LMI-based Stability Analysis for Learning Control

Deep Neural Networks and Locally Weighted Learning

Konstantinos Kokkalis

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

LMI-based Stability Analysis for Learning Control

Deep Neural Networks and Locally Weighted Learning

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Konstantinos Kokkalis

August 14, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

MAX-PLANCK-GESELLSCHAFT

ONASSIS FOUNDATION

The work in this thesis was supported by the Intelligent Control Systems (ICS) Group and the Autonomous Motion Department (AMD) of Max Planck Institute for Intelligent Systems (MPI-IS) Stuttgart/Tübingen, Germany. The financial support of Alexander S. Onassis Public Benefit Foundation and Erasmus+ programme of the European Union during the second year of my studies is also gratefully acknowledged.

TU Delft Delft University of Technology

DCSC

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

LMI-based Stability Analysis for Learning Control

by

Konstantinos Kokkalis

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: August 14, 2018

Supervisor(s):

Dr. Sebastian Trimpe

Dr.ing. Jens Kober

Reader(s):

Prof.dr.ir. Hans Hellendoorn

Dr. Alfredo Núñez Vicencio

Dr. Wei Pan

Abstract

Learning capabilities are a key requisite for an autonomous agent operating in dynamically changing and complex environments, where pre-programming is not anymore possible. Furthermore, it is essential to guarantee that the learning agent will act safely by considering its stability properties. In this thesis, novel conditions are proposed, aiming to examine stability of the learned dynamics for two important model classes; namely *Rectified Linear Unit (ReLU) Deep Neural Networks (DNNs)* and *Locally Weighted Learning (LWL).* For the former method, a theoretical and computational framework is developed by establishing an equivalence between ReLU DNN models and Piecewise Affine (PWA) systems. This allows to leverage well-known tools of PWA system analysis, and consequently compute, characterize equilibria and determine their region of attraction for ReLU DNNs. Due to their increased complexity, a structured search for appropriate stability conditions was performed for LWL methods until the optimal trade-off between conservativeness and computational efficiency was obtained. These stability conditions are given as Linear Matrix Inequality (LMI) problems and they consist the first stability results in literature for these two model classes. Their efficacy is assessed in numerical and real-world dynamical systems and it is shown that the proposed LMIs are not unreasonably conservative, as they can evaluate accurately the stability properties of these two representations. Finally, this work demonstrates how to formulate appropriate stability conditions for learning methods in a principled manner.

Parts of this Thesis have been submitted and are currently under review for presentation and publication at the Conference on Robot Learning (CoRL). The full paper can be found in Appendix C in a camera version. Details of the submission:

K. Kokkalis and S. Trimpe, "Stability Analysis for Deep Neural Network Dynamics Models", 2nd Annual Conference on Robot Learning (CoRL), October 29th-31st, 2018, Zürich, Switzerland (under review).

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my Max-Planck Institute supervisor Dr. Sebastian Trimpe for trusting me this challenging and interesting project. His constant encouragement and advice helped me in every difficulty faced this last year and has taught me more than I could imagine. I could not have hoped for a better supervisor.

I want also to thank my TU Delft supervisor Dr. ing. Jens Kober for guiding me during this project and providing valuable feedback on my progress. I am very grateful for introducing me to Max-Planck Society.

I could not have accomplished anything without the strong support of my parents Eirini and Nikiforos and my sister Anna, who during this couple of years did everything in their power to help me succeed and enjoy my studies. Their love and faith motivated me to keep trying harder and I hope I have made them proud.

Furthermore, I want to thank all my good friends and co-workers for always adding perspective and stimulating me.

And finally, but most importantly, I would like to thank my beloved Kelly for her patience to all my complaints and for cheering me up, even when I thought it was impossible. Her unconditional belief and care throughout this venture have shown me the way to become a better person.

Delft, University of Technology
August 14, 2018

Konstantinos Kokkalis

Chapter 1

Introduction

The current chapter intends to provide a very broad overview of this work and to present its importance for the field of learning control. Starting from a short description of this last notion, the topic of stability guarantees for learning and its relevance for intelligent agents is analyzed. In the next sections, the research problem is carefully identified and a roadmap towards its solution is indicated. Finally, studies relevant to the research questions are assessed and a brief outline of the rest of this Thesis is provided.

