbb{R}^{p_i w_i}$ is a bias vector, and $\boldsymbol{y}^{(i-1)}$ denotes the output of the previous layer with $\boldsymbol{y}^{(0)} := \boldsymbol{x} \in \mathbb{R}^n$.

Now, various activation functions have been proposed. As already stated in the introduction, we here focus on PWA activation functions, i.e., we consider the ReLU activation function

$$\boldsymbol{g}^{(i)}_{\text{ReLU}}(\boldsymbol{z}^{(i)}) = \max\left\{\boldsymbol{0}, \boldsymbol{z}^{(i)}\right\} := \begin{pmatrix} \max\left\{0, z_1^{(i)}\right\} \\ \vdots \\ \max\left\{0, z_{w_i}^{(i)}\right\} \end{pmatrix} \tag{7.8}$$

and the maxout activation function

$$\boldsymbol{g}^{(i)}_{\max}(\boldsymbol{z}^{(i)}) = \begin{pmatrix} \max\limits_{1\le j\le p_i} \left\{z_j^{(i)}\right\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1\le j\le p_i w_i} \left\{z_j^{(i)}\right\} \end{pmatrix}, \tag{7.9}$$

where we use the shorthand notation

$$\max_{1\le j\le p_i} \left\{z_j^{(i)}\right\} := \max\left\{z_1^{(i)}, \dots, z_{p_i}^{(i)}\right\}.$$

We will refer to the resulting NN as ReLU NN and maxout NN, respectively. The proof of [21, Thm. 4.3] shows that maxout NN are PWA functions of the form (7.1) with $\boldsymbol{G}^{(i)} = \boldsymbol{K}^{(i)}_{\text{NN}}$, $\boldsymbol{g}^{(i)} = \boldsymbol{b}^{(i)}_{\text{NN}}$, $r = r_{\text{NN}}$, polyhedral sets $\mathcal{P}^{(i)} = \mathcal{R}^{(i)}_{\text{NN}}$ $\forall i \in \{1, \dots, r_{\text{NN}}\}$ and local gain $\boldsymbol{K}_{\text{NN}}(\boldsymbol{x})$. The number of parameters needed to describe a NN is

$$\#_p := \sum_{i=1}^{\ell} (w_{i-1}+1)p_i w_i + (w_\ell + 1)w_{\ell+1},$$

where ℓ, p_i with $i \in \{1, \dots, \ell\}$ and w_i with $i \in \{1, \dots, \ell+1\}$ describe the topology of the NN.

7.2.3 Approximate MPC

The use of approximate MPC is particularly useful, if the OCP (7.3) is too complex to be solved online and the explicit solution has too many regions and thus a large memory footprint ([115]). In this case, the exact control law may be approximated by a function that is fast to evaluate and has a small memory footprint. A promising candidate for such a function is an NN, as it

combines both required properties and is a universal function approximator [67, Thm. 2.4]. In addition, due to the common PWA structure of some NN [24, Thm. 2], [25, Thm. 2.1] and the control law [20, Thm. 4], they seem to be a natural choice. In fact, for a suitable choice of the weighting matrices and bias vectors, ReLU NN can represent the control law exactly [26, Thm. 1], [7, Thm. 1]. Unfortunately, despite the ability of NN to exactly represent the control law, the approximated version of the control law typically does not preserve desirable properties of the MPC, such as stability and performance. Crucial for preserving these properties is the error function

$$\boldsymbol{e}(\boldsymbol{x}) := \boldsymbol{\pi}(\boldsymbol{x}) - \boldsymbol{\Phi}(\boldsymbol{x}). \tag{7.10}$$

More precisely we have to compute the maximum error

$$\bar{e}_\alpha := \max_{x \in \mathcal{X}} \|\boldsymbol{e}(\boldsymbol{x})\|_\alpha \tag{7.11}$$

and the α-Lipschitz constant of the error

$$\mathcal{L}_\alpha(\boldsymbol{e}, \mathcal{X}) := \sup_{\boldsymbol{x} \neq \boldsymbol{y} \in \mathcal{X}} \frac{\|\boldsymbol{e}(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{y})\|_\alpha}{\|\boldsymbol{x} - \boldsymbol{y}\|_\alpha}.$$

According to [63, Prop. 3.4], the α-Lipschitz constant of a PWA function is equal to the maximum α-norm of the local gain. If we consider a PWA NN, the error is as difference of two PWA functions also PWA [63, Prop. 1.1] and the α-Lipschitz constant is thus

$$\mathcal{L}_\alpha(\boldsymbol{e}, \mathcal{X}) = \max_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{K}_{\text{MPC}}(\boldsymbol{x}) - \boldsymbol{K}_{\text{NN}}(\boldsymbol{x})\|_\alpha . \tag{7.12}$$

Now, if e_α and $\mathcal{L}_\alpha(\boldsymbol{e}, \mathcal{T})$ are below certain values, specified in [16, Eq. (24)–(25)], then the closed-loop system with the approximated NN controller converges exponentially to the origin according to [16, Thm. 3.4]. The problem at this point is that during the training of a NN, the error (7.10) is only evaluated at discrete samples $(\boldsymbol{x}_i^\top \ \boldsymbol{\pi}(\boldsymbol{x}_i)^\top)$ of the control law, and the parameters of the NN are chosen such that the mean squared error (MSE)

$$\hat{e}^2 := \frac{1}{D} \sum_{i=1}^{D} \|\boldsymbol{\pi}(\boldsymbol{x}_i) - \boldsymbol{\Phi}(\boldsymbol{x}_i)\|_2^2 \tag{7.13}$$

over the $D \in \mathbb{N}$ training samples is minimized. We thus do not have any guarantees for the error at points $\boldsymbol{x} \in \mathcal{F}_N$ not included in the training samples. Moreover, a low MSE does not necessarily mean that the values of (7.11) and (7.12) are low. Therefore, to certify stability of the closed-loop system with a pre-trained NN we need a way to compute these values exactly. For $\alpha = \{1, \infty\}$ this is possible by solving a mixed-integer linear program (MILP) [16, Thm. 6.1] if both the output $\boldsymbol{\Phi}(\boldsymbol{x})$ and the local gain $\boldsymbol{K}_{\text{NN}}(\boldsymbol{x})$ of the NN can be computed by solving an MI feasibility problem. Which is proven for ReLU NN in [16, Thm. 6.1]. In the remainder of this paper, we will extend the results to maxout NN.

7.3 Maxout neural networks for approximate MPC

In most of the recent work where the control law (7.5) is approximated by an NN, ReLU NN are used as function approximators (see, e.g., [7, 114, 116]). Maxout NN are rarely considered in this context, although they offer a number of advantages. First, the fact that ReLU NN can represent every PWA function exactly is often used as justification for their use to approximate the PWA control law. However, most ReLU NN that allow an exact description are based on the representation of PWA functions as a sum of the type

$$\hat{F}(\boldsymbol{x}) = \sum_{i=1}^{M} \sigma_i \max_{1 \leq j \leq J} \{\boldsymbol{\beta}_j^{(i)} \boldsymbol{x} + \gamma_j^{(i)}\}. \tag{7.14}$$

From [117] and [70] it is known for which M and J we can find parameters σ_i, $\boldsymbol{\beta}_j$ and γ_j such that $\hat{F}(\boldsymbol{x}) = F(\boldsymbol{x})$ holds for $m = 1$. Since (7.14) is a maxout NN with $\ell = 1$, $w_1 = M$, $p_1 = J$, $w_2 = 1$, these results can be used directly to find suitable topologies for maxout NN that allow an exact description of the control law. For ReLU NN these results are not directly applicable. Therefore, (7.14) is decomposed in, e.g., [24, 25] to find a ReLU topology that allows an exact description. Such a topology is used in, e.g., [7, Thm. 1] to represent the control law. The decomposition step typically leads to a more conservative ReLU topology in terms of number of layers ℓ and neurons per layer w_i compared to a maxout NN that directly represents (7.14). Another advantage of the maxout activation is that it trivially include the ReLU activation as a special case. In fact, a ReLU activation is a maxout activation with $p_i = 2$ where every second affine segment is set to zero (cf. (7.8) and (7.9)). Thus an approach that allow the computation of (7.11) and (7.12) for maxout NN is also applicable to ReLU NN and hence extends the known results.

7.3.1 Exact maxout neural networks

Corollary 7.1. *Let $F(\boldsymbol{x})$ be an arbitrary PWA function of the form (7.1) with one dimensional output, i.e, $m = 1$. Then for a maxout NN $\Phi(\boldsymbol{x})$ with $\ell = 1$, $w_2 = 1$ and $\boldsymbol{b}^{(2)} = 0$ there exist parameters*

(i) $p_1 \in \mathbb{N}$, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{2p_1 \times n}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{2p_1 \times 1}$ *and* $\boldsymbol{W}^{(2)} \in \mathbb{R}^{1 \times 2}$

and

(ii) $w_1 \in \mathbb{N}$, $\boldsymbol{W}^{(1)} \in \mathbb{R}^{w_1(n+1) \times n}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{w_1(n+1) \times 1}$ *and* $\boldsymbol{W}^{(2)} \in \mathbb{R}^{1 \times w_1}$

with $\Phi(\boldsymbol{x}) = F(\boldsymbol{x})$.

Proof. Since the resulting maxout NN are of the form (7.14) with $J = p_1$, $M = 2$ for (i) and $J = n + 1$, $M = w_1$ for (ii), the proof follows from [70, Lem. 1] and [117, Thm. 1], respectively. ■

Corollary 7.1 provides two different topologies (i) and (ii) for maxout NN with one hidden layer that can represent every PWA function with one dimensional output exactly, if p_1 and w_1, respectively are chosen large enough. Although the results are formulated for $m = 1$ they can also be applied to PWA functions with $m > 1$ by applying Corollary 7.1 to every dimension of the output individually. Thus maxout NN can represent the PWA control law (7.5) for arbitrary state and input dimension.

7.3.2 Maxout neural networks as MILP

In this section, we will derive our main result, which is the computation of the output and the local gain of NN with maxout activation by solving an MI feasibility problem. This result then allows to compute (7.11) and (7.12) for the case where the control law (7.5) is approximated by a maxout NN. Both values are according to the descriptions in Section 7.2.3 sufficient to prove stability of the closed-loop system. Moreover, these values provide a more profound way to evaluate the success of the training than by just considering the error at the training samples (7.13) as in, e.g., [7, P1]. We will derive our results based on the observation that the output of a maxout NN with ℓ hidden layers can be modeled by the recursion

$$\begin{aligned} \boldsymbol{y}^{(0)} &= \boldsymbol{x}, \\ \boldsymbol{y}^{(i)} &= \Delta^{(i)}(\boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}),\ 1 \leq i \leq \ell, \\ \boldsymbol{\Phi}(\boldsymbol{x}) &= \boldsymbol{W}^{(\ell+1)}\boldsymbol{y}^{(\ell)} + \boldsymbol{b}^{(\ell+1)} \end{aligned} \tag{7.15}$$

where $\Delta^{(i)}$ is a block diagonal matrix of the form

$$\Delta^{(i)} := \operatorname{diag}(\boldsymbol{\delta}^{(i)}_{1:p_i}, \ldots, \boldsymbol{\delta}^{(i)}_{p_i(w_i-1)+1:p_i w_i}) \tag{7.16}$$

with binary variables $\boldsymbol{\delta}^{(i)} \in \mathbb{B}^{1 \times p_i w_i}$. The matrix $\Delta^{(i)}$ is such that for all elements $\boldsymbol{\delta}^{(i)}_j$ the logical implication

$$\begin{aligned} &[\boldsymbol{\delta}^{(i)}_{k_s} = 1] \Longleftrightarrow [\boldsymbol{W}^{(i)}_j \boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}_j \leq \boldsymbol{W}^{(i)}_{k_s} \boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}_{k_s}, \forall j \in \mathcal{A}^{(i)}_s \setminus k_s,\ k_s \in \mathcal{A}^{(i)}_s], \\ &\forall s \in \{1, \ldots, w_i\},\ \forall i \in \{1, \ldots, \ell\}. \end{aligned} \tag{7.17}$$

holds. Thus, for every neuron in layer i the matrix $\Delta^{(i)}$ selects the largest affine segment among all p_i segments. With the results from [56, Sec. 2] we can model

the logical implication (7.17) by the following MI linear constraints

$$\begin{aligned}
q_s^{(i)} &\leq W_j^{(i)} q^{(i-1)} + b_j^{(i)} + \overline{b}^{(i)}(1 - \delta_j^{(i)}),\\
-q_s^{(i)} &\leq -W_j^{(i)} q^{(i-1)} - b_j^{(i)} - \varepsilon(1 - \delta_j^{(i)}),\\
q^{(0)} &= x,\\
\sum_{j \in \mathcal{A}_s^{(i)}} \delta_j^{(i)} &= 1,\\
&\forall j \in \mathcal{A}_s^{(i)},\ \forall s \in \{1, \dots, w_i\},\ \forall i \in \{1, \dots, \ell\},
\end{aligned} \tag{7.18}$$

with a constant upper bound $\overline{b}^{(i)} \in \mathbb{R}$ and a small $\varepsilon \geq 0$. The variables $q^{(i)} \in \mathbb{R}^{w_i}$ and $\delta^{(i)}$ are real and binary optimization variables, respectively.

Lemma 7.2. *Let $q^{(i)}$ and $\delta^{(i)}$ be such that the constraints (7.18) with $\varepsilon = 0$ hold. Then, the output of the maxout NN (7.6) is given by*

$$\Phi(x) = W^{(\ell+1)} q^{(\ell)} + b^{(\ell+1)}. \tag{7.19}$$

Proof. If we can show that

$$q^{(i)} = y^{(i)} \tag{7.20}$$

holds for all $i \in \{0, \dots, \ell\}$ then (7.19) holds according to (7.6) and (7.7). We prove this by induction. The base case $i = 0$ is true by assumption since we have $q^{(0)} = x = y^{(0)}$. Moreover, the constraints (7.18) are such that for every index set $\mathcal{A}_s^{(i)}$ there exists exactly one $k_s \in \mathcal{A}_s^{(i)}$ with $\delta_{k_s}^{(i)} = 1$ and $\delta_j^{(i)} = 0, \forall j \in \mathcal{A}_s^{(i)} \setminus k_s$. Thus if we assume that the induction hypothesis (7.20) is true for one $i = t$ we obtain for $i = t + 1$ the constraints

$$\begin{aligned}
q_s^{(t+1)} &= W_{k_s}^{(t+1)} y^{(t)} + b_{k_s}^{(t+1)},\ k_s \in \mathcal{A}_s^{(t+1)},\\
q_s^{(t+1)} &\leq W_j^{(t+1)} y^{(t)} + b_j^{(t+1)} + \overline{b}^{(t+1)},\ \forall j \in \mathcal{A}_s^{(t+1)} \setminus k_s,\\
q_s^{(t+1)} &\geq W_j^{(t+1)} y^{(t)} + b_j^{(t+1)},\ \forall j \in \mathcal{A}_s^{(t+1)} \setminus k_s,\\
&\forall s \in \{1, \dots, w_{t+1}\}.
\end{aligned}$$

The first two constraints always hold for a large $\overline{b}^{(i)}$. Hence the constraints (7.18) imply

$$\begin{aligned}
&q_s^{(t+1)} = W_{k_s}^{(t+1)} y^{(t)} + b_{k_s}^{(t+1)} \geq W_j^{(t+1)} y^{(t)} + b_j^{(t+1)},\\
&\forall j \in \mathcal{A}_s^{(t+1)} \setminus k_s,\ \forall s \in \{1, \dots, w_{t+1}\}.
\end{aligned}$$

Which is exactly the relation (7.17) for $i = t + 1$, i.e., the largest affine segment

of the s-th neuron is the k_s-th segment. Thus the relation

$$\begin{pmatrix} \max\limits_{j\in\mathcal{A}_1^{(t+1)}} \{\boldsymbol{W}_j^{(t+1)}\boldsymbol{y}^{(t)}+\boldsymbol{b}_j^{(t+1)}\} \\ \vdots \\ \max\limits_{j\in\mathcal{A}_{w_{t+1}}^{(t+1)}} \{\boldsymbol{W}_j^{(t+1)}\boldsymbol{y}^{(t)}+\boldsymbol{b}_j^{(t+1)}\} \end{pmatrix} = \begin{pmatrix} \boldsymbol{W}_{k_1}^{(t+1)}\boldsymbol{y}^{(t)}+\boldsymbol{b}_{k_1}^{(t+1)} \\ \vdots \\ \boldsymbol{W}_{k_{w_{t+1}}}^{(t+1)}\boldsymbol{y}^{(t)}+\boldsymbol{b}_{k_{w_{t+1}}}^{(t+1)} \end{pmatrix}$$
$$\iff \boldsymbol{y}^{(t+1)} = \boldsymbol{q}^{(t+1)}$$

hold. This proves that (7.20) holds for all $i \in \{0,\dots,\ell\}$. With

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{f}^{(\ell+1)}(\boldsymbol{y}^{(\ell)}) = \boldsymbol{f}^{(\ell+1)}(\boldsymbol{q}^{(\ell)})$$

the output of the maxout NN can be computed via (7.19). ■

Next, we will derive MI linear constraints for the computation of the local gain of a maxout NN. Applying the chain rule for derivative to the recursive formula (7.15) leads to $\boldsymbol{K}_{\mathrm{NN}} : \mathcal{G} \to \mathbb{R}^m$

$$\boldsymbol{K}_{\mathrm{NN}}(\boldsymbol{x}) := \nabla\boldsymbol{\Phi}(\boldsymbol{x})^\top = \boldsymbol{W}^{(\ell+1)} \prod_{i=1}^{\ell} \Delta^{(i)} \boldsymbol{W}^{(i)} \tag{7.21}$$

as expression for the local gain. The gradient is well-defined everywhere except on the boundaries between two neighboring affine segments of the PWA function represented by the maxout NN. These boundaries are given by

$$\begin{aligned} \mathcal{B} :=& \bigcup_{i=1}^{\ell} \bigcup_{l=1}^{w_i} \mathcal{B}_l^{(i)} \text{ with} \\ \mathcal{B}_l^{(i)} :=& \{\boldsymbol{x}\in\mathbb{R}^n \mid \exists k \in \mathcal{A}_l^{(i)} \exists \tilde{k} \in \mathcal{A}_l^{(i)} \setminus k, \tilde{\delta}_k^{(i)} = 1, \\ & \boldsymbol{W}_k^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_k^{(i)} = \boldsymbol{W}_{\tilde{k}}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_{\tilde{k}}^{(i)}\}, \end{aligned} \tag{7.22}$$

where $\tilde{\delta}_k^{(i)}$ is subject to the constraints (7.18) with $\delta^{(i)} = \tilde{\delta}^{(i)}$ and $\varepsilon = 0$. We denote the set where the gradient is well-defined by

$$\mathcal{G} := \mathbb{R}^n \setminus \mathcal{B}.$$

Since the matrix $\Delta^{(i)}$ selects the active, i.e., the largest, affine segment of the maxout neurons, we can model the computation of the local gain $\boldsymbol{K}_{\mathrm{NN}}(\boldsymbol{x})$ by the additional MI linear constraints

$$\begin{aligned} & \underline{w}_{h,r}^{(i)}\delta_h^{(i)} \le \tilde{\xi}_{h,r}^{(i)} \le \overline{w}_{h,r}^{(i)}\delta_h^{(i)}, \\ & -\overline{w}_{h,r}^{(i)}(1-\delta_h^{(i)}) \le \tilde{\xi}_{h,r}^{(i)} - \tilde{W}_{h,r}^{(i)} \le -\underline{w}_{h,r}^{(i)}(1-\delta_h^{(i)}), \\ & \xi_{1,r}^{(i)} = \sum_{\tilde{h}=1}^{p_i} \tilde{\xi}_{\tilde{h},r}^{(i)}, \ \dots, \ \xi_{w_i,r}^{(i)} = \sum_{\tilde{h}=p_i(w_i-1)+1}^{p_i w_i} \tilde{\xi}_{\tilde{h},r}^{(i)}, \\ & \tilde{\boldsymbol{W}}^{(i)} = \boldsymbol{W}^{(i)}\boldsymbol{\xi}^{(i-1)}, \quad \boldsymbol{\xi}^{(0)} = \boldsymbol{I} \\ & \forall h \in \{1,\dots,p_i w_i\}, \forall r \in \{1,\dots,n\}, \forall i \in \{1,\dots,\ell\}, \end{aligned} \tag{7.23}$$

with constant lower and upper bounds $\underline{w}_{h,r}^{(i)}$ and $\overline{w}_{h,r}^{(i)}$, respectively. The MI constraints (7.23) are similar to those used in [16, Eq. 20], except that here we need additional auxiliary variables to account for the p_i different affine segments in each maxout neuron.

Lemma 7.3. *Let $\boldsymbol{q}^{(i)}$, $\boldsymbol{\delta}^{(i)}$, $\tilde{\boldsymbol{W}}^{(i)}$ and $\tilde{\boldsymbol{\xi}}^{(i)}$ be such that the constraints* (7.18) *and* (7.23) *with $\varepsilon > 0$ hold. Then, the local gain of a maxout NN is given by*

$$\boldsymbol{K}_{NN}(\boldsymbol{x}) = \boldsymbol{W}^{(\ell+1)}\boldsymbol{\xi}^{(\ell)}. \tag{7.24}$$

Proof. We first proof that (7.24) is well-defined for $\boldsymbol{x} \in \mathcal{G}$. Since we know that (7.20) holds for all $i \in \{0, \dots, \ell\}$, we can use the same argumentation as in the proof of Lemma 7.2 to rewrite the constraints as follows

$$\begin{aligned}
&\boldsymbol{y}_s^{(i)} = \boldsymbol{W}_{k_s}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_{k_s}^{(i)},\ k_s \in \mathcal{A}_s^{(i)},\\
&\boldsymbol{y}_s^{(i)} \leq \boldsymbol{W}_{j}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_{j}^{(i)} + \overline{b}^{(i)},\ \forall j \in \mathcal{A}_s^{(i)} \setminus k_s,\\
&\boldsymbol{y}_s^{(i)} \geq \boldsymbol{W}_{j}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_{j}^{(i)} + \varepsilon,\ \forall j \in \mathcal{A}_s^{(i)} \setminus k_s,\\
&\quad \forall s \in \{1, \dots, w_i\},\ \forall i \in \{1, \dots, \ell\}.
\end{aligned} \tag{7.25}$$

Again, the first two constraints always hold, if $\overline{b}^{(i)}$ is chosen large enough. Thus the feasibility only depends on the last constraint

$$\boldsymbol{W}_{k_s}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_{k_s}^{(i)} - \boldsymbol{W}_{j}^{(i)}\boldsymbol{y}^{(i-1)} - \boldsymbol{b}_{j}^{(i)} \geq \varepsilon,\ \forall j \in \mathcal{A}_s^{(i)} \setminus k_s,$$

with s and i as in (7.25). Moreover, we can assume without loss of generality that the left-hand side is positive, because if this is not the case for some j, the role of that j can be changed with the role of k_s. Hence the constraints are infeasible if and only if

$$\exists j \in \mathcal{A}_s^{(i)} \setminus k_s : \left| (\boldsymbol{W}_{k_s}^{(i)} - \boldsymbol{W}_{j}^{(i)})\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_{k_s}^{(i)} - \boldsymbol{b}_{j}^{(i)} \right| < \varepsilon \tag{7.26}$$

holds. If we define the set

$$\begin{aligned}
\overline{\mathcal{B}} &:= \bigcup_{i=1}^{\ell} \bigcup_{l=1}^{w_i} \overline{\mathcal{B}}_l^{(i)} \text{ with}\\
\overline{\mathcal{B}}_l^{(i)} &:= \{\boldsymbol{x} \in \mathbb{R}^n \mid \exists k \in \mathcal{A}_l^{(i)} \exists \tilde{k} \in \mathcal{A}_l^{(i)} \setminus k, \tilde{\boldsymbol{\delta}}_k^{(i)} = 1,\\
&\qquad |(\boldsymbol{W}_k^{(i)} - \boldsymbol{W}_{\tilde{k}}^{(i)})\boldsymbol{y}^{(i-1)} + \boldsymbol{b}_k^{(i)} - \boldsymbol{b}_{\tilde{k}}^{(i)}| < \varepsilon\},
\end{aligned}$$

we have $\boldsymbol{x} \in \mathcal{B} \subset \overline{\mathcal{B}} \Rightarrow \boldsymbol{x} \in \overline{\mathcal{B}} \Leftrightarrow$ (7.26), i.e., the constraints are feasible for $\boldsymbol{x} \in \mathbb{R}^n \setminus \overline{\mathcal{B}}$. For small ε we have $\overline{\mathcal{B}} \approx \mathcal{B}$. Thus the domain of (7.21) is approximately the feasible set of (7.18). In addition, the solution to (7.18) is unique. We prove this by contradiction and assume that there exist multiple solutions and thus there exists a $\tilde{k}_s$ with

$$\boldsymbol{\delta}_{\tilde{k}_s}^{(i)} = 1, \quad \tilde{k}_s \in \mathcal{A}_s^{(i)} \setminus k_s.$$

This would imply

$$\boldsymbol{W}_{\tilde{k}_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{\tilde{k}_s}^{(i)}\geq\boldsymbol{W}_{k_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{k_s}^{(i)}+\varepsilon.$$

Further, since for feasible $\boldsymbol{x}$ the inequality

$$\boldsymbol{W}_{k_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{k_s}^{(i)}-\varepsilon\geq\boldsymbol{W}_{\tilde{k}_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{\tilde{k}_s}^{(i)}$$

holds, we obtain the contradiction

$$\boldsymbol{W}_{k_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{k_s}^{(i)}+\varepsilon\leq\boldsymbol{W}_{\tilde{k}_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{\tilde{k}_s}^{(i)}\leq\boldsymbol{W}_{k_s}^{(i)}\boldsymbol{y}^{(i-1)}+\boldsymbol{b}_{k_s}^{(i)}-\varepsilon.$$

Thus the solution exists and is unique for $\boldsymbol{x}\in\mathbb{R}^n\setminus\overline{\mathcal{B}}$. Next, we prove that (7.24) holds for feasible $\boldsymbol{x}$, starting with the case $\delta_h^{(i)}=0$, where we have

$$\tilde{\boldsymbol{\xi}}_{h,r}^{(i)}=0\text{ and }\underline{w}_{h,r}^{(i)}\leq\tilde{\boldsymbol{W}}_{h,r}^{(i)}\leq\overline{w}_{h,r}^{(i)}$$

and for $\delta_h^{(i)}=1$

$$\begin{gathered}\underline{w}_{h,r}^{(i)}\leq\tilde{\boldsymbol{\xi}}_{h,r}^{(i)}\leq\overline{w}_{h,r}^{(i)}\text{ and }\tilde{\boldsymbol{\xi}}_{h,r}^{(i)}=\tilde{\boldsymbol{W}}_{h,r}^{(i)},\\ \forall h\in\{1,\ldots,p_iw_i\},\ \forall r\in\{1,\ldots,n\},\ \forall i\in\{1,\ldots,\ell\}.\end{gathered}$$

We can now rewrite the equality constraint in the third line of (7.23) as follows

$$\begin{aligned}\boldsymbol{\xi}_{1,r}^{(i)}&=\sum_{h=1}^{p_i}\delta_h^{(i)}\tilde{\boldsymbol{W}}_{h,r}^{(i)}=\Delta_{1,:}^{(i)}\tilde{\boldsymbol{W}}_{:,r}^{(i)},\\ &\vdots\\ \boldsymbol{\xi}_{w_i,r}^{(i)}&=\sum_{h=p_i(w_i-1)+1}^{p_iw_i}\delta_h^{(i)}\tilde{\boldsymbol{W}}_{h,r}^{(i)}=\Delta_{w_i:}^{(i)}\tilde{\boldsymbol{W}}_{:,r}^{(i)},\\ &\forall r\in\{1,\ldots,n\},\ \forall i\in\{1,\ldots,\ell\}.\end{aligned}$$

This can be written in a more compact form as matrix multiplication

$$\boldsymbol{\xi}^{(i)}=\Delta^{(i)}\tilde{\boldsymbol{W}}^{(i)}=\Delta^{(i)}\boldsymbol{W}^{(i)}\boldsymbol{\xi}^{(i-1)},\ \forall i\in\{1,\ldots,\ell\}.$$

Starting with $\boldsymbol{\xi}^{(0)}=\boldsymbol{I}$, this recursion leads to

$$\boldsymbol{\xi}^{(\ell)}=\prod_{i=1}^{\ell}\Delta^{(i)}\boldsymbol{W}^{(i)}.$$

By substituting the former relation in (7.21) we can show that the local gain is indeed given by (7.24), which completes the proof. ■

The constraints (7.18) and (7.23) can now be used to model a maxout NN by MI linear constraints and to compute the output (7.19) and the local gain (7.24) by solving an MI feasibility problem. If we further combine the results of Lemma 7.2 and 7.3, we can compute the maximum error and the α-Lipschitz constant of the error.

Theorem 7.4. *Let $\boldsymbol{\Phi}(\boldsymbol{x})$ be a maxout NN and $\alpha = \{1, \infty\}$. Then the maximum error* (7.11) *and the α-Lipschitz constant of the error* (7.12) *can be computed by solving an MILP.*

Proof. The proof follows from [16, Thm. 6.1], where it is shown that both values can be computed by solving an MILP, if the output $\boldsymbol{\Phi}(\boldsymbol{x})$ and the local gain $\boldsymbol{K}_{\mathrm{NN}}(\boldsymbol{x})$ of the NN can be computed by solving an MI feasibility problem. This is always possible for maxout NN according to Lemma 7.2 and 7.3. Thus, by replacing the MI linear constraints for the ReLU NN in [16, Thm. 6.1] with the constraints (7.18) and (7.23), we can compute (7.11) and (7.12) for a maxout NN by solving an MILP. ■

7.4 Numerical examples

We consider two simple examples to highlight our theoretical findings. In both examples, the maximum error (7.11) and the α-Lipschitz constant of the error (7.12) are computed by solving the MILP according to Theorem 7.4 with the MOSEK optimization toolbox for MATLAB (see [45]). The NN are implemented in Python with Keras [118] and Tensorflow [119]. During the training, the MSE (7.13) is minimized with respect to the weighting matrices and bias vectors of the NN using a stochastic gradient descent.

7.4.1 Example system with $n = 1$

We consider the system from [108, Ex. 2] with the dynamics

$$x(k+1) = \tfrac{6}{5}x(k) + u(k)$$

and the constraints $\mathcal{X} = [-10, 10]$ and $\mathcal{U} = [-1, 1]$. As in [108], we choose $Q = 19/5$, $R = 1$, $P = 5$, and $\mathcal{T} = [-1, 1]$. Finally, we select $N = 2$ for illustration purposes here. Explicitly solving the OCP (7.3) then leads to the control law

$$\pi(x) = \begin{cases} -1 & \text{if} \quad x \in \left[-\frac{20}{9}, -1\right], \\ -x & \text{if} \quad x \in [-1, 1], \\ 1 & \text{if} \quad x \in \left[1, \frac{20}{9}\right]. \end{cases} \tag{7.27}$$

For a maxout NN with the topology from Corollary 7.1 (i) with $p_1 = 2$,

$$\begin{aligned} &\boldsymbol{W}^{(1)} = \begin{pmatrix} -1 & 0 & -1 & 0 \end{pmatrix}^\top, \ \boldsymbol{b}^{(1)} = \begin{pmatrix} 0 & -1 & -1 & 0 \end{pmatrix}^\top \text{ and} \\ &\boldsymbol{W}^{(2)} = \begin{pmatrix} 1 & -1 \end{pmatrix} \end{aligned} \tag{7.28}$$

we have

$$\Phi(x) = \max\{-x, -1\} - \max\{-x - 1, 0\} = \pi(x). \tag{7.29}$$

We implemented the constraints (7.18) and (7.23) with $\overline{b}^{(i)} = \overline{w}_{h,r}^{(i)} = -\underline{w}_{h,r}^{(i)} = 10^4$, $\varepsilon = 10^{-5}$ and used them to compute (7.11) and (7.12) with $\alpha = \infty$ according to Theorem 7.4. For a maxout NN with the parameters (7.28), which exactly represents the control law, we computed $\bar{e}_\infty = 4.5 \times 10^{-16}$ and $\mathcal{L}_\infty(e, \mathcal{T}) = 0$. This can be seen as an experimental evidence that the implementation computes correctly the maximum error and the ∞-Lipschitz constant of the error.

To obtain the data in Table 7.1, we sampled randomly 1000 points from the control law (7.27) and trained different maxout NN for 1000 epochs.

Table 7.1. Training results of different maxout NN with $\ell = 1$ and $w_2 = 1$ for $n = 1$.

No.	w_1	p_1	$\#_p$	$\hat{e}$	$\bar{e}_\infty$	$\mathcal{L}_\infty(e, \mathcal{T})$
1.	1	4	10	0.19	0.58	0.74
2.	2	4	19	1.39×10^{-6}	3.22×10^{-6}	2.24×10^{-7}
3.	2	3	15	1.87×10^{-6}	4.36×10^{-6}	2.04×10^{-7}
4.	2	2	11	1.48×10^{-6}	3.26×10^{-6}	1.69×10^{-6}
5.	3	2	16	1.33×10^{-6}	3.73×10^{-6}	1.73×10^{-6}
6.	4	2	21	1.26×10^{-6}	2.93×10^{-6}	3.01×10^{-7}
7.	4	1	13	0.24	0.45	0.65

The results show that a maxout NN with $w_1 = 2$ and $p_1 = 2$, which can represent the control law exactly (cf. (7.29)), is sufficient to get a good approximation, in terms of maximum error and ∞-Lipschitz constant of the error. We only get an approximation with a relatively high error for the first and last topology of Table 7.1. This is not surprising since for $w_1 = 1$, $p_1 = 4$ the maxout NN is a convex PWA function and for $w_1 = 4$, $p_1 = 1$ it is an affine function. In both cases, it is not possible to find an approximation of (7.27) with a maximum error close to zero.

7.4.2 Example system with $n = 2$

We consider the OCP (7.3) with

$$\begin{aligned}
&\boldsymbol{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}, \\
&\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid |\boldsymbol{x}_1| \leq 25,\ |\boldsymbol{x}_2| \leq 5\}, \\
&\mathcal{U} = \{u \in \mathbb{R} \mid |u| \leq 1\},
\end{aligned}$$

and choose $\boldsymbol{Q} = \boldsymbol{I}$, $R = 1$, $N = 3$, $\boldsymbol{P}$ as the solution of the discrete-time algebraic Riccati equation and $\mathcal{T}$ as the maximal output admissible set (see [120] for details). Explicitly solving the OCP leads to a control law with 29 regions. Using the procedure described in [70, Sec. 1], we find a maxout NN of the type (i) from Corollary 7.1 with $p_1 = 38$ and 231 parameters, that exactly describes the control law. For this NN we computed $\bar{e}_\infty = 2.68 \times 10^{-12}$ and $\mathcal{L}_\infty(e, \mathcal{T}) = 3.11 \times 10^{-5}$. A ReLU NN that can exactly represent the control law has according to [7, Thm. 1] 457 parameters. This shows that, as already stated in Section 7.3, ReLU NN for the exact representation of PWA functions are more conservative compared to maxout NN.

Table 7.2 summarizes the training results for different maxout NN trained for 1000 epochs on 10^4 samples of the control law. The values of (7.11) and (7.12) are computed as in the first example, except that in this example $\varepsilon = 10^{-3}$ is chosen.

Table 7.2. Training results of different maxout NN with $\ell = 1$ and $w_2 = 1$ for $n = 2$.

No.	w_1	p_1	$\#_p$	$\hat{e}$	$\bar{e}_\infty$	$\mathcal{L}_\infty(e, \mathcal{T})$
1.	2	38	231	4.47×10^{-3}	4.59×10^{-2}	2.33×10^{-3}
2.	2	10	63	6.04×10^{-3}	1.26×10^{-1}	6.24×10^{-3}
3.	2	3	21	2.45×10^{-2}	1.69×10^{-1}	1.31×10^{-2}
4.	3	3	31	2.19×10^{-2}	1.62×10^{-1}	7.78×10^{-2}
5.	5	3	51	1.97×10^{-2}	1.63×10^{-1}	9.60×10^{-2}
6.	10	3	101	1.31×10^{-2}	1.30×10^{-1}	2.05×10^{-1}
7.	23	3	231	8.95×10^{-3}	9.94×10^{-2}	4.27×10^{-1}

The experimental data indicate that the first maxout NN is the best choice. Note that this is a maxout NN with the topology (i), which theoretically allows an exact representation of the control law, trained on samples of the control law. The seventh maxout NN is of the type (ii), where w_1 is chosen such that the number of parameters is equal to that of the first maxout NN. It has a comparable maximum error but a ∞-Lipschitz constant that is about 183-times higher. This observation may be explained by the fact that maxout NN with more neurons represent PWA functions with more regions compared to maxout NN with less neurons (see [22, Thm. 3.6]). This makes it more likely that there exits one region for which the deviation between the local gains of the NN and the control law is larger, resulting in a larger ∞-Lipschitz constant (cf. (7.12)). The increasing ∞-Lipschitz constant with the number of neurons w_1 in Table 7.2 supports this assumption.

7.5 Conclusions and outlook

We presented a method to compute two values, i.e., the maximum error (7.11) with respect to the optimal control law and the Lipschitz constant (7.12) of the error function (7.10), which are crucial to certify stability of the closed-loop system with a maxout NN controller that approximates the optimal control law of MPC. Our results are derived by showing how the output and the local gain of maxout NN can be computed by solving an MI feasibility problem (cf. Lems. 7.2 and 7.3). The combination of both lemmas leads to Theorem 7.4, which states that the maximum error and the α-Lipschitz constant of the error function can be computed by solving an MILP. The computation of both values has been successfully applied to a number of realizations, including maxout NN which exactly describe the control law.

An interesting direction for future research is the combination of the proposed method with the results from [P2], where maxout NN are designed that allow an exact description of the piecewise quadratic optimal value function in MPC. Such a combination may provide a new method to analyze maxout NN approximations of the optimal value function.

Chapter 8

Efficient computation of Lipschitz constants for MPC with symmetries*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. Lipschitz constants for linear MPC are useful for certifying inherent robustness against unmodeled disturbances or robustness for neural network-based approximations of the control law. In both cases, knowing the minimum Lipschitz constant leads to less conservative certifications. Computing this minimum Lipschitz constant is trivial given the explicit MPC. However, the computation of the explicit MPC may be intractable for complex systems. The paper discusses a method for efficiently computing the minimum Lipschitz constant without using the explicit control law. The proposed method simplifies a recently presented mixed-integer linear program (MILP) that computes the minimum Lipschitz constant. The simplification is obtained by exploiting saturation and symmetries of the control law and irrelevant constraints of the optimal control problem.

Keywords. Model Predictive Control, Optimization, Mixed-Integer Programming, Computational methods

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

8.1 Introduction and problem statement

Model predictive control (MPC) is an established method for the performance-oriented control of dynamical systems subject to state and input constraints (see [17] for an overview). For various applications or realizations of MPC, knowing a Lipschitz constant of the control law is beneficial. For instance, Lipschitz constants can be used to certify inherent robustness of classical MPC against unmodeled disturbances [61, 62]. Based on similar concepts, Lipschitz constants can be used to certify robustness for neural network-based approximations of MPC laws [16, P3].

As pointed out in [96], computing a Lipschitz constant for linear MPC is trivial, if the (piecewise affine) control law is explicitly known. However, the number of affine segments may grow exponentially with the state dimension and the number of constraints in the optimal control problem (OCP). Thus, explicitly computing the control law often becomes intractable for complex systems. This motivates the design of methods that allow the computation of a Lipschitz constant without explicitly computing the control law. Such methods have been previously addressed, e.g, in [16, 96]. In [96], a procedure enumerating potential active sets of the OCP is proposed. However, the procedure is inefficient and, in parts, based on an unproven conjecture (see [96, Conj. 4]). In [16], the problem of finding the smallest Lipschitz constant for MPC is formulated as a mixed-integer linear program (MILP). The procedure is elegant and it can be applied in case where the computation of the explicit control law is numerically intractable (cf. [16, Tab. I]). Still, solving the resulting MILP may be time-consuming since, typically, many binary variables are involved.

In this paper, we aim for a more efficient computation of Lipschitz constants via MILP. To this end, we first slightly improve the MILP formulation from [16] by reducing the initial number of binary variables. Afterwards, we present various preprocessing steps to further reduce the number of binary variables in the MILP. In this context, the most powerful reduction step builds on symmetries that often arise in MPC (see, e.g., [97]). Furthermore, the observation that the control law is constant in many regions for a large prediction horizon also allows a significant reduction in the number of binary variables.

The paper is organized as follows. The notation is given in the remainder of this section. Section 8.2 is devoted to the basics of MPC and presents a known representation of the OCP in terms of an MILP, which can be used to compute the minimum Lipschitz constant of the control law. In Section 8.3, we describe several methods for reducing the number of binary variables needed to compute the minimum Lipschitz constant via MILP. We illustrate the effectiveness of the proposed method with some examples from the literature in Section 8.4 and give a conclusion in Section 8.5.

Notation. For matrices $\boldsymbol{K} \in \mathbb{R}^{m \times n}$, we denote its i-th row by $\boldsymbol{K}_{i,:}$, its j-th column by $\boldsymbol{K}_{:,j}$, and its elements by $\boldsymbol{K}_{i,j}$, respectively. We further use the shorthand notation $\boldsymbol{K}_i$ for $\boldsymbol{K}_{i,:}$ and we indicate different instances of a matrix $\boldsymbol{K}$ by a

superscript, e.g., $\boldsymbol{K}^{(i)}$. The product

$$\boldsymbol{K}\mathcal{X} := \{\boldsymbol{K}\boldsymbol{x} \in \mathbb{R}^m \mid \boldsymbol{x} \in \mathcal{X}\}$$

is defined for a compact and convex set $\mathcal{X}$. For a vector $\boldsymbol{v} \in \mathbb{R}^m$ we define $\mathrm{diag}(\boldsymbol{v})$ as a diagonal matrix with the elements of $\boldsymbol{v}$ being the diagonal elements. Moreover, $\boldsymbol{I}$ refers to the identity matrix and we denote column vectors or matrices whose entries are all 0 respectively 1 by $\mathbf{0}$ respectively $\mathbf{1}$. Finally, all inequalities involving matrices or vectors are understood element-wise.

8.2 Preliminaries

8.2.1 Lipschitz constants of piecewise affine functions

In general a Lipschitz constant of a function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ on a domain $\mathcal{F} \subseteq \mathbb{R}^n$ is a constant L satisfying

$$\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\tilde{\boldsymbol{x}})\| \leq L\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$$

for all $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{F}$ and some vector norm $\|\cdot\|$. We here focus on p-norms and, in particular, the cases $p \in \{1, \infty\}$. We denote corresponding Lipschitz constants by L_p. Further, we aim for as small as possible Lipschitz constants and denote the smallest one by L_p^*. Now, under the assumption that $\boldsymbol{f}$ is a continuous piecewise affine (PWA) function on $\mathcal{F}$, i.e., of the form

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{cases} \boldsymbol{K}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \\ \boldsymbol{K}^{(s)}\boldsymbol{x} + \boldsymbol{b}^{(s)} & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{8.1}$$

with $\boldsymbol{K}^{(i)} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b}^{(i)} \in \mathbb{R}^m$, then it is well-known that

$$L_p^* = \max_{i \in \{1,\ldots,s\}} \|\boldsymbol{K}^{(i)}\|_p, \tag{8.2}$$

where $\|\cdot\|_p$ here refers to the matrix norm induced by the vector p-norm (see, e.g., [63, Prop. 3.4]). As pointed out in [16, Lem. 5.1], the matrix norms $\|\boldsymbol{K}\|_1$ and $\|\boldsymbol{K}\|_\infty$ can be evaluated by solving a linear program (LP). In fact, one finds

$$\begin{aligned} \|\boldsymbol{K}\|_1 = \min_{l, \boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(n)}} \; & l \\ \text{s.t. } \; & \mathbf{1}^\top \boldsymbol{v}^{(j)} \leq l, \;\; -\boldsymbol{v}^{(j)} \leq \boldsymbol{K}_{:,j} \leq \boldsymbol{v}^{(j)}, \;\; \forall j \in \{1, \ldots, n\}. \end{aligned} \tag{8.3}$$

Due to $||\boldsymbol{K}||_\infty = ||\boldsymbol{K}^\top||_1$, the LP for computing the ∞-norm can be formulated in a similar way.

8.2.2 Linear MPC via MILP

MPC for linear discrete-time systems builds on solving an OCP of the form

$$\begin{aligned} V(\boldsymbol{x}) := \min_{\substack{\boldsymbol{x}(0),\dots,\boldsymbol{x}(N)\\ \boldsymbol{u}(0),\dots,\boldsymbol{u}(N-1)}} \; & \varphi(\boldsymbol{x}(N)) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}(k),\boldsymbol{u}(k)) \\ \text{s.t.} \quad & \boldsymbol{x}(0) = \boldsymbol{x}, \\ & \boldsymbol{x}(k+1) = \boldsymbol{A}\,\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \quad \forall k \in \{0,\dots,N-1\}, \\ & (\boldsymbol{x}(k),\boldsymbol{u}(k)) \in \mathcal{X}\times\mathcal{U}, \qquad\quad \forall k \in \{0,\dots,N-1\}, \\ & \boldsymbol{x}(N) \in \mathcal{T} \end{aligned} \tag{8.4}$$

in every time step for the current state $\boldsymbol{x}$. Here, $N \in \mathbb{N}$ refers to the prediction horizon and

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

denote the terminal and stage cost, respectively, where we assume that the weighting matrices $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are positive definite. The dynamics of the linear prediction model are described by $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$. State and input constraints can be incorporated via the polyhedral sets $\mathcal{X}$ and $\mathcal{U}$. Finally, a polyhedral terminal set $\mathcal{T}$ allows to enforce closed-loop stability (see [48] for details). Condensing the OCP (8.4) leads to a parametric quadratic program (QP) of the form

$$\begin{aligned} \boldsymbol{U}^*(\boldsymbol{x}) := \arg\min_{\boldsymbol{U}} \; & \frac{1}{2}\boldsymbol{U}^\top \boldsymbol{H}\boldsymbol{U} + \boldsymbol{x}^\top \boldsymbol{F}^\top \boldsymbol{U} \\ \text{s.t.} \quad & \boldsymbol{G}\boldsymbol{U} \leq \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d} \end{aligned} \tag{8.5}$$

with the decision variable $\boldsymbol{U} \in \mathbb{R}^{mN}$ reflecting the predicted input sequence (i.e, the stacked vectors $\boldsymbol{u}(0),\dots,\boldsymbol{u}(N-1)$) and with $\boldsymbol{H},\boldsymbol{F},\boldsymbol{G},\boldsymbol{E},\boldsymbol{d}$ denoting condensed matrices obtained from $\boldsymbol{A},\boldsymbol{B},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R}$ and the specifications of $\mathcal{X},\mathcal{U},\mathcal{T}$ (see [9, Chap. 3] for details). Now, MPC typically builds on applying the first element of the optimal input sequence, i.e., $\boldsymbol{u}^*(0)$, and repeating the procedure at the next sampling instant. Hence, the resulting control law $f_{\text{MPC}} : \mathcal{F}_{\text{MPC}} \to \mathcal{U}$ can be defined as

$$f_{\text{MPC}}(\boldsymbol{x}) := \boldsymbol{S}\,\boldsymbol{U}^*(\boldsymbol{x}), \tag{8.6}$$

where $\boldsymbol{S} := (\boldsymbol{I}\ \boldsymbol{0}\ \dots\ \boldsymbol{0}) \in \mathbb{R}^{m\times mN}$ serves as a selection matrix and where the set $\mathcal{F}_{\text{MPC}}$ contains all $\boldsymbol{x} \in \mathbb{R}^n$ for which (8.5) (or, equivalently, (8.4)) is feasible. Remarkably, f_{MPC} is of the form (8.1) with regions $\mathcal{R}^{(i)}$ representing polyhedral sets with pairwise disjoint interiors [20, Thm. 4].

Clearly, the solution of (8.5) can be characterized by the Karush-Kuhn-Tucker

(KKT) conditions

$$\begin{aligned} \boldsymbol{H}\boldsymbol{U}^*(\boldsymbol{x}) + \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}^\top \boldsymbol{\lambda}^*(\boldsymbol{x}) &= \boldsymbol{0}, && \text{(8.7a)} \\ \boldsymbol{r}^*(\boldsymbol{x}) &= \boldsymbol{E}\boldsymbol{x} + \boldsymbol{d} - \boldsymbol{G}\boldsymbol{U}^*(\boldsymbol{x}), && \text{(8.7b)} \\ \boldsymbol{r}^*(\boldsymbol{x}) &\geq \boldsymbol{0}, && \text{(8.7c)} \\ \boldsymbol{\lambda}^*(\boldsymbol{x}) &\geq \boldsymbol{0}, && \text{(8.7d)} \\ \operatorname{diag}(\boldsymbol{\lambda}^*(\boldsymbol{x}))\, \boldsymbol{r}^*(\boldsymbol{x}) &= \boldsymbol{0} && \text{(8.7e)} \end{aligned}$$

where $q \in \mathbb{N}$ reflects the dimension of $\boldsymbol{d}$, i.e., the number of constraints. Now, let us assume upper bounds $\overline{\lambda}$ and $\overline{r}$ for the dual optimizers $\boldsymbol{\lambda}^*(\boldsymbol{x})$ respectively the residuals $\boldsymbol{r}^*(\boldsymbol{x})$ are known, i.e., $\boldsymbol{\lambda}^*(\boldsymbol{x}) \leq \overline{\lambda}$ and $\boldsymbol{r}^*(\boldsymbol{x}) \leq \overline{r}$ for all $\boldsymbol{x} \in \mathcal{F}_{\text{MPC}}$. Since (8.5) is feasible for all $x \in \mathcal{F}_{\text{MPC}}$ by definition, such upper bounds exist and are finite. Then, the conditions (8.7c)–(8.7e) can be rewritten as

$$\begin{aligned} 0 \leq \boldsymbol{r}^*(\boldsymbol{x}) &\leq \operatorname{diag}(\overline{r})(\boldsymbol{1} - \boldsymbol{\delta}^*(\boldsymbol{x})), && \text{(8.8a)} \\ 0 \leq \boldsymbol{\lambda}^*(\boldsymbol{x}) &\leq \operatorname{diag}(\overline{\lambda})\, \boldsymbol{\delta}^*(\boldsymbol{x}), && \text{(8.8b)} \\ \boldsymbol{\delta}^*(\boldsymbol{x}) &\in \{0,1\}^q && \text{(8.8c)} \end{aligned}$$

[16, Eq. (13)]. Thus, the QP (8.5) can be solved by solving the MI feasibility problem

$$\text{find } \boldsymbol{U}^*,\, \boldsymbol{\lambda}^*,\, \boldsymbol{r}^*, \text{ and } \boldsymbol{\delta}^* \quad \text{s.t.} \quad \text{(8.7a)–(8.7b) and (8.8)}, \tag{8.9}$$

where we omit dependencies on $\boldsymbol{x}$ for brevity.

8.2.3 Local MPC gain via MILP

Given a solution to (8.9), it is easy to see that the set

$$\mathcal{A}(\boldsymbol{x}) := \{i \in \{1,\dots,q\} \mid \delta_i^*(\boldsymbol{x}) = 1\}$$

reflects the active constraints for (8.5). In principle, $\mathcal{A}(\boldsymbol{x})$ allows to compute the local MPC segment, i.e., $\boldsymbol{K}^*$, $\boldsymbol{b}^*$, and $\mathcal{R}^*$ such that $\boldsymbol{x} \in \mathcal{R}^*$ and

$$f_{\text{MPC}}(\tilde{\boldsymbol{x}}) = \boldsymbol{K}^*\tilde{\boldsymbol{x}} + \boldsymbol{b}^* \quad \forall\, \tilde{\boldsymbol{x}} \in \mathcal{R}^*.$$

However, performing the computation analogously to [20] leads to nonlinear relations between $\boldsymbol{K}^*$ and $\boldsymbol{\delta}^*(\boldsymbol{x})$. Hence, aiming for a combination of (8.2), (8.3) and (8.9) in one MILP, we need to derive $\boldsymbol{K}^*$ differently. A suitable approach has been proposed in the proof of [16, Thm. 5.3]. The underlying idea is to sample the MPC segment (potentially even outside its domain $\mathcal{R}^*$) and to use the samples to characterize $\boldsymbol{K}^*$. The sampling points are chosen as

$$\boldsymbol{x}^{(j)} := \boldsymbol{x} + \boldsymbol{e}^{(j)} \quad \forall j \in \{1,\dots,n\},$$

where $\boldsymbol{e}^{(j)}$ refers to the j-th canonical unit vector. One then constructs corresponding $\boldsymbol{U}^{(j)}$ from the MPC segment around $\boldsymbol{x}$ by introducing the additional

variables $\boldsymbol{U}^{(j)}$, $\boldsymbol{\lambda}^{(j)}$, and $\boldsymbol{r}^{(j)}$ and by augmenting the constraints (8.7a)–(8.7b) and (8.8) with

$$\boldsymbol{H}\boldsymbol{U}^{(j)} + \boldsymbol{F}(\boldsymbol{x} + \boldsymbol{e}^{(j)}) + \boldsymbol{G}^\top \boldsymbol{\lambda}^{(j)} = \boldsymbol{0} \tag{8.10a}$$

$$\boldsymbol{E}(\boldsymbol{x} + \boldsymbol{e}^{(j)}) + \boldsymbol{d} - \boldsymbol{G}\boldsymbol{U}^{(j)} = \boldsymbol{r}^{(j)} \tag{8.10b}$$

$$-M(\boldsymbol{1} - \boldsymbol{\delta}^*(\boldsymbol{x})) \leq \boldsymbol{r}^{(j)} \leq M(\boldsymbol{1} - \boldsymbol{\delta}^*(\boldsymbol{x})) \tag{8.10c}$$

$$-M\boldsymbol{\delta}^*(\boldsymbol{x}) \leq \boldsymbol{\lambda}^{(j)} \leq M\boldsymbol{\delta}^*(\boldsymbol{x}) \tag{8.10d}$$

for all $j \in \{1, \dots, n\}$, where M is a sufficiently large number such that feasible solutions satisfy $\boldsymbol{r}_i^{(j)}, \boldsymbol{\lambda}_i^{(j)} \in (-M, M)$. Then, we obviously have $\boldsymbol{S}\big(\boldsymbol{U}^{(j)} - \boldsymbol{U}^*(\boldsymbol{x})\big) = \boldsymbol{K}^* \boldsymbol{e}^{(j)}$ and consequently

$$\boldsymbol{K}^* = \boldsymbol{S}\begin{pmatrix} \boldsymbol{U}^{(1)} - \boldsymbol{U}^*(\boldsymbol{x}) & \dots & \boldsymbol{U}^{(n)} - \boldsymbol{U}^*(\boldsymbol{x}) \end{pmatrix}. \tag{8.11}$$

We could now compute $\|\boldsymbol{K}^*\|_1$ by solving (8.3) for $\boldsymbol{K}^*$ as in (8.11) subject to the additional constraints (8.7a)–(8.7b), (8.8), and (8.10), which results in an MILP. However, including a maximization analogously to (8.2) (here over $\boldsymbol{x} \in \mathcal{X}$) is non-trivial since (8.3) calls for a minimization. To circumvent this issue, one can consider the dual of (8.3) and observe that the corresponding optimizer is binary [16, Lem. 5.1]. This finally allows to compute the Lipschitz constant L_1^* for an MPC scheme based on an MILP with

$$q + (2m + 1)n \tag{8.12}$$

binary variables (or $q + (2n + 1)m$ for L_∞^*) [16].

8.3 Reducing binary variables for efficiency

The numerical complexity for solving an MILP crucially depends on the number of binary variables. Hence, in order to enable more efficient Lipschitz constant computations for MPC, we aim for a reduction of the number of binary variables in the corresponding MILP. In this context, we first stress that the number in (8.12) consists of two terms resulting from the q constraints of the MPC-related QP in (8.5) and the $(2m + 1)n$ constraints in (8.3). Hence, we have two immediate options for reducing the number of binary variables: First, reducing the number of constraints in (8.5) relevant for the computation of Lipschitz constants and, second, implementing the norm evaluation more efficiently. We investigate these options in Sections 8.3.1 and 8.3.2, respectively. Finally, we show in Section 8.3.3 that exploiting symmetries, which are often present in MPC, can be beneficial for the computation of Lipschitz constants.

8.3.1 More efficient norm computation

Roughly speaking, the computation of Lipschitz constants proposed in [16] builds on reformulations of the QP (8.5) and the LP (8.3) in terms of MI feasibility problems. In [16], the reformulation of (8.3) is realized based on the LP's

dual. Next, we propose a direct reformulation based on the primal LP, which is inspired by MI modeling techniques from, e.g., [56] and which requires fewer binary variables than [16].

Lemma 8.1. *Let $\boldsymbol{K} \in \mathbb{R}^{m \times n}$ and consider the conditions*

$$\boldsymbol{K} = \boldsymbol{K}^+ - \boldsymbol{K}^-, \tag{8.13a}$$

$$\mathbf{0} \leq \boldsymbol{K}^+ \leq M\begin{pmatrix} \boldsymbol{\delta}^{(1)} & \dots & \boldsymbol{\delta}^{(m)} \end{pmatrix}^\top, \tag{8.13b}$$

$$\mathbf{0} \leq \boldsymbol{K}^- \leq M\left(\mathbf{1} - \begin{pmatrix} \boldsymbol{\delta}^{(1)} & \dots & \boldsymbol{\delta}^{(m)} \end{pmatrix}^\top\right), \tag{8.13c}$$

$$\boldsymbol{c}^\top = \mathbf{1}^\top(\boldsymbol{K}^+ + \boldsymbol{K}^-), \tag{8.13d}$$

$$\boldsymbol{c} \leq \mathbf{1}l \leq \boldsymbol{c} + M(\mathbf{1} - \boldsymbol{\delta}^{(m+1)}), \tag{8.13e}$$

$$\mathbf{1}^\top \boldsymbol{\delta}^{(m+1)} = 1, \tag{8.13f}$$

$$\boldsymbol{\delta}^{(1)}, \dots, \boldsymbol{\delta}^{(m+1)} \in \{0,1\}^n \tag{8.13g}$$

for some M being larger than the largest absolute value of the entries in $\boldsymbol{K}$. Then, any solution to

$$\text{find } l, \boldsymbol{K}^+, \boldsymbol{K}^-, \boldsymbol{c}, \boldsymbol{\delta}^{(1)}, \dots, \boldsymbol{\delta}^{(m+1)} \quad \textit{s.t.} \quad (8.13) \tag{8.14}$$

is such that $\|\boldsymbol{K}\|_1 = l$.

Proof. It is easy to see that the conditions (8.13a)–(8.13c) together with $\boldsymbol{\delta}^{(1)}, \dots, \boldsymbol{\delta}^{(m)} \in \{0,1\}^n$ imply

$$(\boldsymbol{K}^+_{i,j}, \boldsymbol{K}^-_{i,j}) := \begin{cases} (\boldsymbol{K}_{i,j}, 0) & \text{if} \quad \boldsymbol{K}_{i,j} \geq 0, \\ (0, -\boldsymbol{K}_{i,j}) & \text{otherwise.} \end{cases}$$

As a consequence, the entries of $\boldsymbol{c}$ as in (8.13d) reflect the absolute column sums of $\boldsymbol{K}$. Due to (8.13e), (8.13f), and $\boldsymbol{\delta}^{(m+1)} \in \{0,1\}^n$, l equals the largest entry of $\boldsymbol{c}$, which is, by definition, identical to $\|\boldsymbol{K}\|_1$. ■

Obviously, the MI feasibility problem (8.14) only requires $(m+1)n$ binary variables and thus mn less than the counterpart in [16]. Again, a similar problem can easily be constructed to compute $\|\boldsymbol{K}\|_\infty$ using $(n+1)m$ binary variables. Likewise, the reduction compared to [16] amounts to mn.

8.3.2 Excluding MPC constraints

We propose two approaches to reduce the number of constraints in (8.5) and consequently the number of binary variables in (8.9). The approaches are conceptually decoupled but it will turn out that their implementation can be efficiently coupled. The first approach builds on the straightforward observation that the i-th constraint in (8.5) (i.e., $\boldsymbol{G}_i\boldsymbol{U} \leq \boldsymbol{E}_i\boldsymbol{x} + \boldsymbol{d}_i$) is irrelevant for the MPC scheme, if there exists no $\boldsymbol{x} \in \mathcal{F}_{\text{MPC}}$ such that $\boldsymbol{G}_i\boldsymbol{U}^*(\boldsymbol{x}) = \boldsymbol{E}_i\boldsymbol{x} + \boldsymbol{d}_i$. According to the following lemma, such a situation can be identified based on an MI feasibility problem similar to (8.9).

Lemma 8.2. *Let $i \in \{1, \dots, q\}$. If the MI feasibility problem* (8.9) *with the additional decision variable $\boldsymbol{x}$ and with the additional constraint $\boldsymbol{\delta}_i^* = 1$ is infeasible, then*

$$\boldsymbol{G}_i \boldsymbol{U}^*(\boldsymbol{x}) - \boldsymbol{E}_i \boldsymbol{x} < \boldsymbol{d}_i \quad \textit{for all } \boldsymbol{x} \in \mathcal{F}_{MPC}.$$

Proof. We first note that the unmodified problem (8.9) has, by construction, the same feasible set $\mathcal{F}_{\text{MPC}}$ as (8.5) with respect to the parameter $\boldsymbol{x}$. Now, the additional constraint $\boldsymbol{\delta}_i^* = 1$ implies $\boldsymbol{r}_i^* = 0$ in (8.8a) and consequently $\boldsymbol{G}_i \boldsymbol{U}^* = \boldsymbol{E}_i \boldsymbol{x} + \boldsymbol{d}_i$ in (8.7b). Hence, infeasibility of the MI feasibility problem in the claim with $\boldsymbol{x}$ as a decision variable immmediately implies that there exists no $\boldsymbol{x} \in \mathcal{F}_{\text{MPC}}$ such that $\boldsymbol{G}_i \boldsymbol{U}^*(\boldsymbol{x}) - \boldsymbol{E}_i \boldsymbol{x} = \boldsymbol{d}_i$. In other words, solving (8.5) for any feasible $\boldsymbol{x} \in \mathcal{F}_{\text{MPC}}$ results in $\boldsymbol{G}_i \boldsymbol{U}^*(\boldsymbol{x}) - \boldsymbol{E}_i \boldsymbol{x} < \boldsymbol{d}_i$. ■

Clearly, with the help of Lemma 8.2, we can eliminate irrelevant constraints by checking the corresponding MI feasibility problem for every (or some) $i \in \{1, \dots, q\}$. At this point, it might seem counterintuitive to consider multiple MI problems in order to simply the overlaying MILP of interest. However, our numerical benchmark in Section 8.4 clearly shows that this approach is meaningfull and that the overall runtime can be (significantly) shortened compared to a direct solution of the unmodified MILP for the computation of Lipschitz constants. Remarkably, Lemma 8.2 could also be used to simplify an MPC scheme offline in order to accelerate the QP solutions online.

The second approach for reducing binary variables associated to constraints differs from the first one in that it is tailored to the problem at hand. It is based on the observation that the MPC law of the form (8.1) often contains many segments with $\boldsymbol{K}^{(i)} = \boldsymbol{0}$ resulting in constant inputs determined by the bias term $\boldsymbol{b}^{(i)}$. In particular, this situation often arises if box-shaped input constraints are present. Clearly, due to $\|\boldsymbol{0}\|_p = 0$, the constant segments are irrelevant for the computation of Lipschitz constants (but they obviously matter for the MPC scheme). Now, according to the following theorem, our procedure to identify and exclude some of these segments is similar to the first approach.

Theorem 8.3. *Let $i \in \{1, \dots, q\}$ and let $\mathcal{F}_{MPC}$ be full-dimensional. Consider the MILP*

$$\max_{\substack{\boldsymbol{U}^*, \boldsymbol{U}^{(1)}, \dots, \boldsymbol{U}^{(m)}, \boldsymbol{x}, \boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}, \\ \boldsymbol{\lambda}^*, \boldsymbol{\lambda}^{(1)}, \dots, \boldsymbol{\lambda}^{(m)}, \boldsymbol{r}^*, \boldsymbol{r}^{(1)}, \dots, \boldsymbol{r}^{(m)}, \boldsymbol{\delta}^*}} \sum_{j=1}^{m} \boldsymbol{U}_j^{(j)} - \boldsymbol{U}_j^* \tag{8.15}$$

subject to (8.7a)–(8.7b), (8.8), $\boldsymbol{\delta}_i^* = 1$, *and*

$$\boldsymbol{H}\boldsymbol{U}^{(j)} + \boldsymbol{F}\boldsymbol{x}^{(j)} + \boldsymbol{G}^\top \boldsymbol{\lambda}^{(j)} = \boldsymbol{0}, \tag{8.16a}$$

$$\boldsymbol{E}\boldsymbol{x}^{(j)} + \boldsymbol{d} - \boldsymbol{G}\boldsymbol{U}^{(j)} = \boldsymbol{r}^{(j)}, \tag{8.16b}$$

$$\boldsymbol{0} \leq \boldsymbol{r}^{(j)} \leq \operatorname{diag}(\overline{\boldsymbol{r}})(\boldsymbol{1} - \boldsymbol{\delta}^*), \tag{8.16c}$$

$$\boldsymbol{0} \leq \boldsymbol{\lambda}^{(j)} \leq \operatorname{diag}(\overline{\boldsymbol{\lambda}})\, \boldsymbol{\delta}^* \tag{8.16d}$$

for every $j \in \{1, \dots, m\}$. If the MILP is feasible and returns 0 as the optimal objective function value, then the i-th constraint can be omitted for the computation of the Lipschitz constant.

Proof. We initially neglect the objective function in (8.15) and investigate the corresponding MI feasibility problem. We further assume feasibility since the theorem is irrelevant otherwise. Now, we consider any feasible set of decision variables and note that, based on the corresponding $\boldsymbol{U}^*$, $\boldsymbol{x}$, $\boldsymbol{\lambda}^*$, $\boldsymbol{r}^*$, and $\boldsymbol{\delta}^*$, we can construct another set of feasible variables by choosing $\boldsymbol{U}^{(j)} := \boldsymbol{U}^*$, $\boldsymbol{x}^{(j)} := \boldsymbol{x}$, $\boldsymbol{\lambda}^{(i)} := \boldsymbol{\lambda}^*$, and $\boldsymbol{r}^{(j)} := \boldsymbol{r}^*$ for every $j \in \{1, \dots, m\}$ (and keeping the other variables). Clearly, the associated objective function value is 0. In other words, given feasibility, the optimal value of the MILP is always non-negative. We next show that the optimal value is always positive if a feasible $\boldsymbol{x}$ exists, for which the corresponding $\mathcal{A}(\boldsymbol{x})$ (containing i by construction) leads to $\boldsymbol{K}^* \neq \boldsymbol{0}$ and a full-dimensional $\mathcal{R}^*$. To this end, we recall that $\mathcal{A}(\boldsymbol{x})$ is determined by $\boldsymbol{\delta}^*$. We further note that the additional constraints (8.16) imply $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)} \in \mathcal{R}^*$ (analogously to $\boldsymbol{x} \in \mathcal{R}^*$). Hence, we find

$$\boldsymbol{U}_j^{(j)} - \boldsymbol{U}_j^* = \boldsymbol{K}_j^* \boldsymbol{x}^{(j)} + \boldsymbol{b}_j^* - \boldsymbol{K}_j^* \boldsymbol{x} - \boldsymbol{b}_j^* = \boldsymbol{K}_j^* (\boldsymbol{x}^{(j)} - \boldsymbol{x})$$

for every $j \in \{1, \dots, m\}$. Now, $\boldsymbol{K}^* \neq \boldsymbol{0}$ implies $\boldsymbol{K}_j^* \neq \boldsymbol{0}$ for at least one j. Due to $\mathcal{R}^*$ being full-dimensional, there exist $\boldsymbol{x}^{(j)}, \boldsymbol{x} \in \mathcal{R}^*$ yielding a positive $\boldsymbol{U}_j^{(j)} - \boldsymbol{U}_j^*$. As a consequence, the optimal value in (8.15) will be positive since the other terms in the cost function have already been shown to be non-negative (due to the m independent $\boldsymbol{x}^{(j)}$)). Conversely, if the MILP returns 0 as an optimal value, the active sets $\mathcal{A}(\boldsymbol{x})$ associated with feasible $\boldsymbol{x}$ either correspond to $\boldsymbol{K}^* = \boldsymbol{0}$, lower dimensional $\mathcal{R}^*$, or both. Now, the former and the latter case are clearly irrelevant for computing the Lipschitz constant. However, also the remaining case $\boldsymbol{K}^* \neq \boldsymbol{0}$ on some lower dimensional domain $\mathcal{R}^*$ is irrelevant if $\mathcal{F}_{\text{MPC}}$ is full-dimensional (as assumed). In fact, due to continuity of $\boldsymbol{f}_{\text{MPC}}$ [20, Thm. 4], the relevant gain $\boldsymbol{K}^*$ will then be captured by some neighboring segment on a full-dimensional domain. ■

Theorem 8.3 provides another condition to potentially exclude the i-th constraint. Remarkably, the two conditions in the previous theorem and Lemma 8.2, while conceptually different, are methodically closely related. To see this, note that the constraints of the corresponding MI problems only differ in terms of (8.16). Now, it is easy to see that the constraints (8.16) are feasible whenever the corresponding constraints (8.7a)–(8.7b) and (8.8) are feasible for $\boldsymbol{x}$. Hence, we immediately find the following relation.

Corollary 8.4. *The MI feasibilty problem in Lemma 8.2 is feasible if and only if the MILP in Theorem 8.3 is feasible.*

As a consequence, one can only investigate the MILP in Theorem 8.3 and exclude the i-th constraint if the MILP is either infeasible or returns 0.

8.3.3 Exploiting symmetries in MPC

Due to common symmetries in the constraints or the cost function, MPC often results in control laws, which likewise offer symmetries. Formalizing these symmetries can, e.g., be carried out analogously to [97, Def. 1]. There, a symmetry is expressed in terms of invertible matrices $(\boldsymbol{\Theta}, \boldsymbol{\Omega})$ satisfying

$$\boldsymbol{\Omega} \boldsymbol{f}_{\text{MPC}}(\boldsymbol{x}) = \boldsymbol{f}_{\text{MPC}}(\boldsymbol{\Theta} \boldsymbol{x}) \tag{8.17}$$

for every $\boldsymbol{x} \in \mathcal{F}_{\text{MPC}}$. Exploiting symmetries is, e.g., useful in the framework of explicit MPC [20] since it allows to reduce the domain for which the explicit control law has to be computed (and stored). To specify this, we first note that multiple symmetries in terms of tuples $(\boldsymbol{\Theta}^{(1)}, \boldsymbol{\Omega}^{(1)}), \dots, (\boldsymbol{\Theta}^{(\sigma)}, \boldsymbol{\Omega}^{(\sigma)})$ can apply simultaneously with the canonical tuple $(\boldsymbol{I}_m, \boldsymbol{I}_n)$ being one of those. Then, we can substitute the constraint $\boldsymbol{x}(0) \in \mathcal{X}$ in (8.4) with $\boldsymbol{x}(0) \in \mathcal{X}_{\text{fun}}$ for any choice of (the so-called fundamental domain) $\mathcal{X}_{\text{fun}} \subseteq \mathcal{X}$ satisfying

$$\mathcal{F}_{\text{MPC}} \subseteq \bigcup_{i=1}^{\sigma} \boldsymbol{\Theta}^{(i)} \mathcal{X}_{\text{fun}}.$$

Clearly, in order to still enable the condensation to (8.5), it additionally makes sense to restrict ourselves to polyhedral sets $\mathcal{X}_{\text{fun}}$. Now, assuming for a moment that $\mathcal{X}$ and $\mathcal{X}_{\text{fun}}$ are characterized by the same number of hyperplanes. Then, it is easy to see that the substitution above does not alter the number of constraints in (8.5). Hence, it is not immediately clear how exploiting symmetries can be beneficial for our purposes. In this context, we first note that substituting $\mathcal{X}$ with a significantly smaller set $\mathcal{X}_{\text{fun}}$ (for the constraint associated with $\boldsymbol{x}(0)$) often results in significantly more excluded constraints by the procedures related to Lemma 8.2 and Theorem 8.3 (see our benchmark in Sect. 8.4). Moreover, symmetries often yield relations like $\boldsymbol{K}^{(i)} = -\boldsymbol{K}^{(j)}$ implying $\|\boldsymbol{K}^{(i)}\|_p = \|\boldsymbol{K}^{(j)}\|_p$. However, symmetries do not always result in such trivial relations, in particular, in light of norms. To see this, note that (8.17) in combination with the structure (8.1) provides relations like

$$\boldsymbol{K}^{(i)} \boldsymbol{x} + \boldsymbol{b}^{(i)} = \boldsymbol{\Omega}^{-1} \boldsymbol{K}^{(j)} \boldsymbol{\Theta} \boldsymbol{x} + \boldsymbol{\Omega}^{-1} \boldsymbol{b}^{(j)}.$$

Hence, instead of evaluating $\boldsymbol{f}_{\text{MPC}}$ for some $\boldsymbol{x} \in \mathcal{R}^{(i)}$, we could also make use of segment j containing $\boldsymbol{\Theta} \boldsymbol{x}$. This would allow us to skip segment i in the context of the Lipschitz constant computation and to consider $\|\boldsymbol{\Omega}^{-1} \boldsymbol{K}^{(j)} \boldsymbol{\Theta}\|_p$ instead. This observation can be exploited in two ways. First, we could simply evaluate σ instances of the final MILP resulting for the tightened constraint $\boldsymbol{x}(0) \in \mathcal{X}_{\text{fun}}$ in order to capture all transformed segments via $\|\boldsymbol{\Omega}^{-1} \boldsymbol{K}^{(j)} \boldsymbol{\Theta}\|_p$ for every of the σ tuples $(\boldsymbol{\Theta}^{(i)}, \boldsymbol{\Omega}^{(i)})$. Second and more efficiently, we can evaluate the final MILP only once and consider only those transformations resulting in invariant norms, i.e.,

$$\|\boldsymbol{\Omega}^{-1} \boldsymbol{K} \boldsymbol{\Theta}\|_p = \|\boldsymbol{K}\|_p \quad \text{for every } \boldsymbol{K} \in \mathbb{R}^{m \times n}. \tag{8.18}$$

A sufficient condition for such transformations is as follows.

Lemma 8.5. *Let* $\|\boldsymbol{\Omega}\|_p = 1$ *and* $\|\boldsymbol{\Theta}\|_p = 1$*, then* (8.18) *holds.*

Proof. We first note that invertability of $\boldsymbol{\Omega}$ and $\boldsymbol{\Theta}$ implies $\|\boldsymbol{\Omega}^{-1}\|_p = \|\boldsymbol{\Omega}\|_p^{-1} = 1$ and $\|\boldsymbol{\Theta}^{-1}\|_p = 1$. Hence,

$$\|\boldsymbol{\Omega}^{-1}\boldsymbol{K}\boldsymbol{\Theta}\|_p \leq \|\boldsymbol{\Omega}^{-1}\|_p \|\boldsymbol{K}\|_p \|\boldsymbol{\Theta}\|_p = \|\boldsymbol{K}\|_p$$

due to sub-multiplicativity. On the other hand,

$$\|\boldsymbol{K}\|_p = \|\boldsymbol{\Omega}\boldsymbol{\Omega}^{-1}\boldsymbol{K}\boldsymbol{\Theta}\boldsymbol{\Theta}^{-1}\|_p \leq \|\boldsymbol{\Omega}\|_p \|\boldsymbol{\Omega}^{-1}\boldsymbol{K}\boldsymbol{\Theta}\|_p \|\boldsymbol{\Theta}^{-1}\|_p = \|\boldsymbol{\Omega}^{-1}\boldsymbol{K}\boldsymbol{\Theta}\|_p.$$

In combination, an inclusion results, which proves (8.18). ■

While restrictive, common symmetries in MPC often satisfy the conditions in Lemma 8.5 (see, e.g., the examples in Sect. 8.4). It remains to comment on the identification of symmetries. In this context, we refer to the methods from [97], which allow to identify tuples $(\boldsymbol{\Theta}, \boldsymbol{\Omega})$ that reflect a symmetry purely based on $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\mathcal{X}$, and $\mathcal{U}$, i.e., without computing the explicit control law.

8.3.4 Combined approaches

We are ready to combine our approaches for a more efficient computation of Lipschitz constants. As already indicated, various combinations of the proposed tools can be considered. We specify two variants that will be used for the numerical benchmark in Section 8.4. The variants differ in whether symmetries are exploited (according to the previous section) or not. Hence, slightly neglecting the additional effort for the identification of symmetries and a fundamental domain $\mathcal{X}_{\text{fun}}$, the crucial difference is that either $\boldsymbol{x}(0) \in \mathcal{X}$ or $\boldsymbol{x}(0) \in \mathcal{X}_{\text{fun}}$ is considered as a constraint for the initial state in (8.4). Apart from this difference, all following steps are identical. In fact, we first condense the OCP to a QP of the form (8.5). We then use the MILP in Theorem 8.3 to reduce the number of constraints. More precisely, we investigate for each constraint i whether the MILP is infeasible or offers the optimal objective value 0. In any of these cases, the i-th constraint is deleted. For simplicity of notation, we do not introduce different instances of the QP parameters for the two variants or during the constraint reduction. In fact, we simply assume that the previous instances are overwritten. Once the reduced QP is obtained, we solve the following MILP in order to compute the Lipschitz constant $L_1^* = l^*$:

$$\max_{\substack{\boldsymbol{x}, l, \boldsymbol{K}^*, \boldsymbol{K}^+, \boldsymbol{K}^-, \boldsymbol{c}, \boldsymbol{U}^*, \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(n)}, \\ \boldsymbol{\delta}^*, \boldsymbol{\delta}^{(1)}, \ldots, \boldsymbol{\delta}^{(m+1)}, \boldsymbol{\lambda}^*, \boldsymbol{\lambda}^{(1)}, \ldots, \boldsymbol{\lambda}^{(n)}, \boldsymbol{r}^*, \boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(n)}}} \quad l \tag{8.19}$$

subject to (8.7a)–(8.7b), (8.8), (8.10), (8.11), and (8.13). Despite the MILP formulation of the reduced QP, a central element is the novel norm computation according to Lemma 8.1. Again, L_∞^* can be computed analogously.

8.4 Numerical benchmark

We demonstrate the effectiveness of the proposed procedures by applying them to examples from the literature summarized in Table 8.1. All MILP in this section are solved using the mixed-integer solver from [45] with constants set to $\overline{\lambda} = \bar{r} = M = 10^4$. For every example, we first compute the minimum Lipschitz constants L_1^* and L_∞^* according to the method in [16, Thm. 5.3], which also served as the starting point for our investigations. The required computation time is listed in the sixth and eighth column of Table 8.2, respectively.

Next, we follow the two variants in Section 8.3.4 in order to apply our novel procedures. Regarding the variant exploiting symmetries, we note that all examples in Table 8.1 offer rotational symmetries. More precisely, the matrices $\mathbf{\Theta}$ and $\mathbf{\Omega}$ are of the form

$$\mathbf{\Theta} = \begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix},$$

with $\varphi \in \{\pi, 2\pi\}$ and $\mathbf{\Omega} \in \{-1, 1\}$ for Systems 1 and 2. For the two following systems, we have $\mathbf{\Theta}$ as before and

$$\mathbf{\Omega} = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} = \mathbf{\Theta}^\top$$

with $\varphi \in \{\pi, 2\pi\}$ for System 3 and $\varphi \in \{\pi/2, \pi, 3\pi/2, 2\pi\}$ for System 4. For System 5, $\mathbf{\Theta}$ likewise reflects rotation matrices (which we omit for brevity) and $\mathbf{\Omega} \in \{-1, 1\}$. In all cases, we easily verify that the condition in Lemma 8.5 (i.e., $||\mathbf{\Omega}||_p = ||\mathbf{\Theta}||_p = 1$) holds. Hence, we can restrict our analysis to the fundamental domains in the ninth column of Table 8.1 without the need to consider multiple instances of the MILP (8.19). Before analyzing the performance of our procedures, we note that we consider two different choices for the terminal set $\mathcal{T}$ for each example. First, we simply choose $\mathcal{T} = \mathcal{X}$ (i.e., no terminal constraints). Second, we consider $\mathcal{T} = \mathcal{S}$ with $\mathcal{S}$ denoting the largest positively invariant set, where the linear quadratic regulator (LQR) can be applied without violating constraints. The computation has been carried out according to [120].

Now, in Table 8.2, we list the number of binary variables of the MILP in [16, Thm. 5.3] in the column $\#_\delta$. Further, we also list the number of regions $\mathcal{R}^{(i)}$ of the explicit control law, computed using the multi-parametric toolbox (MPT, [121]), under $\#_\mathcal{R}$ as an orientation. Key performance indicators of our procedures are listed in columns nine to 13. First, $\#_{\hat{\delta}}^{(1)}$ is the number of binary variables for the simplified MILP without considering symmetries. Second, $\#_{\hat{\delta}}^{(2)}$ reflects the same figure with symmetries. Next, the total computation times for evaluating L_1^* and L_∞^* are listed, respectively. These include the times for all preprocessing steps (such as constraint elimination), which are exclusively listed in the last column for completeness.

As apparent from Table 8.2, the time required to compute the Lipschitz constants can be significantly reduced for all examples. Moreover, one can observe that the computation time for the preprocessing is negligible compared to the time required to compute L_p^*. In fact, although MILP are solved during the preprocessing, the overall time can be reduced in all cases. Furthermore, for the proposed method, the variation of the computation times between the examples is lower. This may indicate, that the proposed method scales better with the model complexity.

Finally, Figure 8.1 highlights in purple the regions that are considered by the MILP (8.19) for computing the Lipschitz constant after reducing the number of binary variables according to Lemma 8.2 and Theorem 8.3 under consideration of symmetries. As apparent from Figure 8.2, Theorem 8.3 allows to identify all regions with a local gain of zero. Moreover, all regions that have the same local gain due to symmetries are also excluded from the computation. Thus only regions that are relevant for computing a Lipschitz constant are considered.

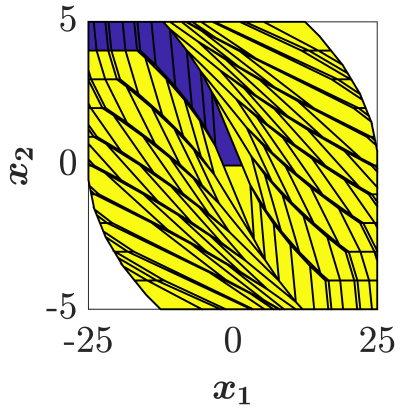


Figure 8.1. Sate space partition of the control law.

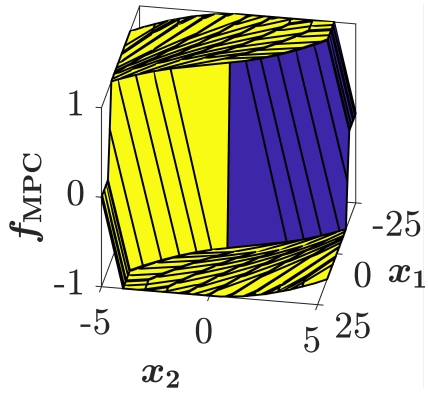


Figure 8.2. Optimal control law for System 1 with $\mathcal{T} = \mathcal{X}$.

Table 8.1. Example systems from the literature.

No.	$\boldsymbol{A}$	$\boldsymbol{B}$	$\mathcal{X}$	$\mathcal{U}$	$\boldsymbol{Q}$	$\boldsymbol{R}$	N	$\mathcal{X}_{\text{fun}}$	Reference
1.	$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$	$\begin{aligned} \lvert x_1 \rvert &\leq 25 \\ \lvert x_2 \rvert &\leq 5 \end{aligned}$	$\lvert u \rvert \leq 1$	$\boldsymbol{I}$	0.1	10	$\begin{aligned} \lvert x_1 \rvert &\leq 25 \\ 0 \leq x_2 &\leq 5 \end{aligned}$	[122, Eqs. (2.8)–(2.9)]
2.	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 4 \end{pmatrix}$	$\begin{aligned} \lvert x_1 \rvert &\leq 5 \\ \lvert x_2 \rvert &\leq 5 \end{aligned}$	$\lvert u \rvert \leq 1$	$\boldsymbol{I}$	4.5	8	$\begin{aligned} \lvert x_1 \rvert &\leq 5 \\ 0 \leq x_2 &\leq 5 \end{aligned}$	[108, Ex. 3]
3.	$\begin{pmatrix} 1.1 & 0.2 \\ -0.2 & 1.1 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.4 \end{pmatrix}$	$\begin{aligned} \lvert x_1 \rvert &\leq 5 \\ \lvert x_2 \rvert &\leq 5 \end{aligned}$	$\begin{aligned} \lvert u_1 \rvert &\leq 1 \\ \lvert u_2 \rvert &\leq 1 \end{aligned}$	$\boldsymbol{I}$	$0.1\boldsymbol{I}$	3	$\begin{aligned} \lvert x_1 \rvert &\leq 5 \\ 0 \leq x_2 &\leq 5 \end{aligned}$	[123, Ex. 2.26]
4.	$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\begin{aligned} \lvert x_1 \rvert &\leq 5 \\ \lvert x_2 \rvert &\leq 5 \end{aligned}$	$\begin{aligned} \lvert u_1 \rvert &\leq 1 \\ \lvert u_2 \rvert &\leq 1 \end{aligned}$	$\boldsymbol{I}$	$\boldsymbol{I}$	10	$\begin{aligned} 0 \leq x_1 &\leq 5 \\ 0 \leq x_2 &\leq 5 \end{aligned}$	[97, Ex. 1]
5.	$\begin{pmatrix} 1 & 0.5 & 0.125 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.02 \\ 0.125 \\ 0.5 \end{pmatrix}$	$\begin{aligned} \lvert x_1 \rvert &\leq 20 \\ \lvert x_2 \rvert &\leq 3 \\ \lvert x_3 \rvert &\leq 1 \end{aligned}$	$\lvert u \rvert \leq 0.5$	$\boldsymbol{I}$	1	3	$\begin{aligned} \lvert x_1 \rvert &\leq 20 \\ \lvert x_2 \rvert &\leq 3 \\ 0 \leq x_3 &\leq 1 \end{aligned}$	[122, Rem. 4.8]

Table 8.2. Computation of Lipschitz constants for different systems.

			MILP from [16, Thm. 5.3]					MILP (8.19) using Lemma 8.1, Theorem 8.3, and symmetries				
No.	$\mathcal{T}$	$\#_{\mathcal{R}}$	$\#_\delta$	L_1^*	Time $[s]$	L_∞^*	Time $[s]$	$\#_\delta^{(1)}$	$\#_\delta^{(2)}$	Time for L_1^* $[s]$	Time for L_∞^* $[s]$	Preprocessing $[s]$
1.	$\mathcal{X}$	211	64	1.89	103.72	1.27	55.51	22	12	0.1735	0.2481	0.0115
1.	$\mathcal{S}$	195	70	1.89	9.98	1.27	43.90	22	12	0.1453	0.2289	0.0156
2.	$\mathcal{X}$	35	52	0.50	15.24	0.50	32.44	32	17	0.6331	0.4759	0.0143
2.	$\mathcal{S}$	35	58	0.50	19.05	0.50	43.24	32	17	0.3275	0.5374	0.0136
3.	$\mathcal{X}$	79	28	16.10	2.59	11.70	3.37	18	15	0.3079	0.5262	0.0587
3.	$\mathcal{S}$	93	32	20.13	1.75	14.63	1.40	16	16	1.2126	1.6429	0.0863
4.	$\mathcal{X}$	491	84	1.69	71.36	1.69	242.74	36	12	0.2590	0.2520	0.0145
4.	$\mathcal{S}$	441	88	1.69	66.14	1.69	85.64	36	12	1.6266	0.1910	0.0248
5.	$\mathcal{X}$	117	30	12.00	2.11	8.00	3.49	28	22	0.3966	0.9440	0.0434
5.	$\mathcal{S}$	107	46	12.00	9.69	8.00	22.87	28	22	1.1988	2.4368	0.0539

8.5 Conclusion

We presented an efficient set of methods to compute the minimum Lipschitz constant of an MPC law. The method adopts a known MILP for the computation of Lipschitz constants and uses various procedures (see Lemma 8.1, Lemma 8.2 and Theorem 8.3) to reduce the number of binary variables in the MILP and thus its complexity. The most powerful reduction builds on exploiting saturation and symmetries of the control law. However, the proposed reduction steps can also be applied to systems without symmetries. This allows an efficient computation of the minimum Lipschitz constant even for complex systems of moderate size. For future research it would be interesting to investigate if the MILP (8.19) can be adopted for the computation of Lipschitz constants of the piecewise quadratic optimal value function $V(\boldsymbol{x})$ of the OP (8.4).

Chapter 9

Piecewise regression via mixed-integer programming for MPC*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. Piecewise regression is a versatile approach used in various disciplines to approximate complex functions from limited, potentially noisy data points. In control, piecewise regression is, e.g., used to approximate the optimal control law of model predictive control (MPC), the optimal value function, or unknown system dynamics. Neural networks are a common choice to solve the piecewise regression problem. However, due to their nonlinear structure, training is often based on gradient-based methods, which may fail to find a global optimum or even a solution that leads to a small approximation error. To overcome this problem and to find a global optimal solution, methods based on mixed-integer programming (MIP) can be used. However, the known MIP-based methods are either limited to a special class of functions, e.g., convex piecewise affine functions, or they lead to complex approximations in terms of the number of regions of the piecewise defined function. Both complicate a usage in the framework of control. We propose a new MIP-based method that is not restricted to a particular class of piecewise defined functions and leads to functions that are fast to evaluate and can be used within an optimization problem, making them well suited for use in control.

Keywords. piecewise regression, mixed-integer programming, model predictive control, neural networks

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

9.1 Introduction

Piecewise regression is used in many disciplines to fit a continuous piecewise defined function to an unknown function based on a finite number of possibly noisy samples. In the context of model predictive control (MPC) for linear systems, piecewise regression is used, e.g., to approximate the piecewise affine (PWA) optimal control law [6, 7] or the piecewise quadratic optimal value function (OVF) [P1]. Both functions are continuous and defined on polyhedral regions, which partition the state space (c.f. [20]). In addition, piecewise regression may be used to find a surrogate model for an unknown nonlinear system that is used as a prediction model in nonlinear MPC. Neural networks (NN) are very popular in piecewise regression as they allow to approximate a large class of functions with arbitrary accuracy [67]. However, due to the nonlinear structure of NN, computing NN parameters, which minimize the approximation error, requires solving a nonlinear optimization problem (OP). This OP is typically solved approximately by gradient-based methods, such as stochastic gradient descent, RMSProb, or Adam [31]. However, these methods are often sensitive to the choice of hyperparameters, and they may even fail to converge for some choices of the learning rate. Moreover, even if the hyperparameters are chosen properly, general convergence guarantees are lacking for the popular RMSProb and Adam algorithms. Furthermore, in case of approximated system dynamics or OVF, trained NN may subsequently be used as a prediction model or to derive optimal control actions, respectively. For both applications, the NN will be included in an OP (either as constraints or costs), and an efficient inclusion is hard for some types of NN. We will consider maxout NN with a special topology that allows to solve the problem of piecewise regression globally and that is suitable for a subsequent usage in an OP.

An alternative for solving the problem of piecewise regression is the direct use of PWA functions. When using PWA functions for regression, either the regions are fixed and a linear regression is performed for each region, or the computation is split into two parts. In the first part, the data is divided into clusters that define the polyhedral regions of the PWA function. In the second part, the parameters of the function are computed by performing a linear regression for every region (see, e.g., [85]). Often, these algorithms alternate between clustering and regression until a termination criterion is met [86]. The main drawback of these methods is that the regression problem is only solved approximately and the approximation quality depends on the initial clustering of the data points. To overcome this problem and to find a global optimal solution, clustering and regression should be done simultaneously. Most of the approaches where clustering and regression are formulated in a combined OP are based on mixed-integer programming (MIP). One of the first MIP-based approaches was introduced in [124], where so-called hinging hyperplanes [125] are fitted via MIP to the data. However, only PWA functions that possess the consistent variation property [83, Def. 5] can be exactly represented by hinging

hyperplanes. Since the consistent variation property typically does not hold in MPC, hinging hyperplanes are not the ideal choice. Another MIP-based method was introduced in [126]. They use binary variables to assign points to regions, we will use similar ideas in our method. In addition, we will also use clustering as in [126, Sec. 3.2.] to cluster points and then assign these clusters to regions of the function. The main weakness of this method is that the function is discontinuous and it is not possible to enforce continuity for problems with $n > 1$ (c.f. [126, Sec. 5.3.]). Moreover, the regions of the function are not necessarily polytopes. Hence, further computations are required to find polytopes that contain the majority of the points associated with an individual region. Furhter MIP-based methods were recently introduced in [33, 34]. However, these methods can only be applied to univariate or bivariate functions. The MIP-based method from [90, Sec. 3.2.] solves some of the aforementioned problems. This method allows the fitting of a convex continuous PWA function to data via a mixed-integer quadratic program (MIQP). The main idea is to represent the convex function as the maximum of its affine segments and use binary variables to determine which affine segment provides the largest value for a given data point. The advantage of this method is that the polyhedral regions of the function are implicitly given by the structure of the function and do not need to be computed separately. However, the method is not extendable to non-convex functions. In [32], an alternative method is given, which allows to combine the benefits of [126] and [90], i.e., it is possible to fit non-convex continuous functions that are defined on polyhedral regions. In [32], the fitted PWA function has one region per data point, this allows to solve the piecewise regression problem by solving a quadratic program (QP) instead of an MIQP as in [90, 124, 126]. Unfortunately, the coupling between the number of regions and the number of data points leads to rather complex functions for large data sets, which is disadvantageous, especially when the function is used in an OP.

Contributions. The main contribution of this paper is to provide a new MIP-based method to globally solve the piecewise regression problem, which combines the advantages of the known approaches. More specifically, our new method

1. allows to freely choose the number of segments as in [126] without the need to compute the regions of the function explicitly,

2. has a simple representation of the fitted function as in [90] without being restricted to convex functions,

3. allows to fit a continuous function as in [32] without coupling the number of regions and the number of data points,

4. results in a model that is suitable for the use in the constraints and the cost function of an OP as in [86].

We also provide two approaches for reducing the complexity of the proposed method to make it applicable to larger data sets.

9.2 Problem formulation and preliminaries

We consider the problem of continuous piecewise regression, i.e., our aim is to fit a continuous piecewise defined function $\Phi(\boldsymbol{x};\boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}$ to samples of an unknown function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$, which typically involves solving the nonlinear OP

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{N_D} \ell\big(y^{(k)} - \Phi(\boldsymbol{x}^{(k)};\boldsymbol{\theta})\big), \tag{9.1}$$

for a given data set $\mathcal{D} := \{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{R}^{n+1} \mid y^{(i)} = f(\boldsymbol{x}^{(i)}), i \in \{1,\dots,N_D\}\}$, where $\boldsymbol{\theta}$ are the parameters of the function and ℓ is a convex loss function. In the following, we briefly summarize the methods from [90] and [32], as our method is most closely related to these and contains both as special cases. In [90], the problem of piecewise affine regression is solved by the MIQP

$$\min_{\boldsymbol{V},\alpha^{(i)},\boldsymbol{\delta}^{(i)}} \sum_{i=1}^{N_D} \big(y^{(i)} - \alpha^{(i)}\big)^2 \tag{9.2a}$$

$$\text{s.t. } \boldsymbol{V}\begin{pmatrix}\boldsymbol{x}^{(i)}\\ 1\end{pmatrix} \le \mathbf{1}\alpha^{(i)}, \quad \boldsymbol{V}\begin{pmatrix}\boldsymbol{x}^{(i)}\\ 1\end{pmatrix} + M(\mathbf{1} - \boldsymbol{\delta}^{(i)}) \ge \mathbf{1}\alpha^{(i)},\ \forall i \in \{1,\dots,N_D\}, \tag{9.2b}$$

$$\sum_{k=1}^{p_1} \boldsymbol{\delta}_k^{(i)} = 1,\ \boldsymbol{\delta}^{(i)} \in \{0,1\}^{p_1},\ \forall i \in \{1,\dots,N_D\}. \tag{9.2c}$$

with $\boldsymbol{V} \in \mathbb{R}^{p_1 \times (n+1)}$ and the subscript j refers to the j-th row of a matrix or column vector. The $N_D p_1$ binary variables select which of the p_1 affine segments $\boldsymbol{V}_j({\boldsymbol{x}^{(i)}}^\top \quad 1)^\top$ with $j \in \{1,\dots,p_1\}$ provides the largest value for a given data point. The constraints (9.2b) ensure that $\max\{\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)})\} = \alpha^{(i)}$ holds, i.e., for a sufficiently large $M \in \mathbb{R}_{>0}$, the MIQP (9.2) solves the nonlinear OP (9.1) with $\ell(\cdot) := \|\cdot\|_2^2$, $\boldsymbol{\theta} := \boldsymbol{V}$, and $\Phi(\boldsymbol{x};\boldsymbol{\theta}) = \max\left\{\boldsymbol{V}\begin{pmatrix}\boldsymbol{x}\\1\end{pmatrix}\right\} := \max\limits_{1\le i\le p_1}\left\{\boldsymbol{V}_i\begin{pmatrix}\boldsymbol{x}\\1\end{pmatrix}\right\}$. Thus, a convex PWA function is fitted to the data set. In contrast to [126], this method allows to make predictions by evaluating the function $\Phi(\boldsymbol{x})$ without explicitly computing the regions of the function. Moreover, the function $\Phi(\boldsymbol{x})$ is continuous by construction. However, the disadvantage is that the fitted function is always convex and PWA. In our method, we also use the binary variables to select the largest segments, but we will overcome the limitation to convex functions by extending the MIQP (9.2).