1-1 Motivation

1-1-1 Learning control

A large part of the current interest in control and robotics research has been directed to the development of systems with increased autonomy and enhanced capabilities. Robots and other intelligent agents of the future are expected to be largely self-sufficient and able to plan and make decisions in order to achieve high-level goals. Nevertheless, still a big portion of today's robotic systems are operating under well-determined environments, doing strictly-defined tasks with very little autonomy and robustness against uncertainty. Since predicting and programming for all possible conditions and goals is thought to be non viable, the reduction of the disparity between current state and future forecasts will rely heavily on the ability of the intelligent systems to process past experiences and *learn* from them.

Learning (for) control comprises one important aspect of the broad field of machine learning, which encompasses various skills such as learning for perception or learning to make decisions [1]. Learning control, in turn, aims to derive optimal policies through several distinct approaches like reinforcement learning [2], imitation learning [3] or supervised learning [4]. Due to lack of a clear definition and inaccurate interchangeability of this term with other types of control like intelligent and adaptive control, it is very difficult to define a learning control method precisely.

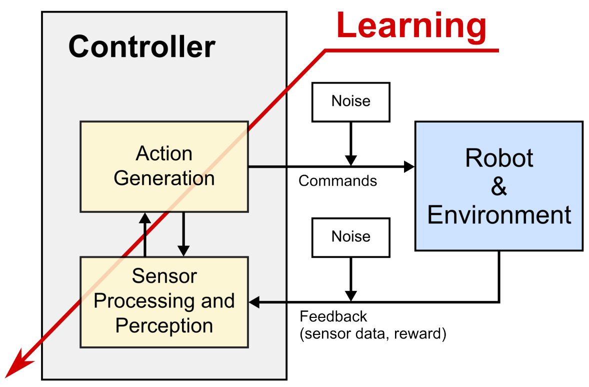


Figure 1-1: Abstract configuration of an autonomous system with learning capabilities[1].

One of the dominant definitions indicates that learning control aims to acquire a control policy for a particular control problem through interaction with the system's environment and processing of past experiences [1]. Traditional control systems without any learning capabilities, on the other hand, are usually restricted to policies that cannot react well in unexpected changes of the environment and many times are based on previously identified models. Frequently, this absence of learning control mechanisms significantly simplifies the extraction of a satisfactory control strategy under known and slowly-changing conditions, but cannot answer the need for more sophisticated policies necessary in complex and constantly changing environments.

What makes learning control methods especially challenging is the aforementioned dynamic relation between the learning process and the behavior of the system, compactly illustrated in Figure 1-1. More precisely, autonomous agent and learning policy are involved in a closed loop, where the performance of the former is processed in order to learn an optimal control strategy and the action chosen by the latter affects the performance of the agent. This dynamic behavior requires optimal learning approaches that decide what is necessary to learn and significant computation power to process the data in an online fashion.

The above figure and the described closed loop relation strongly resembles the classical control paradigm. This is not by mistake and brings to surface the second broad definition of learning control given more than 3 decades ago [5], which identifies it as the *field that solves control theory problems with machine learning methods*. While conventional control theory relies heavily on descriptions and models based on differential and difference equations [6], the introduction of machine learning promises to offer a far more general framework to identify and adjust satisfactory control policies by leveraging data and past experiences of the system. This second definition will mainly be adopted for the rest of this study.

1-1-2 Safety considerations

Machine learning methods, especially with the development of more efficient algorithms and the increase of computational power, have rendered the close approximation of complex and

[1]Courtesy of Sebastian Trimpe and Stefan Schaal https://am.is.tuebingen.mpg.de/research_fields/learning-control

high-dimensional functions realistic. These techniques can identify strongly nonlinear dynamics as well as cost functions and controllers, broadening this way the region of operation of modern control systems. Nonetheless, learning in real control systems faces an additional challenge, usually not present in other learning paradigms: the interaction with the physical environment. As data is gathered and the accurate control policies or dynamics are still unknown, the actions taken by the intelligent system may be highly unsafe, i.e. harming its environment and itself, mainly due to uncertainty. Even if the actions until a point in time have not led to any danger, the same cannot be said with certainty about the future, when new data will be processed and the policy will change. In a relatively recent review of various methods related to robot learning and learning control [1], guaranteeing safe operation has been identified as an open problem with important consequences for the success of robot learning. Thus, it is highly advisable to search for conditions that will ensure that the system will respect some predefined safety requirements.