A method that combines the ability to fit non-convex functions with the continuity property of (9.2) is presented in [32], where the problem of piecewise affine regression is solved by solving a QP (see [32, Eq. (6)]) instead of an MIQP. The constraints in the QP ensure that the function

$$\Phi(\boldsymbol{x}) = \max_{1\le i\le N_D} \{\boldsymbol{a}_i^\top(\boldsymbol{x} - \boldsymbol{x}^{(i)}) + \hat{y}_i + z_i\} - \max_{1\le i\le N_D} \{\boldsymbol{b}_i^\top(\boldsymbol{x} - \boldsymbol{x}^{(i)}) + z_i\} \tag{9.3}$$

is such that we have $\Phi(\boldsymbol{x}^{(i)}) = \hat{y}_i$ and the cost function is the sum of the residuals $(y^{(i)} - \hat{y}_i)^2$, i.e., the function (9.3) is fitted to the data. Since every PWA can be decomposed into the difference of two convex PWA functions and (9.3) is the difference of two convex PWA functions, a general PWA function is fitted to the data points in [32]. Unfortunately, unlike [126] or [90], this method does not allow to choose the number of regions of the PWA function freely. As apparent from (9.3), the number of regions is always equal to the number of data points, resulting in a rather complex function for large data sets.

9.3 A new MIP-based method for piecewise regression

In this section, we present our new method for piecewise regression via MIP, which combines advantageous properties of the known approaches while avoiding some of their limitations. Our starting point is the continuous function $\Phi(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ parameterized by

$$\Phi(\boldsymbol{x}) := \max\{\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x})\} - \max\{\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})\} \tag{9.4}$$

with $\boldsymbol{V} \in \mathbb{R}^{p_1 \times r_1}$ and $\boldsymbol{W} \in \mathbb{R}^{p_2 \times r_2}$ as well as $\boldsymbol{g}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^{r_1}$ and $\boldsymbol{h}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^{r_2}$. We denote $\boldsymbol{V}_j\boldsymbol{g}(\boldsymbol{x})$ with $j \in \{1, \dots, p_1\}$ and $\boldsymbol{W}_j\boldsymbol{h}(\boldsymbol{x})$ with $j \in \{1, \dots, p_2\}$ as segments of the max-functions. In addition, we refer to a segment i as being active if it provides the largest value for a given data point $\boldsymbol{x}$, i.e., if $\boldsymbol{V}_i\boldsymbol{g}(\boldsymbol{x}) \geq \boldsymbol{V}_j\boldsymbol{g}(\boldsymbol{x})$ holds for all $j \in \{1, \dots, p_1\}$. This form of parameterization is chosen because every PWA function can be parameterized in this way if $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ contain affine and constant functions, i.e., for

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix}. \tag{9.5}$$

Moreover, in [P2], it is shown that every PWQ with $n = 1$ can be represented in the form (9.4) if $\boldsymbol{g}(\boldsymbol{x})$ further contains all quadratic monomials of $\boldsymbol{x}$. Alternatively, the function (9.4) can be interpreted as a shallow maxout NN [21] with two neurons of rank p_1 and p_2, respectively. For the training of such NN, the nonlinear OP (9.1) with $\Phi(\boldsymbol{x})$ as in (9.4) is often solved with gradient-based methods. However, these gradient-based methods may get stuck in poor local optima. This problem can be avoided by reformulating the nonlinear OP as

the MIQP

$$\min_{\boldsymbol{V},\boldsymbol{W},\alpha^{(i)},\beta^{(i)},\gamma^{(i)},\boldsymbol{\delta}^{(i)}} \sum_{i=1}^{N_D} \left(y^{(i)} - (\alpha^{(i)} - \beta^{(i)})\right)^2 \tag{9.6a}$$

$$\text{s.t.} \quad \boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\alpha^{(i)}, \quad \boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)}) + M(\mathbf{1} - \boldsymbol{\delta}^{(i)}) \geq \mathbf{1}\alpha^{(i)}, \tag{9.6b}$$

$$\sum_{k=1}^{p_1} \boldsymbol{\delta}_k^{(i)} = 1, \quad \boldsymbol{\delta}^{(i)} \in \{0,1\}^{p_1}, \tag{9.6c}$$

$$\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\beta^{(i)}, \quad \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)}) + M(\mathbf{1} - \boldsymbol{\gamma}^{(i)}) \geq \mathbf{1}\beta^{(i)}, \tag{9.6d}$$

$$\sum_{k=1}^{p_2} \boldsymbol{\gamma}_k^{(i)} = 1, \quad \boldsymbol{\gamma}^{(i)} \in \{0,1\}^{p_2}, \tag{9.6e}$$

for all $i \in \{1, \dots N_D\}$. The binary variable vectors $\boldsymbol{\delta}^{(i)}$ and $\boldsymbol{\gamma}^{(i)}$ in (9.6b) and (9.6d) select which segments of the max-functions are active for the point $\boldsymbol{x}^{(i)}$. The constant $M \in \mathbb{R}_{>0}$, often referred to as big-M, is typically very large. We assume without loss of generality that $p_j \leq N_D$ for $j \in \{1,2\}$, because otherwise there are segments which are inactive for all $\boldsymbol{x}^{(i)}$ with $i \in \{1, \dots, N_D\}$. Note that for the special case $p_2 = 0$ as well as $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (9.5) the MIQP (9.6) becomes equivalent to (9.2). In the following theorem, we will show that the MIQP (9.6) can be used to solve the OP (9.1).

Theorem 9.1. *Let $\Phi(\boldsymbol{x})$ be as in* (9.4)*, $\ell(\cdot) := \|\cdot\|_2^2$, and $\boldsymbol{\theta} := \{\boldsymbol{V}, \boldsymbol{W}\}$. Then there exists an $M \in \mathbb{R}_{>0}$ for which the MIQP* (9.6) *is equivalent to the nonlinear OP* (9.1)*.*

Proof. The constraints (9.6c) are such that for every $i \in \{1, \dots, N_D\}$ there exist exactly one k_i with $\boldsymbol{\delta}_{k_i}^{(i)} = 1$ and $\boldsymbol{\delta}_k^{(i)} = 0$ for all $k \in \{1, \dots, p_1\} \setminus k_i$. Thus, the constraints (9.6b) and (9.6c) can be equivalently written as

$$\boldsymbol{V}_{k_i}\boldsymbol{g}(\boldsymbol{x}^{(i)}) = \alpha^{(i)}, \quad \boldsymbol{V}_k\boldsymbol{g}(\boldsymbol{x}^{(i)}) \leq \boldsymbol{V}_{k_i}\boldsymbol{g}(\boldsymbol{x}^{(i)}),\ \forall i \in \{1, \dots, N_D\},\ \forall k \in \{1, \dots, p_1\} \tag{9.7a}$$

$$\boldsymbol{V}_k\boldsymbol{g}(\boldsymbol{x}^{(i)}) + M \geq \boldsymbol{V}_{k_i}\boldsymbol{g}(\boldsymbol{x}^{(i)}), \qquad \forall i \in \{1, \dots, N_D\},\ \forall k \in \{1, \dots, p_1\}. \tag{9.7b}$$

As a consequence, the constraints (9.7a) are equivalent to $\alpha^{(i)} = \max\{\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)})\}$. By using similar arguments, we can show that similar conditions hold for $\beta^{(i)}$, which leads to the OP

$$\min_{\boldsymbol{V},\boldsymbol{W}} \sum_{i=1}^{N_D} \left(y^{(i)} - (\max\{\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)})\} - \max\{\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)})\})\right)^2 \tag{9.8a}$$

$$\text{s.t.} \quad \max\{\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)})\} = \alpha^{(i)}, \quad \max\{\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)})\} = \beta^{(i)},\ \forall i \in \{1, \dots, N_D\}, \tag{9.8b}$$

$$\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)}) + \mathbf{1}M \geq \mathbf{1}\alpha^{(i)}, \quad \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)}) + \mathbf{1}M \geq \mathbf{1}\beta^{(i)},\ \forall i \in \{1, \dots, N_D\}. \tag{9.8c}$$

This OP is always feasible since the trivial solution ($\boldsymbol{V} = \boldsymbol{0}$, $\boldsymbol{W} = \boldsymbol{0}$, and $\alpha^{(i)} = \beta^{(i)} = 0$ for all $i \in \{1, \dots, N_D\}$) is in the feasible set. Furthermore, since $\boldsymbol{1}\alpha^{(i)} - \boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)})$ and $\boldsymbol{1}\beta^{(i)} - \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)})$ are finite, we can always find a finite $M \in \mathbb{R}_{>0}$ such that the constraints (9.8c) never hold with equality. For such an M, the constraints (9.8b) and (9.8c) can be neglected. Consequently, the OP (9.8) and thus (9.6) are equivalent to the nonlinear OP (9.1) with $\Phi(\boldsymbol{x})$ as in (9.4), $\ell(\cdot) := \|\cdot\|_2^2$, and $\boldsymbol{\theta} := \{\boldsymbol{V}, \boldsymbol{W}\}$. ■

According to Theorem 9.1, the MIQP (9.6) allows to compute a global optimal solution for a special instance of the nonlinear OP (9.1). Remarkably, the MIQP (9.6) can be used to find an approximation of the form (9.4) for every continuous $\boldsymbol{g}_i(\boldsymbol{x})$ and $\boldsymbol{h}_j(\boldsymbol{x})$, e.g., by choosing $\boldsymbol{g}(\boldsymbol{x})$ quadratic and $\boldsymbol{h}(\boldsymbol{x})$ affine, the function represented by (9.4) is a PWQ function. Even more complex functions are possible. For example, Figure 9.1 shows a piecewise defined function with sinusoidal segments, approximated by (9.4) with $\boldsymbol{g}(x) = \begin{pmatrix} 1 & x & \sin(0.2x) \end{pmatrix}^\top$ and $\boldsymbol{h}(x) = \begin{pmatrix} 1 & x \end{pmatrix}^\top$. In all figures the functions $\max\{\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x})\}$ and $\max\{\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})\}$ are represented in blue and green, respectively. The approximation of $f(x)$ by $\Phi(x)$ is illustrated in black, where the data points $(x^{(i)}, f(x^{(i)}))$ used in the MIQP (9.6) are the black crosses.

Regarding the M from the MIQP (9.6), it should be noted that methods from the literature (see, e.g., [127]) to compute the smallest possible value of M cannot be applied here, because the weights $\boldsymbol{V}$ and $\boldsymbol{W}$ are optimization variables and thus not known in advance. This also means that as apparent from the proof of Theorem 9.1, unlike usual in the big-M formulation, the M does not influence the feasibility of the problem. However, the value of M influences the approximation error if it is not sufficiently large. In fact, a small M leads to smaller values in the entries of the weights $\boldsymbol{V}$ and $\boldsymbol{W}$. Thus, M implicitly regularizes the weights. Assuming the PWA case, we have the following result.

Theorem 9.2. *Let $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ be as in* (9.5)*. Then the norm of the matrices $\boldsymbol{V}_{k,1:n}$ and $\boldsymbol{W}_{l,1:n}$ of the OP* (9.6) *are bounded by the constants v_k and w_l, respectively, i.e., $\|\boldsymbol{V}_{k,1:n}\|_2 \leq v_k$, $\forall k \in \{1, \dots, p_1\}$ and $\|\boldsymbol{W}_{l,1:n}\|_2 \leq w_l$, $\forall l \in \{1, \dots, p_2\}$.*

Proof. Assume that $\delta_{k_i}^{(i)} = 1$ holds for $\boldsymbol{x}^{(i)}$, then we have according to (9.7b)

$$\boldsymbol{V}_{k_i,1:n}\boldsymbol{x}^{(i)} + \boldsymbol{V}_{k_i,n+1} \leq \boldsymbol{V}_{k,1:n}\boldsymbol{x}^{(i)} + \boldsymbol{V}_{k,n+1} + M, \; \forall i \in \{1, \dots, N_D\}, \; \forall k \in \{1, \dots, p_1\}. \tag{9.9}$$

We now construct vectors $\tilde{\boldsymbol{x}}^{(1)}, \dots, \tilde{\boldsymbol{x}}^{(n)}$ such that they are an orthonormal basis containing the vector $\tilde{\boldsymbol{x}}^{(\kappa)} := \frac{\boldsymbol{V}_{k_i,1:n}^\top}{\|\boldsymbol{V}_{k_i,1:n}\|_2}$. By representing $\boldsymbol{x}^{(i)}$ in the orthonormal

basis we can rewrite (9.9) as follows

$$\begin{aligned} \boldsymbol{V}_{k_i,1:n} \sum_{l=1}^{n} \tilde{\boldsymbol{x}}^{(l)} \tilde{\boldsymbol{x}}^{(l)\top} \boldsymbol{x}^{(i)} + \boldsymbol{V}_{k_i,n+1} \leq & \boldsymbol{V}_{k,1:n} \boldsymbol{x}^{(i)} \\ & + \boldsymbol{V}_{k,n+1} + M \\ \Leftrightarrow \|\boldsymbol{V}_{k_i,1:n}\|_2 \|\tilde{\boldsymbol{x}}^{(\kappa)}\|_2 \tilde{\boldsymbol{x}}^{(\kappa)\top} \boldsymbol{x}^{(i)} + \boldsymbol{V}_{k_i,1:n} \sum_{l=1, l\neq\kappa}^{n} \tilde{\boldsymbol{x}}^{(l)} \tilde{\boldsymbol{x}}^{(l)\top} \boldsymbol{x}^{(i)} + \boldsymbol{V}_{k_i,n+1} \leq & \boldsymbol{V}_{k,1:n} \boldsymbol{x}^{(i)} \\ & + \boldsymbol{V}_{k,n+1} + M \end{aligned}$$

$$\begin{aligned} \Leftrightarrow \|\boldsymbol{V}_{k_i,1:n}\|_2 \leq & \frac{1}{\tilde{\boldsymbol{x}}^{(\kappa)\top} \boldsymbol{x}^{(i)}} \left(M + \boldsymbol{V}_{k,n+1} - \boldsymbol{V}_{k_i,n+1} + \left(\boldsymbol{V}_{k,1:n} - \boldsymbol{V}_{k_i,1:n} \sum_{l=1, l\neq\kappa}^{n} \tilde{\boldsymbol{x}}^{(l)} \tilde{\boldsymbol{x}}^{(l)\top} \right) \boldsymbol{x}^{(i)} \right) \\ =: & \, v_k \end{aligned}$$

for all $i \in \{1, \ldots, N_D\}$ and for all $k \in \{1, \ldots, p_1\}$. The upper bound w_l for $\|\boldsymbol{W}_{l,1:n}\|_2$ can be derived in a similar way, which completes the proof. ■

Theorem 9.2 shows that it is not necessary to explicitly include a regularization in the cost function as in [32] to ensure small weights since M implicitly regularizes the weights. Figure 9.2 shows the effect of this implicit regularization. For the example, the MIQP (9.6) is solved with $p_1 = p_2 = 2$ and (9.5), for different values of M. In Figure 9.2 (a), the constant is not sufficiently large and thus an exact fit of the data is impossible. For $M = 50$ and $M = 100$, an exact fit is possible. However, $M = 50$ leads to a fit with smaller weights.

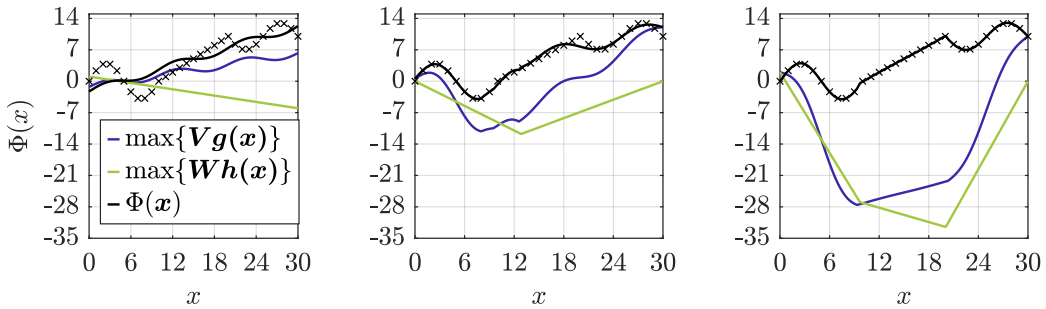


Figure 9.1. $\Phi(x)$ with $p_i = 1$ in (a), $p_i = 2$ in (b), and $p_i = 3$ in (c) for $i \in \{1, 2\}$.

Another advantage of the proposed method is that since for $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (9.5), (9.4) is a maxout NN with affine pre- and post-activation, we can directly use [P3, Lem. 2] to show that (9.4) can be modelled as an MI feasibility problem and is thus well-suited for the use inside an OP.

Corollary 9.3. *Let $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ be as in* (9.5)*. Then, the function* (9.4) *can be modelled as an MI feasibility problem.*

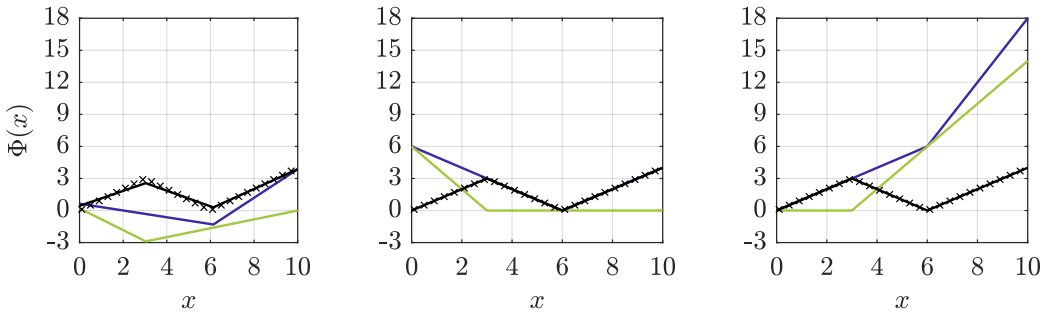


Figure 9.2. PWA $\Phi(x)$ with $M=10$ in (a), $M=50$ in (b), and $M=100$ in (c).

Proof. According to [P3, Lem. 2], an MI feasibility problem with the constraints [P3, Eq. (18)] can be used to replace the nonlinear relation (9.4). ■

This corollary allows the replacement of the nonlinear function (9.4) inside an OP by MI linear constraints. Thus, a nonlinear OCP, where (9.4) is used either as a prediction model or to approximate the cost function, can be reformulated as MIP. This enables the use of standard MI solvers to find a global optimum of the OP.

9.4 Complexity reduction

In this section, we describe two approaches to reduce the complexity of the MIQP (9.6). The approaches mainly focus on reducing the number of binary variables. Before describing our approaches, we first show that the MIQP (9.6) can be solved by a QP without using any binary variables if we are not interested in a small number of segments p_1 and p_2 in (9.4). Consider the special case where we have exactly as many segments as data points, i.e. $p_1 = p_2 = N_D$. In this case, the QP

$$\min_{\boldsymbol{V},\boldsymbol{W},\alpha^{(i)},\beta^{(i)}} \sum_{i=1}^{N_D} \left(y^{(i)} - (\alpha^{(i)} - \beta^{(i)})\right)^2 \tag{9.10a}$$

$$\text{s.t.} \quad \boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\alpha^{(i)}, \quad \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\beta^{(i)}, \quad \forall i \in \{1,\dots N_D\}, \tag{9.10b}$$

$$\boldsymbol{V}_\kappa \boldsymbol{g}(\boldsymbol{x}^{(\kappa)}) \geq \alpha^{(\kappa)}, \quad \boldsymbol{W}_\kappa \boldsymbol{h}(\boldsymbol{x}^{(\kappa)}) \geq \beta^{(\kappa)}, \quad \forall \kappa \in \{1,\dots,N_D\} \tag{9.10c}$$

can be used to compute optimal $\boldsymbol{V}$, $\boldsymbol{W}$, $\alpha^{(i)}$, and $\beta^{(i)}$ of the MIQP (9.6). Since there is one segment for each data point, no binary variables are needed to decide which segment is active, thus (9.6b)-(9.6e) can be replaced by (9.10b) and (9.10c). However, this leads to the fact that the number of data points determines the number of segments. As a result, we have $2N_D(N_D+1)$ inequality constraints and $2N_D + N_D(r_1+r_2)$ optimization variables in (9.10). The QP (9.10) can be reformulated to obtain the QP from [32]. Thus, our

method generalizes known results for piecewise linear regression and includes the OP from [32] as a special case. The elimination of the binary variables in the OP (9.10) is only possible for $p_1 = p_2 = N_D$, which leads to a complex function $\Phi(\boldsymbol{x})$ with a high number of segments, for large data sets. Therefore, in the remainder of this section, we will present two approaches for reducing the number of binary variables in the MIQP (9.6) without increasing the complexity of the fitted function $\Phi(\boldsymbol{x})$, in terms of the number of segments.

9.4.1 Symmetry breaking

Due to symmetries in the MIQP (9.6), there are many equivalent solutions. These symmetries result from the fact that the values of the max-functions in (9.4) are independent of the order of the p_1 and p_2, respectively, segments. In the following proposition, we will present a method to eliminate some of these redundant solutions.

Proposition 9.4. *There exists an optimal solution of the MIQP* (9.6) *with*

$$\boldsymbol{\delta}^{(1)}_{2:p_1} = \mathbf{0},\ \boldsymbol{\delta}^{(2)}_{3:p_1} = \mathbf{0},\ \ldots,\ \boldsymbol{\delta}^{(p_1-1)}_{p_1} = 0 \quad \textit{and} \tag{9.11a}$$

$$\boldsymbol{\gamma}^{(1)}_{2:p_2} = \mathbf{0},\ \boldsymbol{\gamma}^{(2)}_{3:p_2} = \mathbf{0},\ \ldots,\ \boldsymbol{\gamma}^{(p_2-1)}_{p_2} = 0. \tag{9.11b}$$

Proof. Every permutation of the k-indices of an optimal solution of the MIQP (9.6) is also an optimal solution since the maximum is invariant under a permutation of the segments. Assume $\boldsymbol{\delta}^{(i)}$ and $\boldsymbol{\gamma}^{(i)}$ are the binary variables of an optimal solution of the MIQP (9.6). Due to the discussed invariance of the maximum, we can assume that the optimal solution is such that the first segment is active for the first data point, i.e., $\boldsymbol{\delta}^{(1)}_1 = 1$ and $\boldsymbol{\gamma}^{(1)}_1 = 1$. In combination with (9.6c) and (9.6e) this leads to $\boldsymbol{\delta}^{(1)}_{2:p_1} = \mathbf{0}$ and $\boldsymbol{\gamma}^{(1)}_{2:p_2} = \mathbf{0}$. For the second data point, there are two possibilities. Either the same segment is active as for the first data point, or a different segment is active, which we can assume to be the second segment, i.e., $\boldsymbol{\delta}^{(2)}_1 + \boldsymbol{\delta}^{(2)}_2 = 1$ and $\boldsymbol{\gamma}^{(2)}_1 + \boldsymbol{\gamma}^{(2)}_2 = 1$ or equivalent $\boldsymbol{\delta}^{(2)}_{3:p_1} = \mathbf{0}$ and $\boldsymbol{\gamma}^{(2)}_{3:p_2} = \mathbf{0}$. We can proceed in a similar way with the remaining data points until the $(p_1 - 1)$-th and $(p_2 - 1)$-th, respectively, data point, which leads to the remaining constraints in (9.11) and completes the proof. ■

According to Proposition 9.4 we can reduce the number of binary variables by including the constraints (9.11) in the MIQP (9.6). Symmetry breaking is also achieved in [90, Prop. 3.2] by enforcing an order to the parameter of the fitted function through additional constraints. However, unlike our method, this does not reduce the number of binary variables.

9.4.2 Preclustering

Since the number of data points determines the number of binary variables in (9.6), reducing the number of data points is a very effective way to reduce

the computation time needed to solve the MIQP (9.6). Unfortunately, a certain number of data points is often required to obtain an approximation that generalizes well to unseen data. This means that it is impossible to simply reduce the size of the data set. However, we can precompute a clustering of the data set and modify the OP so that instead of assigning individual data points to segments of the max-function, whole clusters of multiple data points are assigned. Assume the data set is partitioned into K disjoint clusters of the form $\mathcal{C}_j := \{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D} \mid i \in \mathcal{I}_j\}$ with $\mathcal{I}_j \subseteq \{1, \dots, N_D\}$, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i \neq j$, and $\cup_{j=1}^{K} \mathcal{C}_j = \mathcal{D}$. Then, we can formulate a modified version of MIQP (9.6) with

$$\boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\alpha^{(i)},\ \boldsymbol{V}\boldsymbol{g}(\boldsymbol{x}^{(i)}) + M(\mathbf{1} - \boldsymbol{\delta}^{(l)}) \geq \mathbf{1}\alpha^{(i)},\ \forall i \in \{1, \dots N_D\}, \tag{9.12a}$$

$$\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)}) \leq \mathbf{1}\beta^{(i)},\ \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}^{(i)}) + M(\mathbf{1} - \boldsymbol{\gamma}^{(l)}) \geq \mathbf{1}\beta^{(i)},\ \forall i \in \{1, \dots N_D\}, \tag{9.12b}$$

instead of (9.6b) and (9.6d), where l is such that we have $(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{C}_l$. The binary variable vectors $\boldsymbol{\delta}^{(l)} \in \{0,1\}^{p_1}$ and $\boldsymbol{\gamma}^{(l)} \in \{0,1\}^{p_2}$ select which segment of the max-functions is active for the whole cluster $\mathcal{C}_l$ and not just for a single data point as in the initial MIQP (9.6). Any algorithm that computes a clustering with the desired properties can be used here, e.g. k-means++ from [91]. Note that although we perform a clustering before solving the optimization problem, the modified MIQP (9.6) with (9.12) instead of (9.6b) and (9.6d) does not computes a linear regression for each cluster individually as in [85], since there is not a distinct segment for each cluster. As long as $p_i < K$ with $i \in \{1, 2\}$, multiple clusters are assigned to the same segment. With the preclustering, we can reduce the number of binary variables from $N_D(p_1 + p_2)$ to $K(p_1 + p_2)$. For $K = N_D$, i.e., each cluster contains exactly one data point, the constraints (9.6b), (9.6d) and (9.12a), (9.12b) are equivalent. Furthermore, analogous to (9.10) for $p_1 = p_2 = K$ the MIQP with preclustering can be reformulated as a QP by replacing (9.10c) in the QP (9.10) with

$$\boldsymbol{V}_\kappa \boldsymbol{g}(\boldsymbol{x}^{(i)}) \geq \alpha^{(i)}, \quad \boldsymbol{W}_\kappa \boldsymbol{h}(\boldsymbol{x}^{(i)}) \geq \beta^{(i)}, \quad \forall i \in \{1, \dots N_D\}, \tag{9.13}$$

where κ in (9.13) is such that we have $(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{C}_\kappa$. The resulting QP has $2N_D(K+1)$ inequality constraints and $2N_D + K(r_1 + r_2)$ optimization variables.

9.5 Numerical examples

In this section, we use the MIQP (9.6) with different $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ to approximate a nonlinear MPC and compare the results with an approximation by an NN.

9.5.1 Nonlinear MPC

Consider the nonlinear OCP

$$V(\boldsymbol{x}) := \min_{\substack{\boldsymbol{x}(0),\dots,\boldsymbol{x}(N)\\ u(0),\dots,u(N-1)}} \boldsymbol{x}(N)^\top \begin{pmatrix} 1.3 & 1.9 \\ 1.9 & 3.0 \end{pmatrix} \boldsymbol{x}(N) + \sum_{k=0}^{N-1} \|\boldsymbol{x}(k)\|_2^2 + \|u(k)\|_2^2 \quad (9.14)$$

$$\begin{aligned} \text{s.t. } & \boldsymbol{x}_1(k+1) = \boldsymbol{x}_1(k) + T_s \boldsymbol{x}_2(k), \\ & \boldsymbol{x}_2(k+1) = \boldsymbol{x}_2(k) - T_s \frac{\rho_0}{m} \mathrm{e}^{-\boldsymbol{x}_1(k)} \boldsymbol{x}_1(k) - T_s \frac{h_d}{m} \boldsymbol{x}_2(k) + T_s \frac{u(k)}{m}, \\ & -1.3 \leq \boldsymbol{x}_2(k) \leq 1.3, \quad -4 \leq u(k) \leq 4, \\ & \boldsymbol{x}(0) = \boldsymbol{x}, \quad \boldsymbol{x}(N)^\top \begin{pmatrix} 1.3 & 1.9 \\ 1.9 & 3.0 \end{pmatrix} \boldsymbol{x}(N) \leq 0.001, \end{aligned}$$

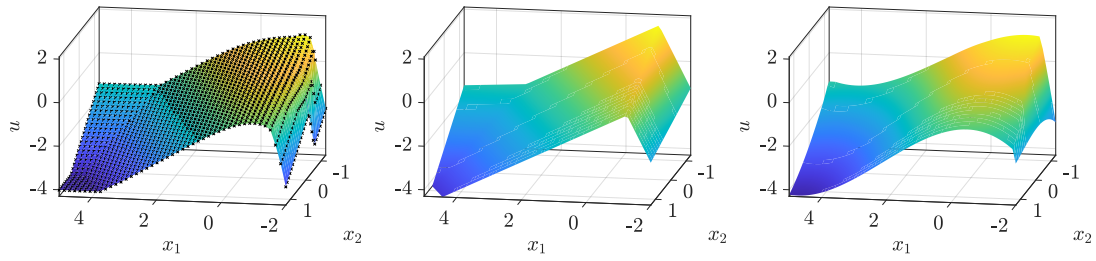
from [128] with parameters $N = 15$, $T_s = 0.4$, $m = 1$, $\rho_0 = 0.33$ and $h_d = 1.1$. All equations and inequalities in (9.14) with index k are valid for all $k \in \{0, \dots, N-1\}$.* The OCP (9.14) is solved for 1350 different values of $\boldsymbol{x}$ on a regular grid of size 45×30 with $\boldsymbol{x}_1$ from -2 to 5 and $\boldsymbol{x}_2$ from -1.3 to 1.3 to generate a data set of the form $\left(\boldsymbol{x}^{(i)}, y^{(i)} = u^*(\boldsymbol{x}^{(i)})\right)$, where $u^*(\boldsymbol{x}^{(i)})$ is the optimal $u(0)$ for the state $\boldsymbol{x} = \boldsymbol{x}^{(i)}$. The nonlinear OCP (9.14) is solved using the software framework CasADi from [129]. Based on the data set, two different maxout NN of the form (9.4) with $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (9.5) are trained using the Adam algorithm [31]. The values of p_1 and p_2 of the two NN are chosen as $p_1 = p_2 = 2$ and $p_1 = p_2 = 50$ (row 1 and 2 of Table 9.1). Such NN can be interpreted as maxout NN with affine pre- and post-activation and 2 neurons of rank 2 and 50, respectively, in one hidden layer. We also used the MIQP (9.6) with $p_1 = p_2 = 2$, different choices of $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ and the symmetry breaking constraints (9.11) to compute global optimal parameters $\boldsymbol{V}$ and $\boldsymbol{W}$ of the function $\Phi(\boldsymbol{x})$. We start with $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ as in (9.5), which results in a function that is equal to the first maxout NN, and add monomials up to the order of three to the functions (row 3 and 4 of Table 9.1). The results are summarized in Table 9.1. The last two columns show the quality of the approximation in terms of the mean squared error (MSE) and the maximum absolute value of the error on a test data set. The second row indicates the method used to compute the parameters. As apparent from Table 9.1, the approximation provided by the MIQP (9.6) with cubic terms in $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ gives the lowest error, even though it has only two segments. By increasing the number of segments p_1 and p_2, the MSE of the first NN approaches the MSE of the 4th function. However, the maximum error barely changes, which may be a problem since it is shown in [16] that the maximum error is crucial for the stability and performance of an approximated controller. Another advantage of the proposed method is that adding additional terms to the $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$

*The sentence had to be added for the reprinted version of the article since the formatting of (9.14) differs from [P5, Eq. (14)].

Table 9.1. Different approximations of the nonlinear optimal control law $u^*(\boldsymbol{x})$.

No.	Method	p_1, p_2	$\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x})$	MSE	$e_{\max}$
1.	NN	2	$(1 \quad \boldsymbol{x}^\top)^\top$	0.0532	1.6054
2.	NN	50	$(1 \quad \boldsymbol{x}^\top)^\top$	0.0252	1.4971
3.	MIQP (9.6)	2	$(1 \quad \boldsymbol{x}^\top)^\top$	0.0503	1.0797
4.	MIQP (9.6)	2	$(1 \quad \boldsymbol{x}^\top \quad x_1^2 \quad x_1x_2 \quad x_2^2 \quad x_1^3 \quad x_1^2x_2 \quad x_1x_2^2 \quad x_2^3)^\top$	0.0240	1.0687

functions can improve the approximation quality without increasing the number of segments. This is beneficial for a fast evaluation of the approximation $\Phi(\boldsymbol{x})$. Figure 9.3 shows how the function 4 from Table 9.1 leads to a better result by approximating the curvature of the optimal control law.



(a) **(b)** **(c)**

Figure 9.3. (a) shows the optimal control law with the sampled data points. (b) and (c) are approximations of the optimal control law by the functions 1 and 4, respectively, from Table 9.1.

9.6 Conclusion

We proposed a new MIP-based method for piecewise regression, which allows to fit an arbitrary piecewise defined function to data points by using the difference of two max-functions (9.4). The presented method is very flexible as it is not restricted to a special class of functions and thus allows to trade off the complexity of the fitted function against the approximation error. In addition, we showed in a numerical example that the method can be used to find a simple approximation, in terms of the number of regions, of the optimal control law of nonlinear MPC.

Chapter 10

Reachability analysis for piecewise affine systems with neural network-based controllers*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. Neural networks (NN) have been successfully applied to approximate various types of complex control laws, resulting in low-complexity NN-based controllers that are fast to evaluate. However, when approximating control laws using NN, performance and stability guarantees of the original controller may not be preserved. Recently, it has been shown that it is possible to provide such guarantees for linear systems with NN-based controllers by analyzing the approximation error with respect to a stabilizing base-line controller or by computing reachable sets of the closed-loop system. The latter has the advantage of not requiring a base-line controller. In this paper, we show that similar ideas can be used to analyze the closed-loop behavior of piecewise affine (PWA) systems with an NN-based controller. Our approach builds on computing over-approximations of reachable sets using mixed-integer linear programming, which allows to certify that the closed-loop system converges to a small set containing the origin while satisfying input and state constraints. We also show how to modify a given NN-based controller to ensure asymptotic stability for the controlled PWA system.

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

Keywords. Model Predictive Control, Neural Networks, Hybrid Systems, Mixed-integer Programming, Reachability, Stability

10.1 Introduction

Piecewise affine (PWA) systems have emerged as an important system class in the control literature as they can be used to model or approximate a large class of hybrid systems or nonlinear systems [65]. Moreover, the equivalence of PWA systems and mixed logical dynamical (MLD) systems has been shown in [64], which allows the use of techniques developed for MLD systems also for PWA systems and vice versa. Stability analysis and control synthesis for PWA systems are an important branch of research, and many methods based on linear matrix inequalities (LMI) have been proposed [101, 130, 131, 132]. However, these methods are often only applicable to linear segments of the PWA system dynamics that contain the origin. Furthermore, it is not possible to handle constraints with these methods. Therefore, model predictive control (MPC) for PWA systems (see, e.g., [133] for an overview) gained attention due to the possibility of directly considering constraints in the optimal control problem (OCP). When using MPC for PWA systems, the switching nature of the system dynamics is typically described by mixed-integer linear constraints [47]. Thus, the OCP with a quadratic cost function becomes a mixed-integer quadratic program (MIQP) that must be solved in every time step. This can be challenging for long prediction horizons or PWA systems with many affine segments because the MIQP may be too complex to be solved within the sampling period. As a consequence, methods have been developed to approximate the optimal control law of MPC ([6, 112]) by functions that are fast to evaluate. A popular choice are neural networks (NN) [40], as they can approximate a large class of functions with arbitrary accuracy. Moreover, NN with PWA activations such as maxout NN [21] share the PWA structure of the MPC law for PWA systems [134, Thm. 1]. Unfortunately, stability and constraint satisfaction are typically not guaranteed for the NN-based controller. In addition, NN are known to react unexpectedly even for small perturbations of the input [135]. Still, for linear systems, some methods to certify stability and constraint satisfaction of the NN-based controller exist. The most commonly used ones are based on mixed-integer linear programming (MILP) ([13, 38]), semi-definite programming [39], or satisfiability modulo theory [37]. However, these methods are either not applicable to PWA systems or require a stabilizing base-line controller.

In this paper, we propose a method for certifying stability and constraint satisfaction for PWA systems with NN-based controllers without using a base-line controller. Our method is based on the computation of over-approximations of k-step reachable sets of the closed-loop system using MILP. Similar ideas have

been used in [38] for linear systems and in [40] for safety verification of PWA systems. However, both existing methods are not suitable for certifying stability and constraint satisfaction for PWA systems with NN-based controllers. In contrast, our method allows us to certify that the closed-loop system converges to a small positively invariant (PI) set containing the origin while satisfying input and state constraints. We also show how to modify the NN-based controller to ensure asymptotic stability for the controlled PWA system.

The paper is organized as follows. In the remainder of this section, we state relevant notation and basic definitions. In Section 10.2, some fundamentals on MPC for PWA systems and NN with PWA activation are given. Section 10.3 contains the main contributions of this paper. Namely, the analysis of the closed-loop system consisting of a PWA system and an NN-based controller by MILP. This MILP is then used to compute PI sets, and over-approximations of reachable sets. These sets are the primary tools used to show the stability of the closed-loop system. A case study is given in Section 10.4 to illustrate the rather technical results. Finally, conclusions are presented in Section 10.5.

10.1.1 Notation and definitions

We define the support function of a polyhedron $\mathcal{X}$ for a row-vector $\boldsymbol{v} \in \mathbb{R}^{1\times n}$ as $h_{\mathcal{X}}(\boldsymbol{v}) := \sup_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{v}\boldsymbol{x}$. For a matrix argument $\boldsymbol{H} \in \mathbb{R}^{w\times n}$, $h_{\mathcal{X}}(\boldsymbol{H})$ is understood as $h_{\mathcal{X}}(\boldsymbol{H}) = \begin{pmatrix} h_{\mathcal{X}}(\boldsymbol{H}_1) & \dots & h_{\mathcal{X}}(\boldsymbol{H}_w) \end{pmatrix}^\top$, where $\boldsymbol{H}_i$ refers to the i-th row of $\boldsymbol{H}$. Support functions have many useful features. For instance, consider a polyhedron of the form $\mathcal{B} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}^{(\mathcal{B})}\boldsymbol{x} \leq \boldsymbol{h}^{(\mathcal{B})}\}$, then $\mathcal{A} \subseteq \mathcal{B}$ if and only if $h_{\mathcal{A}}(\boldsymbol{H}^{(\mathcal{B})}) \leq \boldsymbol{h}^{(\mathcal{B})}$. The scaling of a set $\mathcal{A}$ by a scalar s is defined as $s\mathcal{A} := \{s\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in \mathcal{A}\}$. Natural numbers and natural numbers including 0 are denoted by $\mathbb{N}$ and $\mathbb{N}_0$, respectively. We further define uniform ultimate boundedness (UUB) of a system with respect to a C-set (i.e., a convex and compact set containing the origin in its interior) according to [99, Def. 2.4]: A system is denoted as ultimately bounded in a C-set $\mathcal{T}$, uniformly in $\mathcal{F}$, if for every initial condition $\boldsymbol{x}(0) \in \mathcal{F}$, there exits a $k^*(\boldsymbol{x}(0))$ such that $\boldsymbol{x}(k) \in \mathcal{T}$ holds for all $k \geq k^*(\boldsymbol{x}(0))$.

10.2 Fundamentals of MPC for PWA systems and neural networks with PWA activation

Consider a PWA system of the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\ &:= \boldsymbol{A}^{(i)}\boldsymbol{x}(k) + \boldsymbol{B}^{(i)}\boldsymbol{u}(k) + \boldsymbol{p}^{(i)} \text{ if } \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \in \mathcal{P}^{(i)}, \end{aligned} \tag{10.1}$$

where $\mathcal{P}^{(i)}$ with $i \in \{1, \dots, s\}$ are polyhedral sets with $\text{int}(\mathcal{P}^{(i)}) \cap \text{int}(\mathcal{P}^{(j)}) = \emptyset$ for all $i \neq j$ that partition the state and input space according to $\mathcal{X} \times \mathcal{U} =$

$\cup_{i=1}^{s} \mathcal{P}^{(i)}$. Assume that $\boldsymbol{x} = \boldsymbol{0}$ is an equilibrium of the system for $\boldsymbol{u} = \boldsymbol{0}$. Hence, $\boldsymbol{p}^{(i)} = \boldsymbol{0}$ for all $i \in \mathcal{I}_0 := \{j \in \{1,\dots,s\} \mid \boldsymbol{0} \in \mathcal{P}^{(j)}\}$. Regarding the sets $\mathcal{P}^{(i)}$, we furhter assume that they are given in the hyperplane representation $\mathcal{P}^{(i)} := \{\boldsymbol{\xi} \in \mathbb{R}^{n+m} \mid \boldsymbol{H}^{(i)}\boldsymbol{\xi} \leq \boldsymbol{h}^{(i)}\}$ with $\boldsymbol{H}^{(i)} \in \mathbb{R}^{d_i \times (n+m)}$ and $\boldsymbol{h}^{(i)} \in \mathbb{R}^{d_i}$ being shorthand notation for $\boldsymbol{H}^{(\mathcal{P}^{(i)})}$ and $\boldsymbol{h}^{(\mathcal{P}^{(i)})}$, respectively. A common approach for the control of PWA systems of the form (10.1) is to solve the optimal control problem (OCP)

$$V(\boldsymbol{x}) := \min_{\substack{\hat{\boldsymbol{x}}(0),\dots,\hat{\boldsymbol{x}}(N) \\ \hat{\boldsymbol{u}}(0),\dots,\hat{\boldsymbol{u}}(N-1)}} \|\hat{\boldsymbol{x}}(N)\|_{\boldsymbol{P}}^2 + \sum_{\kappa=0}^{N-1} \|\hat{\boldsymbol{x}}(\kappa)\|_{\boldsymbol{Q}}^2 + \|\hat{\boldsymbol{u}}(\kappa)\|_{\boldsymbol{R}}^2 \tag{10.2a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{x}}(0) = \boldsymbol{x}, \tag{10.2b}$$

$$\hat{\boldsymbol{x}}(\kappa+1) = \boldsymbol{f}_{\text{PWA}}(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)), \forall \kappa \in \{0,\dots,N-1\}, \tag{10.2c}$$

$$(\hat{\boldsymbol{x}}(\kappa), \hat{\boldsymbol{u}}(\kappa)) \in \mathcal{X} \times \mathcal{U}, \ \forall \kappa \in \{0,\dots,N-1\}, \tag{10.2d}$$

$$\hat{\boldsymbol{x}}(N) \in \mathcal{T}, \tag{10.2e}$$

in every time step for the current state $\boldsymbol{x} = \boldsymbol{x}(k)$. The resulting control law $\boldsymbol{f}_{\text{MPC}} : \mathcal{F}_{\text{MPC}} \to \mathcal{U}$ is $\boldsymbol{f}_{\text{MPC}}(\boldsymbol{x}) := \boldsymbol{S}\boldsymbol{U}^*(\boldsymbol{x})$, where $\boldsymbol{S} := (\boldsymbol{I}\ \boldsymbol{0}\ \dots\ \boldsymbol{0}) \in \mathbb{R}^{m \times mN}$ serves as a selection matrix and where the set $\mathcal{F}_{\text{MPC}}$ contains all $\boldsymbol{x} \in \mathbb{R}^n$ for which (10.2) is feasible.

10.2.1 Mixed-integer formulation of PWA system dynamics

To describe the PWA system dynamics of (10.1) by MI linear constraints, we use techniques from [47] and introduce binary variables $\boldsymbol{\gamma}(k) \in \{0,1\}^s$ that model the relation

$$\gamma_i(k) = 1 \Leftrightarrow \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \in \mathcal{P}^{(i)}. \tag{10.3}$$

For sets $\mathcal{P}^{(i)}$ as above, the MI linear constraints

$$\boldsymbol{H}^{(i)} \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \leq \boldsymbol{h}^{(i)} + \boldsymbol{1}M(1 - \gamma_i(k)) \tag{10.4a}$$

$$\boldsymbol{1}^\top \boldsymbol{\gamma}(k) = 1, \quad \boldsymbol{\gamma}(k) \in \{0,1\}^s \tag{10.4b}$$

for all $i \in \{1,\dots,s\}$ and all $k \in \{0,\dots,K-1\}$ with $K \in \mathbb{N}$ can be used to reflect (10.3). The constant M is often referred to as big-M and can be chosen according to [47, Eq. (20)-(21)]. Now, the binary variable $\boldsymbol{\gamma}(k)$ from (10.4) can be used to describe the PWA system dynamics (10.1) by the MI linear constraints

$$\begin{aligned} -\boldsymbol{1}M(1-\gamma_i(k)) &\leq \boldsymbol{A}^{(i)}\boldsymbol{x}(k) + \boldsymbol{B}^{(i)}\boldsymbol{u}(k) + \boldsymbol{p}^{(i)} - \tilde{\boldsymbol{x}}^{(i)}(k+1) \\ &\leq \boldsymbol{1}M(1-\gamma_i(k)) \end{aligned} \tag{10.5a}$$

$$-\boldsymbol{1}M\gamma_i(k) \leq \tilde{\boldsymbol{x}}^{(i)}(k) \leq \boldsymbol{1}M\gamma_i(k) \tag{10.5b}$$

$$\sum_{i=1}^{s} \tilde{\boldsymbol{x}}^{(i)}(k+1) = \boldsymbol{x}(k+1) \tag{10.5c}$$

for all $i \in \{1,\dots,s\}$ and all $k \in \{0,\dots,K-1\}$ (cf. [47, Sec. 3.1]). Thus, the solution of the MI feasibility problem

$$\text{find } \boldsymbol{X}_{K+1}, \boldsymbol{U}_K, \tilde{\boldsymbol{X}}_K^{(1)}, ..., \tilde{\boldsymbol{X}}_K^{(s)}, \boldsymbol{\gamma}(0), ..., \boldsymbol{\gamma}(K-1) \tag{10.6a}$$

$$\text{s.t.} \quad (10.4) \text{ and } (10.5), \text{ with} \tag{10.6b}$$

$$\boldsymbol{X}_{K+1} := \begin{pmatrix} \boldsymbol{x}(0) \\ \vdots \\ \boldsymbol{x}(K) \end{pmatrix}, \boldsymbol{U}_K := \begin{pmatrix} \boldsymbol{u}(0) \\ \vdots \\ \boldsymbol{u}(K-1) \end{pmatrix}, \tilde{\boldsymbol{X}}_K^{(i)} := \begin{pmatrix} \tilde{\boldsymbol{x}}^{(i)}(1) \\ \vdots \\ \tilde{\boldsymbol{x}}^{(i)}(K) \end{pmatrix}$$

for all $i \in \{1,\dots,s\}$, is such that $\boldsymbol{x}(k+1) = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{u}(k))$ holds for all $k \in \{0,\dots,K-1\}$. By replacing the nonlinear system dynamics (10.2c) in the OCP with the MI linear constraints (10.4) and (10.5), the nonlinear OP (10.2) becomes a mixed-integer quadratic program (MIQP), which can be solved using standard solvers, e.g., [45]. However, since the number of binary variables of the resulting MIQP is sN, the online solution of (10.2) in every time step may be intractable for long prediction horizons or PWA systems with a large number of regions. Therefore, we aim for approximations of the control law $\boldsymbol{f}_{\text{MPC}}$ that are fast to evaluate and ensure stability as well as constraint satisfaction.

10.2.2 Mixed-integer formulation of NN with PWA activation

Neural networks are a common choice for approximating various types of complex control laws, as they can be evaluated fast and trained efficiently on large data sets. In general, a feed-forward-NN with ℓ hidden layers and w_i neurons in layer i can be written as a composition of the form

$$\boldsymbol{\Phi}(\boldsymbol{x}) := \boldsymbol{f}^{(\ell+1)} \circ \boldsymbol{g}^{(\ell)} \circ \boldsymbol{f}^{(\ell)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{x}). \tag{10.7}$$

Here, the functions $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{p_i w_i}$ for $i \in \{1,\dots,\ell\}$ refer to preactivations, where the parameter $p_i \in \mathbb{N}$ allows to consider "multi-channel" preactivations as required for maxout (see [21]). Moreover, $\boldsymbol{g}^{(i)} : \mathbb{R}^{p_i w_i} \to \mathbb{R}^{w_i}$ stand for activation functions and $\boldsymbol{f}^{(\ell+1)} : \mathbb{R}^{w_\ell} \to \mathbb{R}^{w_{\ell+1}}$ reflects postactivation. The functions $\boldsymbol{f}^{(i)}$ are typically affine, i.e., $\boldsymbol{f}^{(i)}(\boldsymbol{y}^{(i-1)}) := \boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}$, where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i w_i \times w_{i-1}}$ is a weighting matrix, $\boldsymbol{b}^{(i)} \in \mathbb{R}^{p_i w_i}$ is a bias vector, and $\boldsymbol{y}^{(i-1)}$ denotes the output of the previous layer with $\boldsymbol{y}^{(0)} := \boldsymbol{x} \in \mathbb{R}^n$. Various choices for the activation functions have been proposed. However, since we aim for approximations of the PWA control law $\boldsymbol{f}_{\text{MPC}}(\boldsymbol{x})$ (see [134, Thm. 1]), we here focus on the maxout activation function

$$\boldsymbol{g}^{(i)}(\boldsymbol{z}^{(i)}) := \begin{pmatrix} \max\limits_{1 \le j \le p_i} \{\boldsymbol{z}_j^{(i)}\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1 \le j \le p_i w_i} \{\boldsymbol{z}_j^{(i)}\} \end{pmatrix}, \tag{10.8}$$

as the resulting NN have a PWA input-output relation. Moreover, the maxout activation is more general than other PWA activation functions and includes, e.g., ReLU or leaky ReLU, as special cases. We use the shorthand notation $\max_{1\leq j\leq p_i}\{z_j^{(i)}\} := \max\{z_1^{(i)},\dots,z_{p_i}^{(i)}\}$ and we will refer to the resulting NN as a maxout NN. We will assume that the conditions $\boldsymbol{\Phi}(\mathbf{0}) = \mathbf{0}$ and $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathcal{U}$ hold for all $\boldsymbol{x} \in \mathcal{X}$. Note that these conditions are not restrictive. In fact, for box constraints of the form $\underline{\boldsymbol{u}} \leq \boldsymbol{u}(k) \leq \overline{\boldsymbol{u}}$, they can always be enforced by considering

$$\tilde{\boldsymbol{\Phi}}(\boldsymbol{x}) = -\max\{-\max\{\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{\Phi}(\mathbf{0}), \underline{\boldsymbol{u}}\}, -\overline{\boldsymbol{u}}\}$$

which is again a maxout NN with two additional layers and weights $\tilde{\boldsymbol{W}}^{(i)} = \boldsymbol{W}^{(i)}$, $\tilde{\boldsymbol{b}}^{(i)} = \boldsymbol{b}^{(i)}$ for all $i \in \{1,\dots,\ell\}$ and

$$\tilde{\boldsymbol{W}}^{(\ell+1)} = \begin{pmatrix} \boldsymbol{W}^{(\ell+1)} \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\boldsymbol{b}}^{(\ell+1)} = \begin{pmatrix} \boldsymbol{b}^{(\ell+1)} - \boldsymbol{\Phi}(\mathbf{0}) \\ \underline{\boldsymbol{u}} \end{pmatrix},$$
$$\tilde{\boldsymbol{W}}^{(\ell+2)} = \begin{pmatrix} -\boldsymbol{I}_m \\ \mathbf{0} \end{pmatrix}, \qquad \tilde{\boldsymbol{b}}^{(\ell+2)} = \begin{pmatrix} \mathbf{0} \\ -\overline{\boldsymbol{u}} \end{pmatrix},$$

$\tilde{\boldsymbol{W}}^{(\ell+3)} = -\boldsymbol{I}_m$, and $\tilde{\boldsymbol{b}}^{(\ell+3)} = \mathbf{0}$. Then, we have $\tilde{\boldsymbol{\Phi}}(\mathbf{0}) = \mathbf{0}$ and $\tilde{\boldsymbol{\Phi}}(\boldsymbol{x}) \in \mathcal{U}$ for all $\boldsymbol{x} \in \mathcal{X}$, as required. Thus, in the following, we focus on constraint satisfaction for the state constraints and assume that the input constraints hold whenever the state constraints hold. A similar method was used in [38, Prop. 1] for guaranteed input constraint satisfaction for ReLU NN-based control laws. Methods for input constraint satisfaction for arbitrary polyhedral regions $\mathcal{U}$ can be found in [15, Sec. III-B]. For analyzing a closed-loop system consisting of a PWA system and an NN-based controller, we need an MI formulation of the maxout NN similar to the MI feasibility problem (10.6). According to [P3, Lem. 2] the MI linear constraints

$$\boldsymbol{q}_l^{(i)}(k) \leq \boldsymbol{W}_j^{(i)}\boldsymbol{q}^{(i-1)}(k) + \boldsymbol{b}_j^{(i)} + \overline{b}^{(i)}(1-\boldsymbol{\delta}_j^{(i)}(k)), \tag{10.9a}$$
$$-\boldsymbol{q}_l^{(i)}(k) \leq -\boldsymbol{W}_j^{(i)}\boldsymbol{q}^{(i-1)}(k) - \boldsymbol{b}_j^{(i)} - \varepsilon(1-\boldsymbol{\delta}_j^{(i)}(k)), \tag{10.9b}$$
$$\sum_{j\in\mathcal{A}_l^{(i)}} \boldsymbol{\delta}_j^{(i)}(k) = 1, \tag{10.9c}$$
$$\forall j \in \mathcal{A}_l^{(i)},\ \forall l \in \{1,\dots,w_i\}, \forall i \in \{1,\dots,\ell\}, \tag{10.9d}$$

with $\mathcal{A}_l^{(i)} := \{p_i(l-1)+1,\dots,p_i l\}$ and $\boldsymbol{\delta}^{(i)}(k) \in \{0,1\}^{p_i w_i}$ are such that the output of the NN is $\boldsymbol{\Phi}(\boldsymbol{q}^{(0)}(k)) = \boldsymbol{W}^{(\ell+1)}\boldsymbol{q}^{(\ell)}(k) + \boldsymbol{b}^{(\ell+1)}$. Thus, the desired MI linear constraints are given by (10.9), and the output of a maxout NN can be computed by solving the MI feasibility problem

$$\text{find } \boldsymbol{q}^{(1)}(k),\dots,\boldsymbol{q}^{(\ell)}(k), \text{ and } \boldsymbol{\delta}^{(1)}(k),\dots,\boldsymbol{\delta}^{(\ell)}(k) \tag{10.10a}$$
$$\text{s.t.} \quad (10.9) \tag{10.10b}$$

with $k \in \{0,\dots,K-1\}$.

10.3 Closed-loop analysis of PWA systems with neural network-based controller

By combining the MI linear constraints (10.4), (10.5) and (10.9) for the PWA system dynamics and the maxout NN, respectively, we can describe the closed-loop system consisting of a PWA system and a maxout NN-based controller by the MI feasibility problem

$$\begin{aligned} \text{find } & \boldsymbol{X}_{K+1}, \boldsymbol{U}_K, \tilde{\boldsymbol{X}}_K^{(1)}, ..., \tilde{\boldsymbol{X}}_K^{(s)}, \gamma(0), ..., \gamma(K-1) && \text{(10.11a)} \\ & \boldsymbol{q}^{(1)}(0), \dots, \boldsymbol{q}^{(\ell)}(0), \dots, \boldsymbol{q}^{(1)}(K-1), \dots, \boldsymbol{q}^{(\ell)}(K-1) \\ & \boldsymbol{\delta}^{(1)}(0), \dots, \boldsymbol{\delta}^{(\ell)}(0), \dots, \boldsymbol{\delta}^{(1)}(K-1), \dots, \boldsymbol{\delta}^{(\ell)}(K-1) \\ \text{s.t.} \quad & (10.4), (10.5), (10.9), && \text{(10.11b)} \\ & \boldsymbol{x}(0) = \boldsymbol{x} && \text{(10.11c)} \\ & \boldsymbol{q}^{(0)}(k) = \boldsymbol{x}(k) && \text{(10.11d)} \\ & \boldsymbol{u}(k) = \boldsymbol{W}^{(\ell+1)} \boldsymbol{q}^{(\ell)}(k) + \boldsymbol{b}^{(\ell+1)} && \text{(10.11e)} \end{aligned}$$

for all $k \in \{0, \dots, K-1\}$, which leads to the following lemma.

Lemma 10.1. *Let $\boldsymbol{f}_{PWA}$ be as in* (10.1) *and let $\boldsymbol{\Phi}(\boldsymbol{x})$ be a maxout NN as in* (10.7)–(10.8)*. Then, any solution to the MI feasibility problem* (10.11) *is such that*

$$\boldsymbol{x}(k+1) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k) + \boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(k)) + \boldsymbol{p}^{(i)} \tag{10.12}$$

holds for all $k \in \{0, \dots, K-1\}$.

Proof. For $\boldsymbol{X}_{K+1}$ and $\boldsymbol{U}_K$ subject to (10.4) and (10.5) we have $\boldsymbol{x}(k+1) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k) + \boldsymbol{B}^{(i)} \boldsymbol{u}(k) + \boldsymbol{p}^{(i)}$ for all $k \in \{1, \dots, K-1\}$, with (10.11e) this leads to

$$\boldsymbol{x}(k+1) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k) + \boldsymbol{B}^{(i)} \left(\boldsymbol{W}^{(\ell+1)} \boldsymbol{q}^{(\ell)}(k) + \boldsymbol{b}^{(\ell+1)} \right) + \boldsymbol{p}^{(i)}.$$

Moreover, according to [P3, Lem. 2] we have $\boldsymbol{\Phi}(\boldsymbol{q}^{(0)}(k)) = \boldsymbol{W}^{(\ell+1)} \boldsymbol{q}^{(\ell)}(k) + \boldsymbol{b}^{(\ell+1)}$ for $\boldsymbol{q}^{(0)}(k), \dots, \boldsymbol{q}^{(\ell)}(k)$ that satisfy (10.9). With (10.11d) this finally gives us $\boldsymbol{x}(k+1) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k) + \boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(k)) + \boldsymbol{p}^{(i)}$ for all $k \in \{1, \dots, K-1\}$ and thus proves the claim. ■

Lemma 10.1 allows to describe a PWA system with a maxout NN as controller by MI linear constraints.

10.3.1 Reachability analysis

By adding a cost function to the MI feasibility problem (10.11), we can analyze the k-step reachable set of the closed-loop system by the MILP

$$
\begin{aligned}
c_k^*(\boldsymbol{v}, \mathcal{X}) := \max_{\substack{\boldsymbol{X}_{K+1}, \boldsymbol{U}_K, \tilde{\boldsymbol{X}}_K^{(1)}, \ldots, \tilde{\boldsymbol{X}}_K^{(s)}, \boldsymbol{\gamma}(0), \ldots, \boldsymbol{\gamma}(K-1) \\ \boldsymbol{q}^{(1)}(0), \ldots, \boldsymbol{q}^{(\ell)}(0), \ldots, \boldsymbol{q}^{(1)}(K-1), \ldots, \boldsymbol{q}^{(\ell)}(K-1) \\ \boldsymbol{\delta}^{(1)}(0), \ldots, \boldsymbol{\delta}^{(\ell)}(0), \ldots, \boldsymbol{\delta}^{(1)}(K-1), \ldots, \boldsymbol{\delta}^{(\ell)}(K-1)}} & \boldsymbol{v}\boldsymbol{x}(k) && \text{(10.13a)} \\
\text{s.t.} \quad & \text{(10.11b), (10.11d), (10.11e), and } \boldsymbol{x}(0) \in \mathcal{X} && \text{(10.13b)}
\end{aligned}
$$

with $\boldsymbol{v} \in \mathbb{R}^{1 \times n}$.

Lemma 10.2. *Consider the k-step reachable set*

$$
\begin{aligned}
\mathcal{R}_k(\mathcal{X}) := \{ \boldsymbol{x}(k) \in \mathbb{R}^n \mid \boldsymbol{x}(k) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k-1) + \\
\boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(k-1)) + \boldsymbol{p}^{(i)}, \boldsymbol{x}(k-1) \in \mathcal{R}_{k-1}(\mathcal{X}) \}
\end{aligned}
$$

with $\mathcal{R}_0(\mathcal{X}) := \mathcal{X}$ and let $c_k^*(\boldsymbol{v}, \mathcal{X})$ be as in (10.13)*. Then, the relation $c_k^*(\boldsymbol{v}, \mathcal{X}) = \boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{v})$ holds.*

Proof. According to Lemma 10.1, the constraints of the MILP (10.13) are such that we have $\boldsymbol{x}(k+1) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k) + \boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(k)) + \boldsymbol{p}^{(i)}$ for all $k \in \{1, \ldots, K-1\}$. Thus, the MILP becomes

$$
\begin{aligned}
c_k^*(\boldsymbol{v}, \mathcal{X}) = \max_{\boldsymbol{X}_{K+1}, \boldsymbol{U}_K} & \boldsymbol{v}\boldsymbol{x}(k) && \text{(10.14a)} \\
\text{s.t.} \quad & \boldsymbol{x}(0) \in \mathcal{X} = \mathcal{R}_0(\mathcal{X}) && \text{(10.14b)} \\
& \boldsymbol{x}(1) = \boldsymbol{A}^{(i)} \boldsymbol{x}(0) + \boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(0)) + \boldsymbol{p}^{(i)} && \text{(10.14c)} \\
& \boldsymbol{x}(2) = \boldsymbol{A}^{(i)} \boldsymbol{x}(1) + \boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(1)) + \boldsymbol{p}^{(i)} && \text{(10.14d)} \\
& \quad \vdots \\
& \boldsymbol{x}(k) = \boldsymbol{A}^{(i)} \boldsymbol{x}(k-1) + \boldsymbol{B}^{(i)} \boldsymbol{\Phi}(\boldsymbol{x}(k-1)) + \boldsymbol{p}^{(i)}.
\end{aligned}
$$

By definition of $\mathcal{R}_1(\mathcal{X})$, we can summarize the constraints (10.14b) and (10.14c) as $\boldsymbol{x}(1) \in \mathcal{R}_1(\mathcal{X})$. Then, the constraint (10.14d) and $\boldsymbol{x}(1) \in \mathcal{R}_1(\mathcal{X})$ can be summarized by $\boldsymbol{x}(2) \in \mathcal{R}_2(\mathcal{X})$. Continuing this way until the $(k+1)$-th constraint, we can summarize all constraints by $\boldsymbol{x}(k) \in \mathcal{R}_k(\mathcal{X})$. This results in

$$
\begin{aligned}
c_k^*(\boldsymbol{v}, \mathcal{X}) = \max_{\boldsymbol{x}(k)} & \boldsymbol{v}\boldsymbol{x}(k) && \text{(10.15a)} \\
\text{s.t.} \quad & \boldsymbol{x}(k) \in \mathcal{R}_k(\mathcal{X}), && \text{(10.15b)}
\end{aligned}
$$

which is the support function of $\mathcal{R}_k(\mathcal{X})$ evaluated for the row-vector $\boldsymbol{v}$, i.e., $c_k^*(\boldsymbol{v}, \mathcal{X}) = h_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{v})$. ■

This result can be used to compute polyhedral over-approximations of the k-step reachable set according to the following lemma.

Lemma 10.3. *Let $\mathcal{R}_k(\mathcal{X})$ be as in Lemma 10.2. Then, the set*

$$\overline{\mathcal{R}}_k(\mathcal{X}) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{c}\} \tag{10.16}$$

with $\boldsymbol{C} \in \mathbb{R}^{l \times n}$ and $\boldsymbol{c} := \begin{pmatrix} c_k^*(\boldsymbol{C}_1, \mathcal{X}) & \dots & c_k^*(\boldsymbol{C}_l, \mathcal{X}) \end{pmatrix}^\top$ is such that $\mathcal{R}_k(\mathcal{X}) \subseteq \overline{\mathcal{R}}_k(\mathcal{X})$ and $h_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}) = h_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C})$.

Proof. We first prove that $\mathcal{R}_k(\mathcal{X}) \subseteq \overline{\mathcal{R}}_k(\mathcal{X})$ holds, which is the case if and only if $\boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}) \leq \boldsymbol{c}$. According to Lemma 10.2, we have $c_k^*(\boldsymbol{C}_i, \mathcal{X}) = \boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}_i)$ for all $i \in \{1, \dots, l\}$ and thus $\boldsymbol{c} = \boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C})$ which implies $\boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}) \leq \boldsymbol{c}$. To prove $h_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}) = h_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C})$, we first note that

$$\begin{aligned} \boldsymbol{h}_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C}_i) = \max_{\boldsymbol{x} \in \overline{\mathcal{R}}_k(\mathcal{X})} \boldsymbol{C}_i\boldsymbol{x} &= \max_{\boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C})} \boldsymbol{C}_i\boldsymbol{x} \\ &\leq \max_{\boldsymbol{C}_i\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}_i)} \boldsymbol{C}_i\boldsymbol{x} \end{aligned}$$

holds and thus $\boldsymbol{h}_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C}_i) \leq \boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}_i)$. Moreover, due to $\mathcal{R}_k(\mathcal{X}) \subseteq \overline{\mathcal{R}}_k(\mathcal{X})$, the inequality $\boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}_i) \leq \boldsymbol{h}_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C}_i)$ holds. In combination, we find $\boldsymbol{h}_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}_i) = \boldsymbol{h}_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C}_i)$ for all $i \in \{1, \dots, l\}$. ■

While in general it is computationally demanding or even impossible to compute the exact k-step reachable set for a PWA system with an NN-based controller, Lemma 10.3 provides a way to compute over-approximations $\overline{\mathcal{R}}_k(\mathcal{X})$ of $\mathcal{R}_k(\mathcal{X})$ which are tight in the direction of the vectors $\boldsymbol{C}_i$. In other words, both sets have the same expansion in the direction of $\boldsymbol{C}_i$, since $h_{\mathcal{R}_k(\mathcal{X})}(\boldsymbol{C}_i) = h_{\overline{\mathcal{R}}_k(\mathcal{X})}(\boldsymbol{C}_i)$ holds.

The following corollary describes a feature of the over-approximation $\overline{\mathcal{R}}_1(\mathcal{X})$ that is frequently used in this paper.

Corollary 10.4. *Let $\overline{\mathcal{R}}_1(\mathcal{X})$ be defined as in* (10.16) *and assume that $\mathcal{A} \subseteq \mathcal{B}$ holds for two polyhedral sets $\mathcal{A} \subset \mathbb{R}^n$ and $\mathcal{B} \subset \mathbb{R}^n$. Then, $\overline{\mathcal{R}}_1(\mathcal{A}) \subseteq \overline{\mathcal{R}}_1(\mathcal{B})$ holds.*

Proof. By assumption, we have $\mathcal{A} \subseteq \mathcal{B} \Leftrightarrow \mathcal{A} \cap \mathcal{B} = \mathcal{A}$. In combination with (10.14) we can conclude that $c_k^*(\boldsymbol{C}_i, \mathcal{A}) = c_k^*(\boldsymbol{C}_i, \mathcal{A} \cap \mathcal{B}) \leq c_k^*(\boldsymbol{C}_i, \mathcal{B})$ holds for all $i \in \{1, \dots, l\}$. The latter is equivalent to $h_{\overline{\mathcal{R}}_1(\mathcal{A})}(\boldsymbol{C}) \leq h_{\overline{\mathcal{R}}_1(\mathcal{B})}(\boldsymbol{C}) = \boldsymbol{h}^{(\overline{\mathcal{R}}_1(\mathcal{B}))}$ and thus $\overline{\mathcal{R}}_1(\mathcal{A}) \subseteq \overline{\mathcal{R}}_1(\mathcal{B})$ holds. ■

In the remainder of this section, we describe methods that can be used to check whether a given set is positively invariant (PI) for the closed system (10.12) and with which PI sets with various desirable properties, e.g., small or large, can be computed.

Lemma 10.5. *Let $\overline{\mathcal{R}}_1(\mathcal{F})$ be defined as in* (10.16) *and assume that $\overline{\mathcal{R}}_1(\mathcal{F}) \subseteq \mathcal{F}$ holds for an arbitrary polyhedral set $\mathcal{F} \subset \mathbb{R}^n$. Then,*

(i) the set $\mathcal{F}$ is PI for the closed-loop system (10.12),

(ii) the set

$$\overline{\mathcal{R}_k(\mathcal{F})} := \underbrace{\overline{\mathcal{R}}_1(\overline{\mathcal{R}}_1(\dots\overline{\mathcal{R}}_1}_{k \text{ times}}(\mathcal{F}))) \tag{10.17}$$

is PI with $\mathcal{R}_k(\mathcal{F}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})} \subseteq \mathcal{F}$ *and* $\overline{\mathcal{R}}_1(\overline{\mathcal{R}_k(\mathcal{F})}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})}$ *for all* $k \in \mathbb{N}$.

Proof. (i) Since $\mathcal{R}_1(\mathcal{F}) \subseteq \overline{\mathcal{R}}_1(\mathcal{F})$ holds according to Lemma 10.3, we have $\mathcal{R}_1(\mathcal{F})$
$\subseteq \overline{\mathcal{R}}_1(\mathcal{F}) \subseteq \mathcal{F}$ and consequently $\mathcal{R}_1(\mathcal{F}) \subseteq \mathcal{F}$. By using the definition of $\mathcal{R}_1(\mathcal{F})$ we obtain $\mathcal{R}_1(\mathcal{F}) = \{\boldsymbol{x}^+ \in \mathbb{R}^n \mid \boldsymbol{x}^+ = \boldsymbol{A}^{(i)}\boldsymbol{x} + \boldsymbol{B}^{(i)}\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{p}^{(i)}, \boldsymbol{x} \in \mathcal{F}\} \subseteq \mathcal{F}$. Thus, for all $\boldsymbol{x} \in \mathcal{F}$, the successor state $\boldsymbol{x}^+$ is in the set $\mathcal{F}$, which proves that $\mathcal{F}$ is PI for the closed-loop system (10.12) (cf. [75, Sec. 3.2]).

(ii) We prove the claim by induction. For $k = 1$ we have $\overline{\mathcal{R}_1(\mathcal{F})} = \overline{\mathcal{R}}_1(\mathcal{F})$ by definition and thus $\overline{\mathcal{R}}_1(\overline{\mathcal{R}_1(\mathcal{F})}) = \overline{\mathcal{R}}_1(\overline{\mathcal{R}}_1(\mathcal{F})) \subseteq \overline{\mathcal{R}}_1(\mathcal{F}) = \overline{\mathcal{R}_1(\mathcal{F})} \subseteq \mathcal{F}$, where we used that $\overline{\mathcal{R}_1(\mathcal{F})} = \overline{\mathcal{R}}_1(\mathcal{F}) \subseteq \mathcal{F}$ holds by assumption and thus $\overline{\mathcal{R}}_1(\overline{\mathcal{R}}_1(\mathcal{F})) \subseteq \overline{\mathcal{R}}_1(\mathcal{F})$ holds according to Corollary 10.4. Consequently, $\overline{\mathcal{R}_1(\mathcal{F})}$ is according to (i) PI and $\mathcal{R}_1(\mathcal{F}) \subseteq \overline{\mathcal{R}}_1(\mathcal{F}) = \overline{\mathcal{R}_1(\mathcal{F})} \subseteq \mathcal{F}$ holds. Thus, the induction hypothesis $\overline{\mathcal{R}}_1(\overline{\mathcal{R}_k(\mathcal{F})}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})}$ and $\mathcal{R}_k(\mathcal{F}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})} \subseteq \mathcal{F}$ hold for one $k = i$. For $k = i + 1$ we have

$$\begin{aligned}\overline{\mathcal{R}}_1(\overline{\mathcal{R}_{i+1}(\mathcal{F})}) &= \overline{\mathcal{R}}_1\left(\overline{\mathcal{R}}_1(\overline{\mathcal{R}_i(\mathcal{F})})\right)\\ &\subseteq \overline{\mathcal{R}}_1(\overline{\mathcal{R}_i(\mathcal{F})}) = \overline{\mathcal{R}_{i+1}(\mathcal{F})}.\end{aligned}$$

Moreover, we have $\overline{\mathcal{R}_{i+1}(\mathcal{F})} = \overline{\mathcal{R}}_1(\overline{\mathcal{R}_i(\mathcal{F})}) \subseteq \overline{\mathcal{R}}_1(\mathcal{F}) \subseteq \mathcal{F}$ and $\mathcal{R}_{i+1}(\mathcal{F}) = \mathcal{R}_1(\mathcal{R}_i(\mathcal{F})) \subseteq \overline{\mathcal{R}}_1(\mathcal{R}_i(\mathcal{F})) \subseteq \overline{\mathcal{R}}_1(\overline{\mathcal{R}_i(\mathcal{F})}) = \overline{\mathcal{R}_{i+1}(\mathcal{F})}$ by the induction hypothesis and Corollary 10.4. In combination with the proof of (i), we can conclude that $\overline{\mathcal{R}_k(\mathcal{F})}$ is a PI set with $\mathcal{R}_k(\mathcal{F}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})} \subseteq \mathcal{F}$ and $\overline{\mathcal{R}}_1(\overline{\mathcal{R}_k(\mathcal{F})}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})}$ for all $k \in \mathbb{N}$. ■

Lemma 10.5, allows computing potentially smaller PI sets $\overline{\mathcal{R}_k(\mathcal{F})}$ that are contained in a given larger PI set $\mathcal{F}$ and can be reached from all $\boldsymbol{x} \in \mathcal{F}$ in at most k time steps, since $\mathcal{R}_k(\mathcal{F}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})}$ holds. Note that, for linear systems, there always exits a k such that $\mathcal{R}_k(\mathcal{F}_{\max}) \subseteq \mathcal{F}_{\min}$ holds if $\mathcal{F}_{\max}$ and $\mathcal{F}_{\min}$ are PI sets with $\mathcal{F}_{\min} \subseteq \mathcal{F}_{\max}$. Thus, $\mathcal{F}_{\min}$ is reachable for all $\boldsymbol{x} \in \mathcal{F}_{\max}$. However, for PWA systems, there may exist PI sets with $\mathcal{F}_{\min} \subseteq \mathcal{F}_{\max}$ for which $\mathcal{F}_{\min}$ is not reachable from all $\boldsymbol{x} \in \mathcal{F}_{\max}$, i.e., $\mathcal{R}_k(\mathcal{F}_{\max}) \not\subseteq \mathcal{F}_{\min}$ for all k. Therefore, $\mathcal{R}_k(\mathcal{F}) \subseteq \overline{\mathcal{R}_k(\mathcal{F})}$ in Lemma 10.5 (ii) is crucial and does not follow from the fact that $\overline{\mathcal{R}_k(\mathcal{F})}$ and $\mathcal{F}$ are PI sets with $\overline{\mathcal{R}_k(\mathcal{F})} \subseteq \mathcal{F}$. In addition, using $\overline{\mathcal{R}_k(\mathcal{F})}$ has several computational advantages over $\overline{\mathcal{R}}_k(\mathcal{F})$. The set $\overline{\mathcal{R}_k(\mathcal{F})}$ is guaranteed to be PI if $\mathcal{F}$ is PI. While $\overline{\mathcal{R}}_k(\mathcal{F})$ may not be PI for some k even if $\mathcal{F}$ is PI. More importantly, $\overline{\mathcal{R}_k(\mathcal{F})}$ relies on repeatedly computing over-approximations of one-step reachable sets. This is computationally more efficient than computing the over-approximation of the k-step reachable set $\overline{\mathcal{R}}_k(\mathcal{F})$.

10.3.2 Convergence to a set containing the origin

We first aim for the computation of a large PI set $\mathcal{F}_{\max}$ with $\mathcal{F}_{\max} \subseteq \mathcal{X}$ and thus $\boldsymbol{\Phi}(\boldsymbol{x}) \subseteq \mathcal{U}$ for all $\boldsymbol{x} \in \mathcal{F}_{\max}$, which will serve as feasible set for the NN-based controller. Ideally, $\mathcal{F}_{\max}$ should be the maximal PI set, which is the union of all sets $\mathcal{F}$ satisfying $\mathcal{R}_1(\mathcal{F}) \subseteq \mathcal{F}$. However, an exact computation of the maximal PI set for (10.12) requires knowledge of all affine segments and polyhedral sets of the PWA function represented by the NN [74, Alg. 4.1]. Computing these affine segments and sets is computationally intractable even for small NN. Therefore, the following algorithm provides a method for computing a polyhedral inner approximation of the maximum PI set.

Algorithm 10.6. *Compute a large PI set $\mathcal{F}_{max} \subseteq \mathcal{X}$.*

1: *initialize $\mathcal{F} \leftarrow \mathcal{X}$ and $\boldsymbol{C} \leftarrow \boldsymbol{H}^{(\mathcal{X})}$*
2: ***while*** $\overline{\mathcal{R}}_1(\mathcal{F}) \not\subseteq \mathcal{F}$ ***do***
3: *set $\mathcal{F} \leftarrow \overline{\mathcal{R}}_1(\mathcal{F}) \cap \mathcal{X}$*
4: *set $\mathcal{F}_{max} \leftarrow \mathcal{F}$*
5: ***return*** $\mathcal{F}_{max}$

In contrast to [74, Alg. 4.1], we use here in line 3 of the algorithm the over-approximation (10.17) of the one-step reachable set, which allows the application to PWA systems of the form (10.12).

Proposition 10.7. *Let $\overline{\mathcal{R}_k(\mathcal{X})}$ be defined as in* (10.17) *and assume that there exists a $\bar{k}$ such that $\overline{\mathcal{R}_{\bar{k}+1}(\mathcal{X})} \subseteq \overline{\mathcal{R}_{\bar{k}}(\mathcal{X})} \subseteq \mathcal{X}$ holds. Then, Algorithm 10.6 terminates with a PI set $\mathcal{F}_{max} \subseteq \mathcal{X}$ after at most $\bar{k}$ iterations.*

Proof. The set computed in the $(k+1)$-th iteration of Algorithm 10.6 is $\mathcal{F}_{k+1} = \overline{\mathcal{R}}_1(\mathcal{F}_k) \cap \mathcal{X}$ with $\mathcal{F}_0 = \mathcal{X}$. We first note, that by construction $\mathcal{F}_{k+1} \subseteq \mathcal{F}_k \subseteq \mathcal{X}$ and $\mathcal{F}_k \subseteq \overline{\mathcal{R}_k(\mathcal{X})}$ hold for all $k \in \mathbb{N}$. Thus, we have $\overline{\mathcal{R}}_1(\mathcal{F}_{\bar{k}}) \subseteq \overline{\mathcal{R}}_1(\overline{\mathcal{R}_{\bar{k}}(\mathcal{X})}) = \overline{\mathcal{R}_{\bar{k}+1}(\mathcal{X})} \subseteq \overline{\mathcal{R}_{\bar{k}}(\mathcal{X})} \subseteq \mathcal{X}$, which implies $\overline{\mathcal{R}}_1(\mathcal{F}_{\bar{k}}) = \overline{\mathcal{R}}_1(\mathcal{F}_{\bar{k}}) \cap \mathcal{X} = \mathcal{F}_{\bar{k}+1}$. Consequently, $\overline{\mathcal{R}}_1(\mathcal{F}_{\bar{k}}) = \mathcal{F}_{\bar{k}+1} \subseteq \mathcal{F}_{\bar{k}} \subseteq \mathcal{X}$ holds, which proves that $\mathcal{F}_{\bar{k}}$ is a PI set contained in $\mathcal{X}$. Thus Algorithm 10.6 will terminate after at most $\bar{k}$ iterations, since $\overline{\mathcal{R}}_1(\mathcal{F}_{\bar{k}}) \subseteq \mathcal{F}_{\bar{k}}$ holds. ■

Remark 10.8. *The condition from Proposition 10.7, which guarantees finite termination of Algorithm 10.6, requires that a PI and reachable set $\overline{\mathcal{R}_{\bar{k}}(\mathcal{X})}$ exists that is contained in $\mathcal{X}$. Surely, the set $\overline{\mathcal{R}_{\bar{k}}(\mathcal{X})}$ could also be used as feasible set for the NN-based controller. However, since $\bar{k}$ is an upper bound, the Algorithm 10.6 will typically terminate in less than $\bar{k}$ iterations with a PI set $\mathcal{F}_{max}$ that is larger than $\overline{\mathcal{R}_{\bar{k}}(\mathcal{X})}$.*

With Proposition 10.7, we can guarantee constraint satisfaction of the closed-loop system with NN-based controller (10.12) for all trajectories starting from $\mathcal{F}_{\max}$, since $\mathcal{F}_{\max}$ is a PI set contained in $\mathcal{X}$. Moreover, by assumption, we then further have $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathcal{U}$. To show that the closed-loop system with NN-based controller reaches and remains in a possibly small set containing the origin for

all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$, we will next state a result that allows the computation of small PI sets.

Proposition 10.9. *Let $\overline{\mathcal{R}_k(\mathcal{F}_{max})}$ be defined as in (10.17) with $\mathcal{F}_{max}$ being the PI set computed with Algorithm 10.6 and assume that there exists a k^* such that*

$$\frac{1}{1+\epsilon}\overline{\mathcal{R}_{k^*}(\mathcal{F}_{max})} \subseteq \overline{\mathcal{R}}_1\left(\frac{1}{1+\epsilon}\overline{\mathcal{R}_{k^*}(\mathcal{F}_{max})}\right) \tag{10.18}$$

holds for a given $\epsilon > 0$. Then, the set

$$\mathcal{F}_{min} := \overline{\mathcal{R}_{k^*}(\mathcal{F}_{max})} \tag{10.19}$$

is PI with

$$\overline{\mathcal{R}_\infty(\mathcal{F}_{max})} \subseteq \overline{\mathcal{R}_{k^*}(\mathcal{F}_{max})} \subseteq (1+\epsilon)\overline{\mathcal{R}_\infty(\mathcal{F}_{max})}, \tag{10.20}$$

where $\overline{\mathcal{R}_\infty(\mathcal{F}_{max})}$ is short for $\lim_{k\to\infty}\overline{\mathcal{R}_k(\mathcal{F}_{max})}$.

Proof. Since $\overline{\mathcal{R}}_1(\mathcal{F}_{\max}) \subseteq \mathcal{F}_{\max}$ holds for $\mathcal{F}_{\max}$ computed with Algorithm 10.6, the set $\mathcal{F}_{\min} = \overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})}$ is PI for all $k^* \in \mathbb{N}$ according Lemma 10.5 (ii). It remains to prove that the inclusion (10.20) holds. The left-hand side of the inclusion holds since we have $\overline{\mathcal{R}_{k+1}(\mathcal{F}_{\max})} = \overline{\mathcal{R}}_1(\overline{\mathcal{R}_k(\mathcal{F}_{\max})}) \subseteq \overline{\mathcal{R}_k(\mathcal{F}_{\max})}$ for all $k \in \mathbb{N}$ as apparent from Lemma 10.5 (ii) and thus $\overline{\mathcal{R}_\infty(\mathcal{F}_{\max})} \subseteq \overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})}$. For the right-hand side, we prove that $\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} \subseteq (1+\epsilon)\overline{\mathcal{R}_{k^*+k}(\mathcal{F}_{\max})}$ holds for all $k \in \mathbb{N}_0$. For $k = 0$ we have $\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} \subseteq (1+\epsilon)\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})}$. Thus, $\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} \subseteq (1+\epsilon)\overline{\mathcal{R}_{k^*+i}(\mathcal{F}_{\max})}$ holds for one $i \in \mathbb{N}_0$. For $k = i+1$ we obtain

$$\begin{aligned}\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} &\subseteq (1+\epsilon)\overline{\mathcal{R}}_1\left(\frac{1}{1+\epsilon}\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})}\right)\\ &\subseteq (1+\epsilon)\overline{\mathcal{R}}_1(\overline{\mathcal{R}_{k^*+i}(\mathcal{F}_{\max})})\\ &= (1+\epsilon)\overline{\mathcal{R}_{k^*+i+1}(\mathcal{F}_{\max})}\end{aligned}$$

by using assumption (10.18), the induction hypothesis, and Corollary 10.4. Which proves that $\overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} \subseteq (1+\epsilon)\overline{\mathcal{R}_{k^*+k}(\mathcal{F}_{\max})}$ holds for all $k \in \mathbb{N}_0$. In summary, we have $\overline{\mathcal{R}_\infty(\mathcal{F}_{\max})} \subseteq \overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} \subseteq (1+\epsilon)\lim_{k\to\infty}\overline{\mathcal{R}_{k^*+k}(\mathcal{F}_{\max})}$, which completes the proof. ■

Remark 10.10. *Condition (10.18) has a rather intuitive interpretation. According Lemma 10.5 (ii) we have $\overline{\mathcal{R}}_1(\overline{\mathcal{R}_k(\mathcal{F}_{max})}) \subseteq \overline{\mathcal{R}_k(\mathcal{F}_{max})}$ for all $k \in \mathbb{N}$ and thus the reachable sets $\overline{\mathcal{R}_k(\mathcal{F}_{max})}$ usually becomes smaller when k grows and finally converge to $\overline{\mathcal{R}_\infty(\mathcal{F}_{max})}$. However, if the reachable sets starting from $\mathcal{F}$ grow, i.e., if we have $\mathcal{F} \subseteq \overline{\mathcal{R}}_1(\mathcal{F})$, then this implies $\mathcal{F} \subseteq \overline{\mathcal{R}_\infty(\mathcal{F}_{max})}$. Thus condition (10.18) states that k^* is such that a small scaling of the set $\overline{\mathcal{R}_{k^*}(\mathcal{F}_{max})}$ with $1/(1+\epsilon)$ leads to a growing reachable set (cf. right-hand side of (10.18)) and thus $1/(1+\epsilon)\overline{\mathcal{R}_{k^*}(\mathcal{F}_{max})} \subseteq \overline{\mathcal{R}_\infty(\mathcal{F}_{max})}$, which is exactly what we aim for. For the limit case $\epsilon \to 0$ condition (10.18) becomes $\overline{\mathcal{R}_k(\mathcal{F}_{max})} \subseteq \overline{\mathcal{R}}_1(\overline{\mathcal{R}_k(\mathcal{F}_{max})})$ which is equivalent to $\overline{\mathcal{R}_k(\mathcal{F}_{max})} =$*

$\overline{\mathcal{R}_{k+1}(\mathcal{F}_{max})} = \overline{\mathcal{R}_{\infty}(\mathcal{F}_{max})}$. *This means that for* $\epsilon \to 0$ *we need finite determinedness of* $\overline{\mathcal{R}_{\infty}(\mathcal{F}_{max})}$ *or* $k^* \to \infty$ *for* (10.18) *to hold. For the other limit case* $\epsilon \to \infty$, *the left-hand side of* (10.18) *becomes a set only containing the origin. Moreover, since* $\boldsymbol{\Phi}(\mathbf{0}) = \mathbf{0}$ *and* $\boldsymbol{f}_{PWA}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$ *holds, the right-hand side of* (10.18) *also becomes a set only containing the origin and thus* (10.18) *holds for all* $k^* \geq 0$. *This means that with* ϵ, *we can influence the number of computation steps needed to compute* (10.19).

We now combine the previous results to state the main contribution of this section.

Theorem 10.11. *Let* $\mathcal{F}_{max}$ *be the set computed with Algorithm 10.6 and* $\mathcal{F}_{min}$ *by defined as in Proposition 10.9. Then, the system* (10.1) *with* $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ *is ultimately bounded in* $\mathcal{F}_{min}$, *uniformly in* $\mathcal{F}_{max}$. *Moreover,* $\boldsymbol{x}(k) \in \mathcal{X}$ *and* $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ *holds for all* $k \in \mathbb{N}_0$.

Proof. Since $\mathcal{F}_{\max}$ is a PI set for the closed-loop system (10.12), we have for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$ that $\boldsymbol{x}(k) \in \mathcal{F}_{\max} \subseteq \mathcal{X}$ holds for all $k \in \mathbb{N}$. Moreover, by assumption we have $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathcal{U}$ for all $\boldsymbol{x} \in \mathcal{X}$ and thus $\boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ for all $k \in \mathbb{N}_0$. According to Lemma 10.5 (ii) and Proposition 10.9, $\mathcal{F}_{\min}$ is a PI set with $\mathcal{R}_{k^*}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} = \mathcal{F}_{\min}$. Thus, $\boldsymbol{x}(k) \in \mathcal{R}_{k^*}(\mathcal{F}_{\max}) \subseteq \mathcal{F}_{\min}$ holds for all $k \geq k^*$ and for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$. This proves, that the system (10.1) with $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ is ultimately bounded in $\mathcal{F}_{\min}$, uniformly in $\mathcal{F}_{\max}$. ■

10.3.3 Stability of the origin

Consider the modified dual-mode control law

$$\boldsymbol{\pi}(\boldsymbol{x}) := \begin{cases} \boldsymbol{\kappa}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in s\mathcal{F}_0, \\ \boldsymbol{\Phi}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{F}_{\max} \setminus s\mathcal{F}_0, \end{cases} \tag{10.21}$$

where $\mathcal{F}_{\max}$ is the PI set computed with Algorithm 10.6 and $\boldsymbol{\kappa}(\boldsymbol{x})$ is a stabilizing PWA control law with region of attraction $\mathcal{F}_0 := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{x} \leq \xi^*\}$ for the PWA system (10.1). The set $\mathcal{F}_0$ is such that ξ^* is the largest ξ for which $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{x} \leq \xi, \xi > 0\} \subseteq \cup_{i \in \mathcal{I}_0} \mathcal{P}^{(i)}$ holds. In addition, $0 < s \leq 1$ is a scaling factor that is chosen such that $\mathcal{F}_{\min} \subseteq s\mathcal{F}_0$ holds. The control law $\boldsymbol{\kappa}(\boldsymbol{x})$, $\boldsymbol{S}$, and $\mathcal{F}_0$ can be computed by, e.g., solving the linear matrix inequality (LMI) described in [101, Eq. (9)-(10)]. Then $V(\boldsymbol{x}) := \boldsymbol{x}^\top \boldsymbol{S}\boldsymbol{x}$ is a common Lyapuov function for the system (10.1) with $\boldsymbol{u} = \boldsymbol{\kappa}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \cup_{i \in \mathcal{I}_0} \mathcal{P}^{(i)}$. Thus the set $s\mathcal{F}_0$ is PI (see [101, Sec. 3.2]).

Theorem 10.12. *Let* $\boldsymbol{\pi}(\boldsymbol{x})$ *and* $\mathcal{F}_{min}$ *be defined as in* (10.21) *and Proposition 10.9, respectively and assume that* $\mathcal{F}_{min} \subseteq s\mathcal{F}_0$ *with* $0 < s \leq 1$ *holds. Then, the origin is asymptotically stable for the system* (10.1) *with* $\boldsymbol{u}(k) = \boldsymbol{\pi}(\boldsymbol{x}(k))$. *The region of attraction is* $\mathcal{F}_{max}$.

Proof. We distinguish the two cases $\boldsymbol{x} \in s\mathcal{F}_0$ and $\boldsymbol{x} \in \mathcal{F}_{\max} \setminus s\mathcal{F}_0$. In the first case, we have $\boldsymbol{u} = \boldsymbol{\pi}(\boldsymbol{x}) = \boldsymbol{\kappa}(\boldsymbol{x})$, which asymptotically stabilizes the origin

of the PWA system while ensuring $\boldsymbol{x}(k) \in s\mathcal{F}_0 \subseteq \mathcal{X}$ for all $k \in \mathbb{N}_0$ ([101, Lem. 1]). To prove asymptotic stability for states $\boldsymbol{x} \in \mathcal{F}_{\max} \setminus s\mathcal{F}_0$ it remains to show that trajectories starting in $\boldsymbol{x} \in \mathcal{F}_{\max} \setminus s\mathcal{F}_0$ enter $s\mathcal{F}_0$ after a finite number of time-steps by applying $\boldsymbol{u} = \boldsymbol{\pi}(\boldsymbol{x}) = \boldsymbol{\Phi}(\boldsymbol{x})$. By assumption we have $\mathcal{R}_{k^*}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}_{k^*}(\mathcal{F}_{\max})} = \mathcal{F}_{\min} \subseteq s\mathcal{F}_0$. Thus, for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max} \subseteq \mathcal{X}$ there exist a $\tilde{k} \leq k^*$ such that $\boldsymbol{x}(\tilde{k}) \in s\mathcal{F}_0$ holds when applying $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ for all $k < \tilde{k}$. Since $s\mathcal{F}_0$ is PI for (10.1) with $\boldsymbol{u} = \boldsymbol{\kappa}(\boldsymbol{x})$ we have $\boldsymbol{x}(k) \in s\mathcal{F}_0$ and consequently $\boldsymbol{u}(k) = \boldsymbol{\kappa}(\boldsymbol{x}(k))$ for all $k \geq \tilde{k}$, which asymptotically stabilizes the system. In summary, the control law (10.21) asymptotically stabilizes the system (10.1) for all $\boldsymbol{x} \in \mathcal{F}_{\max}$. ■

The parameter s is a design parameter which can be used to scale the set in which the controller $\boldsymbol{\kappa}(\boldsymbol{x})$ acts. Typically, we want to have $\boldsymbol{\pi}(\boldsymbol{x}) = \boldsymbol{\Phi}(\boldsymbol{x})$ as long as possible and thus choose $s = \min_{\mathcal{F}_{\min} \subseteq c\mathcal{F}_0} c$, which is the smallest s that satisfies the conditions of Theorem 10.12.