For conventional control theory, safety of the dynamical system is guaranteed by assessing different stability properties. In most cases, a relatively simple model of the dynamics is considered under a number of assumptions for its uncertainty. Based on this model stability analysis is performed, i.e., conditions are determined such that the autonomous system is stable under a specific definition of stability. Usually these conditions can be formulated in a strict way such that a closed-form solution can be derived or an optimization problem can be solved. However, applications of machine learning to control rarely provide such stability guarantees for the controlled system, allowing it to continue its autonomous operation with a minimum level of confidence.

1-1-3 Stability for learning-based dynamics

One of the most common ways in which control theory evaluates stability properties is through a variety of broad and generic tools going under the name of *Lyapunov stability theory.* Introduced in 1982 by the Russian mathematician and engineer Lyapunov, this theory can be in principle applied to any linear and nonlinear representation, since it originates from a simple high-level idea and a number of precise conditions for stability can be derived imposing gradually a number of restricting assumptions. Lyapunov theory proposes constructive methods to study stability, since a *Lyapunov function* needs to be determined for a specific dynamic representation in order to assess its stability properties.

Learning very simple representations and more specifically linear models offers significant advantages such as efficient learning algorithms, rich control literature available and derivation of easily verifiable stability conditions through numerous well-established techniques such as analysis of eigenvalues [7, 8]. However, this simplicity comes at a cost, since it is commonly accepted that these representations are not rich enough to describe many commonly observed nonlinearities [9]. A very interesting summary of different nonlinear models and phenomena can be found in the first chapter of [10], where it is shown that even simple mechanical and electrical systems are in principal nonlinear. Therefore, for modern autonomous agents and robots, the linear models cannot be sufficiently accurate for larger operation regions.

Although Lyapunov theory is generally applicable for nonlinear dynamics such as learning-based model representations, it has been criticized for its difficulty to be applied in systems even with mildly complicated dynamics and/or without physical insight due to its constructive nature [11]. More precisely, if the nonlinear model comes from first principles with its parameters computed through identification, then it may be possible only in a fraction of cases to exploit the connection of notions in Lyapunov theory with physical properties of nonlinear systems like energy and dissipation in order to assess stability. For these cases, the applicability of Lyapunov theory for nonlinear systems relies more on sophisticated "guessing" based on experience than on a principled way of deriving stability results. However, when learning algorithms are implemented to approximate the dynamics, usually no prior knowledge and equivalence to physical properties of the system are available.

One possibility to push back the drawbacks of these two cases i.e. linear and general nonlinear representation is to synthesize a global nonlinear model by multiple local linear models, compressing this way large amounts of data in a small number of parameters. It is obvious that in this case the linear control theory cannot be directly applied. Nevertheless, the structure of the dynamics did not become largely more complex compared to linear systems, but remained relatively generic, hoping this way to maintain a good *bias-variance tradeoff* [12]. The last property is rather significant for every supervised algorithm and in brevity expresses the conflict between overfitting to the training set and underfitting due to inappropriate assumptions in the model. Furthermore, such systems relying in composition will allow to systematically study stability properties instead of guessing, and thus make stability analysis an algorithm.

1-1-4 A motivating example: Deep Neural Networks

A learning architecture that has received a lot of attention in the field of machine learning the recent years is: Deep Neural Networks (DNNs). An illustration of a general DNN approximating a nonlinear function $f : \mathbb{R}^2 \to \mathbb{R}$ is given in Figure 1-2. A Deep Neural Network can be seen as the composition of several (hidden) layers, each one applying successively an affine transformation and a point-wise nonlinear function to the inputs of the layer. The latter essentially makes the approximation of nonlinear functions possible and adds the nonlinearity in the representation. The *deep* characterization results from the inclusion of multiple such layers, while Neural Networks with a single hidden layer are called *shallow.*

With these very simple components and newly developed training algorithms, DNNs have managed to approximate very complex and high dimensional mappings between inputs and outputs. Success of deep learning in applications such as natural language processing, image and speech recognition has been unprecedented, while there is a strong belief that a lot more progress is left to be made [13]. This shows the efficacy of the proposed linear model composition as presented in the previous subsection.

Despite their rapid development and their good prediction performance, there is a lack of understanding of what DNNs actually represent and numerous studies have been devoted the recent years to explain the success of this particular learning architecture, e.g., [14–16]. For example, if the dynamics of a system are approximated with DNNs, it is very difficult to gain any insight into its dynamic behavior and properties, affecting also the ability to design and evaluate successful control policies based on this very rich formulation.