10.4 Case study

Consider the OCP (10.2) with $\boldsymbol{Q} = \boldsymbol{P} = \boldsymbol{I}$, $\boldsymbol{R} = 1$ and $N = 10$ for a PWA system of the form (10.1) with

$$\boldsymbol{A}^{(1)} = \begin{pmatrix} -0.04 & -0.461 \\ -0.139 & 0.341 \end{pmatrix}, \boldsymbol{A}^{(2)} = \begin{pmatrix} 0.936 & 0.323 \\ 0.788 & -0.049 \end{pmatrix},$$
$$\boldsymbol{A}^{(3)} = \begin{pmatrix} -0.857 & 0.815 \\ 0.491 & 0.62 \end{pmatrix}, \boldsymbol{A}^{(4)} = \begin{pmatrix} -0.022 & 0.644 \\ 0.758 & 0.271 \end{pmatrix},$$
$$\boldsymbol{B}^{(1)} = \boldsymbol{B}^{(2)} = \boldsymbol{B}^{(3)} = \boldsymbol{B}^{(4)} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top,$$

and $\boldsymbol{H}^{(1)} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \boldsymbol{H}^{(2)} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$,
$\boldsymbol{H}^{(3)} = -\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(4)} = -\boldsymbol{H}^{(2)}$, as well as $\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \boldsymbol{h}^{(3)} = \boldsymbol{h}^{(4)} = \boldsymbol{0}$ from [101, Sec. 7] with the additional constraints $\|\boldsymbol{x}(k)\|_\infty \leq 10$ for all $k \in \{0, \dots, N\}$ and $|u(k)| \leq 1$ for all $k \in \{0, \dots, N-1\}$. We solved the OCP for 1000 randomly sampled $\boldsymbol{x} \in \mathcal{X}$ to generate a data set with the feasible points $(\boldsymbol{x}^{(i)}, \boldsymbol{f}_{\text{MPC}}(\boldsymbol{x}^{(i)}))$. This data set is used to train a 3×3 maxout NN, i.e., $\ell = 3$ and $w_i = 3$ for all $i \in \{1, 2, 3\}$. Figure 10.1 illustrates trajectories of the closed-loop system with the resulting NN-based controller and different PI sets. In (a), the outer box $[-10, 10] \times [-10, 10]$ represents the state constraints $\mathcal{X}$. The next smaller box $[-7.15, 8.76] \times [-10, 8.37]$ is $\mathcal{F}_{\max}$ in which the gray arrows in (a) and (b) represent the evolution of the closed-loop system with NN-based controller, i.e., $\boldsymbol{u} = \boldsymbol{\Phi}(\boldsymbol{x})$. The set $\mathcal{F}_{\max}$ is computed according to Algorithm 10.6. The computation terminates after 1 iteration, whereas we have $\overline{k} = 5 \geq 1$, which is in line with Remark 10.8. Figure 10.1 (b) shows a detailed view of a region around the origin of (a). The smallest box $[-0.25, 0.32] \times [-0.33, 0.27]$ around

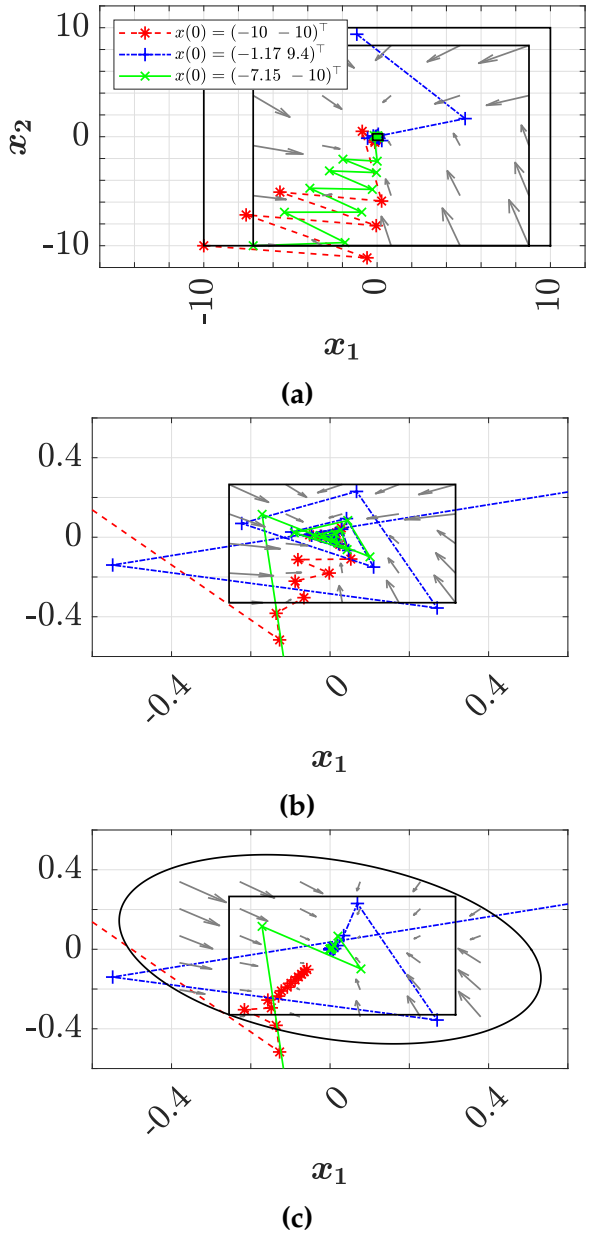


Figure 10.1. Trajectories of the closed-loop system with the NN-based controller and the sets $\mathcal{X}$, $\mathcal{F}_{\text{max}}$, and $\mathcal{F}_{\text{min}}$, respectively. The initial states of the trajectories are specified in the legend and apply to all figures.

the origin is $\mathcal{F}_{\text{min}}$. This set is computed according to Proposition 10.9 with $\epsilon = 10^{-3}$, which results in $k^* = 81$. Therefore, we have $\mathcal{F}_{\text{min}} = \overline{\mathcal{R}_{81}(\mathcal{F}_{\text{max}})}$ with $\overline{\mathcal{R}_\infty(\mathcal{F}_{\text{max}})} \subseteq \overline{\mathcal{R}_{81}(\mathcal{F}_{\text{max}})} \subseteq (1 + 10^{-3})\overline{\mathcal{R}_\infty(\mathcal{F}_{\text{max}})}$. The Figures 10.1 (a) and (b) confirm Theorem 10.11, since it can be observed that the trajectory starting in $\mathcal{F}_{\text{max}}$, marked with green x, remains in that set and enters $\mathcal{F}_{\text{min}}$ after a finite number of time steps (cf. (b)). For the trajectory, marked with red asterisk, we have $\boldsymbol{x}(0) \notin \mathcal{F}_{\text{max}}$, which leads to a constraint violation of $\boldsymbol{x}(1)$. However, since $\mathcal{F}_{\text{max}}$ is only an inner approximation of the maximum PI set contained in $\mathcal{X}$, there also exist trajectories (blue plus) with $\boldsymbol{x}(0) \notin \mathcal{F}_{\text{max}}$, that enter $\mathcal{F}_{\text{min}}$

while satisfying state and input constraints. The results of Section 10.3.3 are illustrated in Figure 10.1 (c). There three trajectories with the same $\boldsymbol{x}(0)$ as in (a) are shown for the system (10.1) with $\boldsymbol{u} = \boldsymbol{\pi}(\boldsymbol{x})$. The ellipse is the region $s\mathcal{F}_0$ with $s = 5.32 \times 10^{-2}$, in which, according to (10.21), the control law $\boldsymbol{u} = \boldsymbol{\kappa}(\boldsymbol{x})$ is applied and asymptotically stabilizes the system. The gray arrows represent the evolution of the corresponding closed-loop system. For the green trajectory, which starts in $\mathcal{F}_{\text{max}}$, we can guarantee that it enters $s\mathcal{F}_0$ after at most k^* time steps and converges to the origin.

10.5 Conclusions

We presented various methods for analyzing PWA systems with an NN-based controller. The combination of the methods leads to the main results of the paper, Theorems 10.11 and 10.12, which allow certifying stability and constraint satisfaction based on PI sets. The proposed approaches all build on on the computation of over-approximations of k-step reachable sets. This fact indicates a possible direction for future research. In fact, it may be feasible to replace the MILP (10.13) with another OP without binary variables that can be solved faster at the cost of a more conservative over-approximation. Such a modification would not affect the validity of the results since the results of Section 10.3 hold independent of the method used to compute the over-approximations.

Chapter 11

A mixed-integer framework for analyzing neural network-based controllers for piecewise affine systems with bounded disturbances*

Dieter Teichrib† and Moritz Schulze Darup†

Abstract. We present a method for representing the closed-loop dynamics of piecewise affine (PWA) systems with bounded additive disturbances and neural network-based controllers as mixed-integer (MI) linear constraints. We show that such representations enable the computation of robustly positively invariant (RPI) sets for the specified system class by solving MI linear programs. These RPI sets can subsequently be used to certify stability and constraint satisfaction. Furthermore, the approach allows to handle nonlinear systems based on suitable PWA approximations and corresponding error bounds, which can be interpreted as the bounded disturbances from above.

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

11.1 Introduction

Piecewise affine (PWA) systems are an important system class in control theory as they can be used to model a large class of hybrid systems or to approximate nonlinear systems [65]. When PWA models are used to describe systems that are inherently PWA as, e.g., mixed logical dynamical (MLD) systems [64], no error has to be considered. In these cases, analysis of the closed-loop system and control design can be performed directly using techniques developed for PWA system [101], [132], [133], [P6] or MLD systems [64]. However, when a PWA model approximates a nonlinear system, an approximation error has to be considered when analysing the system behaviour and designing a controller. In [136], an adaption of tube-based model predictive control (MPC) [137] is introduced, which is applicable to PWA with bounded additive disturbances. The resulting optimal control problem (OCP) is a mixed-integer program (MIP). As common in MPC, the OCP is solved in every time step, and only the first element of the optimal input sequence is applied to the system. Repeatedly solving the OCP in each time step within the sampling period of the system may be intractable for a long prediction horizon or PWA systems with many regions. Therefore, methods have been developed in [112], [6], [P5] where an approximate solution of the OCP is used instead of the exact one. Neural networks (NN) are often used in this context (see, e.g, [6, 40]), as they are fast to evaluate, and there exist several software libraries ([138, 139]) that allow an efficient training on large data sets. However, guarantees for stability and constraint satisfaction of the original MPC do typically not hold for the approximated NN-based controller. Fortunately, in recent years, some methods to certify stability and constraint satisfaction for systems with NN-based controllers have been developed, most of them for linear systems [13, 39] and recently for PWA systems without disturbances [P6]. These approaches cannot be applied to PWA systems with bounded disturbances and NN-based controllers.

In this paper, we present a method based on MIP that can be used to certify stability and constraint satisfaction for PWA systems with bounded additive disturbances and NN-based controllers. Methods based on MIP are also considered in, e.g., [13] or [38]. However, these methods either require a stabilizing base-line controller or are only applicable to linear systems. Our method builds on the computation of polyhedral over-approximations of the k-step reachable set as in [P6] and, therefore, does not require a stabilizing base-line controller. These over-approximations are subsequently used to compute different robustly positively invariant (RPI) sets for certifying stability and constraint satisfaction of the closed-loop system. Furthermore, we investigate under which conditions the RPI sets for the PWA system with bounded additive disturbance allow us to show that a nonlinear system converges to a small positively invariant (PI) set while satisfying state and input constraints.

The paper is organized as follows. In the remainder of this section, we state

notation and some basic definitions. In Section 11.2, we give some fundamentals on mixed-integer (MI) based reachability analysis of PWA systems with NN-based controllers. The MI-based reachability analysis is extended in Section 11.3 to PWA systems with bounded additive disturbances, allowing us to handle PWA approximations of nonlinear systems. In Section 11.4, we present the main contributions of our paper, which include the computation of RPI sets for PWA systems with bounded additive disturbances and NN-based controller as well as results on stability and constraint satisfaction. Two illustrative case studies are given in Section 11.5. Finally, the paper is concluded in Section 11.6. To improve the readability of the paper, the proofs of all lemmas are in the appendix.

11.1.1 Notation and basic definitions

We define the support function of a polyhedron $\mathcal{X}$ for a row-vector $\boldsymbol{v} \in \mathbb{R}^{1\times n}$ as $h_{\mathcal{X}}(\boldsymbol{v}) := \sup_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{v}\boldsymbol{x}$. For a matrix argument $\boldsymbol{H} \in \mathbb{R}^{w\times n}$, $h_{\mathcal{X}}(\boldsymbol{H})$ is understood as $h_{\mathcal{X}}(\boldsymbol{H}) = \begin{pmatrix} h_{\mathcal{X}}(\boldsymbol{H}_1) & \dots & h_{\mathcal{X}}(\boldsymbol{H}_w) \end{pmatrix}^\top$, where $\boldsymbol{H}_i$ refers to the i-th row of $\boldsymbol{H}$. The scaling of a set $\mathcal{A}$ by a scalar s is defined as $s\mathcal{A} := \{s\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in \mathcal{A}\}$. For compact and convex sets, the operation $\mathcal{A} \oplus \mathcal{B} := \{\boldsymbol{a} + \boldsymbol{b} \mid \boldsymbol{a} \in \mathcal{A},\ \boldsymbol{b} \in \mathcal{B}\}$ is the Minkowski sum. Natural numbers and natural numbers, including 0, are denoted by $\mathbb{N}$ and $\mathbb{N}_0$, respectively. The operator $\text{col}(\boldsymbol{x}(j)_{j=K_1}^{K_2})$ stacks the column vectors $\boldsymbol{x}(j)$ with $j \in \{K_1, \dots, K_2\}$ in a single column vector. We further define $d(\boldsymbol{z}, \mathcal{X}) := \inf_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{z} - \boldsymbol{x}\|_2$ as the distance of a point to a set and $\mathcal{B}_\delta := \{\boldsymbol{x} \in \mathbb{R}^n \mid d(\boldsymbol{x}, \boldsymbol{0}) \leq \delta\}$ as a 2-norm ball with radius $\delta > 0$. Moreover, we sometimes use the short-hand notation $\boldsymbol{0}$ for a set containing only the origin.

11.2 Fundamentals on mixed-integer based reachability analysis

In this paper, we consider discrete-time PWA systems

$$\boldsymbol{x}(k+1) = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{u}(k)) + \boldsymbol{d}(k) \tag{11.1}$$

with

$$\boldsymbol{f}_{\text{PWA}} := \boldsymbol{A}^{(i)}\boldsymbol{x}(k) + \boldsymbol{B}^{(i)}\boldsymbol{u}(k) + \boldsymbol{p}^{(i)} \text{ if } \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k) \end{pmatrix} \in \mathcal{P}^{(i)}.$$

with $i \in \{1, \dots, s\}$. The additive disturbance $\boldsymbol{d}(k)$ is assumed to be bounded by state and input-dependent polyhedral sets, i.e., $\boldsymbol{d}(k) \in \mathcal{D}^{(i)} \subset \mathbb{R}^n$. The polyhedral sets

$$\mathcal{P}^{(i)} := \{\boldsymbol{\xi} \in \mathbb{R}^{n+m} \mid \boldsymbol{H}^{(i)}\boldsymbol{\xi} \leq \boldsymbol{h}^{(i)}\} \tag{11.2}$$

with $\boldsymbol{H}^{(i)} \in \mathbb{R}^{d_i\times(n+m)}$ and $\boldsymbol{h}^{(i)} \in \mathbb{R}^{d_i}$ for all $i \in \{1, \dots, s\}$ are pairwise disjoint, i.e, $\text{int}(\mathcal{P}^{(i)}) \cap \text{int}(\mathcal{P}^{(j)}) = \emptyset$ for all $i \neq j$, and partition the state $\mathcal{X} \subset \mathbb{R}^n$ and

input space $\mathcal{U} \subset \mathbb{R}^m$ according to $\mathcal{X} \times \mathcal{U} = \cup_{i=1}^{s} \mathcal{P}^{(i)}$ where s is the number of polyhedral sets. We aim to develop methods for computing large, robustly positively invariant (RPI) sets contained in $\mathcal{X}$, in which the controlled system (11.1) with an NN-based controller

$$\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k)) \tag{11.3}$$

can be operated safely for all time. Where safe means that the state and input constraints are satisfied. Moreover, we aim to compute small RPI sets to which the controlled system converges. The methods mainly build on the k-step reachable set

$$\begin{aligned}\mathcal{R}_k^{\mathcal{D}}(\mathcal{F}) := \{\boldsymbol{x}^+ \in \mathbb{R}^n \mid \boldsymbol{x}^+ = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x})) + \boldsymbol{d}, \\ \boldsymbol{x} \in \mathcal{R}_{k-1}^{\mathcal{D}}(\mathcal{F}), \boldsymbol{d} \in \mathcal{D}^{(i)}\}\end{aligned} \tag{11.4}$$

with $\mathcal{R}_0^{\mathcal{D}}(\mathcal{F}) := \mathcal{F}$ or over-approximations of this set. Based on the k-step reachable set (11.4), we can define an RPI set as follows.

Definition 11.1 ([100, Def. 1]). *A Set $\mathcal{F} \subseteq \mathbb{R}^n$ is RPI if and only if*

$$\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{F}. \tag{11.5}$$

To compute the reachable sets (11.4) via mixed-integer programming (MIP), we will summarize in the following subsections MI linear constraints that allow a description of NN-based controllers (11.3) and PWA systems without disturbances, i.e., $\boldsymbol{d}(k) = \boldsymbol{0}$. This description is extended to the case $\boldsymbol{d}(k) \in \mathcal{D}^{(i)}$ in Section 11.3.

11.2.1 MI formulation of neural networks

We consider feed-forward-NN of the form

$$\boldsymbol{\Phi}(\boldsymbol{x}) := \boldsymbol{f}^{(\ell+1)} \circ \boldsymbol{g}^{(\ell)} \circ \boldsymbol{f}^{(\ell)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{x}) \tag{11.6}$$

where ℓ is the number of hidden layers and w_i the number of neurons in layer i. Here, the functions $\boldsymbol{f}^{(i)} : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{p_i w_i}$ for $i \in \{1, \dots, \ell\}$ refer to preactivations, where the parameter $p_i \in \mathbb{N}$ allows to consider "multi-channel" preactivations as required for maxout activation (see [21]). Moreover, $\boldsymbol{g}^{(i)} : \mathbb{R}^{p_i w_i} \to \mathbb{R}^{w_i}$ stand for activation functions and $\boldsymbol{f}^{(\ell+1)} : \mathbb{R}^{w_\ell} \to \mathbb{R}^{w_{\ell+1}}$ reflects postactivation. We assume affine functions $\boldsymbol{f}^{(i)}$, i.e.,

$$\boldsymbol{f}^{(i)}(\boldsymbol{y}^{(i-1)}) := \boldsymbol{W}^{(i)} \boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}, \tag{11.7}$$

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i w_i \times w_{i-1}}$ is a weighting matrix, $\boldsymbol{b}^{(i)} \in \mathbb{R}^{p_i w_i}$ is a bias vector, and $\boldsymbol{y}^{(i-1)}$ denotes the output of the previous layer with $\boldsymbol{y}^{(0)} := \boldsymbol{x} \in \mathbb{R}^n$. Various choices for the activation functions have been proposed. However, since we

aim for a mixed-integer formulation of the NN, we focus on PWA activation functions. More precisely, we consider the maxout activation function

$$g^{(i)}(z^{(i)}) := \begin{pmatrix} \max\limits_{1\leq j\leq p_i} \{z_j^{(i)}\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1\leq j\leq p_i w_i} \{z_j^{(i)}\} \end{pmatrix}, \tag{11.8}$$

since the maxout activation is more general than other PWA activation functions and includes, e.g., ReLU or leaky ReLU, as special cases. We refer to the resulting NN as a maxout NN. We assume that $\boldsymbol{\Phi}(x) \in \mathcal{U}$ holds for all $x \in \mathcal{X}$. Note that for polyhedral state constraints as considered here, a given NN can always be modified to satisfy the required condition by adding layers to the NN that perform a projection onto the input constraints. These layers can be constructed using methods from, e.g., [15, Sec. III-B] or [P6, Sec. II-B]. For analyzing a PWA system with an NN-based controller, we need a description of the maxout NN in terms of MI linear constraints. According to [P3, Lem. 2] the MI linear constraints

$$q_l^{(i)}(k) \leq W_j^{(i)} q^{(i-1)}(k) + b_j^{(i)} + \overline{b}^{(i)}(1-\delta_j^{(i)}(k)), \tag{11.9a}$$

$$-q_l^{(i)}(k) \leq -W_j^{(i)} q^{(i-1)}(k) - b_j^{(i)}, \tag{11.9b}$$

$$\sum_{j\in\mathcal{A}_l^{(i)}} \delta_j^{(i)}(k) = 1, \tag{11.9c}$$

$$\forall j \in \mathcal{A}_l^{(i)},\ \forall l \in \{1,\dots,w_i\}, \forall i \in \{1,\dots,\ell\}, \tag{11.9d}$$

with $\mathcal{A}_l^{(i)} := \{p_i(l-1)+1,\dots,p_i l\}$ and $\delta^{(i)}(k) \in \{0,1\}^{p_i w_i}$ are such that the output of the NN is $\boldsymbol{\Phi}(q^{(0)}(k)) = W^{(\ell+1)} q^{(\ell)}(k) + b^{(\ell+1)}$. Thus, the desired MI linear constraints are given by (11.9).

11.2.2 MI formulation of PWA systems without disturbances

In this section, we briefly summarize known methods from [47] for constructing MI linear constraints for PWA systems of the form (11.1) with $d(k) = 0$. The first step is to introduce the MI linear constraints

$$H^{(i)} \begin{pmatrix} x(k) \\ u(k) \end{pmatrix} \leq h^{(i)} + \mathbf{1}M(1-\gamma_i(k)) \tag{11.10a}$$

$$\mathbf{1}^\top \gamma(k) = 1, \quad \gamma(k) \in \{0,1\}^s \tag{11.10b}$$

for all $i \in \{1,\dots,s\}$ and all $k \in \{0,\dots,K-1\}$ with $K \in \mathbb{N}$ and $\gamma(k) \in \{0,1\}^s$, which are according to [47] such that

$$\gamma_i(k) = 1 \Leftrightarrow \begin{pmatrix} x^\top(k) & u^\top(k) \end{pmatrix}^\top \in \mathcal{P}^{(i)} \tag{11.11}$$

holds. The constant M in the constraints (11.10) is often referred to as big-M and can be chosen according to [47, Eq. (20)-(21)]. Now, the binary variable $\gamma(k)$ can be used to describe the PWA system dynamics (11.1) with $\boldsymbol{d}(k) = \mathbf{0}$ by the MI linear constraints

$$\begin{aligned} -\mathbf{1}M(1-\gamma_i(k)) \leq \boldsymbol{A}^{(i)}\boldsymbol{x}(k)+\boldsymbol{B}^{(i)}\boldsymbol{u}(k)+\boldsymbol{p}^{(i)}-\tilde{\boldsymbol{x}}^{(i)}(k+1) \\ \leq \mathbf{1}M(1-\gamma_i(k)) \end{aligned} \tag{11.12a}$$

$$-\mathbf{1}M\gamma_i(k) \leq \tilde{\boldsymbol{x}}^{(i)}(k) \leq \mathbf{1}M\gamma_i(k) \tag{11.12b}$$

$$\boldsymbol{x}(k+1) = \sum_{j=1}^{s} \tilde{\boldsymbol{x}}^{(j)}(k+1) \tag{11.12c}$$

for all $i \in \{1,\dots,s\}$ and all $k \in \{0,\dots,K-1\}$ (cf. [47, Sec. 3.1]), which are such that $\boldsymbol{x}(k+1) = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}(k),\boldsymbol{u}(k))$ holds for all $k \in \{0,\dots,K-1\}$.

11.3 Mixed-integer linear constraints for PWA systems with bounded disturbances

For computing reachable sets for (11.1) by mixed-integer linear programs (MILP), we extend the MI linear constraints (11.12) to systems with state and input-dependent disturbances of the form

$$\boldsymbol{d}(k) \in \mathcal{D}^{(i)} \text{ if } \begin{pmatrix}\boldsymbol{x}(k)^\top & \boldsymbol{u}(k)^\top\end{pmatrix}^\top \in \mathcal{P}^{(i)}. \tag{11.13}$$

This type of systems can be described by the MI linear constraints (11.10), (11.12b), (11.12c), and

$$\begin{aligned} -\mathbf{1}M(1-\gamma_i(k)) \leq \boldsymbol{A}^{(i)}\boldsymbol{x}(k) + \boldsymbol{B}^{(i)}\boldsymbol{u}(k) + \boldsymbol{p}^{(i)} + \boldsymbol{d}(k) \\ - \tilde{\boldsymbol{x}}^{(i)}(k+1) \leq \mathbf{1}M(1-\gamma_i(k)) \end{aligned} \tag{11.14a}$$

$$\boldsymbol{H}^{(\mathcal{D}^{(i)})}\boldsymbol{d}(k) \leq \boldsymbol{h}^{(\mathcal{D}^{(i)})} + \mathbf{1}M(1-\gamma_i(k)). \tag{11.14b}$$

Lemma 11.2. *Consider the system dynamics* (11.1) *with* (11.13)*, a maxout NN* $\boldsymbol{\Phi}(\boldsymbol{x})$ *as in* (11.6)–(11.8)*, and define* $\boldsymbol{X}_{K+1} = col(\boldsymbol{x}(j)_{j=0}^{K})$*,* $\boldsymbol{U}_K = col(\boldsymbol{u}(j)_{j=0}^{K-1})$*,* $\boldsymbol{D}_K = col(\boldsymbol{d}(j)_{j=0}^{K-1})$*, and* $\tilde{\boldsymbol{X}}_K^{(i)} = col(\tilde{\boldsymbol{x}}^{(i)}(j)_{j=1}^{K})$ *for all* $i \in \{1,\dots,s\}$*. Then, any solution to the MI feasibility problem*

$$\begin{aligned} \textit{find}\, \boldsymbol{X}_{K+1},\boldsymbol{U}_K,\boldsymbol{D}_K,\tilde{\boldsymbol{X}}_K^{(1)},\dots,\tilde{\boldsymbol{X}}_K^{(s)},\gamma(0),\dots,\gamma(K-1) \\ \boldsymbol{q}^{(1)}(0),\dots,\boldsymbol{q}^{(\ell)}(0),\dots,\boldsymbol{q}^{(1)}(K-1),\dots,\boldsymbol{q}^{(\ell)}(K-1) \\ \boldsymbol{\delta}^{(1)}(0),\dots,\boldsymbol{\delta}^{(\ell)}(0),\dots,\boldsymbol{\delta}^{(1)}(K-1),\dots,\boldsymbol{\delta}^{(\ell)}(K-1) \end{aligned} \tag{11.15a}$$

$$\textit{s.t. } (11.9), (11.10), (11.12b), (11.12c), (11.14), \tag{11.15b}$$

$$\boldsymbol{q}^{(0)}(k) = \boldsymbol{x}(k), \tag{11.15c}$$

$$\boldsymbol{u}(k) = \boldsymbol{W}^{(\ell+1)}\boldsymbol{q}^{(\ell)}(k) + \boldsymbol{b}^{(\ell+1)} \tag{11.15d}$$

for all $k \in \{0, \dots, K-1\}$, *is such that*

$$\boldsymbol{x}(k+1) = \boldsymbol{f}_{PWA}(\boldsymbol{x}(k), \boldsymbol{\Phi}(\boldsymbol{x}(k))) + \boldsymbol{d}(k), \tag{11.16a}$$

$$\boldsymbol{d}(k) \in \mathcal{D}^{(i)}, \tag{11.16b}$$

hold for all $k \in \{0, \dots, K-1\}$ *and for all* $\boldsymbol{x}(0) \in \mathbb{R}^n$.

For OP where the PWA system dynamics (11.16a) with bounded disturbance (11.16b) is in the constraints, Lemma 11.2 can be used to replace the nonlinear constraints (11.16a)-(11.16b) by the MI linear constraints (11.15b)-(11.15d), thus transforming the nonlinear OP to an MILP that can be solved with standard software, e.g., [45]. This can be used to formulate a MILP for the evaluation of the support function of the k-step reachable set for the PWA system (11.1) with (11.3), i.e. a PWA system with bounded disturbance and an NN-based controller.

Lemma 11.3. *Let the* k*-step reachable set be defined as in* (11.4)*. Then the MILP*

$$c_k^*(\boldsymbol{v}, \mathcal{X}) := \max_{\substack{\boldsymbol{X}_{k+1}, \boldsymbol{U}_k, \boldsymbol{D}_k, \tilde{\boldsymbol{X}}_k^{(1)}, \dots, \tilde{\boldsymbol{X}}_k^{(s)}, \boldsymbol{\gamma}(0), \dots, \boldsymbol{\gamma}(k-1) \\ \boldsymbol{q}^{(1)}(0), \dots, \boldsymbol{q}^{(\ell)}(0), \dots, \boldsymbol{q}^{(1)}(k-1), \dots, \boldsymbol{q}^{(\ell)}(k-1) \\ \boldsymbol{\delta}^{(1)}(0), \dots, \boldsymbol{\delta}^{(\ell)}(0), \dots, \boldsymbol{\delta}^{(1)}(k-1), \dots, \boldsymbol{\delta}^{(\ell)}(k-1)}} \boldsymbol{v}\boldsymbol{x}(k) \tag{11.17a}$$

$$\text{s.t.} \quad \text{(11.15b)–(11.15d)} \tag{11.17b}$$

with $\boldsymbol{v} \in \mathbb{R}^{1 \times n}$, *is such that* $c_k^*(\boldsymbol{v}, \mathcal{X}) = \boldsymbol{h}_{\mathcal{R}_k^{\mathcal{D}}(\mathcal{X})}(\boldsymbol{v})$ *holds.*

Based on the support function $\boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{X})}$ of the 1-step reachable set $\mathcal{R}_1^{\mathcal{D}}(\mathcal{X})$, we define an over-approximation

$$\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{X}) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{C}\boldsymbol{x} \leq \boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{X})}(\boldsymbol{C})\} \tag{11.18}$$

with $\boldsymbol{C} \in \mathbb{R}^{l \times n}$, which is according to [P6, Lem. 3] such that $\mathcal{R}_1^{\mathcal{D}}(\mathcal{X}) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{X})$ and $\boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{X})}(\boldsymbol{C}) = \boldsymbol{h}_{\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{X})}(\boldsymbol{C})$ hold. The hyperplanes in the matrix $\boldsymbol{C}$ are a design parameter. A typical choice are hyperplanes that lead to a hypercube, i.e., $\boldsymbol{C} = (\boldsymbol{I} \ -\boldsymbol{I})^\top$. By using Lemma 11.3, the set (11.18) can be computed by solving l-times the MILP (11.17). Moreover, with Lemma 11.3, it is possible to check whether a given polyhedral set $\mathcal{F}$ is RPI. According to Definition 11.1, checking if a given set is RPI requires verifying the condition $\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{F}$. For a polyhedral set $\mathcal{F}$ with l hyperplanes, (11.5) holds if and only if $\boldsymbol{h}_{\mathcal{R}_1^{\mathcal{D}}(\mathcal{F})}(\boldsymbol{H}^{(\mathcal{F})}) \leq \boldsymbol{h}^{(\mathcal{F})}$, which can be verified by solving l-times the MILP (11.17).

11.4 Analysis of PWA systems with bounded disturbances and NN-based controllers

The analysis in this section is based on the over-approximation

$$\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{F}) := \underbrace{\overline{\mathcal{R}}_1^{\mathcal{D}}(\overline{\mathcal{R}}_1^{\mathcal{D}}(\dots \overline{\mathcal{R}}_1^{\mathcal{D}}}_{k \text{ times}}(\mathcal{F}))) \tag{11.19}$$

with $\overline{\mathcal{R}}_0^{\mathcal{D}}(\mathcal{F}) := \mathcal{F}$ of the k-step reachable set (11.4). According to [P6, Lem. 5] and [P6, Cor. 4] the set $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{F})$ is such that

$$\mathcal{R}_k^{\mathcal{D}}(\mathcal{A}) \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{A}) \text{ for all } k \in \mathbb{N}_0 \text{ and} \tag{11.20a}$$

$$\overline{\mathcal{R}}_1(\mathcal{A}) \subseteq \overline{\mathcal{R}}_1(\mathcal{B}) \text{ for } \mathcal{A} \subseteq \mathcal{B}. \tag{11.20b}$$

These two features are crucial for the following results and are often used in the proofs. The over-approximation (11.19) builds on the repeated computation of the one-step reachable set. Thus, evaluating the support function of $\overline{\mathcal{R}}_{\hat{k}}^{\mathcal{D}}(\mathcal{F})$ requires to solve the MILP (11.17) with $k = 1$ and $N_1 := s + \sum_{i=1}^{\ell} p_i w_i$ binary variables $\hat{k}$-times. Whereas evaluating the support function of $\mathcal{R}_{\hat{k}}^{\mathcal{D}}(\mathcal{F})$ requires solving a larger MILP (11.17) with $k = \hat{k}$ and thus $\hat{k}N_1$ binary variables only once. Meaning the computational complexity of evaluating the support function of $\overline{\mathcal{R}}_{\hat{k}}^{\mathcal{D}}(\mathcal{F})$ grows linearly with $\hat{k}$, whereas for $\mathcal{R}_{\hat{k}}^{\mathcal{D}}(\mathcal{F})$ it grows in the worst case exponentially with $\hat{k}$, due to the increasing number of binary variables. Thus, for computational reasons, all results are formulated for the over-approximation (11.19) of the k-step reachable set (11.4). However, since the following proofs apply to sets satisfying the conditions (11.20), the results still hold if $\overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{F})$ is replaced by the exact k-step reachable set $\mathcal{R}_k^{\mathcal{D}}(\mathcal{F})$.

11.4.1 Computation of a safe set

We start with computing a large RPI set in which the closed-loop system can be operated safely, i.e., without violating any constraints, for all time. The largest set with the required specifications is the maximum RPI set. However, this set cannot be computed using the techniques introduced in Section 11.3. Therefore, inspired by [P6, Alg. 1] we define the following set

$$\mathcal{F}_{k+1} = \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_k) \cap \mathcal{X} \text{ with } \mathcal{F}_0 := \mathcal{X}. \tag{11.21}$$

Proposition 11.4. *Let $\mathcal{F}_k$ be as in* (11.21) *and assume that there exists a k^* such that $\overline{\mathcal{R}}_{k^*+1}^{\mathcal{D}}(\mathcal{X}) \subseteq \mathcal{X}$. Then, there exist a $\hat{k} \leq k^*$ such that $\mathcal{F}_{max} := \mathcal{F}_{\hat{k}}$ is a RPI set with $\mathcal{F}_{max} \subseteq \mathcal{X}$.*

Proof. We first note that $\mathcal{F}_k \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{X})$ and $\mathcal{F}_{k+1} \subseteq \mathcal{F}_k \subseteq \mathcal{X}$ for all $k \in \mathbb{N}_0$. Thus we have

$$\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_{k^*}) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}(\overline{\mathcal{R}}_{k^*}^{\mathcal{D}}(\mathcal{X})) = \overline{\mathcal{R}}_{k^*+1}^{\mathcal{D}}(\mathcal{X}) \subseteq \mathcal{X} \tag{11.22}$$

which implies

$$\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}_{k^*}) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_{k^*}) = \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_{k^*}) \cap \mathcal{X} = \mathcal{F}_{k^*+1} \subseteq \mathcal{F}_{k^*}. \tag{11.23}$$

As apparent from (11.22) and (11.23) $\mathcal{F}_{k^*}$ is RPI if $\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_{k^*}) \subseteq \mathcal{X}$ holds. Thus the condition $\overline{\mathcal{R}}_{k^*+1}^{\mathcal{D}}(\mathcal{X}) \subseteq \mathcal{X}$ is only sufficient for $\mathcal{F}_{k^*}$ to be RPI, i.e., there may exist a $\hat{k} \leq k^*$ for which $\mathcal{F}_{\hat{k}}$ is RPI. ■

According to Proposition 11.4, a large RPI set can be computed by iterating (11.21) until $\overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_k) \subseteq \mathcal{F}_k$ holds and thus $\mathcal{F}_k$ is RPI. This is the case after, at most, k^* iterations.

11.4.2 Computation of a terminal set

The next step is to compute a small terminal set, which is RPI and to which all trajectories starting in $\mathcal{F}_{\max}$ converge. For stable linear systems, a set with the required specifications is given by the minimal RPI set $\mathcal{R}_{\min} := \overline{\mathcal{R}}_\infty^{\mathcal{D}}(\mathbf{0})$, which is typically not finitely determined. Thus, often RPI over-approximations $\underline{\mathcal{R}}_{\min} := (1+\underline{\epsilon})\overline{\mathcal{R}}_{\underline{k}}^{\mathcal{D}}(\mathbf{0})$ with $\underline{\epsilon} > 0$ are considered, where $\underline{k}$ is chosen depending on $\underline{\epsilon}$ such that $\underline{\mathcal{R}}_{\min}$ is RPI and an over-approximation of the minimal RPI set (cf. [100]). However, the set $\underline{\mathcal{R}}_{\min}$ can only be used as terminal set if there exists a k with $\mathcal{R}_k^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \underline{\mathcal{R}}_{\min}$, which is not guaranteed for PWA systems. Thus, there may exist $\underline{\mathcal{R}}_{\min}$ that are not reachable for the closed-loop system from $\mathcal{F}_{\max}$. Therefore, we alternatively consider the set

$$\overline{\mathcal{R}}_{\min} := \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \tag{11.24}$$

where $\bar{k}$ is chosen so that

$$\frac{1}{1+\bar{\epsilon}}\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}\left(\frac{1}{1+\bar{\epsilon}}\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})\right) \tag{11.25}$$

holds for a given $\bar{\epsilon} > 0$. Then, according to [P6, Prop. 7], $\overline{\mathcal{R}}_{\min}$ is an RPI over-approximation of $\overline{\mathcal{R}}_\infty^{\mathcal{D}}(\mathcal{F}_{\max})$ with

$$\overline{\mathcal{R}}_\infty^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_{\min} \subseteq (1+\bar{\epsilon})\overline{\mathcal{R}}_\infty^{\mathcal{D}}(\mathcal{F}_{\max}).$$

Moreover, since $\boldsymbol{x}(k) \in \mathcal{R}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) = \overline{\mathcal{R}}_{\min}$ holds for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$ and $k \geq \bar{k}$, all trajectories starting in $\mathcal{F}_{\max}$ will enter and remain in $\overline{\mathcal{R}}_{\min}$ after at most $\bar{k}$ time steps. Thus the set $\overline{\mathcal{R}}_{\min}$ is reachable for the closed-loop system for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$.

11.4.3 Stability of the closed-loop system

Now, we combine the previous results to state one of the main results of this section that proves stability of the set $\overline{\mathcal{R}}_{\min}$ for the PWA system with bounded disturbance (11.1) and NN-based controller (11.3). Where stability of a set is defined based on [137, Sec. 1] as follows: A set $\mathcal{T}$ is (Lyapunov) stable if for every $\epsilon > 0$, there exists a $\delta > 0$ such that, if $d(\boldsymbol{x}(0), \mathcal{T}) < \delta$, then $d(\boldsymbol{x}(k), \mathcal{T}) < \epsilon$ for all $k \in \mathbb{N}_0$. If $\mathcal{T}$ is stable and $\lim_{k\to\infty} d(\boldsymbol{x}(k), \mathcal{T}) = 0$ holds for all $\boldsymbol{x}(0) \in \mathcal{F}$, then $\mathcal{T}$ is asymptotically stable with region of attraction $\mathcal{F}$.

Theorem 11.5. *Let $\mathcal{F}_{max}$ and $\overline{\mathcal{R}}_{min}$ by defined as in Proposition 11.4 and (11.24), respectively and assume that*

$$\alpha\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{max}) \subseteq \overline{\mathcal{R}}_{\bar{k}-1}^{\mathcal{D}}(\mathcal{F}_{max}) \tag{11.26}$$

holds for $\alpha > 1$. Then, the set $\overline{\mathcal{R}}_{min}$ is asymptotically stable for the system (11.1) with $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$. The region of attraction is $\mathcal{F}_{max}$. Moreover, $\boldsymbol{x}(k) \in \mathcal{X}$ and $\boldsymbol{u}(k) \in \mathcal{U}$ holds for all $k \in \mathbb{N}_0$ with $\boldsymbol{x}(0) \in \mathcal{F}_{max}$.

Proof. Since $\mathcal{F}_{\max} \subseteq \mathcal{X}$ is an RPI set for the closed-loop system (11.16a), we have for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$ that $\boldsymbol{x}(k) \in \mathcal{F}_{\max} \subseteq \mathcal{X}$ holds for all $k \in \mathbb{N}$. Moreover, by assumption we then have $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathcal{U}$ for all $\boldsymbol{x} \in \mathcal{X}$ and thus $\boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ for all $k \in \mathbb{N}_0$. To prove asymptotic stability, we first define $\delta_{\max} := \max \delta$ subject to $\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta \subseteq \alpha\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})$ and choose $\delta = \min\{\epsilon, \delta_{\max}\}$ with $\epsilon > 0$. Note that $\delta_{\max}$ can be explicitly computed using the support function and is given by $\delta_{\max} = (\alpha - 1)\min_{\boldsymbol{\eta}} h_{\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})}(\boldsymbol{\eta})$. For polyhedral $\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})$ with hyperplanes $\boldsymbol{C}$, $\delta_{\max}$ simplifies to $\delta_{\max} = (\alpha - 1)\min_i h_{\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})}(\boldsymbol{C}_i)$. Since $\alpha > 1$ and $\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})$ has, due to $\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\boldsymbol{0}) \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})$, a nonempty interior, we have $\delta_{\max} > 0$. We further have

$$\overline{\mathcal{R}}_k^{\mathcal{D}}\left(\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta\right) \subset \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\epsilon \tag{11.27}$$

for all $k \in \mathbb{N}$, for all $\epsilon > 0$ and $\delta = \min\{\epsilon, \delta_{\max}\}$. For $k = 1$, (11.27) holds due to

$$\begin{aligned}&\overline{\mathcal{R}}_1^{\mathcal{D}}\left(\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta\right) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}(\alpha\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})) \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})\\ \subset&\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\epsilon\end{aligned}$$

for all $\epsilon > 0$ and $\delta = \min\{\epsilon, \delta_{\max}\} \leq \epsilon$. Thus, (11.27) holds for one i. For $k = i + 1$ we have

$$\begin{aligned}&\overline{\mathcal{R}}_1^{\mathcal{D}}\left(\overline{\mathcal{R}}_i^{\mathcal{D}}\left(\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta\right)\right) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}\left(\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta\right)\\ \subset&\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\epsilon\end{aligned}$$

for all $\epsilon > 0$. As a result, (11.27) holds for all $k \in \mathbb{N}$. Now, if $d(\boldsymbol{x}(0), \overline{\mathcal{R}}_{\min}) < \delta$, then $\boldsymbol{x}(0) \in \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta$. Consequently, according to (11.27) we have $\boldsymbol{x}(k) \in \mathcal{R}_k^{\mathcal{D}}\left(\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta\right) \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}\left(\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\delta\right) \subset \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \oplus \mathcal{B}_\epsilon$ and thus $d(\boldsymbol{x}(k), \overline{\mathcal{R}}_{\min}) < \epsilon$ for all $k \in \mathbb{N}$ and for all $\epsilon > 0$. For $k = 0$ we directly obtain $d(\boldsymbol{x}(0), \overline{\mathcal{R}}_{\min}) < \delta \leq \epsilon$. Thus, in summary for every $\epsilon > 0$ there exist a $\delta = \min\{\epsilon, \delta_{\max}\}$ such that if $d(\boldsymbol{x}(0), \overline{\mathcal{R}}_{\min}) < \delta$, then $d(\boldsymbol{x}(k), \overline{\mathcal{R}}_{\min}) < \epsilon$ for all $k \in \mathbb{N}_0$. This proves stability of the system. Since $\mathcal{R}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) = \overline{\mathcal{R}}_{\min}$ hold, we further have $\boldsymbol{x}(k) \in \overline{\mathcal{R}}_{\min}$ for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$ and for all $k \geq \bar{k}$. Therefore $\lim_{k\to\infty} d(\boldsymbol{x}(k), \overline{\mathcal{R}}_{\min}) = 0$ for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$ and thus $\overline{\mathcal{R}}_{\min}$ is asymptotic stable with region of attraction $\mathcal{F}_{\max}$. ■

11.4.4 Analysis of nonlinear systems

The proposed approach for reachability analysis can be used to analyze nonlinear systems that can be approximated by PWA systems with bounded additive disturbances. Therefore, we define the k-step reachable set of the nonlinear system

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \tag{11.28}$$

with an NN-based controller as in (11.3) as follows

$$\mathcal{R}_k(\mathcal{F}) := \{\boldsymbol{x}^+ \in \mathbb{R}^n \mid \boldsymbol{x}^+ = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\Phi}(\boldsymbol{x})), \boldsymbol{x} \in \mathcal{R}_{k-1}(\mathcal{F})\}.$$

with $\mathcal{R}_0(\mathcal{F}) = \mathcal{F}$ and state a result that relates the reachable sets of (11.28) and the reachable sets of (11.1).

Lemma 11.6. *Assume $\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{F} \subseteq \mathcal{X}$ and that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{f}_{PWA}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{D}^{(i)}$ holds for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$. Then $\mathcal{R}_k(\mathcal{F}) \subseteq \mathcal{R}_k^{\mathcal{D}}(\mathcal{F})$ for all $k \in \mathbb{N}_0$.*

Combining this lemma with the results stated in the sections 11.4.1 and 11.4.2 shows that the nonlinear system (11.28) is uniformly ultimately bounded (UUB) in the sense of [99, Def. 2.4]: A system is denoted as ultimately bounded in the C-set $\mathcal{T}$ (i.e., a convex and compact set containing the origin in its interior), uniformly in $\mathcal{F}$, if for every initial condition $\boldsymbol{x}(0) \in \mathcal{F}$, there exits a $k^*(\boldsymbol{x}(0))$ such that $\boldsymbol{x}(k) \in \mathcal{T}$ holds for all $k \geq k^*(\boldsymbol{x}(0))$.

Theorem 11.7. *Let $\mathcal{F}_{max}$ and $\overline{\mathcal{R}}_{min}$ by defined as in Proposition 11.4 and (11.24), respectively and assume that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{f}_{PWA}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{D}^{(i)}$ holds for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$. Then, the system (11.28) with $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ is ultimately bounded in $\overline{\mathcal{R}}_{min}$, uniformly in $\mathcal{F}_{max}$. Moreover, $\boldsymbol{x}(k) \in \mathcal{X}$ and $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ holds for all $k \in \mathbb{N}_0$.*

Proof. Since we have $\mathcal{R}_1^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \mathcal{F}_{\max} \subseteq \mathcal{X}$ and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{f}_{\mathrm{PWA}}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{D}^{(i)}$ for all $\boldsymbol{x} \in \mathcal{X}$ and for all $\boldsymbol{u} \in \mathcal{U}$ Lemma 11.6 applies here. In combination with (11.20a) this results in $\mathcal{R}_k(\mathcal{F}_{\max}) \subseteq \mathcal{R}_k^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_k^{\mathcal{D}}(\mathcal{F}_{\max})$ for all $k \in \mathbb{N}_0$. Thus we have $\mathcal{R}_1(\mathcal{F}_{\max}) \subseteq \mathcal{R}_1^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_1^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \mathcal{F}_{\max}$, i.e. $\mathcal{F}_{\max}$ is PI for the nonlinear system (11.28) with NN-based controller (11.3). Therefore, we have $\boldsymbol{x}(k) \in \mathcal{F}_{\max} \subseteq \mathcal{X}$ for all $k \in \mathbb{N}_0$ and by assumption $\boldsymbol{\Phi}(\boldsymbol{x}(k)) \in \mathcal{U}$ for all $k \in \mathbb{N}_0$. The nonlinear system with (11.3) will further converge to $\overline{\mathcal{R}}_{\min}$ for all $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$, since we have $\boldsymbol{x}(k) \in \mathcal{R}_{\bar{k}}(\mathcal{F}_{\max}) \subseteq \mathcal{R}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) \subseteq \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) = \overline{\mathcal{R}}_{\min}$ for all $k \geq \bar{k}$ and $\boldsymbol{x}(0) \in \mathcal{F}_{\max}$. This proves that the system (11.28) with $\boldsymbol{u}(k) = \boldsymbol{\Phi}(\boldsymbol{x}(k))$ is ultimately bounded in $\overline{\mathcal{R}}_{\min}$, uniformly in $\mathcal{F}_{\max}$. ■

11.5 Case studies

In the case study, we consider two different cases. The first case is a PWA system with bounded additive disturbance controlled by an NN-based controller that approximates the MPC control law of the undisturbed system. In

the second case study, we consider a nonlinear system approximated by a PWA system.

11.5.1 PWA system with additive disturbance

Consider the OCP [P6, Eq. 2] for a nominal PWA system, i.e, with $\boldsymbol{d}(k) = \boldsymbol{0}$ for all $k \in \mathbb{N}_0$ of the form (11.1) with

$$\begin{aligned}
\boldsymbol{A}^{(1)} &= \begin{pmatrix} -0.04 & -0.461 \\ -0.139 & 0.341 \end{pmatrix}, \boldsymbol{A}^{(2)} = \begin{pmatrix} 0.936 & 0.323 \\ 0.788 & -0.049 \end{pmatrix}, \\
\boldsymbol{A}^{(3)} &= \begin{pmatrix} -0.857 & 0.815 \\ 0.491 & 0.62 \end{pmatrix}, \boldsymbol{A}^{(4)} = \begin{pmatrix} -0.022 & 0.644 \\ 0.758 & 0.271 \end{pmatrix}, \\
\boldsymbol{B}^{(1)} &= \boldsymbol{B}^{(2)} = \boldsymbol{B}^{(3)} = \boldsymbol{B}^{(4)} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top, \\
\boldsymbol{H}^{(1)} &= \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \boldsymbol{H}^{(2)} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},
\end{aligned}$$

$\boldsymbol{H}^{(3)} = -\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(4)} = -\boldsymbol{H}^{(2)}$, $\boldsymbol{h}^{(1)} = \boldsymbol{h}^{(2)} = \boldsymbol{h}^{(3)} = \boldsymbol{h}^{(4)} = \boldsymbol{0}$, $\boldsymbol{p}^{(1)} = \boldsymbol{p}^{(2)} = \boldsymbol{p}^{(3)} = \boldsymbol{p}^{(4)} = \boldsymbol{0}$, $\boldsymbol{Q} = \boldsymbol{P} = \boldsymbol{I}$, $\boldsymbol{R} = 1$, and $N = 10$ from [101, Sec. 7] with the additional constraints $\|\boldsymbol{x}(k)\|_\infty \leq 10$ for all $k \in \{0, \dots, N\}$ and $|u(k)| \leq 1$ for all $k \in \{0, \dots, N-1\}$. We solved the OCP for 1000 randomly sampled $\boldsymbol{x} \in \mathcal{X}$ to generate a data set with the feasible points $(\boldsymbol{x}^{(i)}, u^*(\boldsymbol{x}^{(i)}))$, where $u^*(\boldsymbol{x}^{(i)})$ is the first element of the optimal input sequence. This data set is used to train a 3×3 maxout NN, i.e., $\ell = 3$ and $p_i = w_i = 3$ for all $i \in \{1, 2, 3\}$. Afterwards, we analyzed the PWA system with NN-based controller and an additive disturbance of $\|\boldsymbol{d}(k)\|_\infty \leq 0.15$, i.e, $\mathcal{D}^{(i)} = \{\boldsymbol{d} \in \mathbb{R}^n \mid \|\boldsymbol{d}\|_\infty \leq 0.15\}$ for $i \in \{1, \dots, 4\}$ by computing the sets $\mathcal{F}_{\max}$ from Proposition 11.4 and $\overline{\mathcal{R}}_{\min}$ (11.24) with $\bar{\epsilon} = 10^{-3}$. The computation of $\mathcal{F}_{\max}$ terminates after the first iteration, i.e., $\mathcal{F}_{\max} = \mathcal{F}_1$, where we choose $\boldsymbol{C} = (\boldsymbol{I} \ -\boldsymbol{I})$ for (11.18). The computation of $\overline{\mathcal{R}}_{\min}$ with $\boldsymbol{C}_i = \begin{pmatrix} \cos(\pi/4(i-1)) & \sin(\pi/4(i-1)) \end{pmatrix}$ for all $i \in \{1, \dots, 8\}$ can be seen in Figure 11.1. The black sets illustrate how the sequence of sets $\overline{\mathcal{R}}_i^{\mathcal{D}}(\mathcal{F}_{\max})$ with $i \in \{1, \dots, \bar{k}\}$ shrinks until (11.25) holds and the computation terminates with $\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) = \overline{\mathcal{R}}_{\min}$ (blue set) with $\bar{k} = 51$.