Furthermore, it has been established that it is difficult to train these deep architectures for a variety of reasons such as existence of multiple local minima in the optimization and saturating nonlinearities [17]. This implies that the result of the non-convex optimization, i.e., training

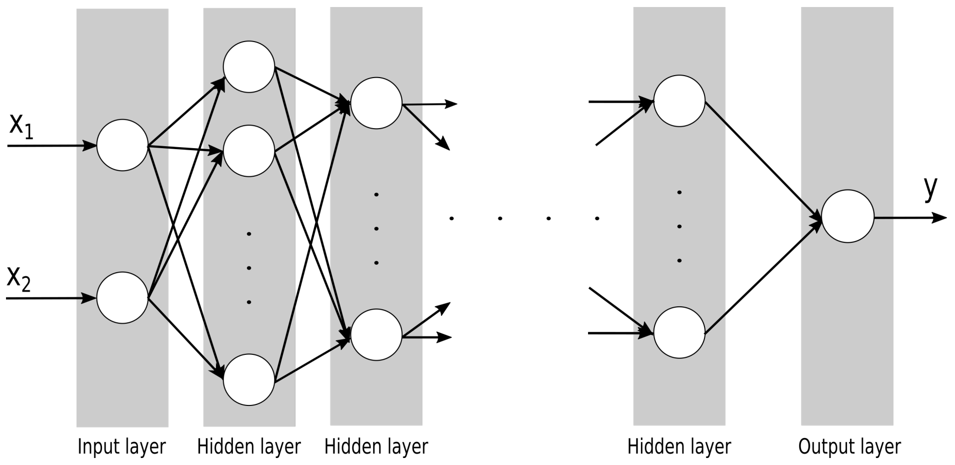


Figure 1-2: Deep Neural Network with multiple hidden layers approximating a mapping with 2 inputs and 1 output.

will be probably not close to optimal and an interpretation of the network properties based on well-known theory of dynamics could help the selection of a better set of network weights for an examined application or even possibly the development of more efficient optimization algorithms to learn dynamics.

1-2 Problem statement

The need to ensure **stability for learning control methods** provides the general motivation for this study and Lyapunov theory can propose general results to accomplish so. Since no previous knowledge and insight into the dynamics are assumed, a systematic methodology should be developed in order to perform stability analysis. To keep this analysis tractable, while being able to approximate generic and complex mappings, the representations of dynamics under examination should include a significant number of simpler models that combined create the necessary variety of nonlinearities. A similar decomposition should also be performed for the general stability conditions, where one or more simple conditions will correspond to a single linear model. Then, if all the individual stability conditions are fulfilled, a stability property of the overall system will be deduced.

Depending on the learning method, both local and overall dynamics representations change largely, rendering the derivation of a common set of conditions for all learning methods improbable. It is, thus, rather important to determine the methods that would be good candidates and provide the criteria for their selection. The two candidates that were chosen for this work are:

1. Deep Neural Networks (DNNs) with Rectified Linear Units;
2. Locally Weighted Learning (LWL) models.

DNNs were partly examined as a motivational example (see Section 1-1-4) and are is well-known for their ability to approximate high-dimensional mappings with good prediction accuracy, as well as to learn hierarchical representations with different levels of abstraction. This implies that DNNs are a very generic tool, able to approximate a wide variety of different dynamics, usually not achievable with other learning methods. Furthermore, they can be perceived as the synthesis of smaller components usually going under the name of neurons and layers and their models can very concisely be described by a set of relations. The choice of activation function affects largely the mappings that can be approximated and since most recent DNNs use Rectified Linear Units (ReLUs) in their hidden layers, the stability analysis for DNNs will focus on this particular class of activation functions.

Another motivating factor for developing such a theory for DNNs is the high impact that could have on understanding and gaining insight into this learning method, which has received unprecedented attention in many applications during the recent years. Finally, due to this popularity, there is already a variety of training algorithms efficient enough to process the very large training data sets, usually present in control applications.

On the other hand, DNNs are not an optimal choice for online learning. For many applications like speech and image recognition, there is no need to adjust the mapping based on new knowledge, but for control applications it is not uncommon that new experiences are constantly arriving from a possibly non-stationary environment and the system is expected to adjust itself in relatively short amount of time. For DNNs, learning from the beginning will become quickly infeasible and important knowledge may have to be discarded together with the corresponding training samples in order to maintain a reasonably sized training data set. A learning paradigm that handles these important considerations is *incremental learning*, which relies on a compact representation of obtained knowledge, while adjusting what has already been learned based on new training data, instead of storing indiscriminately all the previous training samples. Furthermore, DNNs require prior knowledge on the number of parameters to be learned and the size of the representation in general, a fact that may decelerate learning.

These concerns for online learning will be addressed with the second candidate to be examined; LWL. The methods that comprise this class are usually incremental and they also have a compact and easy way to describe dynamics. Furthermore, no assumptions on the number of local models and by implication on the number of parameters are required. Nevertheless, many of these methods do not achieve prediction performance as good as DNNs, especially in higher dimensional cases, where they suffer from the *curse of dimensionality.* Finally, they have not been so thoroughly studied in literature as the later and they include a large number of hyperparameters required to be tuned before training.

With the choice of these two learning methods it is hoped that an overall approach to the problem of stability guarantees for learning control methods can be proposed. Though other learning methods could be selected, these two candidates appear to be complementary to each other and manage to fulfill all the requirements for nonlinear representation composed of simpler linear models. Further, it is expected that the analysis tools and results introduced in the rest of this study will be generalizable to other popular learning methods with similar dynamics descriptions.

In conclusion, the core research problem, around which this study will be centered, can be formalized as follows:

Development of mathematically solid methodologies to evaluate stability properties of dynamics given in the form of ReLU Deep Neural Networks and Locally Weighted Learning models.

1-3 Approach

After having broadly described the goal of this thesis in the precious section, it will now be outlined how to approach this research question. As claimed a couple of times already, Lyapunov theory plays a key role in stability analysis for nonlinear systems, but constitutes a constructive method that requires a number of assumptions before one obtains results of some use. In this section, it is further explained how these results will be derived from Lyapunov theory in a principled way for the aforementioned two representations.

The large number of local models poses a significant numerical challenge, since the number of stability conditions will increase accordingly. In order to keep the computational cost tractable and be able to deduce stability properties in a reliable manner, it is essential to transform the stability conditions in an optimization problem that can be solved efficiently. One of the most popular choices for optimization problems in control theory are Linear Matrix Inequality (LMI) problems, which belong to the general class of Convex Programming (CP). Due to the convexity of the problem and the increase of available computation power, a relatively large number of these conditions can be included in the optimization problem and it will still remain tractable.

This general framework of stability analysis is certainly not a new idea and has been used in many different control settings and model representations, in particular for *Piecewise Affine (PWA)* and *Takagi-Sugeno (T-S) fuzzy* systems. For both of these classes of systems, the overall model is usually a combination of locally activated, linear (in the state vector) models. Their difference can be noted in the fact that local models in the case of T-S fuzzy models may overlap, while for PWA they do not. The focus of this study will be to leverage ideas from these two powerful frameworks to propose new conditions that examine the stability properties of DNNs and LWL methods. Although the stability theory developed for PWA and T-S fuzzy systems provide a stepping stone, further adjustments of the theory will be needed.

For DNNs, their close representational similarity with PWA systems will be exploited in order to obtain novel stability results for DNNs. The large number of local models present for DNNs compared to PWA systems has a number of consequences, especially with respect to computation cost, which will be taken into consideration in this study. On the other hand, LWL comprises several methods, one of which, Receptive Field Weighted Regression (RFWR), shares some common features with T-S fuzzy systems. After elaborating on the choice of this particular method, several assumptions required for Lyapunov stability analysis are sequentially assessed with respect to their conservativeness. While relevant theory of T-S fuzzy systems will be examined and evaluated in order to gain some insight, novel results will be additionally proposed in order to face the challenges posed by LWL methods.

It is worth mentioning that in this work only the learned representations will be analyzed with respect to stability and their relation with the real dynamics will not be assessed. Although in the end someone is interested in the real properties, it has been remarked that the stability conditions for a specific model representation can usually be extended to deduce the corresponding properties for the true dynamics via robust stability. This extension could be accomplished using some measure of the error between the real and the model dynamics, or assumptions on the uncertainty and its source. Therefore, the analysis of the nominal case i.e. learned dynamics can be seen as the important and general first step before evaluating the true dynamics.

1-4 Related work

In this section, studies relevant to the aforementioned research problem will be stated. Starting from an overview of learning control with guarantees, the application of DNNs and LWL methods for learning control will be examined.