Since we have (11.26) for $\alpha = 1 + 9 \times 10^{-7}$, Theorem 11.5 applies here. This means that all trajectories of the PWA system with disturbance starting in $\mathcal{F}_{\max}$ remain in that set and enter the stable set $\mathcal{F}_{\min}$ after a finite number of time steps. This is illustrated in Figure 11.2, where the green and blue "tubes" represent 100 trajectories with random disturbance for each of the two initial states $\boldsymbol{x}(0) = (-7.30 \ 8.52)^\top$ and $\boldsymbol{x}(0) = (-7.30 \ -10)^\top$, respectively. As apparent from the figure, even with disturbance, the state constraints are satisfied, and all simulated trajectories enter $\overline{\mathcal{R}}_{\min}$ (small black set). Moreover, we can observe an interesting phenomenon. For some blue trajectories, the additive disturbance causes them to be in a different region of the PWA function than the nominal system at the respective time step. This leads to the significant

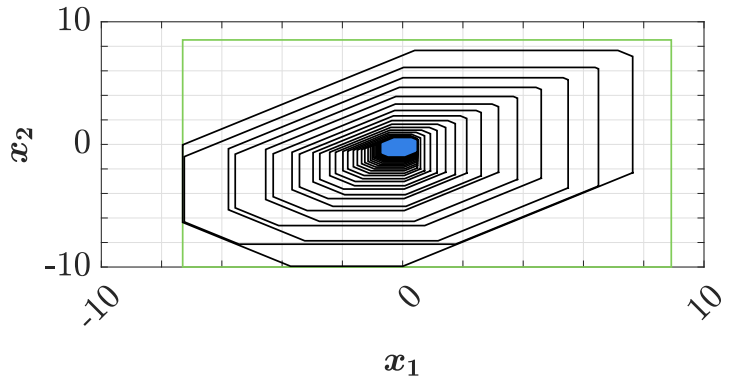


Figure 11.1. Illustration of the computation of the set $\overline{\mathcal{R}}_{\min} = \overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max})$ in blue starting from the set $\mathcal{F}_{\max}$ in green. The black sets illustrate the shrinking sequence of sets $\overline{\mathcal{R}}_{i}^{\mathcal{D}}(\mathcal{F}_{\max})$ with $i \in \{1, \dots \bar{k}-1\}$ during the computation of $\overline{\mathcal{R}}_{\min}$. The computation terminates with the set $\overline{\mathcal{R}}_{\bar{k}}^{\mathcal{D}}(\mathcal{F}_{\max}) = \overline{\mathcal{R}}_{\min}$ when (11.25) holds.

deviation between the nominal trajectory (black) and disturbed blue trajectories shortly before the set $\overline{\mathcal{R}}_{\min}$. However, since the computation of the RPI sets $\mathcal{F}_{\max}$ and $\overline{\mathcal{R}}_{\min}$ is based on reachable sets (cf. Figure 11.1) of the disturbed PWA system, these cases are included in the analysis.

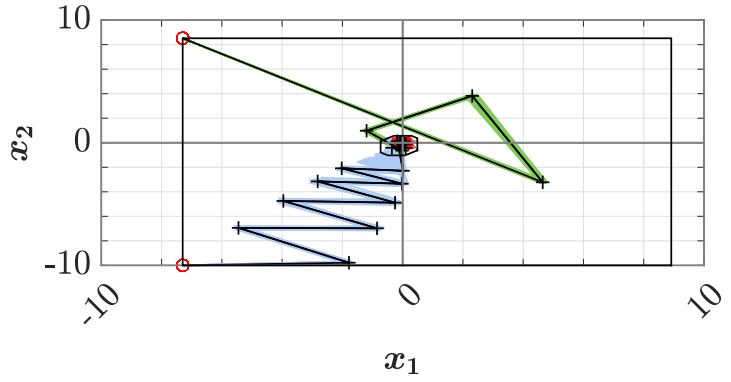


Figure 11.2. The green and blue "tubes" illustrate in each case 100 trajectories of the closed-loop system with a random additive disturbance $\|\boldsymbol{d}(k)\|_\infty \leq 0.15$ in every time step. The black lines represent trajectories of the nominal system (i.e. $\boldsymbol{d}(k) = \mathbf{0}$). The small set around the origin is $\overline{\mathcal{R}}_{\min}$ and the box $[-7.3, 8.91] \times [-10, 8.52]$ is the set $\mathcal{F}_{\max}$. The thick grey lines indicate the regions of the PWA system.

11.5.2 PWA approximation of a nonlinear system

We consider a system of the form (11.28) with

$$f(\boldsymbol{x}, u) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} u + \begin{pmatrix} 0.025 \\ 0.025 \end{pmatrix} \boldsymbol{x}^\top \boldsymbol{x}, \tag{11.29}$$

which is the nonlinear double integrator from [102, Sec. 6]. We solved the OCP from [140, Sec. IV] with the constraints $\|\boldsymbol{x}(k)\|_\infty \leq 6$ and $|u(k)| \leq 2$ for 900 initial values $\boldsymbol{x} = \boldsymbol{x}(0) \in \mathcal{X}$ on a regular grid of size 30×30 with $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from -6 to 6 to generate a data set with feasible points $(\boldsymbol{x}^{(i)}, u^*(\boldsymbol{x}^{(i)}))$ where $u^*(\boldsymbol{x}^{(i)})$ is the first element of the optimal input sequence. The data set is used to train a 5×5 maxout NN, i.e., $\ell = 5$ $p_i = w_i = 5$ for all $i \in \{1, \dots, 5\}$. For analyzing the closed-loop system with the methods from 11.4.4, we approximate the system dynamics of the nonlinear double integrator by a PWA function $\boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, u) : \mathbb{R}^{n+m} \to \mathbb{R}^n$ with 9 regions. The 9 regions divide the state space into a regular chessboard pattern with a side length of 4, i.e., $\mathcal{P}^{(i)} = \{(\boldsymbol{x}^\top \; u)^\top \mid \|\boldsymbol{x}\|_\infty \leq 2\} \oplus \{(\boldsymbol{x}^\top \; u)^\top \mid \boldsymbol{x} = (k\,l)^\top\}$ where the tuple $(k\,l)$ takes the 9 values from $\{-4, 0, 4\} \times \{-4, 0, 4\}$. In each region, we performed a least squares fit of the nonlinear function (11.29) to compute the parameters $\boldsymbol{A}^{(i)}$, $\boldsymbol{B}^{(i)}$, and $\boldsymbol{p}^{(i)}$ of the PWA approximation (11.1). Which results in an approximation with a maximum error of 0.1186, i.e., $\|\boldsymbol{f}(\boldsymbol{x}, u) - \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, u)\|_\infty \leq 0.1186$ for all $(\boldsymbol{x}^\top \; u)^\top \in \mathcal{X} \times \mathcal{U}$. Thus we choose $\mathcal{D}^{(i)} = \{\boldsymbol{d} \in \mathbb{R}^n \mid \|\boldsymbol{d}\|_\infty \leq 0.1186\}$ for all $i \in \{1, \dots, 9\}$. Since this leads to a PWA approximation with $\boldsymbol{f}(\boldsymbol{x}, u) - \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}, u) \in \mathcal{D}^{(i)}$ for all $(\boldsymbol{x}^\top \; u)^\top \in \mathcal{X} \times \mathcal{U}$, Theorem 11.7 applies here. For the PWA system we computed the sets $\mathcal{F}_{\text{max}}$ and $\overline{\mathcal{R}}_{\text{min}}$ with $\boldsymbol{C}_i = \begin{pmatrix} \cos(\pi/4(i-1)) & \sin(\pi/4(i-1)) \end{pmatrix}$ for all $i \in \{1, \dots, 8\}$ according to Proposition 11.4 and (11.24). Where the computation of $\mathcal{F}_{\text{max}}$ terminates after 19 iterations and the computation of $\overline{\mathcal{R}}_{\text{min}}$ with $\bar{\epsilon} = 10^{-3}$ after $\bar{k} = 20$ iterations. The validity of Theorem 11.7 can be observed in Figure 11.3. As apparent from the grey arrows, representing the evolution of the nonlinear system (11.29) with NN-based controller (11.3), and the red example trajectory, all trajectories starting in $\mathcal{F}_{\text{max}} \subseteq \mathcal{X}$ (green set) remain in that set and enter and remain in $\overline{\mathcal{R}}_{\text{min}}$ (blue set) after a finite number of time steps.

11.6 Conclusions

We presented an extension of MI-based analysis for PWA systems with NN-based controllers to the case where the PWA system is assumed to be affected by a bounded additive disturbance. The inclusion of disturbances allows to certify stability and constraint satisfaction for PWA systems with bounded additive disturbances and NN-based controllers (cf. Theorem 11.5) and UUB of nonlinear systems approximated by PWA systems (cf. Theorem 11.7).

Appendix

Proof of Lemma 11.2

For $\boldsymbol{X}_{K+1}, \boldsymbol{U}_K, \boldsymbol{D}_K, \tilde{\boldsymbol{X}}_K^{(1)}, ..., \tilde{\boldsymbol{X}}_K^{(s)}$ subject to (11.10), (11.12b), (11.12c), and (11.14a) we have $\boldsymbol{x}(k+1) = f_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{u}(k)) + \boldsymbol{d}(k)$ for all $k \in \{1, \dots, K-1\}$ and for all

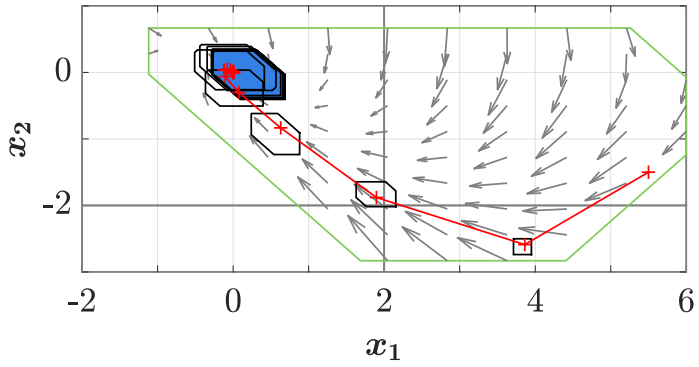


Figure 11.3. Nonlinear double integrator with an NN-based controller. The grey arrows represent the evolution of the closed-loop system within the set $\mathcal{F}_{\max}$ (green set). The red line represents a trajectory $\boldsymbol{x}(0), \ldots, \boldsymbol{x}(\bar{k})$ of the system (11.29) with (11.3) starting at $\boldsymbol{x}(0) = (5.5 \ -1.5)^\top$. The small sets around the states $\boldsymbol{x}(i)$ are the reachable sets $\overline{\mathcal{R}}_i^{\mathcal{D}}(\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = (5.5 \ -1.5)^\top\})$ with $i \in \{1, \ldots, \bar{k}\}$ in which the disturbed PWA system (which is an approximation of (11.29)) and thus the nonlinear system is guaranteed to be in time step i. Thick grey lines represent the regions of the PWA system.

$\boldsymbol{x}(0) \in \mathbb{R}^n$, according to [47, Sec. 3.1]. Since the relation (11.11) holds for (11.10) we further infer that $\boldsymbol{d}(k) \in \mathcal{D}^{(i)}$ (11.16b) is equivalent to (11.14b). Moreover, with (11.9) and (11.15c) we have $\boldsymbol{\Phi}(\boldsymbol{q}^{(0)}(k)) = \boldsymbol{\Phi}(\boldsymbol{x}(k)) = \boldsymbol{W}^{(\ell+1)}\boldsymbol{q}^{(\ell)}(k) + \boldsymbol{b}^{(\ell+1)}$ according to [P3, Lem. 2]. Combining this with (11.15d) finally results in $\boldsymbol{x}(k+1) = f_{\text{PWA}}(\boldsymbol{x}(k), \boldsymbol{\Phi}(\boldsymbol{x}(k))) + \boldsymbol{d}(k)$ (11.16a) for all $k \in \{1, \ldots, K-1\}$ and for all $\boldsymbol{x}(0) \in \mathbb{R}^n$, which completes the proof. ■

Proof of Lemma 11.3

According to Lemma 11.2 the constraints (11.15b)–(11.15d) can be replaced by (11.16). Thus, the MILP (11.17) becomes

$$
\begin{aligned}
c_k^*(\boldsymbol{v}, \mathcal{X}) = \max_{\boldsymbol{X}_{k+1}, \boldsymbol{D}_k} \ & \boldsymbol{v}\boldsymbol{x}(k) \\
\text{s.t.} \quad & \boldsymbol{x}(0) \in \mathcal{X} = \mathcal{R}_0(\mathcal{X}) \\
& \boldsymbol{x}(j+1) = \boldsymbol{A}^{(i)}\boldsymbol{x}(j) + \boldsymbol{B}^{(i)}\boldsymbol{\Phi}(\boldsymbol{x}(j)) + \boldsymbol{p}^{(i)} + \boldsymbol{d}(j) \\
& \boldsymbol{d}(j) \in \mathcal{D}(\boldsymbol{x}(j))
\end{aligned}
$$

for all $j \in \{0, \ldots, k-1\}$. By using the definition (11.4), we can replace the constraints for $j = 0$ and $\boldsymbol{x}(0) \in \mathcal{R}_0(\mathcal{X})$ by $\boldsymbol{x}(1) \in \mathcal{R}_1^{\mathcal{D}}(\mathcal{X})$. Then we can replace the constraints for $j = 1$ and $\boldsymbol{x}(1) \in \mathcal{R}_1(\mathcal{X})$ by $\boldsymbol{x}(2) \in \mathcal{R}_2^{\mathcal{D}}(\mathcal{X})$. Continuing in this way, we can successively replace all constraints until only $\boldsymbol{x}(k) \in \mathcal{R}_k^{\mathcal{D}}(\mathcal{X})$ remains. This finally results in

$$
c_k^*(\boldsymbol{v}, \mathcal{X}) = \max_{\boldsymbol{x}(k)} \boldsymbol{v}\boldsymbol{x}(k) \quad \text{s.t.} \quad \boldsymbol{x}(k) \in \mathcal{R}_k^{\mathcal{D}}(\mathcal{X}),
$$

which is by definition the support function of $\mathcal{R}_k^{\mathcal{D}}(\mathcal{X})$ evaluated for v, i.e., $c_k^*(v,\mathcal{X}) = h_{\mathcal{R}_k^{\mathcal{D}}(\mathcal{X})}(v)$. ■

Validity of Lemma 11.6

We first prove the following intermediate result.

Lemma 11.8. *Let $f(\mathcal{X},\mathcal{U})$ and $f_{PWA}^{\mathcal{D}}(\mathcal{X},\mathcal{U})$ be defined as $f(\mathcal{X},\mathcal{U}) := \{x^+ \in \mathbb{R}^n \mid x^+ = f(x,u), x \in \mathcal{X}, u \in \mathcal{U}\}$ and $f_{PWA}^{\mathcal{D}}(\mathcal{X},\mathcal{U}) := \{x^+ \in \mathbb{R}^n \mid x^+ = f_{PWA}(x,u) + d, x \in \mathcal{X}, u \in \mathcal{U}, d \in \mathcal{D}^{(i)}\}$, respectively. Further assume that $f(x,u) - f_{PWA}(x,u) \in \mathcal{D}^{(i)}$ for all $x \in \mathcal{X}$ and for all $u \in \mathcal{U}$. Then, $f(\mathcal{F},\mathcal{G}) \subseteq f_{PWA}^{\mathcal{D}}(\mathcal{F},\mathcal{G})$ for $\mathcal{F} \subseteq \mathcal{X}$ and $\mathcal{G} \subseteq \mathcal{U}$.*

Proof. We need to prove that if $x^+ \in f(\mathcal{F},\mathcal{G})$ then $x^+ \in f_{\text{PWA}}^{\mathcal{D}}(\mathcal{F},\mathcal{G})$. For $x^+ \in f(\mathcal{F},\mathcal{G})$ we have

$$x^+ = f(x,u),\ x \in \mathcal{F},\ u \in \mathcal{G}. \tag{11.31}$$

By assumption there exist a $d = f(x,u) - f_{\text{PWA}}(x,u) \in \mathcal{D}^{(i)}$ for all $x \in \mathcal{F} \subseteq \mathcal{X}$ and for all $u \in \mathcal{G} \subseteq \mathcal{U}$. Thus (11.31) can be reformulated as follows

$$x^+ = f_{\text{PWA}}(x,u) + d,\ x \in \mathcal{F},\ u \in \mathcal{G},\ d \in \mathcal{D}^{(i)},$$

which is exactly the definition of $f_{\text{PWA}}^{\mathcal{D}}(\mathcal{F},\mathcal{G})$, i.e., we have $x \in f_{\text{PWA}}^{\mathcal{D}}(\mathcal{F},\mathcal{G})$ if $x \in f(\mathcal{F},\mathcal{G})$ for $\mathcal{X} \subseteq \mathcal{F}$ and $\mathcal{G} \subseteq \mathcal{U}$. Thus $f(\mathcal{F},\mathcal{G}) \subseteq f_{\text{PWA}}^{\mathcal{D}}(\mathcal{F},\mathcal{G})$ holds for $\mathcal{F} \subseteq \mathcal{X}$ and $\mathcal{G} \subseteq \mathcal{U}$. ■

Now, we use the intermediate result to prove Lemma 11.6.

Proof of Lemma 11.6

We prove the claim by induction. For $k = 1$ we have

$$\begin{aligned}\mathcal{R}_1(\mathcal{F}) &= \{x^+ \in \mathbb{R}^n \mid x^+ = f(x,u), x \in \mathcal{F}, u \in \mathcal{U}, u = \Phi(x)\} \\ &= f(\mathcal{F},\mathcal{G})\end{aligned}$$

with $\mathcal{G} := \{u \in \mathbb{R}^m \mid u \in \mathcal{U},\ u = \Phi(x)\} \subseteq \mathcal{U}$ and $\mathcal{F} \subseteq \mathcal{X}$. Then we have according to Lemma 11.8

$$\begin{aligned}f(\mathcal{F},\mathcal{G}) &\subseteq f_{\text{PWA}}^{\mathcal{D}}(\mathcal{F},\mathcal{G}) \\ &= \{x^+ \in \mathbb{R}^n \mid x^+ = f_{\text{PWA}}(x,u) + d, \\ &\qquad x \in \mathcal{F}, u \in \mathcal{U}, u = \Phi(x), d \in \mathcal{D}^{(i)}\} = \mathcal{R}_1^{\mathcal{D}}(\mathcal{F})\end{aligned}$$

In summary $\mathcal{R}_1(\mathcal{F}) \subseteq \mathcal{R}_1^{\mathcal{D}}(\mathcal{F})$ holds for $\mathcal{F} \subseteq \mathcal{X}$. For $k = i+1$ we have $\mathcal{R}_1(\mathcal{R}_i(\mathcal{F})) \subseteq \mathcal{R}_1(\mathcal{R}_i^{\mathcal{D}}(\mathcal{F})) \subseteq \mathcal{R}_1^{\mathcal{D}}(\mathcal{R}_i^{\mathcal{D}}(\mathcal{F})) = \mathcal{R}_{i+1}^{\mathcal{D}}(\mathcal{F})$ and thus $\mathcal{R}_{i+1}(\mathcal{F}) \subseteq \mathcal{R}_{i+1}^{\mathcal{D}}(\mathcal{F})$ if $\mathcal{R}_i^{\mathcal{D}}(\mathcal{F}) \subseteq \mathcal{X}$, which is satisfied since $\mathcal{F} \subseteq \mathcal{X}$ is RPI by assumption. This proves that $\mathcal{R}_k(\mathcal{F}) \subseteq \mathcal{R}_k^{\mathcal{D}}(\mathcal{F})$ holds for all $k \in \mathbb{N}_0$ and for RPI sets $\mathcal{F} \subseteq \mathcal{X}$. ■

Chapter 12

Polynomial function approximations with leading integer coefficients for efficient encrypted implementations*

Dieter Teichrib[1†], Janis Adamek[1†], Philipp Binfet[†],
and Moritz Schulze Darup[†]
[1] Both authors contributed equally to this paper.

Abstract. Computations on encrypted data can, in principle, be performed using homomorphic encryption. However, due to certain limitations, only algorithms based on polynomial functions can be efficiently implemented in an encrypted setting. Consequently, polynomial approximations of non-polynomial functions are essential for efficient encrypted computations. In particular, low- to moderate-degree polynomial approximations of activation functions in neural networks are of special interest.

We show that the accuracy of *encryption-friendly* approximations can be improved through a simple yet effective extension of state-of-the-art methods. Specifically, we show that enforcing a leading integer coefficient enables the use of polynomials of one degree higher than all existing approaches. Incorporating this novel integer constraint into classical regression problems initially leads to mixed-integer programs (MIPs). However, we develop tailored solution schemes that avoid MIP solving. Using these schemes, we compute new polynomial approximations for various test cases and demonstrate the effectiveness of our method compared to existing approaches.

†All authors are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

Keywords. Polynomial regression, optimization, homomorphic encryption, Chebyshev regression, privacy-preserved machine learning

12.1 Introduction and problem statement

Homomorphic encryption (HE) enables computations on encrypted data (see [141] for an overview). The unique capability of encrypted computations has unlocked a wide range of fascinating real-world applications across various fields such as privacy-preserving machine learning (ML) [72, 142], secure cloud computing [143], encrypted database queries [144, 145], encrypted financial services [146, 147], secure energy grid management [148, 149], secure voting systems [150], and encrypted control of networked systems [151]. In all these applications, HE enables new exciting features such as learning on encrypted data, encrypted regression, encrypted classification, or encrypted decision making.

Yet, while the number and performance of homomorphically encrypted applications is increasing, encrypted computations are still challenging. In fact, the set of available operations offered by HE schemes is typically quite limited and mainly includes encrypted multiplications and additions. Moreover, the number of consecutive encrypted multiplications (or, more precisely, the multiplicative depth) is usually restrictive [71]. These limitations hinder the encrypted evaluation of many functions or algorithms. Nevertheless, polynomials of moderate degree can be evaluated efficiently on encrypted data. As a consequence, accurate polynomial approximations of non-polynomial functions are currently intensively investigated in the context of encrypted implementations [152, 153]. In particular, due to their heavy usage in ML, popular activation functions in artificial neural networks such as rectified linear units (ReLU) [72, 92], sigmoid functions [154], or hyperbolic tangent are of special interest. The recent ReLU approximation challenge [155] underlines the significance of this research field even for commercial applications. In addition, polynomial approximations are relevant for further methods in encrypted ML, like logistic regression [156, 157] and are also utilized in other fields such as encrypted control [94].

Given the demand for effective polynomial approximations in the context of HE, our problem of interest can be formalized as follows. We aim for *encryption-friendly* polynomials of the form

$$p(x) := c_0 + c_1 x + \cdots + c_n x^n \tag{12.1}$$

approximating functions $f(x)$ on closed, non-empty intervals $[a, b]$. Now, for implementing (12.1) in an encrypted fashion using an HE scheme allowing for a multiplicative depth $d \in \mathbb{N}$, a tree-based realization – also known as exponentiation by squaring – is the current standard. It allows for implementing

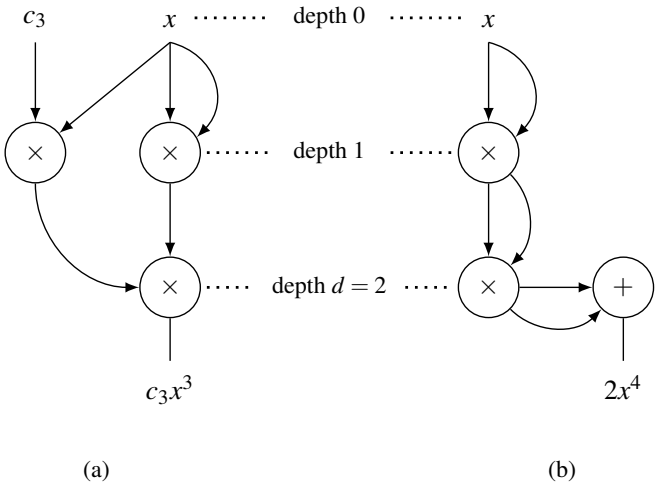


Figure 12.1. Computational circuits with multiplicative depth $d = 2$ for (a) a monomial of degree $3 = 2^d - 1$ with a generic coefficient and (b) a monomial of degree $4 = 2^d$ with an integer coefficient.

monomials of degree up to $n = 2^d - 1$ with generic coefficients $c_i \in \mathbb{R}$ [44], as illustrated in Figure 12.1.a for the example of c_3x^3 and $d = 2$. In this paper, we propose a new method for increasing the degree of polynomials that can be implemented in an encrypted fashion under a multiplicative depth d from $2^d - 1$ to 2^d. For moderate multiplicative depths such as $d \in \{1, 2, \ldots, 5\}$, this additional degree can make a significant difference as we illustrate with various numerical examples in Section 12.4. To the best of the authors' knowledge, such an approach has not been considered in the existing literature.

Technically, our method builds on restricting the polynomial's leading coefficient c_n to the set of integers $\mathbb{Z}$. In fact, as illustrated in Figure 12.1.b and detailed in Section 12.2.1, c_nx^n can then be evaluated via $x^n + \cdots + x^n$ with no additional multiplication. Now, as specified in Section 12.2.2, searching for optimal $c_n \in \mathbb{Z}$ in the context of regression problems initially leads to mixed-integer programs (MIPs). However, we show that the resulting MIPs can be solved efficiently by solving only continuous (i.e., non-integer) optimization problems (OP). Remarkably, as presented in Section 12.3, our method can also be adapted to Chebyshev regression, which significantly extends its applicability. In summary, our main contributions are (i) a novel approach for encryption-friendly polynomial approximations building on leading integer coefficients, (ii) an efficient procedure for identifying optimal $c_n \in \mathbb{Z}$, and (iii) an illustration of the effectiveness of our method with various numerical examples. Before detailing our approach in the following sections, we briefly specify relevant notation.

Notation. We denote natural numbers, the set of integers, and real numbers by $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$, respectively. Furthermore, with $\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$, and $\lfloor \cdot \rceil$, we refer to rounding down, rounding up, and rounding to the nearest integer (with rounding towards zero in case of tie).

12.2 Optimal polynomial approximations with leading integer coefficients

As summarized in the introduction, we are aiming for effective polynomial function approximations of the form (12.1) in the context of encrypted implementations. We briefly motivated that, given this special context, leading integer coefficients are beneficial. We specify this benefit in Section 12.2.1 before discussing the resulting regression problem and our proposed solution in Section 12.2.2.

12.2.1 Benefit of leading integer coefficient

In HE schemes, the multiplicative depth $d \in \mathbb{N}$ is typically limited. It refers to the maximum number of consecutive encrypted multiplications supported by the given scheme. More precisely, assuming the desired computations are specified by a computational circuit as in Figure 12.1, then d is an upper limit for the number of multiplications (i.e., nodes "$\times$" in Fig. 12.1) along any directed branch from an input node to an output node. If this limit is exceeded, computations usually lead to (highly) erroneous results. As a consequence, sticking to the limit is mandatory in encrypted computations.

Unsurprisingly, a limited multiplicative depth implies a limit on implementable monomials $c_n x^n$. For instance, naively computing $c_n x^n$ via $c_n \times x \times \cdots \times x$ requires n consecutive multiplications. The number of consecutive multiplications required to evaluate a monomial $c_n x^n$ can be easily reduced in optimized implementations. It is well-known that, for generic coefficients $c_n \in \mathbb{R}$, a tree-based realization requires the fewest possible number of consecutive multiplications and allows evaluating monomials of degree up to $n = 2^d - 1$ given a multiplicative depth d. For instance, given $d = 2$, we can evaluate $c_3 x^3$ via $(c_3 \times x) \times (x \times x)$ as illustrated in Figure 12.1.a. Analogously, given $d = 3$ and defining $t := (x \times x)$, we can evaluate $c_7 x^7$ via $[(c_7 \times x) \times t] \times [t \times t]$, where terms in round brackets are evaluated at depth 1 and terms in square brackets at depth 2 (cf. Fig. 12.1). Up to now, this approach has built the basis for polynomial approximations in homomorphically encrypted computations. However, an improved realization of polynomials of degree up to $n = 2^d$ is possible when the leading coefficient c_n is an integer, as proposed by our approach and detailed below.

In fact, if $c_n \in \mathbb{Z}$, then one of the consecutive multiplications for computing $c_n x^n$ can be avoided and replaced by additions (which are not limited by the multiplicative depth). For instance, after having computed x^n using the classical tree-based approach, $c_n x^n$ can be evaluated via $x^n + \cdots + x^n$. Remarkably, tree-based approaches can also be utilized to compute this summation more efficiently. For example, $4x^2$ can be calculated via $s + s$ with $s := x^2 + x^2$ (being computed as in Fig. 12.1.b) using only two consecutive additions instead of three as for the naive approach. In general, for integer c_n, $c_n x^n$ can be evaluated

using $\lceil \log_2(c_n) \rceil$ additions, which is important if larger $c_n \in \mathbb{Z}$ are considered. In summary, integer c_n are beneficial for encrypted implementations of polynomials as they allow to increase the implementable degree by one. However, as discussed next, efficiently designing polynomials with $c_n \in \mathbb{Z}$ for approximating non-polynomial functions is non-trivial.

12.2.2 Efficient polynomial regression with leading integer coefficient

In order to leverage the increased polynomial degree in polynomial regression, we have to ensure $c_n \in \mathbb{Z}$. In principle, including this novel constraint is straightforward. In fact, assume N sampling points x_i are given and consider any standard regression problem of the form

$$\min_{c_0,\dots,c_n} J(c_0,\dots,c_n,x_1,\dots,x_N). \tag{12.2}$$

Then, adding the integer constraint $c_n \in \mathbb{Z}$ results in the desired restriction. For standard performance measures such as the mean squared error (MSE)

$$J(c_0,\dots,c_n,x_1,\dots,x_N) := \frac{1}{N}\sum_{i=1}^{N}(f(x_i)-p(x_i))^2, \tag{12.3}$$

the resulting optimization problem is a mixed-integer quadratic program (MIQP), which can be solved using standard software such as MOSEK [45] or Gurobi [46]. Yet, as we show next, the optimal solution can be derived more efficiently without relying on mixed-integer optimization. In fact, by exploiting that only one decision variable is restricted to integers, an optimal solution can be found by solving at most three continuous OP. An appropriate approach can be derived from the following theorem, whose proof is provided in the appendix.

Theorem 12.1. *Assume the cost function J in* (12.2) *is convex in the coefficients c_i and assume $\boldsymbol{c}^* := (c_0^*,\dots,c_n^*)$ is an optimizer for* (12.2) *(without constraints). Then, there exists an optimizer $\hat{\boldsymbol{c}}^* := (\hat{c}_0^*,\dots,\hat{c}_n^*)$ for the constrained problem*

$$\min_{\hat{c}_0,\dots,\hat{c}_n} J(\hat{c}_0,\dots,\hat{c}_n,x_1,\dots,x_N) \quad s.t. \quad \hat{c}_n \in \mathbb{Z}. \tag{12.4}$$

with $\hat{c}_n^* = \lfloor c_n^* \rfloor$ or $\hat{c}_n^* = \lceil c_n^* \rceil$.

From Theorem 12.1, we infer the following procedure for solving the MIP (12.4). Initially, we solve the unconstrained (and continuous) OP (12.2). If the resulting optimizer $\boldsymbol{c}^*$ is such that $c_n^* \in \mathbb{Z}$, we immediately found a solution to (12.4). Otherwise, we solve two variants of the OP (12.2), where we fix c_n to $\lfloor c_n^* \rfloor$ or $\lceil c_n^* \rceil$, respectively. The solution with the smaller cost function value then reflects a solution to (12.4). Hence, the MIP (12.4) can indeed be solved by (at most) three continuous OPs. As we show next, the procedure can even be

shortened if J is symmetric with respect to the unconstrained optimizer $\boldsymbol{c}^*$. Among others, this is the case for the MSE (12.3) and the mean absolute error (MAE) considered further below in (12.13). A formal proof for the following theorem is (again) provided in the appendix.

Theorem 12.2. *Let J and $\boldsymbol{c}^*$ be as in Theorem 12.1. Additionally, let J possess the point symmetry $J(\boldsymbol{c}^* + \boldsymbol{c}) = J(\boldsymbol{c}^* - \boldsymbol{c})$ for every $\boldsymbol{c} \in \mathbb{R}^{n+1}$. Then, there exists an optimizer $\hat{\boldsymbol{c}}^*$ for* (12.4) *with $\hat{c}_n^* = \lfloor c_n^* \rceil$.*

Clearly, given the symmetry in Theorem 12.2, we can find a solution to (12.4) by solving at most two continuous OPs. In fact, assuming the solution to the unconstrained OP (12.2) is such that $c_n^* \notin \mathbb{Z}$, we only need to solve one additional variant of (12.2), where we fix $c_n := \lfloor c_n^* \rceil$. We conclude this section by formalizing that solving (12.4) for a certain degree n can only lead to an improvement (or tie) compared to the optimal solution of the unconstrained OP (12.2) for a smaller degree.

Corollary 12.3. *Consider any positive $\hat{n} \in \mathbb{N}$ and let $\hat{\boldsymbol{c}}^*$ be an optimizer for* (12.4) *and $n := \hat{n}$. Furthermore, let $\boldsymbol{c}^*$ be an optimizer for* (12.2) *and an degree $n \in \mathbb{N}$ smaller than $\hat{n}$. Then, $J(\hat{\boldsymbol{c}}^*) \leq J(\boldsymbol{c}^*)$, where J refers to the same cost function instantiated for the two different degrees (but same number of sampling points N).*

Proof. Let $\Delta n > 0$ be the difference of the degrees considered for the OPs (12.4) respectively (12.2). Then, the statement trivially follows from the fact that the optimizer $\boldsymbol{c}^*$ augmented by Δn zeros is feasible for (12.4) and $n := \hat{n}$. ■

While Corollary 12.3 relates the two optimal cost function values through a non-strict inequality, in practice, we often observe that including the leading integer coefficient results in a significant improvement compared to unconstrained solutions of smaller degree. In such cases, we call the leading coefficient *beneficial* according to the following definition.

Definition 12.4. *A leading integer coefficient is beneficial, if there exists an optimizer $\hat{\boldsymbol{c}}^*$ for* (12.4) *with $\hat{c}_n^* \neq 0$ but none with $\hat{c}_n^* = 0$.*

We will illustrate and discuss the existence of beneficial leading integer coefficients with numerical examples in Section 12.4. Prior to this, we extend our results to Chebyshev regression in the following section.

12.3 Extension to Chebyshev regression

From a numerical perspective, not only the integer constraint in (12.4) is challenging but also the relatively high polynomial degree n of (up to) 2^d. A standard approach addressing the latter issue is to consider more suitable polynomial families rather than the canonical power basis in (12.1). In particular, fitting Chebyshev polynomials of the form

$$\tilde{p}(z) := \tilde{c}_0 + \tilde{c}_1 T_1(z) + \cdots + \tilde{c}_n T_n(z) \tag{12.5}$$

with $T_i(z)$ referring to the i-th Chebyshev polynomial (of the first kind) typically works well for high polynomial degrees. This is mainly because Chebyshev regression avoids the large growth of higher-order coefficients as often observed in the canonical power basis. Still, using this approach requires to map the interval $[a, b]$ onto $[-1, 1]$ using, e.g., the mapping function

$$g(x) := \left(x - \frac{a+b}{2}\right) \frac{2}{b-a}. \tag{12.6}$$

Formally, (12.2) is then substituted by

$$\min_{\tilde{c}_0,\ldots,\tilde{c}_n} \tilde{J}\left(\tilde{c}_0, \ldots, \tilde{c}_n, g(x_1), \ldots, g(x_N)\right) \tag{12.7}$$

and, for MSE-based cost, (12.3) is replaced by

$$\tilde{J}(\tilde{c}_0, \ldots, \tilde{c}_n, z_1, \ldots, z_N) := \frac{1}{N} \sum_{i=1}^{N} \left(f(g^{-1}(z_i)) - \tilde{p}(z_i)\right)^2$$

with

$$g^{-1}(z) := \frac{b-a}{2} z + \frac{a+b}{2}$$

being the inverse mapping to (12.6). Now, solving (12.7) leads to an optimizer $\tilde{c}^*$. For the implementation of the corresponding polynomial approximation of f, two scenarios are possible: First, given a sample x, one can compute $z := g(x)$ via (12.6) and, subsequently, evaluate (12.5). Second, one can expand $\tilde{p}(g(x))$ in x and, subsequently, evaluate the resulting polynomial for the given x.

For the encrypted implementation considered here, the first approach is unsuitable. In fact, mapping x via g would require a multiplication and, hence, would consume one level of the available multiplicative depth. Thus, we here focus on the second approach. Regarding the expansion, it is well known that only the polynomial $T_n(z)$ involves a monomial of the highest degree n. More specifically, this monomial reads $2^{n-1}z^n$. Taking the mapping $z = g(x)$ into account, we find that the leading coefficient of the expanded polynomial (of the form (12.1)) is given by

$$c_n = \tilde{c}_n 2^{n-1} \left(\frac{2}{b-a}\right)^n = \frac{\tilde{c}_n}{2} \left(\frac{4}{b-a}\right)^n. \tag{12.8}$$

Combining this observation with our integer constraint $c_n \in \mathbb{Z}$ from above, leads to the adapted constraint

$$\frac{\tilde{c}_n}{2} \left(\frac{4}{b-a}\right)^n \in \mathbb{Z} \tag{12.9}$$

for the OP (12.7). Clearly, this constraint is trivially fulfilled for $\tilde{c}_n = 0$. However, in this case, we also obtain $c_n = 0$, i.e., a polynomial of degree $n - 1$.

Satisfying the constraint (12.9) for $\tilde{c}_n \neq 0$ is slightly delicate. In fact, for moderate to high degrees n, the unconstrained OP (12.7) typically leads to $\tilde{c}_n^*$ with an absolute value (significantly) smaller than 1. This can be problematic due to relations between $\tilde{c}_n^*$ and $\hat{c}_n^*$ implied by Theorems 12.1 and 12.2. In fact, assuming Theorem 12.2 applies, a beneficial leading integer coefficient $\hat{c}_n^* \in \mathbb{Z} \setminus \{0\}$ can only exist if

$$\frac{\tilde{c}_n^*}{2} \left(\frac{4}{b-a} \right)^n \notin [-0.5, 0.5] \tag{12.10}$$

since, otherwise, applying $\lfloor \cdot \rceil$ to the left-hand side of (12.10), results in $\hat{c}_n^* = 0$. Theorem 12.1 allows to derive a similar condition* for the existence of an optimizer $\hat{\boldsymbol{c}}^*$ with $\hat{c}_n^* \in \mathbb{Z} \setminus \{0\}$, which even applies to asymmetric J. Now, for small $\tilde{c}_n^* \in (-1, 1)$, condition (12.10) can only be satisfied for large factors $4^n/(b-a)^n > 1$, which requires

$$b - a < 4. \tag{12.11}$$

Remarkably, it turns out that (12.11) is indeed necessary for a beneficial leading integer coefficient for all test cases in the following section.

12.4 Numerical case studies

As previously discussed, Corollary 12.3 implies that our method cannot be outperformed by any existing approach, as they are all limited to polynomial approximations of at least one degree lower. Yet, there are cases (specified below) where our method performs equally well as an existing approach. Consequently, the relation in Corollary 12.3 is tight and admits no further improvement. Thus, we need numerical experiments to demonstrate the practical benefit of our method. To this end, we evaluate our method on various test cases involving different functions, domains, performance measures, and multiplicative depths. With regard to functions to be approximated, we consider

$$f_1(x) := \max\{x, 0\}, \tag{12.12a}$$
$$f_2(x) := \tanh(5x), \tag{12.12b}$$
$$f_3(x) := \frac{1}{1 + e^{-10x}} \tag{12.12c}$$

inspired by activation functions commonly used in artificial neural networks. In fact, (12.12a) refers to a ReLU, while (12.12b) and (12.12c) reflect scaled* versions of the hyperbolic tangent and sigmoid functions, respectively. Regarding

*According to Thm. 12.1, the left-hand side of (12.10) being not contained in $(-1, 1)$ is sufficient for the existence of an optimizer $\hat{\boldsymbol{c}}^*$ with $\hat{c}_n^* \in \mathbb{Z} \setminus \{0\}$.

*The scaling is required since the unscaled hyperbolic tangent and sigmoid functions are almost linear on the domains considered for the test cases, which renders the approximation task rather trivial.

function domains, we consider different intervals $[a, b]$ with $a < 0 < b$ (due to the link to activation functions) and $b - a \leq 4$ due to (12.11). More specifically, we investigate the intervals $\mathcal{I}_1 := [-2, 2]$, $\mathcal{I}_2 := [-1, 1]$, $\mathcal{I}_3 := [-0.5, 0.5]$, $\mathcal{I}_4 := [-2, 1]$, $\mathcal{I}_5 := [-1, 2]$, $\mathcal{I}_6 := [-1, 0.5]$, and $\mathcal{I}_7 := [-0.5, 1]$, where we note that the first three are symmetric (i.e, $a = -b$) whereas the others are not. As performance measures, we consider the MSE (12.3) and the MAE

$$J(c_0, \ldots, c_n, x_1, \ldots, x_N) := \frac{1}{N} \sum_{i=1}^{N} |f(x_i) - p(x_i)| \tag{12.13}$$

or, more precisely, their counterparts for Chebyshev regression. Regarding the multiplicative depth, we take $d \in \{1, 2, 3, 4, 5\}$ and the corresponding maximum polynomial degrees $n := 2^d$ into account. Finally, regarding the number of sampling points, we consider $N := 10n$ points on a regular grid in each case. In summary, we consider $3 \times 7 \times 2 \times 5 = 210$ different test cases (reflecting the different combinations of functions, domains, performance measures, and multiplicative depths).

For each test case, we perform a Chebyshev regression and solve (12.7) subject to (i) the integer constraint (12.9) and (ii) the restriction $\tilde{c}_n = 0$. For the former, given the symmetry of the performance measures, we use Theorem 12.2 for an efficient solution. Regarding the latter, we note that it reflects the solution to the unconstrained OP of degree $n - 1$. Having solved both OPs, we compare the two resulting performances (i.e., optimal function values) and compute the relative performance improvement resulting from the leading integer coefficient. The corresponding results are presented in Tables 12.1 to 12.6, where each individual table contains data for one function and one performance measure (but all depths d and intervals $\mathcal{I}_j$). In each table, when there is no improvement for a specific test case, we write "–" instead of "0.0" in order to highlight the cases, where significant improvements are achieved.

Table 12.1. Performance improvement (in %) for $f_1(x)$ and MSE.

d	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$
1	–	–	92.63	–	–	–	–
2	–	–	75.39	–	–	–	–
3	–	47.03	51.39	–	–	48.09	48.09
4	–	30.83	30.83	–	–	18.36	18.36
5	–	17.03	17.03	–	–	–	–

Now, the data in the tables offers numerous insights. First, the results confirm that existing methods never outperform our method, although ties do occur. Second, we do not observe an improvement for the interval $\mathcal{I}_1$ in any test case. This result is in line with our analysis in Section 12.3 since condition (12.11) is violated for $\mathcal{I}_1$ (due to $b - a = 4$) and since we indeed find small $\tilde{c}_n^* \in (-1, 1)$ in every test case. For all other intervals $\mathcal{I}_j$, there exist multiple

Table 12.2. Performance improvement (in %) for $f_1(x)$ and MAE.

d	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$
1	–	–	73.68	–	–	1.09	1.09
2	–	–	52.53	–	–	–	–
3	–	34.60	34.67	–	–	33.16	33.16
4	–	20.21	20.21	–	–	14.89	14.89
5	–	11.15	11.15	2.58	2.58	2.67	2.67

Table 12.3. Performance improvement (in %) for $f_2(x)$ and MSE.

d	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$
1	–	–	–	–	–	59.08	59.08
2	–	–	–	0.41	0.41	81.86	81.86
3	–	–	–	–	–	28.25	28.25
4	–	–	–	27.63	27.63	70.72	70.72
5	–	–	–	57.84	57.84	56.27	56.27

Table 12.4. Performance improvement (in %) for $f_2(x)$ and MAE.

d	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$
1	–	–	–	–	–	45.84	45.84
2	–	–	–	2.58	2.58	58.30	58.30
3	–	–	–	–	–	3.41	3.41
4	–	–	–	1.48	1.48	39.91	39.91
5	–	–	0.07	35.69	35.69	42.48	42.48

Table 12.5. Performance improvement (in %) for $f_3(x)$ and MSE.

d	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$
1	–	–	–	–	–	55.00	55.00
2	–	–	–	–	–	49.98	49.98
3	–	–	–	–	–	5.01	5.01
4	–	–	–	–	–	0.07	0.07
5	–	–	–	52.65	52.65	–	–

test cases, where the leading integer coefficient is beneficial. In general, improvements are more frequent for smaller intervals (i.e., smaller $b - a$) and for higher multiplicative depths d implying higher degrees n. Both are reasonable with regard to condition (12.10). Another trend is that, although improvements are more likely for larger values of d, the extent of these improvements tends to decrease as d increases. This makes sense since the positive effect of an additional degree diminishes with increasing degrees. To underline the significance

Table 12.6. Performance improvement (in %) for $f_3(x)$ and MAE.

d	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$
1	–	–	–	–	–	45.84	45.84
2	–	–	–	–	–	56.48	56.48
3	–	–	–	–	–	3.41	3.41
4	–	–	–	–	–	39.91	39.91
5	–	–	0.34	35.69	35.69	42.48	42.48

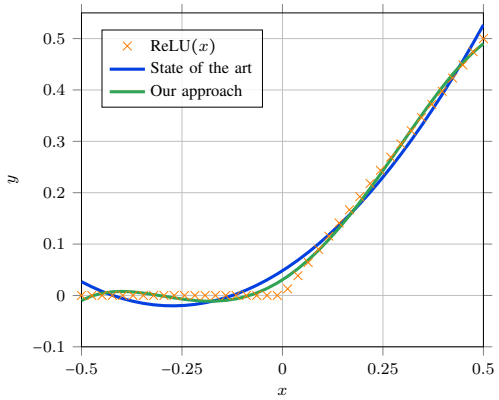


Figure 12.2. Exemplary comparison of our solution with the current state of the art for the MSE polynomial regression problem of the ReLU function with a multiplicative depth of $d = 3$ on $\mathcal{I}_3$.

of the improvements, we visualized the comparison between our solution and the current state of the art for the MSE regression task of the ReLU function in Figure 12.2 for $d = 3$ on $\mathcal{I}_3$. The improvement of 51.39% in Table 12.1 due to the higher polynomial degree is visually detectable.

Apart from the general trends discussed above, we can observe some more specific trends for the individual functions and intervals. First, note that the performance measures for the intervals $\mathcal{I}_4$ and $\mathcal{I}_5$ respectively the intervals $\mathcal{I}_6$ and $\mathcal{I}_7$ are identical within each row of every table. This observation can be explained as follows. Clearly, the matching intervals can be transformed into each other by reflecting them across the origin. Now, the three test functions offer similar symmetries. In fact, it is easy to see that

$$f_1(x) = f_1(-x) + x, \tag{12.14a}$$
$$f_2(x) = -f_2(-x), \tag{12.14b}$$
$$f_3(x) = -f_3(-x) + 1 \tag{12.14c}$$

for every $x \in \mathbb{R}$. As a consequence, whenever we found a polynomial of degree $n \geq 1$ approximating one of the test functions f_i on a certain interval $[a, b]$ with a certain performance measure based on N data points on a regular grid, using the relations (12.14), we can derive a polynomial of the same degree which approximates f_i on $[-b, -a]$ with the same performance. In fact, for

degrees $n \geq 1$, we can compensate for the "asymmetric offsets" x in (12.14a) respectively 1 in (12.14c). Hence, it is no surprise that the optimization-based results in all tables are equivalent for the matching intervals. The relations (12.14b) and (12.14c) further reveal that f_2 and $f_3(x) - 0.5$ are odd functions (whereas $f_1(x) - x/2 = |x|/2$ is even). Now, polynomial approximations of odd functions on symmetric intervals tend to yield odd polynomials. As a consequence, we here find $\tilde{c}_n^* \approx 0$ for almost every test case involving f_2 or f_3 and any of the intervals $\mathcal{I}_1$, $\mathcal{I}_2$, or $\mathcal{I}_3$. This explains the blocks of "–" in the Tables 12.3–12.6 (with only few exceptions for $d = 5$ potentially resulting from numerical limitations*). In summary, a beneficial leading integer coefficient was found in 86 of the 210 test cases. In 70 of the test cases (i.e., one third) the improvement exceeded 3%.