1-4-1 Learning control with guarantees

Despite being described as open in the previous sections, the problem of learning control with guarantees is not new and there have been some studies trying to address it. Most of them have been released in the previous couple of decades and examine stability of dynamics representations relying on recurrent and shallow feedforward Neural Networks. However, a few recent studies focused on Gaussian Processes are presented at end of this subsection.

If the NN includes one or more feedback loops between inputs and outputs and/or hidden layers, then it is referred to as a recurrent network [18]. The apparent connection with nonlinear dynamic systems has rendered the stability analysis for recurrent NNs an important consideration and the literature on this topic has been focused on different structures like cellular NNs [19, 20], delayed cellular NNs [21, 22], and Hopfield NNs [23–25]. Despite the success of these types of recurrent neural networks in some applications relative to pattern recognition, image processing and associate memory, they have not yet managed to achieve the wide acceptance of DNNs. The computational tools applied for stability analysis in these cases are quite different from the ones proposed in this study.

Due to the piecewise linear nature of the ReLU activation function, literature on stability of feedforward NNs with piecewise linear activation functions is especially interesting for this study. Nonetheless, to the best of the author's knowledge, only a couple of studies have been concerned with this issue. In [26, 27], stability conditions based on a piecewise affine activation function, called in the context of these studies Piecewise Affine Perceptron (PAP), were proposed relying on a piecewise quadratic Lyapunov function. Although the activation function mentioned is piecewise linear (as the ReLU), more linear regimes are considered for PAP and its response tries to resemble the one of sigmoid activation functions, and thus also suffers from a vanishing gradient as the latter (in the case of multiple layers). Finally, an important remark applicable not only for PAP, but for every activation function is the non-existence of any stability result and framework for DNNs, which is one goal of this study.

A more recent approach to the topic of stability guarantees for learning control, explored especially the last years, has been centered around Gaussian Processes (GPs) as the main method to learn and represent dynamics. An extensive study on stability guarantees of dynamics learned with GPs can be found in [28]. One of the characteristics of GPs that distinguishes them from other learning methods is the availability of uncertainty information, which can be used to derive more robust control policies. In [29], a linearized model and its uncertainty estimates around an operating point were computed from the GP dynamics and based on them a linear controller was derived solving a LMI problem. Finally, regions of attraction around equilibria of the nonlinear dynamics (expressed as GPs) are computed in [30] taking also into consideration uncertainty relying mainly on Lyapunov theory and Bayesian optimization.

1-4-2 Learning control for DNNs and LWL methods

For supervised learning, Neural Networks with one hidden layer and logistic or sigmoid activation functions have been studied extensively for modeling and control of dynamical systems [31, 32], but, unexpectedly, the use of DNNs in this area has been very limited. In [33], deep neural networks with ReLUs were trained to approximate the solution to the Hamilton-Jacobi-Bellman equations, while in [34] DNN provided policies equivalent to PID controllers. In [35], a DNN was trained to provide both state and control input for a whole trajectory. In the two last studies sigmoid activation functions were considered without providing any intuition why this choice would give better results than ReLUs. It should be noted that DNNs have found many applications in reinforcement learning [36], where their main use is the approximation of different value functions.

Despite being firstly examined in fields like statistics and regression of nonlinear relations, LWL methods have become very popular in a number of learning control applications, where they were used to approximate all kinds of mappings necessary for robotics and control. Locally Weighted Regression (LWR) was the first method of the class introduced for control and has been applied in a wide variety of tasks, learning inverse and forward models, as well as variations of linear quadratic controllers [37]. Besides its computational advantages, this method experienced a number of constraints, caused mainly by the fact that every sample had to be kept in memory, while it was required to do a variation of least squares in every iteration *(lazy learning)*.

LWL algorithms that were more appropriate for online learning have been developed and tested in a few applications since then. In [38], such a method, Locally Weighted Projection Regression (LWPR) was chosen to approximate challenging nonlinear forward dynamics in high-dimensions and applied in conjunction with the iterative Linear Quadratic Gaussian (iLQG) algorithm [39] to control a complex manipulator with many degrees of freedom. An interesting result with some stability implications was proposed in [40], where a Single Input Single Output (SISO) continuous function was approximated with a LWL method, while a corresponding adaptive law was attempting to ensure successful reference tracking. Unfortunately, the extension to Multiple Input Multiple Output (MIMO) cases is straightforward only under specific relatively strict conditions. In [41], a short review of most studies that have addressed learning control problems with LWL methods can be found.