12.5 Conclusions and outlook

We proposed a simple but effective method for more powerful polynomial function approximations in the context of HE. More precisely, we showed how to increase the supported polynomial degree (with respect to a limited multiplicative depth of the HE scheme) by one compared to state-of-the-art approaches. As illustrated with a comprehensive numerical case study involving popular activation functions for (deep) neural network-based machine learning, this additional degree can lead to significant improvements in approximation accuracy for low to moderate degree polynomials.

Technically, our approach builds on the consideration of polynomial approximations with leading integer coefficients c_n, where n is of the form 2^d with $d \in \mathbb{N}$. In fact, this feature enables the evaluation of monomials $c_n x^n$ in an encrypted fashion using a HE scheme supporting a multiplicative depth d (which was impossible before for more general $c_n \in \mathbb{R} \setminus \mathbb{Z}$). Performing a regression with the restriction $c_n \in \mathbb{Z}$ can be formulated as an MIP and, in principle, be solved using standard software. Yet, we showed that the solution to the MIP can also be obtained by solving at most two (for symmetric cost functions J) or three (for asymmetric J) continuous OP (see Thms. 12.1 and 12.2).

Future work aims for studying the beneficial effect of the novel polynomial approximations when used in compositions as it is the case, e.g., for privacy-preserving evaluations of deep neural networks or iterative optimization solvers.

*Note that, for $d = 5$, we have $n = 2^d = 32$. Hence, computing c_n from $\tilde{c}_n$ according to (12.8) involves the factor $9.22 \cdot 10^{18}$. As a consequence, depending on the machine precision, we may find $\lfloor c_n^* \rceil \neq 0$ also for $\tilde{c}_n^* \approx 0$.

Appendix

Formal proofs of key results

Proof of Theorem 12.1. The statement of the theorem is trivially satisfied for the special case $c_n^* = \lfloor c_n^* \rfloor = \lceil c_n^* \rceil \in \mathbb{Z}$. In all other cases, we have $\lfloor c_n^* \rfloor < c_n^* < \lceil c_n^* \rceil$. For these cases, we consider a relaxed version of the OP (12.4), where the constraint $\hat{c}_n \in \mathbb{Z}$ is replaced by

$$\hat{c}_n \in \{c \in \mathbb{R} \mid c \leq \lfloor c_n^* \rfloor\} \cup \{c \in \mathbb{R} \mid c \geq \lceil c_n^* \rceil\}. \tag{12.15}$$

Clearly, this OP can be solved by independently solving

$$\min_{\underline{c}_0,\ldots,\underline{c}_n} J(\underline{c}_0,\ldots,\underline{c}_n,x_1,\ldots,x_N) \quad \text{s.t.} \quad \underline{c}_n \leq \lfloor c_n^* \rfloor \tag{12.16}$$

and

$$\min_{\overline{c}_0,\ldots,\overline{c}_n} J(\overline{c}_0,\ldots,\overline{c}_n,x_1,\ldots,x_N) \quad \text{s.t.} \quad \overline{c}_n \geq \lceil c_n^* \rceil, \tag{12.17}$$

and then selecting the solution with the smaller cost function value. Now, let $\underline{\boldsymbol{c}}^*$ and $\overline{\boldsymbol{c}}^*$ be optimizers for (12.16) and (12.17), respectively, and assume $J(\underline{\boldsymbol{c}}^*) \leq J(\overline{\boldsymbol{c}}^*)$, where $J(\boldsymbol{c})$ is short for $J(c_0,\ldots,c_n,x_1,\ldots,x_N)$. Then, existence of an optimizer $\hat{\boldsymbol{c}}^*$ for (12.4) satisfying $\hat{c}_n^* = \lfloor c_n^* \rfloor$ can be shown as follows. First, convexity and feasibility of (12.16) implies that the Karush-Kuhn-Tucker (KKT) conditions are not only sufficient but also necessary for optimality. Hence, any optimizer $\underline{\boldsymbol{c}}^*$ satisfies

$$\nabla_{\underline{\boldsymbol{c}}} J(\underline{\boldsymbol{c}}^*) + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \lambda^* \end{pmatrix} = \mathbf{0}, \tag{12.18a}$$

$$\underline{c}_n^* - \lfloor c_n^* \rfloor \leq 0, \tag{12.18b}$$

$$\lambda^* \geq 0, \tag{12.18c}$$

$$\lambda^*(\underline{c}_n^* - \lfloor c_n^* \rfloor) = 0 \tag{12.18d}$$

with $\lambda^* \in \mathbb{R}$ reflecting the (optimal) Lagrange multiplier. Next, taking (12.18c) into account, we distinguish the two cases (i) $\lambda^* > 0$ and (ii) $\lambda^* = 0$. In the first case, we immediately find $\underline{c}_n^* = \lfloor c_n^* \rfloor \in \mathbb{Z}$. Hence, $\underline{\boldsymbol{c}}^*$ is also feasible for (12.4). Moreover, since the set on the right-hand side in (12.15) is a superset of $\mathbb{Z}$, $J(\underline{\boldsymbol{c}}^*) \leq J(\overline{\boldsymbol{c}}^*)$ excludes the existence of an $\hat{\boldsymbol{c}}^*$ satisfying $J(\hat{\boldsymbol{c}}^*) < J(\underline{\boldsymbol{c}}^*)$. Hence, $\hat{\boldsymbol{c}}^* := \underline{\boldsymbol{c}}^*$ is an optimizer for (12.4) with the desired property (i.e., $\hat{c}_n^* = \lfloor c_n^* \rfloor$). It remains to consider the case $\lambda^* = 0$. In this case, (12.18a) implies $\nabla_{\underline{\boldsymbol{c}}} J(\underline{\boldsymbol{c}}^*) = \mathbf{0}$. Clearly, we also $\nabla_{\underline{\boldsymbol{c}}} J(\boldsymbol{c}^*) = \mathbf{0}$ for the unconstrained OP (12.2). Hence, due to convexity of J, we deduce $J(\underline{\boldsymbol{c}}^*) = J(\boldsymbol{c}^*)$. Furthermore, convexity implies

$$J((1-\alpha)\boldsymbol{c}^* + \alpha\underline{\boldsymbol{c}}^*) \leq (1-\alpha)J(\boldsymbol{c}^*) + \alpha J(\underline{\boldsymbol{c}}^*) = J(\boldsymbol{c}^*) \tag{12.19}$$

for every $\alpha \in [0,1]$. Due to global optimality of $\boldsymbol{c}^*$, $J(\boldsymbol{c}^*)$ is also a lower bound for the left-hand side in (12.19) implying

$$J((1-\alpha)\boldsymbol{c}^* + \alpha\underline{\boldsymbol{c}}^*) = J(\boldsymbol{c}^*).$$

Now, taking $\underline{c}_n^* \leq \lfloor c_n^* \rfloor < c_n^*$ into account, it becomes clear that there exist an $\alpha \in (0,1]$ such that

$$\hat{\boldsymbol{c}}^* := (1-\alpha)\boldsymbol{c}^* + \alpha\underline{\boldsymbol{c}}^*$$

satisfies $\hat{c}_n^* = \lfloor c_n^* \rfloor \in \mathbb{Z}$ and $J(\hat{\boldsymbol{c}}^*) = J(\boldsymbol{c}^*)$. Hence, $\hat{\boldsymbol{c}}^*$ is an optimizer for (12.4) with the desired property. Finally, the case $J(\underline{\boldsymbol{c}}^*) > J(\overline{\boldsymbol{c}}^*)$ leading to an optimizer for (12.4) with $\hat{c}_n^* = \lceil c_n^* \rceil$ can be handled analogously. ■

Proof of Theorem 12.2. We already know from Theorem 12.1 that there exists an optimizer $\hat{\boldsymbol{c}}^*$ for (12.4) with $\hat{c}_n^* = \lfloor c_n^* \rfloor$ or $\hat{c}_n^* = \lceil c_n^* \rceil$. Hence, it remains to show that, given the specified symmetry, out of the two options, the nearest integer $\lfloor c_n^* \rceil$ reflects a solution. To this end, we define the two (non-negative) quantities $\Delta\underline{c}_n := c_n^* - \lfloor c_n^* \rfloor$ and $\Delta\overline{c}_n := \lceil c_n^* \rceil - c_n^*$, and first consider $\Delta\underline{c}_n \leq \Delta\overline{c}_n$. Then, $\lfloor c_n^* \rceil = \lfloor c_n^* \rfloor$. In order to prove the claim, we next show that there does not exist a $\overline{\boldsymbol{c}}$ with $\overline{c}_n = \lceil c_n^* \rceil$ such that $J(\overline{\boldsymbol{c}}) < J(\underline{\boldsymbol{c}})$ for every $\underline{\boldsymbol{c}}$ with $\underline{c}_n = \lfloor c_n^* \rfloor$. To do so, we assume such a $\overline{\boldsymbol{c}}$ exists and derive a contradiction. Since $0 \leq \Delta\underline{c}_n \leq \Delta\overline{c}_n$, there exists an $\alpha \in [0,1]$ such that $\alpha\Delta\overline{c}_n = \Delta\underline{c}_n$. Due to convexity of J, we further have

$$J((1-\alpha)\boldsymbol{c}^* + \alpha\overline{\boldsymbol{c}}) \leq (1-\alpha)J(\boldsymbol{c}^*) + \alpha J(\overline{\boldsymbol{c}}).$$

Taking $J(\boldsymbol{c}^*) \leq J(\overline{\boldsymbol{c}})$ due to optimality into account, we additionally obtain $J((1-\alpha)\boldsymbol{c}^* + \alpha\overline{\boldsymbol{c}}) \leq J(\overline{\boldsymbol{c}})$. Now, we define $\Delta\boldsymbol{c} := \overline{\boldsymbol{c}} - \boldsymbol{c}^*$ and note that $(1-\alpha)\boldsymbol{c}^* + \alpha\overline{\boldsymbol{c}} = \boldsymbol{c}^* + \alpha\Delta\boldsymbol{c}$. Thus, due to the symmetry of J, we find $J(\boldsymbol{c}^* + \alpha\Delta\boldsymbol{c}) = J(\boldsymbol{c}^* - \alpha\Delta\boldsymbol{c})$. It is easy to see that the $(n+1)$-th component of $\Delta\boldsymbol{c}$ equals $\Delta\overline{c}_n$. Thus, the $(n+1)$-th component of $\boldsymbol{c}^* - \alpha\Delta\boldsymbol{c}$ is $c_n^* - \alpha\Delta\overline{c}_n = c_n^* - \Delta\underline{c}_n = \lfloor c_n^* \rfloor$. In summary, $\underline{\boldsymbol{c}} := \boldsymbol{c}^* - \alpha\Delta\boldsymbol{c}$ is such that $\underline{c}_n = \lfloor c_n^* \rfloor$ and

$$J(\underline{\boldsymbol{c}}) = J(\boldsymbol{c}^* + \alpha\Delta\boldsymbol{c}) = J((1-\alpha)\boldsymbol{c}^* + \alpha\overline{\boldsymbol{c}}) \leq J(\overline{\boldsymbol{c}}),$$

which contradicts the assumption on $\overline{\boldsymbol{c}}$. The remaining case $\Delta\underline{c}_n > \Delta\overline{c}_n$ can be handled analogously. ■

Chapter 13

On the representation of piecewise quadratic functions by neural networks*

Dieter Teichrib† and Moritz Schulze Darup†

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

†Dieter Teichrib and Moritz Schulze Darup are with the Control and Cyberphysical Systems Group, Department of Mechanical Engineering, TU Dortmund University, Germany. E-mails: {dieter.teichrib,moritz.schulzedarup}@tu-dortmund.de

13.1 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 [67]. 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 [20]. There are several approaches [6, 16, P6] 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 [24, 25] and, thus, of the optimal control law [26, P3]. This connection between NNs and the optimal control law is used in [7] 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 [158, 159] 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 [10], [160], [161]. An application example is the use of model-based reinforcement learning (RL) or approximate dynamic programming (ADP) [3, 27, 76, 77] 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 [8, 27]. One of the first uses of ADP in control is presented [8], 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 [8, 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 [20]. Thus, when applying

ADP to constrained linear systems, a PWQ function has to be learned [27, 77]. In [77], 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 [27] 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 [27]. 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 [162], such as Q-learning [163, 164], 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 [20]. 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 [28], [82], [29], [165], [P2] on PWQ functions for special cases. In [P1] and [P2], 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 [28], [82], [29], 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 13.2-13.2.2. 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 13.2, 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 13.3. In Section 13.4, we use the results from Section 13.3 to derive maxout and relu NNs that allow an exact representation of PWQ functions with a polyhedral domain partition. In Section 13.5, 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 13.6 with a conclusion and a summary.

13.1.1 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 $\boldsymbol{I}_n$.

13.2 Preliminaries and background

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

$$\varphi(\boldsymbol{x}) := \begin{cases} \varphi^{(1)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ \varphi^{(s)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{13.1}$$

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

$$\mathcal{R}^{(i)} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^{(i)}\boldsymbol{x} + \boldsymbol{b}^{(i)} \leq \boldsymbol{0}\} \tag{13.2}$$

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 $i \in \{1, \ldots, s\}$. 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 $j \in \{1, \ldots, d^{(i)}\}$ and for all $i \in \{1, \ldots, s\}$. 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 $i \in \{1, \ldots, s\}$. Moreover, the function (13.1) 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

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

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

$$\mathcal{H}^{(p,r)} := \mathcal{R}^{(p)} \cap \mathcal{R}^{(r)} \subset \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)} = 0\}, \tag{13.3}$$

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 [82, 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

$$\varphi^{(p)}(\boldsymbol{x}) - \varphi^{(r)}(\boldsymbol{x}) = e^{(p,r)}(\boldsymbol{x}) g^{(p,r)}(\boldsymbol{x}) \tag{13.4}$$

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 (13.4) 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 $g^{(p,r)}(\boldsymbol{x})$ [82]. 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 13.3 to derive NNs for representing PWQ functions. Prior to this, we introduce our notion of NNs.

13.2.1 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

$$\boldsymbol{\Phi}(\boldsymbol{\xi}) = \boldsymbol{f}^{(l+1)} \circ \boldsymbol{g}^{(l)} \circ \boldsymbol{f}^{(l)} \circ \cdots \circ \boldsymbol{g}^{(1)} \circ \boldsymbol{f}^{(1)}(\boldsymbol{\xi}). \tag{13.5}$$

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 [21]). 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.,

$$\boldsymbol{f}^{(i)}(\boldsymbol{y}^{(i-1)}) = \boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{v}^{(i)}, \tag{13.6}$$

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

$$\boldsymbol{g}_{\mathrm{relu}}^{(i)}(\boldsymbol{z}^{(i)}) = \max\left\{\boldsymbol{0}, \boldsymbol{z}^{(i)}\right\} := \begin{pmatrix} \max\left\{0, z_1^{(i)}\right\} \\ \vdots \\ \max\left\{0, z_{w_i}^{(i)}\right\} \end{pmatrix} \tag{13.7}$$

and the maxout activation function

$$\boldsymbol{g}_{\max}^{(i)}(\boldsymbol{z}^{(i)}) = \begin{pmatrix} \max\limits_{1\leq j\leq p_i} \left\{z_j^{(i)}\right\} \\ \vdots \\ \max\limits_{p_i(w_i-1)+1\leq j\leq p_i w_i} \left\{z_j^{(i)}\right\} \end{pmatrix}, \tag{13.8}$$

where we use the shorthand notation $\max\limits_{1\leq j\leq p_i} \left\{z_j^{(i)}\right\} := \max\left\{z_1^{(i)}, \ldots, z_{p_i}^{(i)}\right\}$. 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 [25], [21] and thus not suitable for representing PWQ functions. However, in Section 13.4-13.4.1, we will modify these NNs such that they can be used to represent continuous PWQ functions.

13.2.2 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 [P1, P2, P6, 70], we investigate under which conditions we can find a continuous PWA function $h(\boldsymbol{x}) : \Omega \to \mathbb{R}$ of the form

$$h(\boldsymbol{x}) := \begin{cases} h^{(1)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(1)}, \\ \vdots & \vdots \\ h^{(s)}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}^{(s)}, \end{cases} \tag{13.9}$$

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

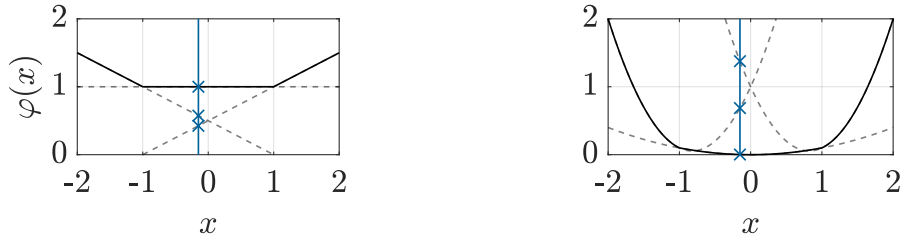
$$\varphi(\boldsymbol{x}) = \max_{1\leq i\leq s}\{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} - \max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\} \tag{13.10}$$

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

$$\tilde{\varphi}(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) \tag{13.11}$$

and $\tilde{\varphi}^{(i)}(\boldsymbol{x}) := \varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ for all $i \in \{1, \dots, s\}$. The representation (13.10) is particularly well-suited for our purpose due to several reasons. First, the difference between two max-expressions (13.10) can be represented by a maxout NN. Moreover, results on the connection between maxout NNs and relu NNs [25] can then be used to derive relu NN for representing (13.10). The connection between (13.10), relu NNs, and maxout NNs described in [24, 25, 70] motivates the focus on the activation functions (13.7) and (13.8). We discuss this connection in more detail in Section 13.4-13.4.1. Second, for certain functions, it is already known that a representation of the form (13.10) 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 [70]. 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 (13.10). In Figure 13.1-(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 [166]. However, for the PWQ case, it is not possible to represent the convex PWQ functions as the maximum of their local functions. In Figure 13.1-(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 (13.10) is not possible.



(a) **(b)**

Figure 13.1. 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 $\varphi^{(1)}(x)$, $\varphi^{(2)}(x)$, and $\varphi^{(3)}(x)$ outside their respective regions $\mathcal{R}^{(1)} = \{x \in \mathbb{R} | -2 \leq x \leq -1\}$, $\mathcal{R}^{(2)} = \{x \in \mathbb{R} | -1 \leq x \leq 1\}$, and $\mathcal{R}^{(3)} = \{x \in \mathbb{R} | 1 \leq x \leq 2\}$. 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 $\max\{\varphi^{(1)}(x), \varphi^{(2)}(x), \varphi^{(3)}(x)\} = \varphi(x)$.

Nevertheless, there are some results for the representation of continuous

PWQ functions. In [29], 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 $n-k$ [83]. 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 [82], 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 [83, Def. 5]. For our case this means that the function $g^{(\tilde{p},\tilde{r})}(\boldsymbol{x})$ from (13.4) 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 [81] and [P2], results on the representation of piecewise polynomial and PWQ functions, respectively, for $n = 1$ are presented. In [P2], it is additionally shown that for PWQ functions with $n = 1$, there exists a continuous and convex PWA $h(x)$ such that (13.10) holds.

In Section 13.4, 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 (13.10). 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.

13.2.3 Model predictive control

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

$$
\begin{aligned}
V_N(\boldsymbol{x}) := \min_{\substack{\boldsymbol{x}(0),\dots,\boldsymbol{x}(N)\\ \boldsymbol{u}(0),\dots,\boldsymbol{u}(N-1)}} \; & \ell_N(\boldsymbol{x}(N)) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}(k),\boldsymbol{u}(k)) && (13.12)\\
\text{s.t.} \quad & \boldsymbol{x}(0) = \boldsymbol{x},\\
& \boldsymbol{x}(k+1) = \boldsymbol{A}\,\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), && \forall k \in \{0,\dots,N-1\},\\
& (\boldsymbol{x}(k),\boldsymbol{u}(k)) \in \mathcal{X} \times \mathcal{U}, && \forall k \in \{0,\dots,N-1\},\\
& \boldsymbol{x}(N) \in \mathcal{T}
\end{aligned}
$$

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 [48] for details). For the considered setup, it is well-known that the control policy $\pi : \mathcal{F}_N \to \mathcal{U}$ with $\pi(\boldsymbol{x}) := \boldsymbol{u}^*(0)$ is a continuous PWA function of the form (13.9) with domain $\Omega = \mathcal{F}_N$ where $\mathcal{F}_N \subseteq \mathcal{X}$ is the feasible set of (13.12) [20, Thm. 4]. Moreover, the OVF $V_N(\boldsymbol{x})$ is a continuous and convex PWQ function of the form (13.1) with domain $\Omega = \mathcal{F}_N$.

Typically, when MPC is applied to a system, the OCP (13.12) 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 [20] or learning-based predictive control [7], where we exploit the PWA structure of the control policy or the PWQ structure of the OVF.

13.2.4 Learning-based predictive control

In principle, the PWA function describing the control policy $\pi(\boldsymbol{x})$ 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 (13.12) 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 (13.12) [20]. 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 (13.5) for an approximation in the policy space. In this case, samples $(\boldsymbol{x}, \pi(\boldsymbol{x}))$ of the PWA control policy $\pi(\boldsymbol{x})$ 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 [6, 7, 14, 16], for linear systems and in [167] for nonlinear systems. Moreover, in [7], 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 [10], [3], [27], where the PWQ OVF $V_N(\boldsymbol{x})$ or variants of it are approximated by NNs. In ADP [27, 77] the OVF is approximated based on samples $(\boldsymbol{x}, V_N(\boldsymbol{x}))$ for which the OCP (13.12) has been solved. Using the OVF an optimal input can be computed by $\boldsymbol{u}^*(\boldsymbol{x}) := \arg\min_{\boldsymbol{u}} \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u})$ [27]. 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., [3, P.

13])

$$Q_N(\boldsymbol{x}, \boldsymbol{u}) := \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{N-1}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}) \tag{13.13}$$

is learned iteratively by an NN with, e.g., Q-learning [163, 164], which is a variant of temporal difference learning [79], [11]. The Q-function (13.13) 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 (13.12), the OVF and, thus, the Q-function (13.13) are PWQ functions with a polyhedral domain partition [20, 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., [10] and [160]. However, in these papers, PWA NN are used to approximately represent PWQ functions. There are also approaches with PWQ NN. In [27] the authors propose a "local-global" topology in the form $\boldsymbol{x}^\top \boldsymbol{P}^* \boldsymbol{x} + \Phi(\boldsymbol{x})$ (cf. [27, 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 [27] 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 (13.10) 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 (13.10) 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.

13.3 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 (13.10). 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 (13.9).

13.3.1 Sufficient conditions

We start the derivation of sufficient conditions that ensure (13.10) by noting that if

$$\max_{1\leq i\leq s}\{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) \text{ and} \tag{13.14}$$

$$\max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\} = h(\boldsymbol{x}) \tag{13.15}$$

hold for all $\boldsymbol{x} \in \Omega$, then we have $\max_{1\leq i\leq s}\{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} - \max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}) + h(\boldsymbol{x}) - h(\boldsymbol{x}) = \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ and thus we can represent the PWQ function $\varphi(\boldsymbol{x})$ in the desired form (13.10). Therefore, in the remainder of this section, we will investigate under which conditions the PWA function $h(\boldsymbol{x})$ is such that (13.14) and (13.15) hold. We first present an intermediate result, which we will use to prove that (13.14) holds for all $\boldsymbol{x} \in \Omega \subset \mathbb{R}^n$.

Lemma 13.1. *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* (13.1), (13.9), *and* (13.11), *respectively. Consider*

$$d^{(p,r)} = \begin{cases} \frac{\tilde{c}^{(p)} - \tilde{c}^{(r)}}{b^{(p,r)}} & \text{if } b^{(p,r)} \neq 0, \\ \frac{\tilde{l}_t^{(p)} - \tilde{l}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} & \text{if } b^{(p,r)} = 0 \end{cases} \tag{13.16}$$

and

$$\boldsymbol{k}_q^{(p,r)} = \frac{1}{\boldsymbol{h}_t^{(p,r)}}\left(2\Delta\boldsymbol{Q}_{t,q}^{(p,r)} - \frac{\boldsymbol{h}_q^{(p,r)}}{\boldsymbol{h}_t^{(p,r)}}\Delta\boldsymbol{Q}_{t,t}^{(p,r)}\right) \tag{13.17}$$

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

$$\begin{cases} \frac{\gamma^{(p)} - \gamma^{(r)}}{b^{(p,r)}} \leq \frac{c^{(r)} - c^{(p)}}{b^{(p,r)}} - \boldsymbol{k}^{(p,r)}\boldsymbol{v}_l{}^\top & \text{if } b^{(p,r)} \neq 0, \\ \frac{\boldsymbol{\beta}_t^{(p)} - \boldsymbol{\beta}_t^{(r)}}{\boldsymbol{h}_t^{(p,r)}} \leq \frac{\boldsymbol{l}_t^{(r)} - \boldsymbol{l}_t^{(p)}}{\boldsymbol{h}_t^{(p,r)}} - \boldsymbol{k}^{(p,r)}\boldsymbol{v}_l{}^\top & \text{if } b^{(i,j)} = 0 \end{cases} \tag{13.18}$$

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,

$$\begin{aligned} \Delta\tilde{\varphi}^{(p,r)}(\boldsymbol{x}) :=& \tilde{\varphi}^{(p)}(\boldsymbol{x}) - \tilde{\varphi}^{(r)}(\boldsymbol{x}) \\ =& \left(\boldsymbol{x}^\top \boldsymbol{h}^{(p,r)}{}^\top + b^{(p,r)}\right)\left(\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)}\right) \end{aligned} \tag{13.19}$$

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

$$\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)} \leq 0 \tag{13.20}$$

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

Proof. According to [82, Thm. 1] for

$$\Delta\tilde{\varphi}^{(p,r)}(\boldsymbol{x}) = \boldsymbol{x}^\top \Delta\boldsymbol{Q}^{(p,r)}\boldsymbol{x} + \left(\tilde{\boldsymbol{l}}^{(p)} - \tilde{\boldsymbol{l}}^{(r)}\right)\boldsymbol{x} + \tilde{c}^{(p)} - \tilde{c}^{(r)} \tag{13.21}$$

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

$$\begin{aligned}
&\boldsymbol{x}^\top \Delta\boldsymbol{Q}^{(p,r)}\boldsymbol{x} + \left(\tilde{\boldsymbol{l}}^{(p)} - \tilde{\boldsymbol{l}}^{(r)}\right)\boldsymbol{x} + \tilde{c}^{(p)} - \tilde{c}^{(r)} \\
=&\left(\boldsymbol{x}^\top {\boldsymbol{h}^{(p,r)}}^\top + b^{(p,r)}\right)\left(\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)}\right) \\
=&\boldsymbol{x}^\top \frac{1}{2}\left({\boldsymbol{h}^{(p,r)}}^\top \boldsymbol{k}^{(p,r)} + {\boldsymbol{k}^{(p,r)}}^\top \boldsymbol{h}^{(p,r)}\right)\boldsymbol{x} \\
&+ \left(\boldsymbol{h}^{(p,r)} d^{(p,r)} + \boldsymbol{k}^{(p,r)} b^{(p,r)}\right)\boldsymbol{x} + b^{(p,r)} d^{(p,r)}
\end{aligned} \tag{13.22}$$

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 (13.3). Now, by comparing the coefficients of the right-hand and left-hand side of the quadratic functions in (13.22), we can identify

$$\frac{1}{2}\left({\boldsymbol{h}^{(p,r)}}^\top \boldsymbol{k}^{(p,r)} + {\boldsymbol{k}^{(p,r)}}^\top \boldsymbol{h}^{(p,r)}\right) = \Delta\boldsymbol{Q}^{(p,r)} \tag{13.23a}$$

$$\boldsymbol{h}^{(p,r)} d^{(p,r)} + \boldsymbol{k}^{(p,r)} b^{(p,r)} = \tilde{\boldsymbol{l}}^{(p)} - \tilde{\boldsymbol{l}}^{(r)} \tag{13.23b}$$

$$b^{(p,r)} d^{(p,r)} = \tilde{c}^{(p)} - \tilde{c}^{(r)}. \tag{13.23c}$$

With (13.23b) and (13.23c) we can write $d^{(p,r)}$ as (13.16). Note that for (13.16) 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 (13.3) 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 (13.18) as

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

and subsequently substitute the reformulation of $d^{(p,r)}$ by (13.16), which leads to

$$\boldsymbol{k}^{(p,r)} {\boldsymbol{v}_l}^\top + d^{(p,r)} \leq 0 \tag{13.24}$$

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 (13.18) can be expressed in terms of the coefficients of the functions (13.1) and (13.9). By using (13.23a) we find $\boldsymbol{k}_t^{(p,r)} = {}^{\Delta}\boldsymbol{Q}_{t,t}^{(p,r)} / \boldsymbol{h}_t^{(p,r)}$ 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 (13.17). Now, since (13.24) holds for all vertices $\boldsymbol{v}_l$ of the polyhedron Ω we have (13.20) 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 13.1 to prove that (13.14) holds. This requires to show

$$\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x}) := \tilde{\varphi}^{(i)}(\boldsymbol{x}) - \tilde{\varphi}^{(j)}(\boldsymbol{x}) \geq 0 \tag{13.25}$$

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 (13.19) from Lemma 13.1 to decompose the quadratic function $\Delta\tilde{\varphi}^{(p,r)}(\boldsymbol{x})$ into two affine factors. Then, to prove (13.25), 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 Figure 13.2. Thus, we cannot directly use Lemma 13.1 to prove (13.25) for all regions. Fortunately, it is possible to rewrite $\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x})$ in terms of differences of neighbouring regions by

$$\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x}) = \sum_{k=1}^{K} \Delta\tilde{\varphi}^{(p_k,r_k)}(\boldsymbol{x}) \tag{13.26}$$

with $(p_k,r_k) \in \mathcal{N}$ for all $k \in \{1,\dots,K\}$. This allows us to use (13.19) from Lemma 13.1 for every summand in (13.26). 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 (13.26) to be a valid reformulation of $\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x})$, we need

$$p_K = i, \quad r_1 = j \quad \text{and} \quad p_k = r_{k+1} \tag{13.27}$$

for all $k \in \{1,\dots,K-1\}$. As illustrated in Figure 13.2-(a) and 13.2-(b), there are several possible ways for choosing a valid $\mathcal{C}^{(i,j)}$ that satisfies (13.27). The paths from Figure 13.2-(a) and 13.2-(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 Figure 13.2-(a), this results in $\mathcal{C}^{(4,2)} = \{(3,2),(4,3)\}$ and for the path from Figure 13.2-(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 13.2. *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* (13.26) *and*

$$\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)} \leq 0 \tag{13.28}$$

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 13.2 may or may not hold. In Figure 13.2 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 (13.26) and (13.28) 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 Figure 13.2-(a) and 13.2-(b) illustrate the two possible ways for determining $\mathcal{C}^{(4,2)}$. For Fingure 13.2-(a) and 13.2-(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 Figure 13.2-(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 Figure 13.2-(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 (13.28) does not hold for all $\boldsymbol{x} \in \mathcal{R}^{(4)}$. Thus, Assumption 13.2 does not hold for the partition shown in Figure 13.2.

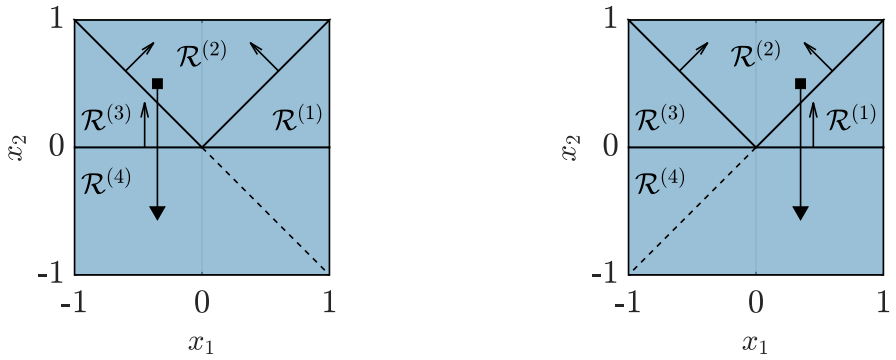
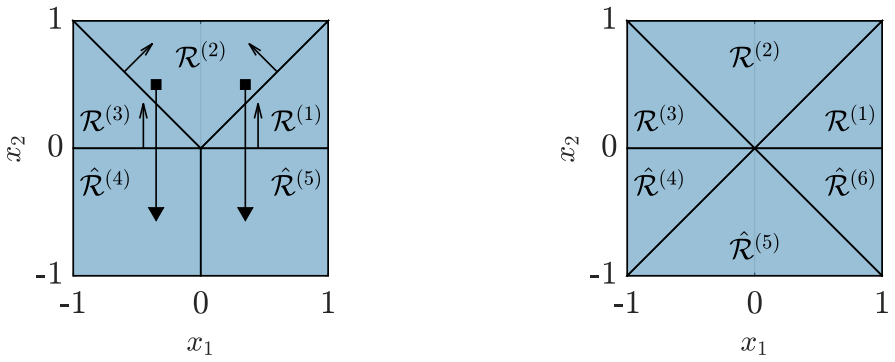


Figure 13.2. Illustration of an exemplary domain partition with paths from $\mathcal{R}^{(2)}$ to $\mathcal{R}^{(4)}$. Each path suggests a way of expressing the difference $\Delta\tilde{\varphi}^{(4,2)}(\boldsymbol{x}) = \tilde{\varphi}^{(4)}(\boldsymbol{x}) - \tilde{\varphi}^{(2)}(\boldsymbol{x})$ in terms of differences of neighbouring regions by (13.26). 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 $\boldsymbol{h}^{(1,4)}$, $\boldsymbol{h}^{(2,1)}$, $\boldsymbol{h}^{(2,3)}$, and $\boldsymbol{h}^{(3,4)}$ of the hyperplanes crossed by the paths.

However, if we add the hyperplane $\{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{x}_1 = 0\}$ in $\mathcal{R}^{(4)}$ as illustrated in Figure 13.3-(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 Figure 13.2 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 Figure 13.3-(a), for both sets we have (13.26) 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 $i \in \{1,\ldots,s\}$ and $j \in \{1,\ldots,s\} \setminus i$ we are able to find a set $\mathcal{C}^{(i,j)}$ such that (13.26) and (13.28) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p,r) \in \mathcal{C}^{(i,j)}$. Thus, Assumption 13.2 holds for the modified partition.

In Figure 13.3-(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 13.2 holds.



(a) **(b)**

Figure 13.3. Illustration of two possible refinements of the partition shown in Figure 13.2. The partitions in (a) and (b) are both such that Assumption 13.2 holds. The arrows on the boundaries between the regions in (a) represent the normal vectors of the hyperplanes crossed by the paths.

As shown in Figure 13.2 and 13.3, if we have a PWQ function $\varphi(\boldsymbol{x})$ for which Assumption 13.2 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 13.2. Adding of hyperplanes can be done as in Figure 13.3-(a), i.e., by carefully adding hyperplanes such that Assumption 13.2 holds for a specific partition. Fortunately, as we will show in the following proposition, the approach illustrated in Figure 13.3-(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 13.2 holds. For ease of reading, the proof is provided in the appendix.

Proposition 13.3. *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 13.2 holds.*

Intuitively, Proposition 13.3 states that Assumption 13.2 holds if all hyperplanes of the partition are extended to the whole domain, as shown in Figure 13.3-(b). This extension of all hyperlanes typically leads to a partition with a large number of regions s. However, according to Proposition 13.3 the extension of all hyperplanes is only sufficient for Assumption 13.2 to hold. As illustrated in Figure 13.3-(a), we may be able to find a refinement of the partition with fewer regions for which Assumption 13.2 holds. This will lead to a representation of the form (13.10) with a smaller s.

We will now use Lemma 13.1 to show that the factors (13.20) and (13.28) of each summand (13.19) in (13.26) are negative. This will help us to show that (13.25) holds, which is sufficient for (13.14).

Lemma 13.4. *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* (13.1), (13.9), *and* (13.11), *respectively. Assume that* (13.18) *as in Lemma 13.1 holds for all $l \in \{1, \dots, V\}$ and for all $(p, r) \in \mathcal{N}$. Then,* (13.14) *holds for all $\boldsymbol{x} \in \Omega$.*

Proof. To ensure (13.14) we need

$$\Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x}) \geq 0 \tag{13.29}$$

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 (13.29) as the sum over differences of neighbouring regions using (13.26). According to Lemma 13.1 for the summands in (13.26) we have

$$\Delta\tilde{\varphi}^{(p,r)}(\boldsymbol{x}) = (\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)})(\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)}) \tag{13.30}$$

with (13.20) 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 (13.20) holds for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(i,j)}$. We further have (13.28) for all $\boldsymbol{x} \in \mathcal{R}^{(i)}$ and for all $(p, r) \in \mathcal{C}^{(i,j)}$ by Assumption 13.2. Consequently, for all $j \in \{1, \dots, s\} \setminus \{i\}$ and for all $i \in \{1, \dots, s\}$ the inequality

$$\underbrace{(\boldsymbol{h}^{(p,r)}\boldsymbol{x} + b^{(p,r)})}_{\leq 0 \forall \boldsymbol{x} \in \mathcal{R}^{(i)}} \underbrace{(\boldsymbol{k}^{(p,r)}\boldsymbol{x} + d^{(p,r)})}_{\leq 0 \forall \boldsymbol{x} \in \mathcal{R}^{(i)}} \geq 0 \tag{13.31}$$

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 (13.26) holds, we have

$$\begin{aligned} \Delta\tilde{\varphi}^{(i,j)}(\boldsymbol{x}) &= \sum_{k=1}^{K} \Delta\tilde{\varphi}^{(p_k,r_k)}(\boldsymbol{x}) \\ &= \sum_{k=1}^{K} (\boldsymbol{h}^{(p_k,r_k)}\boldsymbol{x} + b^{(p_k,r_k)})(\boldsymbol{k}^{(p_k,r_k)}\boldsymbol{x} + d^{(p_k,r_k)}) \geq 0 \end{aligned}$$

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 (13.1), (13.9), and (13.11) 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 13.5. *The condition* (13.18)*, which is sufficient for a representation in the form* (13.14)*, 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* (13.1) *with* $\boldsymbol{Q}^{(i)} = \boldsymbol{0}$ *for all* $i \in \{1, \ldots, s\}$*, a representation in the form*

$$\varphi(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{\varphi^{(i)}(\boldsymbol{x})\} \tag{13.32}$$

is possible if $\varphi(\boldsymbol{x})$ *is convex [70]. For this special case, Lemma 13.4 states that* (13.32) *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* (13.16) *the former is equivalent to*

$$d^{(p,r)} \leq 0 \tag{13.33}$$

for all $(p, r) \in \mathcal{N}$*. Using [82, 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* (13.33) *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 13.4 and leads to the result known from the literature [70].*

We will now show how to find a PWA function $h(\boldsymbol{x})$ such that (13.18) holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \ldots, V\}$. 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 (13.10).

13.3.2 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 (13.9) such that the function (13.11) can be represented as the maximum of its local functions by (13.14). According to Lemma 13.4, a sufficient condition for such a representation is (13.18). Therefore, we will show in the following that it is always possible to construct a convex lifting [110] via $h(\boldsymbol{x})$ such that (13.18) and consequently (13.14) 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 [110, Thm. III.9], i.e, if and only if the OP [110, 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, (13.15) holds, then, by combining (13.14) and (13.15), we can show that every PWQ function can be represented as the difference of two max-expressions by (13.10). To prove the next results, we assume from now on that the following assumption holds.

Assumption 13.6. *The polyhedral domain partition* $\cup_{i=1}^{s} \mathcal{R}^{(i)} = \Omega$ *of* $\varphi(\boldsymbol{x})$ *as in* (13.1) *is convexly liftable.*

Remark 13.7. *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 [110, 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 [110, Thm. III.12], i.e., Assumption 13.6 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* $\boldsymbol{x} \in \mathcal{R}^{(i)}$ *[110, Alg. 4]. Thus, Proposition 13.3 applies and consequently Assumption 13.2 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 13.2 and 13.6 hold. Thus, we can pose both assumptions without loss of generality.*

In the following, we show that the condition (13.18), which is sufficient for (13.14), can always be satisfied by using a continuous and convex PWA function $h(\boldsymbol{x})$.

Lemma 13.8. *Let the PWQ function* $\varphi(\boldsymbol{x})$ *and* $\tilde{\varphi}(\boldsymbol{x}) := \varphi(\boldsymbol{x}) + h(\boldsymbol{x})$ *be defined as in* (13.1) *and* (13.11)*, respectively. Then, there exists a continuous and convex PWA function* $h(\boldsymbol{x})$ *as in* (13.9) *such that* (13.18) *holds for all* $l \in \{1, \dots, V\}$ *and for all* $(p, r) \in \mathcal{N}$*.*

Proof. For a convexly liftable partition, [110, Thm. III.9] states that for all $\delta > 0$, there exists a continuous and convex PWA function of the form (13.9) with

$$\boldsymbol{\beta}^{(p)} \boldsymbol{x}^{(p)} + \gamma^{(p)} \geq \boldsymbol{\beta}^{(r)} \boldsymbol{x}^{(p)} + \gamma^{(r)} + \delta \tag{13.34}$$

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

$$(\boldsymbol{\beta}^{(p)} - \boldsymbol{\beta}^{(r)}) \boldsymbol{x} + \gamma^{(p)} - \gamma^{(r)} = \rho^{(p,r)} (\boldsymbol{h}^{(p,r)} \boldsymbol{x} + b^{(p,r)}) \tag{13.35}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and for all $(p, r) \in \mathcal{N}$ according to [82, Thm. 1]. By comparing the coefficients of the right-hand and left-hand side of the affine functions in (13.35), we find

$$\boldsymbol{\beta}^{(p)} - \boldsymbol{\beta}^{(r)} = \rho^{(p,r)} \boldsymbol{h}^{(p,r)} \quad \text{and} \tag{13.36a}$$

$$\gamma^{(p)} - \gamma^{(r)} = \rho^{(p,r)} b^{(p,r)}. \tag{13.36b}$$

Thus, we can identify

$$\rho^{(p,r)} = \begin{cases} \frac{\gamma^{(p)}-\gamma^{(r)}}{b^{(p,r)}} & \text{if } b^{(p,r)} \neq 0, \\ \frac{\beta_t^{(p)}-\beta_t^{(r)}}{h_t^{(p,r)}} & \text{if } b^{(p,r)} = 0. \end{cases} \tag{13.37}$$

Substituting $\beta^{(p)} = \beta^{(r)} + \rho^{(p,r)} h^{(p,r)}$ and $\gamma^{(p)} = \gamma^{(r)} + \rho^{(p,r)} b^{(p,r)}$ from (13.36a) and (13.36b), respectively in the left-hand side of (13.34) gives $\beta^{(p)} x^{(p)} + \gamma^{(p)} = (\beta^{(r)} + \rho^{(p,r)} h^{(p,r)}) x^{(p)} + \gamma^{(r)} + \rho^{(p,r)} b^{(p,r)}$ which results in

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

and thus

$$\begin{aligned} &\rho^{(p,r)} \underbrace{(h^{(p,r)} x^{(p)} + b^{(p,r)})}_{<0 \text{ for } x^{(p)} \in \operatorname{int}(\mathcal{R}^{(p)})} \geq \delta > 0 \\ \Leftrightarrow\ &\rho^{(p,r)} \leq \frac{\delta}{h^{(p,r)} x^{(p)} + b^{(p,r)}} < 0 \end{aligned} \tag{13.38}$$

is feasible for all $\delta > 0$. With (13.37) and (13.38) we can use

$$\begin{cases} \frac{\gamma^{(p)}-\gamma^{(r)}}{b^{(p,r)}} \leq \frac{\delta}{h^{(p,r)} x^{(p)} + b^{(p,r)}} < 0 & \text{if } b^{(p,r)} \neq 0, \\ \frac{\beta_t^{(p)}-\beta_t^{(r)}}{h_t^{(p,r)}} \leq \frac{\delta}{h^{(p,r)} x^{(p)} + b^{(p,r)}} < 0 & \text{if } b^{(p,r)} = 0. \end{cases} \tag{13.39}$$

to overestimate the left-hand side of (13.18). If we now take into account that the inequality (13.39) is feasible for all $\delta > 0$ and that the right-hand side in (13.18) is finite then we can always choose a $\delta > 0$ such that

$$\begin{cases} \frac{\delta}{h^{(p,r)} x^{(p)} + b^{(p,r)}} \leq \frac{c^{(r)}-c^{(p)}}{b^{(p,r)}} - k^{(p,r)} v_l{}^\top & \text{if } b^{(p,r)} \neq 0, \\ \frac{\delta}{h^{(p,r)} x^{(p)} + b^{(p,r)}} \leq \frac{l_t^{(r)}-l_t^{(p)}}{h_t^{(p,r)}} - k^{(p,r)} v_l{}^\top & \text{if } b^{(p,r)} = 0 \end{cases} \tag{13.40}$$

holds for all $(p,r) \in \mathcal{N}$ and for all $l \in \{1,\dots,V\}$. In combination with (13.39) this means that there exists a continuous and convex PWA function $h(x)$ such that (13.18) holds for all $l \in \{1,\dots,V\}$ and for all $(p,r) \in \mathcal{N}$. ■

Remark 13.9. *Lemma 13.8 states that it is always possible to compute a convex lifting via a PWA function $h(x)$ such that* (13.18) *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 [110, Alg. 3] with a $c := \delta$ for which $\delta > 0$ and* (13.40) *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 13.4 and 13.8, 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 (13.14) holds for all $x \in \Omega$. In the following section, we will use this result to prove that PWQ functions can be represented as the difference of two max-expressions.

13.4 Representation and approximations of PWQ functions

In Section 13.3, 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 (13.10) for arbitrary continuous PWQ functions $\varphi(\boldsymbol{x})$ with polyhedral partition. The two results we will use to prove the existence are Lemma 13.4 and 13.8. 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 (13.14). Moreover, since $h(\boldsymbol{x})$ is a convex PWA function, it can be represented as the maximum of its local affine functions by (13.15). Then, by combining (13.14) and (13.15), we can eventually prove the existence of (13.10).

Theorem 13.10. *For every continuous PWQ function $\varphi(\boldsymbol{x})$ of the form* (13.1) *with a polyhedral domain partition as in* (13.2)*, there exists a PWA function $h(\boldsymbol{x})$ as in* (13.9) *such that*

$$\varphi(\boldsymbol{x}) = \max_{1 \leq i \leq s} \{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\} - \max_{1 \leq i \leq s} \{h^{(i)}(\boldsymbol{x})\} \tag{13.41}$$

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

Proof. Lemma 13.8 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 (13.18) holds for all $l \in \{1, \ldots, V\}$ and for all $(p, r) \in \mathcal{N}$. Then, we can deduce from Lemma 13.4 that (13.14) holds for all $\boldsymbol{x} \in \Omega$. Moreover, for continuous and convex PWA functions $h(\boldsymbol{x})$, we have (13.15) for all $\boldsymbol{x} \in \Omega$ according to [70]. 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 (13.41) will serve as a starting point for deriving NNs that allow an exact representation of arbitrary continuous PWQ functions with a polyhedral domain partition.

13.4.1 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 13.10. Then, based on the derived maxout NN, we use methods for decomposing max-expressions to derive suitable relu NNs.

13.4.1.1 Maxout neural networks

Theorem 13.10 can directly be used to derive maxout NNs that allow an exact representation of PWQ functions by exploiting the structure of (13.41). 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.,

$$\boldsymbol{\xi}(\boldsymbol{x}) := \begin{pmatrix} \boldsymbol{x}^\top & x_1^2 & x_1x_2 & \dots & x_1x_n & \dots & x_n^2 \end{pmatrix}^\top. \tag{13.42}$$

Theorem 13.11. *Every continuous PWQ function of the form* (13.1) *with a polyhedral domain partition as in* (13.2) *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* (13.42)*, postactivation weights $\boldsymbol{W}_1^{(2)} = 1$, $\boldsymbol{W}_2^{(2)} = -1$, and postactivtion bias $v^{(2)} = 0$.*

Proof. According to (13.5), (13.6), and (13.8), 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 (13.41) are linear combinations of 1 and the monomials contained in $\boldsymbol{\xi}$ from (13.42). 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 13.10, 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 13.4 shows the structure of the maxout NN from Theorem 13.11 with the extended input vector (13.42) and two neurons. The first neuron represents (13.14), and the second represents (13.15).

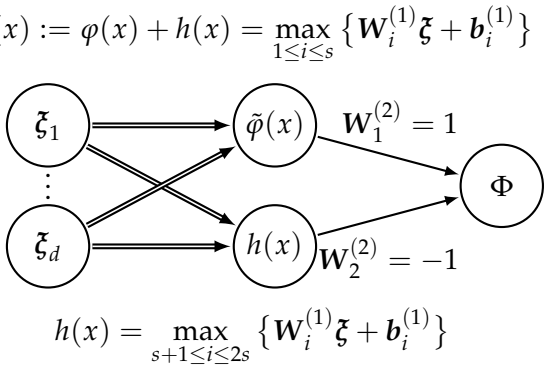


Figure 13.4. Maxout NN with two neurons and extended input vector (13.42) of dimension $d := n^2+3n/2$ for the exact representation of PWQ functions. The double arrows indicate the multi-channel preactivation with $p_1 = s$, i.e, the s elements in each max-expression (13.41).