In conclusion, for both of these learning methods there are currently no (up to the author's knowledge) any previous studies related to safety guarantees or stability conditions of the representing dynamics in any form.

1-5 Outline of the Thesis

The remainder of this thesis is organized as follows.

In Chapter 2, some general theoretical results and tools needed for the rest of the study are introduced. Firstly, well-established stability properties as well as theorems to examine their validity for general dynamic models are revisited relying on *Lyapunov stability theory*. Then, the concept of *conservativeness* and its importance for assessing a stability analysis result is explained. Finally, mathematical insight into Linear Matrix Inequalities and the reasons for their popularity are provided in the last section of the chapter.

Chapter 3, entitled *Deep Neural Networks* provides the first specific stability results for ReLU DNNs, the most common type of DNNs. After showing the connection of this architecture to PWA systems, stability conditions and a general framework are developed in order to study its most important stability properties. This framework is later evaluated in multiple examples, where the efficiency and the significance of this stability analysis is shown.

Locally Weighted Learning is studied with respect to stability in Chapter 4, as a more appropriate method than DNNs for online learning. Since there are multiple candidate methods to select from in this class, the most suitable one is selected and will be the basis for the stability analysis. Starting gradually from simpler results, a number of stability theorems are proposed and evaluated with respect to their conservativeness, both in theory and practical examples.

Several contributions and concluding remarks of this study along with directions for future research are presented in Chapter 5.

Chapter 2

Background

In this chapter, important theory around basic notions mentioned in the Introduction such as *stability* and *Linear Matrix Inequalities (LMIs)* will be briefly presented. The intention of this chapter is **not** to present a thorough and extensive analysis on these complex topics, which usually require some effort to get a solid grasp on, but rather provide crucial and compact definitions that will facilitate comprehension of the rest of this study. Nevertheless, appropriate references to studies that can provide further insight will be given in every stage of this chapter. Starting from a very general description of dynamics, the intuition behind Lyapunov stability theory will be shortly described as several important definitions and theorems around different notions of stability will be recalled. In the second section, the general formulation of LMIs, their connection with control theory and the reasons of their popularity in these applications will be given.

2-1 Mathematical notation

Before proceeding with the introduction of more complex mathematical notions and definitions, some short important notation rules aiming to facilitate comprehension of the mathematical analysis needs to be given. Most of them have been adapted from the Matrix Cookbook [42] and the interested reader can find in the last pages an additional list with symbols or abbreviations used throughout this study. Unless noted otherwise for a specific case:

- If A and x are a matrix and a vector respectively, then A_i, A^j, A_i^j and x_i, x^j, x_i^j are their indexed versions for some purpose.
- if x is the state or a closely related vector, then $x(k)$ is its value at time step k.
- If x is a vector, then $(x)_i$ is its i^{th} element.
- If A is a matrix, then $(A)_{ij}$ is its $(i,j)^{\text{th}}$ entry.
- If x is a vector, $||x||$ is its **Euclidean** norm.

- $\mathbf{1_n}$ is a vector with n entries equal to 1.
- $\mathbf{0_n}$ is a vector with n entries equal to 0.
- $\mathbf{0_{n\times m}}$ is a matrix with n rows and m columns whose entries are all equal to 0.

2-2 Lyapunov stability theory

2-2-1 Dynamics representation

Consider the dynamics of a nonlinear system, whose properties should be studied in order to control it. These dynamics can either be described by a continuous-time or a discrete-time relation. The emergence of digital computers in control applications, as well as the interest to derive models from real data makes the discrete-time formulation more appropriate for this study and from now on only discrete-time dynamics will be considered. Therefore, assume that the state progression of the system can be fully described by the following very general relation:

$$x(k+1) = f(x(k)), \quad x(0) = x_{\text{init}} \tag{2-1}$$

where $x(k) \in \mathbb{R}^n$ is the state vector of the system in time step k, f is a nonlinear function, x_{init} is the initial condition and n is the state dimensionality. Furthermore, the above dynamics are assumed to be *time-invariant* i.e. the f function does not depend (explicitly) on time step k.

When the stability of the above system is studied, it is usually attempted to assess how the system will behave after long periods of time, possibly after some perturbation in the initial conditions. It is possible that the system will remain in a state relatively close to x_{init} or it will diverge from it as time tends to infinity. The first condition is usually characteristic of a *stable* system and the second of an *unstable*.