13.4.1.2 Relu neural networks

For deriving relu NNs that allow an exact representation of PWQ functions, we can use results from [25], 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 13.12. *Every continuous PWQ Function of the form* (13.1) *with a polyhedral domain partition as in* (13.2) *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* ξ *as in* (13.42)*.*

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

$$\max\{x_1,x_2\} = \max\{x_1 - x_2,0\} + \max\{x_2,0\} - \max\{-x_2,0\}.$$

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

$$f(x) := (\max\{x_1,x_2\},\max\{x_3,x_4\},\dots,\max\{x_{2^n-1},x_{2^n}\})^\top$$

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 $f(x)$ is a relu NN with width 3. Now, we can compute the maximum $\max\{x_1,\dots,x_{2^n}\}$ by calling n-times the function $f(x)$. According to [25, Lem. D.1], the composition

$$\underbrace{f \circ \cdots \circ f}_{n \text{ times}}(x) = \max\{x_1,\dots,x_{2^n}\}$$

can be represented by a relu NN of depth n and width $3 \times 2^{n-1}$ since $f(x)$ 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 (13.41) from Theorem 13.10 and $\boldsymbol{\xi}$ as in (13.42), 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 [25, Lem. D.2]. ■

Theorem 13.12 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 [24], 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 13.13. *Every continuous PWQ Function of the form* (13.1) *with a polyhedral domain partition as in* (13.2) *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* (13.42).

Proof. The quadratic local functions $\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})$ and the linear local functions $h^{(i)}(\boldsymbol{x})$ in (13.41) are linear combinations of 1 and the $d := \frac{n^2+3n}{2}$ input variables $\boldsymbol{\xi}$ from (13.42). Thus, according to Theorem 13.10, 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

$$\begin{aligned} &\max_{1 \le i \le s} \{\boldsymbol{K}_i \boldsymbol{\xi} + \boldsymbol{k}_i\} - \max_{1 \le i \le s} \{\boldsymbol{L}_i \boldsymbol{\xi} + \boldsymbol{l}_i\} \\ &= \max_{1 \le i \le s} \{\varphi^{(i)}(\boldsymbol{x}) + h^{(i)}(\boldsymbol{x})\} - \max_{1 \le i \le s} \{h^{(i)}(\boldsymbol{x})\} = \varphi(\boldsymbol{x}). \end{aligned} \tag{13.43}$$

Moreover, since the domain Ω of the PWQ function (13.1) is polyhedral, the affine mapping

$$\boldsymbol{T}_j(\boldsymbol{\xi}) = \frac{\boldsymbol{\xi}_j - \min_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j}{\max_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j - \min_{\boldsymbol{x} \in \Omega} \boldsymbol{\xi}_j}$$

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 (13.43), 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 [24, Thm. 2] since $\hat{\varphi}(\boldsymbol{\zeta})$ is a function that maps from $[0,1]^d$ to $\mathbb{R}_+$. Moreover, we then have

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

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 $\hat{\varphi}(\boldsymbol{\zeta}) = \max_{1\leq i\leq s}\{\hat{\boldsymbol{K}}_i\boldsymbol{\zeta} + \hat{\boldsymbol{k}}_i\} - \max_{1\leq i\leq s}\{\hat{\boldsymbol{L}}_i\boldsymbol{\zeta} + \hat{\boldsymbol{l}}_i\}$. According to [25, 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 Theorem 13.12 and 13.13, we can see that the NN from Theorem 13.12 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 13.13 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 Theorem 13.12 and 13.13, respectively. Thus, for small $s \leq 2^{\frac{n^2}{3}+n+2}$, the NN from Theorem 13.12 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 13.13 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 [168] for approximating complex functions.

13.4.2 Approximations of PWQ functions

In Section 13.4-13.4.1, 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 (13.42). 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 13.4-13.4.1 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 13.11, 13.12, and 13.13 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

$$e^* := \min_{\boldsymbol{\theta}} \sum_{i=1}^{N_D} \ell_{\mathrm{NN}} \left(y^{(i)} - \Phi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \right) \tag{13.44}$$

with a convex loss function ℓ_{NN} approximately using a gradient-based optimization algorithm, e.g., stochastic gradient descent, RMSProb, or Adam [31] 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 13.11, 13.12, and 13.13 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 13.5-13.5.2, 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.

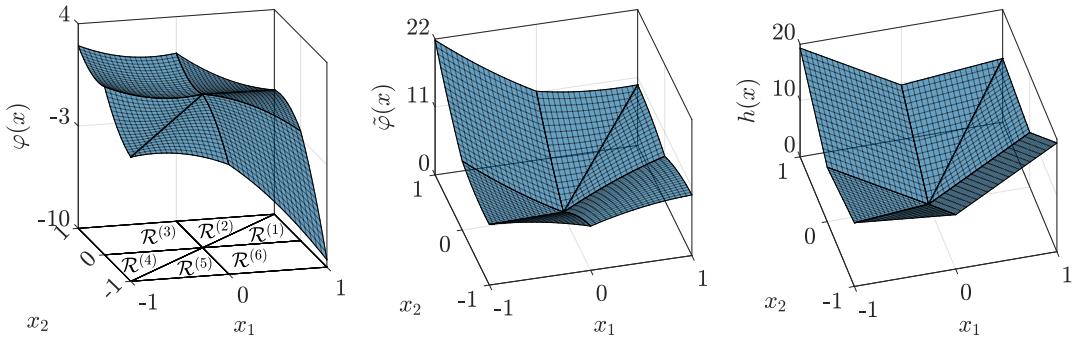
Another way to use the results of Section 13.4-13.4.1 for computing low error approximations of PWQ functions is to use the fact that for quadratic loss functions ℓ_{NN} and the maxout NN from Theorem 13.11, the OP (13.44) can be reformulated as a mixed-integer quadratic program (MIQP) using the results from our preliminary study [P6]. The MIQP can be solved globally using standard software, e.g., MOSEK [45] or Gurobi [46].

Corollary 13.14 ([P6, Thm. 1])**.** *For a maxout NN as in Theorem 13.11, the OP* (13.44) *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 13.11). 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 (13.44) 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 $i \in \{1, \dots, N_D\}$.

13.5 Numerical examples

We will provide different numerical examples. In the first example, we illustrate the representation (13.10) for an exemplary PWQ function with $n = 2$. In the second part of this section, we compare the in Section 13.4 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 (13.12) for different example systems.



(a) **(b)** **(c)**

Figure 13.5. Exemplary PWQ function $\varphi(\boldsymbol{x})$ with six regions in (a). The functions $\tilde{\varphi}(\boldsymbol{x})$ in (b) and $h(\boldsymbol{x})$ in (c) are such that they can be represented in the form (13.14) and (13.15), respectively, i.e., we have $\varphi(\boldsymbol{x}) = \tilde{\varphi}(\boldsymbol{x}) - h(\boldsymbol{x}) = \max_{1\leq i\leq s}\{\tilde{\varphi}^{(i)}(\boldsymbol{x})\} - \max_{1\leq i\leq s}\{h^{(i)}(\boldsymbol{x})\}$. The PWA function $h(\boldsymbol{x})$ in (c) is computed as described in Remark 13.9 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).

13.5.1 Exact representation of PWQ functions

In the first example, we illustrate the theoretical results of Section 13.3 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 (13.2) 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 Figure 13.5-(a). We choose the function such that the domain partition is degenerate according to the definition from [29] and the description in Section 13.2-13.2.2 since at $\boldsymbol{x}^\top = \begin{pmatrix} 0 & 0 \end{pmatrix}$, we have an intersection of $k = 3$ hyperplanes that is not an empty set. In addition, the consistent variation property (CVP) [82] 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 (13.17) is $\boldsymbol{k}^{(1,6)} = \begin{pmatrix} 4 & -4 \end{pmatrix} \neq \begin{pmatrix} 0 & 4 \end{pmatrix} = \boldsymbol{k}^{(3,4)}$ 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 13.10 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 (13.10). For determining the representation of $\varphi(\boldsymbol{x})$ in the form (13.10), we first compute a continuous and convex PWA function $h(\boldsymbol{x})$ such that (13.18) holds for all $(p, r) \in \mathcal{N}$ and for all $l \in \{1, \dots, V\}$, which is according to Lemma 13.8 always possible. We compute the function $h(\boldsymbol{x})$ as described in Remark 13.9 by computing a convex lifting according to [110, Alg. 3] with a $c := \delta$ for which $\delta > 0$ and (13.40) 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

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

shown in Figure 13.5-(c). Since $h(\boldsymbol{x})$ is now such that (13.18) 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 13.4. 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 13.10, 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 Figure 13.5.

13.5.2 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 (13.12) for different example systems using the NNs from Section 13.4-13.4.1. The considered example systems, together with the parameters for the OCP (13.12) and the number of regions of the corresponding OVF, are summarized in Table 13.1. The number of regions s in Table 13.1 is determined by explicitly computing the OVF of (13.12) using the multi-parametric toolbox 3 (MPT3) [121]. For each example system from Table 13.1, we solved the OCP (13.12) for different values of $\boldsymbol{x}$ on a regular grid with a step size of 0.1 to generate the data sets $\mathcal{D}_{\xi} := \{(\boldsymbol{\xi}^{(i)}, y^{(i)}) \mid y^{(i)} = V_N(\boldsymbol{\xi}^{(i)}) \forall i \in \{1, \ldots, N_D\}\}$ with an extended input vector $\boldsymbol{\xi}$ as in (13.42). This results in data sets with 6411, 10199, and 31690 data points for examples 1., 2., and 3., respectively, from Table 13.1. As is common in the machine learning literature [169], 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

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

for all $j \in \{1, \ldots, \frac{n^2+3n}{2}\}$. Afterwards, we trained the NNs listed in Table 13.2 with the corresponding normalized data set $\hat{\mathcal{D}}_{\xi} := \{(\hat{\boldsymbol{\xi}}^{(1)}, \hat{y}^{(1)}), \ldots, \hat{\boldsymbol{\xi}}^{(N_D)}, \hat{y}^{(N_D)})\}$ using the stochastic gradient descent optimizer from the Keras API [170] with a learning rate of 0.01 and a momentum of 0.9. For each case from Table 13.2, 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 [171]. 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 13.2. The MAEs of the different cases from Table 13.2 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 13.2, 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 ξ as in (13.42). According to Theorem 13.11, 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 13.2.

Table 13.1. Example systems from the literature.

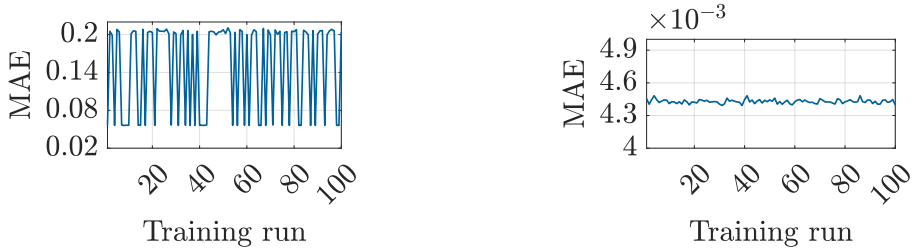
Ex. No.	$\boldsymbol{A}$	$\boldsymbol{B}$	$\mathcal{X}$	$\mathcal{U}$	$\boldsymbol{Q}$	$\boldsymbol{R}$	N	s	Reference
1.	$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$	$\begin{matrix} \lvert x_1 \rvert \leq 25 \\ \lvert x_2 \rvert \leq 5 \end{matrix}$	$\lvert u \rvert \leq 1$	$\boldsymbol{I}_2$	0.1	2	121	[122, Eqs. (2.8)–(2.9)]
2.	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 4 \end{pmatrix}$	$\begin{matrix} \lvert x_1 \rvert \leq 5 \\ \lvert x_2 \rvert \leq 5 \end{matrix}$	$\lvert u \rvert \leq 1$	$\boldsymbol{I}_2$	4.5	10	81	[108, Ex. 3]
3.	$\begin{pmatrix} 1 & 0.5 & 0.125 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.02 \\ 0.125 \\ 0.5 \end{pmatrix}$	$\begin{matrix} \lvert x_1 \rvert \leq 20 \\ \lvert x_2 \rvert \leq 3 \\ \lvert x_3 \rvert \leq 1 \end{matrix}$	$\lvert u \rvert \leq 0.5$	$\boldsymbol{I}_3$	1	2	15646	[122, Rem. 4.8]

Table 13.2. 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.	(13.42)	1	2	121	4.43×10^{-3}
3.	2.	$\boldsymbol{x}$	1	2	81	5.78×10^{-2}
4.	2.	(13.42)	1	2	81	5.70×10^{-3}
5.	3.	$\boldsymbol{x}$	1	2	15646	1.35×10^{-1}
6.	3.	(13.42)	1	2	15646	9.90×10^{-4}

Now, the experiments offer several insights. First, the NNs with $\boldsymbol{\xi}$ as in (13.42) 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 $5.78/5.7 \times 10 = 10.14$ between the test cases 3. and 4. and the highest observed improvement is $1.35/9.9 \times 10^3 = 136.36$ between the test cases 1. and 2.. Thus, we have a significant improvement of a minimum of one order of magnitude when using the NNs proposed in Section 13.4-13.4.1.

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 Figure 13.6-(a) and Figure 13.6-(b), respectively.



(a) **(b)**

Figure 13.6. Minimum MAE reached in each of the 100 training runs when approximating the OVF for the first example system from Table 13.1 with the NN from the test cases 1. and 2. from Table 13.2 in (a) and (b), respectively. The weights of the NN are initialized randomly at the beginning of each training run.

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 Figure 13.6-(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 13.11, for which an exact representation of the OVF is possible,

we can observe in Figure 13.6-(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 13.1 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 [172], which is a linear system with $n = 10$ and $m = 3$, with a PWQ NN from the literature [27]. For the MIMO30 system we consider the OCP (13.12) 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 $i \in \{1, \dots, 10\}$ and $|\boldsymbol{u}_i| \leq 1$ for all $i \in \{1, \dots, 3\}$, respectively. The resulting OCP is such that the MPT3 [121] 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 (13.12) for 10000 random values of $\boldsymbol{x} \in \mathcal{F}_N$, where $\mathcal{F}_n \subseteq \mathcal{X}$ is the feasible set of (13.12). We use this dataset to train two different relu NNs. The first relu NN has a topology as in Theorem 13.13. Note that, since we are not able to solve the OCP (13.12) explicitly in this example, the number of regions s of the OVF is unknown, and thus we cannot fully utilize the results of Theorem 13.13. 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 13.13 by choosing a relu NN with an input vector $\boldsymbol{\xi}$ as in (13.42) and $l = 1$ layer with $w_1 = {}^{n^2+3n}/_2 + 3 = 68$ neurons. We compare this relu NN with the PWQ relu NN from [27] with the "local-global" topology as described in Section 13.2-13.2.4. 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 13.2. The PWQ NN from [27] archives on average a MAE of 4.46×10^{-4} and the NN from Theorem 13.13 a MAE of 1.02×10^{-4}. Thus, even for a high-dimensional example where the results of Section 13.4-13.4.1 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.

13.6 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 (13.10). The representation is based on constructing a convex lifting of the domain partition that satisfies certain conditions specified in Lemma 13.4, which ensure (13.14).

We showed in Lemma 13.8 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 13.10, where we state the existence of a representation of the form (13.10) for PWQ functions. Based on this representation, we have derived different maxout and relu NN in the Theorems 13.11, 13.12 and 13.13. 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 13.5-13.5.1 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 13.5-13.5.2 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 13.3. 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 13.2 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 (13.19) such that (13.28) 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 (13.28) 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 (13.28) 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 (13.28) 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 (13.26) such that (13.28) 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 (13.28) 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 (13.26) such that (13.28) 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. ■

Acknowledgment

Financial support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the grant 556837821 is gratefully acknowledged.

Bibliography

[1] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals, and Systems* 2 (1989), pp. 303–314.

[2] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi. "A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges". In: *IEEE Transactions on Cybernetics* 54.12 (2024), pp. 7173–7186.

[3] D. P. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[5] Y. Wang. "A new concept using LSTM Neural Networks for dynamic system identification". In: *2017 American Control Conference (ACC)*. 2017, pp. 5324–5329.

[6] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari. "Approximating Explicit Model Predictive Control Using Constrained Neural Networks". In: *Proc. of the 2018 American Control Conference*. 2018, pp. 1520–1527.

[7] B. Karg and S. Lucia. "Efficient Representation and Approximation of Model Predictive Control Laws via Deep Learning". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3866–3878.

[8] S. J. Bradtke, B. E. Ydstie, and A. G. Barto. "Adaptive linear quadratic control using policy iteration". In: *Proceedings of 1994 American Control Conference - ACC '94*. Vol. 3. 1994, pp. 3475–3479.

[9] J. Maciejowski. *Predictive Control with Constraints.* England.: Prentice Hall, 2002.

[10] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov. "Value function approximation and model predictive control". In: *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. 2013, pp. 100–107.

[11] D. P. Bertsekas. "Model Predictive Control and Reinforcement Learning: A Unified Framework Based on Dynamic Programming". In: *IFAC-PapersOnLine* 58.18 (2024). 8th IFAC Conference on Nonlinear Model Predictive Control 2024, pp. 363–383.

[12] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön. "Deep networks for system identification: A survey". In: *Automatica* 171 (2025), p. 111907.

[13] R. Schwan, C. N. Jones, and D. Kuhn. "Stability Verification of Neural Network Controllers Using Mixed-Integer Programming". In: *IEEE Transactions on Automatic Control* 68.12 (2023), pp. 7514–7529.

[14] X. Li and K. You. "ReLU Neural Networks for Approximating Model Predictive Control: Complexity and Stability Guarantees". In: *Proc. of the 2025 American Control Conference*. 2025, pp. 27–32.

[15] L. Markolf and O. Stursberg. "Polytopic Input Constraints in Learning-Based Optimal Control Using Neural Networks". In: *arXiv:2105.03376v1 [eess.SY]* (2021).

[16] F. Fabiani and P. J. Goulart. "Reliably-Stabilizing Piecewise-Affine Neural Network Controllers". In: *IEEE Transactions on Automatic Control* 68.9 (2023), pp. 5201–5215.

[17] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd Edition. Nob Hill Publishing, 2017.

[18] B. Karg and S. Lucia. "Deep learning-based embedded mixed-integer model predictive control". In: *2018 European Control Conference (ECC)*. 2018, pp. 2075–2080.

[19] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. "Tune: A Research Platform for Distributed Model Selection and Training". In: *arXiv preprint arXiv:1807.05118* (2018).

[20] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1 (2002), pp. 3–20.

[21] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. "Maxout Networks". In: *Proc. of the 30th International Conference on Machine Learning*. Vol. 28. 2013, pp. 1319–1327.

[22] G. Montúfar, Y. Ren, and L. Zhang. *Sharp bounds for the number of regions of maxout networks and vertices of Minkowski sums*. arXiv:2104.08135v2 [math.CO]. 2021.

[23] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. "On the Number of Linear Regions of Deep Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2924–2932.

[24] B. Hanin. *Universal Function Approximation by Deep Neural Nets with Bounded Width and ReLU Activations*. arXiv:1708.02691v3 [stat.ML]. 2017.

[25] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. *Understanding Deep Neural Networks with Rectified Linear Units*. arXiv:1611.01491v6 [cs.LG]. 2018.

[26] M. Schulze Darup. "Exact representation of piecewise affine functions via neural networks". In: *Proc. of the 2020 European Control Conference*. 2020, pp. 1073–1078.

[27] K. He, S. Shi, T. van den Boom, and B. De Schutter. "Approximate dynamic programming for constrained linear systems: A piecewise quadratic approximation approach". In: *Automatica* 160 (2024), p. 111456.

[28] J.-N. Lin and R. Unbehauen. "On the quadratic extension of the canonical piecewise-linear network". In: *Proc. of the 1992 IEEE International Symposium on Circuits and Systems*. Vol. 1. 1992, 316–319 vol.1.

[29] W. Li, J.-N. Lin, and R. Unbehauen. "Canonical representation of piecewise-polynomial functions with nondegenerate linear-domain partitions". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 45.8 (1998), pp. 838–848.

[30] Y. Bengio. "Practical Recommendations for Gradient-Based Training of Deep Architectures". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by G. Montavon, G. B. Orr, and K.-R. Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478.

[31] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. arXiv:1412.6980 [cs.LG]. 2017.

[32] A. Siahkamari, A. Gangrade, B. Kulis, and V. Saligrama. "Piecewise Linear Regression via a Difference of Convex Functions". In: *Proc. of the 37th International Conference on Machine Learning*. Vol. 119. 2020, pp. 8895–8904.

[33] S. Rebennack and V. Krasko. "Piecewise Linear Function Fitting via Mixed-Integer Linear Programming". In: *INFORMS Journal on Computing* 32.2 (2020), pp. 507–530.

[34] J. A. Warwicker and S. Rebennack. "A unified framework for bivariate clustering and regression problems via mixed-integer linear programming". In: *Discrete Applied Mathematics* 336 (2023), pp. 15–36.

[35] P. Pauli, N. Funcke, D. Gramlich, M. A. Msalmi, and F. Allgöwer. "Neural network training under semidefinite constraints". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022, pp. 2731–2736.

[36] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer. "Training Robust Neural Networks Using Lipschitz Bounds". In: *IEEE Control Systems Letters* 6 (2022), pp. 121–126.

[37] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. "Safety Verification of Deep Neural Networks". In: *CoRR* abs/1610.06940 (2016).

[38] B. Karg and S. Lucia. "Stability and feasibility of neural network-based controllers via output range analysis". In: *Proc. of the 2020 Conference on Decision and Control*. 2020, pp. 4947–4954.

[39] M. Fazlyab, M. Morari, and G. J. Pappas. "Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming". In: *IEEE Transactions on Automatic Control* 67.1 (2022), pp. 1–15.

[40] W. Xiang, H.-D. Tran, J. A. Rosenfeld, and T. T. Johnson. "Reachable Set Estimation and Safety Verification for Piecewise Linear Systems with Neural Network Controllers". In: *Proc. of the 2018 American Control Conference*. 2018, pp. 1574–1579.

[41] M. Schulze Darup, A. Redder, and D. E. Quevedo. "Encrypted cloud-based MPC for linear systems with input constraints". In: *Proceedings of the 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018*. 2018, pp. 635–642.

[42] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Transactions on Information Theory* 31.4 (1985), pp. 469–472.

[43] P. Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology - Eurocrypt '99*. Vol. 1592. Springer, 1999, pp. 223–238.

[44] K. Han and D. Ki. "Better Bootstrapping for Approximate Homomorphic Encryption". In: *Topics in Cryptology – CT-RSA 2020*. Springer International Publishing, 2020, pp. 364–390.

[45] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 10.0.* 2022.

[46] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024.

[47] A. Bemporad and M. Morari. "Control of systems integrating logic, dynamics, and constraints". In: *Automatica* 35.3 (1999), pp. 407–427.

[48] D. Q. Mayne, J. B. Rawlings, C. Rao, and P. O. M. Scokaert. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36 (2000), pp. 789–814.

[49] J.-B. Hiriart-Urruty. *Convex Analysis and Minimization Algorithms I*. Springer-Verlag Berlin Heidelberg, 1993.

[50] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[51] I. I. Dikin. "Iterative solution of problems of linear and quadratic programming". English. In: *Sov. Math., Dokl.* 8 (1967), pp. 674–675.

[52] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[53] T.-T. Lu and S.-H. Shiou. "Inverses of 2 × 2 block matrices". In: *Computers & Mathematics with Applications* 43.1 (2002), pp. 119–129.

[54] A. H. Land and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3 (1960), pp. 497–520.

[55] E. L. Lawler and D. E. Wood. "Branch-And-Bound Methods: A Survey". In: *Operations Research* 14.4 (1966), pp. 699–719.

[56] M. Fischetti and J. Jo. "Deep neural networks and mixed-integer linear optimization". In: *Constraints* 23.3 (2018), pp. 296–309.

[57] C. Ragsdale and A. Stam. "A note on solving quadratic programs using mixed-integer programming". In: *Computers & Operations Research* 16.4 (1989), pp. 393–395.

[58] S. Qin and T. A. Badgwell. "A survey of industrial model predictive control technology". In: *Control Engineering Practice* 11.7 (2003), pp. 733–764.

[59] S. Di Cairano and I. V. Kolmanovsky. "Automotive applications of model predictive control". In: *Handbook of model predictive control* (2019), pp. 493–527.

[60] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe. "Model Predictive Control in Aerospace Systems: Current State and Opportunities". In: *Journal of Guidance, Control, and Dynamics* 40.7 (2017), pp. 1541–1566.

[61] P. Scokaert, J. Rawlings, and E. Meadows. "Discrete-time stability with perturbations: application to model predictive control". In: *Automatica* 33.3 (1997), pp. 463–470.

[62] D. Limon Marruedo, T. Alamo, and E. Camacho. "Stability analysis of systems with bounded additive uncertainties based on invariant sets: Stability and feasibility of MPC". In: *Proc. of the 2002 American Control Conference*. Vol. 1. 2002, 364–369 vol.1.

[63] V. V. Gorokhovik, O. I. Zorko, and G. Birkhoff. "Piecewise affine functions and polyhedral sets". In: *Optimization* 31.3 (1994), pp. 209–221.

[64] A. Bemporad, G. Ferrari-Trecate, and M. Morari. "Observability and controllability of piecewise affine and hybrid systems". In: *IEEE Transactions on Automatic Control* 45.10 (2000), pp. 1864–1876.

[65] S.-i. Azuma, J.-i. Imura, and T. Sugie. "Lebesgue piecewise affine approximation of nonlinear systems". In: *Nonlinear Analysis: Hybrid Systems* 4.1 (2010), pp. 92–102.

[66] A. Basu, M. Conforti, M. Di Summa, and H. Jiang. "Complexity of Branch-and-Bound and Cutting Planes in Mixed-Integer Optimization — II". In: *Combinatorica* 42.1 (2022), pp. 971–996.

[67] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[68] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018.

[69] G. Cybenko. "Dynamic load balancing for distributed memory multiprocessors". In: *Journal of parallel and distributed computing* 7.2 (1989), pp. 279–301.

[70] A. Kripfganz and R. Schulze. "Piecewise affine functions as a difference of two convex functions". In: *Optimization* 18.1 (1987), pp. 23–29.

[71] N. Schlüter, P. Binfet, and M. Schulze Darup. "A brief survey on encrypted control: From the first to the second generation and beyond". In: *Annual Reviews in Control* (2023), p. 100913.

[72] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, et al. "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network". In: *IEEE Access* 10 (2022), pp. 30039–30054.

[73] J. Ferlez and Y. Shoukry. "AReN: Assured ReLU NN architecture for model predictive control of LTI systems". In: *Proc. of the 23rd International Conference on Hybrid Systems: Computation and Control*. 2020, pp. 1–11.

[74] S. Rakovic, P. Grieder, M. Kvasnica, D. Mayne, and M. Morari. "Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances". In: *Proc. of the 2004 Conference on Decision and Control*. Vol. 2. 2004, 1418–1423 Vol.2.

[75] F. Blanchini. "Set invariance in control". In: *Automatica* 35.11 (1999), pp. 1747–1767.

[76] J. M. Lee and J. H. Lee. "Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes". In: *Automatica* 41.7 (2005), pp. 1281–1288.

[77] H. Nosair and J. Min Lee. "Parametric Approximation of Piecewise Quadratic Value Functions for the Control of Complex Systems". In: *IFAC Proceedings Volumes* 41.2 (2008). 17th IFAC World Congress, pp. 3252–3257.

[78] S. V. Rakovic and M. Baric. "Parameterized Robust Control Invariant Sets for Linear Systems: Theoretical Advances and Computational Remarks". In: *IEEE Transactions on Automatic Control* 55.7 (2010), pp. 1599–1614.

[79] N. Hansen, X. Wang, and H. Su. "Temporal Difference Learning for Model Predictive Control". In: *International Conference on Machine Learning*. 2022.

[80] P. Hartman. "On functions representable as a difference of convex functions." In: *Pacific Journal of Mathematics* 9.3 (1959), pp. 707–713.

[81] J. Caravantes, M. A. Gomez-Molleda, and L. Gonzalez-Vega. "A canonical form for the continuous piecewise polynomial functions". In: *Journal of Computational and Applied Mathematics* 283 (2015), pp. 17–27.

[82] J.-N. Lin and R. Unbehauen. "Canonical representation: from piecewise-linear function to piecewise-smooth functions". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 40.7 (1993), pp. 461–468. DOI: 10.1109/81.257301.

[83] L. Chua and A.-C. Deng. "Canonical piecewise-linear representation". In: *IEEE Transactions on Circuits and Systems* 35.1 (1988), pp. 101–111.

[84] N. Schlüter and M. Schulze Darup. *Novel convex decomposition of piecewise affine functions*. 2021.

[85] A. Magnani and S. P. Boyd. "Convex piecewise-linear fitting". In: *Optimization and Engineering* 10.1 (2009), pp. 1–17.

[86] A. Bemporad. "A Piecewise Linear Regression and Classification Algorithm With Application to Learning and Model Predictive Control of Hybrid Systems". In: *IEEE Transactions on Automatic Control* 68.6 (2023), pp. 3194–3209.

[87] G. Balazs, A. György, and C. Szepesvari. "Near-optimal max-affine estimators for convex regression". In: *Proc. of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA: PMLR, 2015, pp. 56–64.

[88] N. Goldberg, S. Rebennack, Y. Kim, V. Krasko, and S. Leyffer. "MINLP formulations for continuous piecewise linear function fitting". In: *Computational Optimization and Applications* 79.1 (2021), pp. 223–233.

[89] J. A. Warwicker and S. Rebennack. "Efficient continuous piecewise linear regression for linearising univariate non-linear functions". In: *IISE Transactions* 57.3 (2025), pp. 231–245.

[90] A. Toriello and J. P. Vielma. "Fitting piecewise linear continuous functions". In: *European Journal of Operational Research* 219.1 (2012), pp. 86–95.

[91] D. Arthur and S. Vassilvitskii. "K-Means++: The Advantages of Careful Seeding". In: *Proc. of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[92] J. Lee, E. Lee, J.-W. Lee, Y. Kim, Y.-S. Kim, and J.-S. No. "Precise approximation of convolutional neural networks for homomorphically encrypted data". In: *IEEE Access* 11 (2023), pp. 62062–62076.

[93] A. Sabbagh Molahosseini, L. Seabra de Sousa, and C.-H. Chang. *Embedded Systems Design with Special Arithmetic and Number Systems*. Springer, 2017.

[94] M. Kvasnica, J. Löfberg, and M. Fikar. "Stabilizing polynomial approximation of explicit MPC". In: *Automatica* 47.10 (2011), pp. 2292–2297.

[95] J. A. Paulson and A. Mesbah. "Approximate Closed-Loop Robust Model Predictive Control With Guaranteed Stability and Constraint Satisfaction". In: *IEEE Control Systems Letters* 4.3 (2020), pp. 719–724.

[96] M. Schulze Darup, M. Jost, G. Pannocchia, and M. Mönnigmann. "On the maximal controller gain in linear MPC". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 9218–9223.

[97] C. Danielson and F. Borrelli. "Symmetric Linear Model Predictive Control". In: *IEEE Transactions on Automatic Control* 60.5 (2015), pp. 1244–1259.

[98] Z. Shi, Y. Wang, H. Zhang, Z. Kolter, and C.-J. Hsieh. "Efficiently computing local lipschitz constants of neural networks via bound propagation". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. Curran Associates Inc., 2022.

[99] F. Blanchini. "Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions". In: *IEEE Transactions on Automatic Control* 39.2 (1994), pp. 428–433.

[100] S. V. Rakovic, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne. "Invariant approximations of the minimal robust positively invariant set". In: *IEEE Transactions on automatic control* 50.3 (2005), pp. 406–410.

[101] D. Mignone, G. Ferrari-Trecate, and M. Morari. "Stability and stabilization of piecewise affine and hybrid systems: an LMI approach". In: *Proc. of the 2000 Conference on Decision and Control*. Vol. 1. 2000, 504–509 vol.1.

[102] M. Lazar, D. Muñoz de la Peña, W. Heemels, and T. Alamo. "On input-to-state stability of min–max nonlinear model predictive control". In: *Systems & Control Letters* 57.1 (2008), pp. 39–48.

[103] F. L. Lewis and D. Vrabie. "Reinforcement learning and adaptive dynamic programming for feedback control". In: *IEEE Circuits and Systems Magazine* 9.3 (2009), pp. 32–50.

[104] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones. "Learning a feasible and stabilizing explicit model predictive control law by robust optimization". In: *Proc. of the 2011 Conference on Decision and Control and European Control Conference*. 2011, pp. 513–519.

[105] M. Bhardwaj, S. Choudhury, and B. Boots. *Blending MPC & Value Function Approximation for Efficient Reinforcement Learning*. arXiv:2012.05909v1 [cs.LG]. 2020.

[106] K. F. Cheung and C. S. Leung. "Rotational quadratic function neural networks". In: *Proc. of the 1991 IEEE International Joint Conference on Neural Networks*. Vol. 1. 1991, pp. 869–874.

[107] F. Fan, J. Xiong, and G. Wang. *Universal Approximation with Quadratic Deep Networks*. arXiv:1808.00098v3 [cs.LG]. 2019.

[108] M. Schulze Darup and M. Cannon. "Some observations on the activity of terminal constraints in linear MPC". In: *Proc. of the 2016 European Control Conference*. 2016, pp. 4977–4983.

[109] I. The MathWorks. *Deep Learning Toolbox*. Natick, Massachusetts, United State, 2020.

[110] N. A. Nguyen, M. Gulan, S. Olaru, and P. Rodriguez-Ayerbe. "Convex lifting: Theory and control applications". In: *IEEE Transactions on Automatic Control* 63.5 (2017), pp. 1243–1258.

[111] P. Karamanakos, E. Liegmann, T. Geyer, and R. Kennel. "Model Predictive Control of Power Electronic Systems: Methods, Results, and Challenges". In: *IEEE Open Journal of Industry Applications* 1 (2020), pp. 95–114.

[112] C. Jones and M. Morari. "Approximate explicit MPC using bilevel optimization". In: *Proc. of the 2009 European Control Conference*. 2009, pp. 2396–2401.

[113] A. Bemporad and C. Filippi. "Suboptimal Explicit Receding Horizon Control via Approximate Multiparametric Quadratic Programming". In: *Journal of Optimization Theory and Applications* 117 (2003), pp. 9–38.

[114] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari. "Large scale model predictive control with neural networks and primal active sets". In: *Automatica* 135 (2022), p. 109947.

[115] M. Kvasnica and M. Fikar. "Clipping-Based Complexity Reduction in Explicit MPC". In: *IEEE Transactions on Automatic Control* 57.7 (2012), pp. 1878–1883.

[116] R. Drummond, S. Duncan, M. Turner, P. Pauli, and F. Allgower. "Bounding the Difference Between Model Predictive Control and Neural Networks". In: *Proc. of the 4th Annual Learning for Dynamics and Control Conference*. Vol. 168. 2022, pp. 817–829.

[117] S. Wang and X. Sun. "Generalization of hinging hyperplanes". In: *IEEE Transactions on Information Theory* 51.12 (2005), pp. 4425–4431.

[118] F. Chollet et al. *Keras*. Available: `https://keras.io`. 2015.

[119] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Available: `https://www.tensorflow.org/`. 2015.

[120] E. G. Gilbert and K. T. Tan. "Linear Systems with State and Control Constraints: The Theory and Application of Maximal Output Admissible Sets". In: *IEEE Trans. Autom. Control* 36.9 (1991), pp. 1008–1020.

[121] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. "Multi-Parametric Toolbox 3.0". In: *Proc. of the European Control Conference*. 2013, pp. 502–510.

[122] 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". In: *IEEE Transactions on Automatic Control* 32.3 (1987), pp. 251–254.

[123] M. Schulze Darup. "Numerical methods for the investigation of stabilizability of constrained systems". PhD thesis. Ruhr-Universität Bochum, Universitätsbibliothek, 2014.

[124] J. Roll, A. Bemporad, and L. Ljung. "Identification of piecewise affine systems via mixed-integer programming". In: *Automatica* 40.1 (2004), pp. 37–50.

[125] L. Breiman. "Hinging hyperplanes for regression, classification, and function approximation". In: *IEEE Transactions on Information Theory* 39.3 (1993), pp. 999–1013.

[126] D. Bertsimas and R. Shioda. "Classification and Regression via Integer Optimization". In: *Operations Research* 55.2 (2007), pp. 252–271.

[127] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese. "On Mathematical Programming with Indicator Constraints". In: *Math. Program.* 151.1 (2015), pp. 191–223.

[128] L. Magni, G. De Nicolao, R. Scattolini, and F. Allgöwer. "Robust model predictive control for nonlinear discrete-time systems". In: *International Journal of Robust and Nonlinear Control* 13.3-4 (2003), pp. 229–246.

[129] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[130] G. Ferrari-Trecate, F. Cuzzola, D. Mignone, and M. Morari. "Analysis and control with performance of piecewise affine and hybrid systems". In: *Proc. of the 2001 American Control Conference*. Vol. 1. 2001, 200–205 vol.1.

[131] G. Ferrari-Trecate, F. A. Cuzzola, D. Mignone, and M. Morari. "Analysis of discrete-time piecewise affine and hybrid systems". In: *Automatica* 38.12 (2002), pp. 2139–2146.

[132] P. Grieder, M. Kvasnica, M. Baotić, and M. Morari. "Stabilizing low complexity feedback control of constrained piecewise affine systems". In: *Automatica* 41.10 (2005), pp. 1683–1694.

[133] E. Camacho, D. Ramirez, D. Limon, D. Muñoz de la Peña, and T. Alamo. "Model predictive control techniques for hybrid systems". In: *Annual Reviews in Control* 34.1 (2010), pp. 21–31.

[134] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. "An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems". In: *Proc. of the 2003 American Control Conference*. Vol. 6. 2003, 4717–4722 vol.6.

[135] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. *Intriguing properties of neural networks*. 2014.

[136] M. S. Ghasemi and A. A. Afzalian. "Robust tube-based MPC of constrained piecewise affine systems with bounded additive disturbances". In: *Nonlinear Analysis: Hybrid Systems* 26 (2017), pp. 86–100.

[137] D. Q. Mayne, M. M. Seron, and S. V. Raković. "Robust model predictive control of constrained linear systems with bounded disturbances". In: *Automatica* 41 (2005), pp. 219–224.

[138] Martín Abadi and et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[139] A. Paszke and et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: (2019).

[140] L. Lüken and S. Lucia. "Learning Iterative Solvers for Accurate and Fast Nonlinear Model Predictive Control via Unsupervised Training". In: *Proc. of the 2024 European Control Conference*. 2024, pp. 1843–1850.

[141] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj. "Survey on Fully Homomorphic Encryption, Theory, and Applications". In: *Proceedings of the IEEE* 110.10 (2022), pp. 1572–1609.

[142] R. Podschwadt, D. Takabi, P. Hu, M. H. Rafiei, and Z. Cai. "A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption". In: *IEEE Access* 10 (2022), pp. 117477–117500.

[143] S. J. Mohammed and D. B. Taha. "From cloud computing security towards homomorphic encryption: A comprehensive review". In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 19.4 (2021), pp. 1152–1161.

[144] W.-K. Lin, E. Mook, and D. Wichs. "Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 595–608.

[145] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu. "Private database queries using somewhat homomorphic encryption". In: *Applied Cryptography and Network Security: 11th International Conference, ACNS 2013*. Springer. 2013, pp. 102–118.

[146] J. Byun, H. Ko, and J. Lee. "A Privacy-preserving mean–variance optimal portfolio". In: *Finance Research Letters* 54 (2023), p. 103794.

[147] V. S. Naresh and D. Ayyappa. "PPDNN-CRP: CKKS-FHE Enabled Privacy-Preserving Deep Neural Network Processing for Credit Risk Prediction". In: *Computational Economics* (2024), pp. 1–25.

[148] Z.-P. Yuan, P. Li, Z.-L. Li, and J. Xia. "A Fully Distributed Privacy-Preserving Energy Management System for Networked Microgrid Cluster Based on Homomorphic Encryption". In: *IEEE Transactions on Smart Grid* (2023).

[149] Z. Cheng, F. Ye, X. Cao, and M.-Y. Chow. "A homomorphic encryption-based private collaborative distributed energy management system". In: *IEEE Transactions on Smart Grid* 12.6 (2021), pp. 5233–5243.

[150] P. R. Naidu, D. R. Bolla, G. Prateek, S. S. Harshini, S. A. Hegde, and V. V. S. Harsha. "E-Voting system using blockchain and homomorphic encryption". In: *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*. 2022, pp. 1–5.

[151] M. Schulze Darup, A. B. Alexandru, D. E. Quevedo, and G. J. Pappas. "Encrypted control for networked systems: An illustrative introduction and current challenges". In: *IEEE Control Systems Magazine* 41.3 (2021), pp. 58–78.

[152] E. Lee, J.-W. Lee, J.-S. No, and Y.-S. Kim. "Minimax approximation of sign function by composite polynomial for homomorphic comparison". In: *IEEE Transactions on Dependable and Secure Computing* 19.6 (2021), pp. 3711–3727.

[153] S. Panda. "Polynomial Approximation of Inverse sqrt Function for FHE". In: *Cyber Security, Cryptology, and Machine Learning*. Springer International Publishing, 2022, pp. 366–376.

[154] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon. "Logistic regression model training based on the approximate homomorphic encryption". In: *BMC medical genomics* 11 (2018), pp. 23–31.

[155] G. Arakelov, N. Kaskov, D. Pianykh, and Y. Polyakov. *FHERMA: Building the Open-Source FHE Components Library for Practical Use*. Cryptology ePrint Archive, Paper 2024/612. 2024.

[156] C. Bonte and F. Vercauteren. "Privacy-preserving logistic regression training". In: *BMC medical genomics* 11 (2018), pp. 13–21.

[157] M. Tian, J. Liu, Z. Chen, and S. Wang. "Privacy-preserving logistic regression with improved efficiency". In: *Journal of Information Security and Applications* 85 (2024), p. 103848.

[158] S. Kang and L. Chua. "A global representation of multidimensional piecewise-linear functions with linear partitions". In: *IEEE Transactions on Circuits and Systems* 25.11 (1978), pp. 938–940.

[159] C. Wen and X. Ma. "A Canonical Piecewise-Linear Representation Theorem: Geometrical Structures Determine Representation Capability". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 58.12 (2011), pp. 936–940. DOI: 10.1109/TCSII.2011.2172715.

[160] M. Bhardwaj, S. Choudhury, and B. Boots. "Blending MPC & Value Function Approximation for Efficient Reinforcement Learning". In: *International Conference on Learning Representations*. 2021.

[161] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado. "Learning Lyapunov functions for hybrid systems". In: *Proc. of the 24th International Conference on Hybrid Systems: Computation and Control*. Association for Computing Machinery, 2021.

[162] A. B. Kordabad, D. Reinhardt, A. S. Anand, and S. Gros. "Reinforcement Learning for MPC: Fundamentals and Current Challenges". In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 5773–5780.

[163] M. Dawood, S. Pan, N. Dengler, S. Zhou, A. P. Schoellig, and M. Bennewitz. "Safe Multi-Agent Reinforcement Learning for Behavior-Based Cooperative Navigation". In: *IEEE Robotics and Automation Letters* 10.6 (2025), pp. 6256–6263.

[164] Y. Zheng, Y. Liu, G. Zeng, H. Zhang, S. Li, Y. Tang, Q. Zhao, M. Luo, and Y. Wang. "Distributed Q-Learning Model Predictive Control Based on ADMM for continuous nonlinear systems". In: *Digital Chemical Engineering* 16 (2025), p. 100257.

[165] Y. Cui, T.-H. Chang, M. Hong, and J.-S. Pang. "A Study of Piecewise Linear-Quadratic Programs". In: *J. Optim. Theory Appl.* 186.2 (2020), pp. 523–553.

[166] L. Bittner. "Some representation theorems for functions and sets and their application to nonlinear programming". In: *Numerische Mathematik* 16 (1970), pp. 32–51.

[167] 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". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3050–3057.

[168] M. Telgarsky. "Benefits of depth in neural networks". In: *29th Annual Conference on Learning Theory*. Ed. by V. Feldman, A. Rakhlin, and O. Shamir. Vol. 49. PMLR, 2016, pp. 1517–1539.

[169] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao. "Normalization Techniques in Training DNNs: Methodology, Analysis and Application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.8 (2023), pp. 10173–10196.

[170] F. Chollet et al. *Keras*. https://keras.io. 2015.

[171] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proc. of the 13th International Conference on Artificial Intelligence and Statistics*. Vol. 9. 2010, pp. 249–256.

[172] M. Jost, G. Pannocchia, and M. Mönnigmann. *Simulation studies on online constraint removal with a Lyapunov function*. arXiv:1411.0981 [math.OC]. 2015.

Publications of the Author

[P1] D. Teichrib and M. Schulze Darup. "Tailored neural networks for learning optimal value functions in MPC". In: *Proc. of the 2021 Conference on Decision and Control*. 2021, pp. 5281–5287. DOI: 10.1109/CDC45484.2021.9683528.

[P2] D. Teichrib and M. Schulze Darup. "Tailored max-out networks for learning convex PWQ functions". In: *Proc. of the 2022 European Control Conference*. 2022, pp. 2272–2278. DOI: 10.23919/ECC55457.2022.9838225.

[P3] D. Teichrib and M. Schulze Darup. "Error bounds for maxout neural network approximations of model predictive control". In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 10113–10119. DOI: 10.1016/j.ifacol.2023.10.883.

[P4] D. Teichrib and M. Schulze Darup. "Efficient Computation of Lipschitz Constants for MPC with Symmetries". In: *Proc. of the 2023 Conference on Decision and Control*. 2023, pp. 6685–6691. DOI: 10.1109/CDC49753.2023.10383472.

[P5] D. Teichrib and M. Schulze Darup. "Piecewise regression via mixed-integer programming for MPC". In: *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*. Ed. by A. Abate, M. Cannon, K. Margellos, and A. Papachristodoulou. Vol. 242. Proceedings of Machine Learning Research. PMLR, 2024, pp. 337–348.

[P6] D. Teichrib and M. Schulze Darup. "Reachability analysis for piecewise affine systems with neural network-based controllers". In: *Proc. of the 2024 Conference on Decision and Control*. 2024, pp. 894–900. DOI: 10.1109/CDC56724.2024.10886414.

[P7] D. Teichrib and M. Schulze Darup. "A mixed-integer framework for analyzing neural network-based controllers for piecewise affine systems with bounded disturbances". In: *Proc. of the 2025 European Control Conference*. 2025, pp. 904–910. DOI: 10.23919/ECC65951.2025.11187202.

[P8] D. Teichrib, J. Adamek, P. Binfet, and M. Schulze Darup. "Polynomial Function Approximations With Leading Integer Coefficients for Efficient Encrypted Implementations". In: *IEEE Access* 13 (2025), pp. 157455–157462. DOI: 10.1109/ACCESS.2025.3606013.

[P9] D. Teichrib and M. Schulze Darup. "On the Representation of Piecewise Quadratic Functions by Neural Networks". In: *IEEE Open Journal of Control Systems* 4 (2025), pp. 447–462. DOI: 10.1109/OJCSYS.2025.3607844.

Additional Publications of the Author

[P10] M. Schulze Darup and D. Teichrib. "Efficient computation of RPI sets for tube-based robust MPC". In: *Proc. of the 18th European Control Conference*. 2019, pp. 325–330. DOI: 10.23919/ECC.2019.8796265.

[P11] M. Klädtke, D. Teichrib, N. Schlüter, and M. Schulze Darup. "A deterministic view on explicit data-driven (M)PC". In: *Proc. of the 2022 IEEE Conference on Decision and Control*. 2022, pp. 499–504. DOI: 10.1109/CDC51059.2022.9993384.

[P12] H. Nikbakht, M. T. Khoshmehr, B. van Someren, D. Teichrib, M. Hammer, J. Förstner, and B. I. Akca. "Asymmetric, non-uniform 3-dB directional coupler with 300-nm bandwidth and a small footprint". In: *Opt. Lett.* 48.2 (2023), pp. 207–210. DOI: 10.1364/OL.476537.

[P13] J. Adamek, A. Aikata, A. Al Badawi, A. Alexandru, A. Arakelov, G. Arakelov, P. Binfet, V. Correa, J. Dumezy, S. Gomenyuk, V. Kononova, D. Lekomtsev, V. Maloney, C.-H. Nguyen, Y. Polyakov, D. Pianykh, H. Shaul, M. Schulze Darup, D. Teichrib, and D. Tronin. "FHERMA Cookbook: FHE Components for Privacy-Preserving Applications". In: *Proc. of the 13th Workshop on Encrypted Computing & Applied Homomorphic Computing*. WAHC '25. Taipei, Taiwan: Association for Computing Machinery, 2025, pp. 68–76. DOI: 10.1145/3733811.3767313.