These very broad definitions, although useful to understand, are made more precise by *Lyapunov stability theory*, which aims to assess stability with respect to specific points in the state space commonly called *equilibria* or *fixed points* x_e. Except for Lyapunov theory, other ways to assess stability of a system such as *Input-output stability* have also been proposed but will not be considered in this study.

2-2-2 Definitions of stability

Assume that (2-1) has an equilibrium, x_{e} (not necessarily unique), whose stability properties should be analyzed. Firstly, an equilibrium of (2-1) is strictly determined by the following relation:

$$x_e = f(x_e)$$

There are, however, numerous notions of stability of x_e according to Lyapunov, with the most general being the notion of *stability in the sense of Lyapunov.*

Definition 2.1 (Stability in the sense of Lyapunov, [9, Def. 5.4]). *Consider the autonomous nonlinear dynamics* (2-1)*. If for each* $\epsilon \in \mathbb{R}_{>0}$*, there is* $\delta = \delta(\epsilon) > 0$ *such that:*

$$||x_{\text{init}} - x_e|| \le \delta \implies ||x(k) - x_e|| \le \epsilon \quad \forall k \in \mathbb{N}^1,$$

[1]$\mathbb{N}$ is the set of natural numbers

then equilibrium x_e *is* stable in the sense of Lyapunov (i.s.L)*.*

The above definition implies for a stable equilibrium x_e that if the trajectory starts in a ball with radius δ around x_e, then it will necessarily remain in a ball around x_e with radius ϵ. Therefore, by starting sufficiently close to the equilibrium, it is guaranteed that the trajectory will main sufficiently close in the long run. Another very popular notion of stability is the *asymptotic stability*, formally defined as:

Definition 2.2 (Asymptotic stability, [9, Def. 5.4])**.** *Consider the autonomous nonlinear dynamics* (2-1)*. If* x_e *is stable in the sense of Lyapunov and there exists a* $\delta \in \mathbb{R}_{>0}$ *such that:*

$$||x_\mathrm{init} - x_e|| \leq \delta \implies \lim_{k\to\infty} ||x(k) - x_e|| = 0$$

then equilibrium x_e *is* asymptotically stable*.*

In this case, except for being stable i.s.L, equilibrium x_e is also *attractive*, which means that if the state starts relatively close to the equilibrium, it tends to converge to it as time tends to infinity. Asymptotic stability is obviously is a stronger notion of stability than stability i.s.L, since the former implies the latter, but the opposite is not generally true.

Although asymptotic stability guarantees that the state will converge in the end to the equilibrium it does not provide any information about how quickly this will be achieved. For this reason, *exponential stability* is defined:

Definition 2.3 (Exponential stability, [43, Def. 2.2.1])**.** *Consider the autonomous nonlinear dynamics* (2-1)*. If there exists some* $\delta, \theta \in \mathbb{R}_{>0}$*, and* $\rho \in [0, 1)$ *such that:*

$$||x_\mathrm{init} - x_e|| \leq \delta \implies ||x(k) - x_\mathrm{e}|| \leq \theta\rho^k ||x_\mathrm{init} - x_\mathrm{e}||, \quad \forall k \in \mathbb{N},$$

then equilibrium x_e *is* exponentially stable*.*

Exponential stability ensures that trajectories starting from an initial condition will converge exponentially quickly to the corresponding equilibrium, which is a stronger notion than asymptotic stability.

All three of these definitions do not necessarily include the whole state space, but neighborhoods around the equilibrium, something that makes them *local* definitions. The *global* notions can be deduced by adding the requirement that the same conditions apply for every $x_\mathrm{init} \in \mathbb{R}^n$.

2-2-3 Lyapunov functions

Although different definitions of stability have been described, it is still not clear how this stability will be determined, given dynamics (2-1). Lyapunov theory tries to answer this problem by finding functions of the state that express some measure of energy dissipation and based on them assess how the energy of the system fluctuates through time. This way it is not necessary to find all the solutions of the finite differences equation (2-1), but a more structured and principled approach can be followed.

Searching for this kind of functions that will help deduce on stability is not easy and there is no standard way to do it without making some assumptions on the dynamics of the system and the nature of the function itself. Usually, as the assumptions get stronger, the easier it is to search for *Lyapunov functions* i.e. scalar functions whose existence guarantee stability for this particular system. Generally speaking, in order for a candidate function $V(x)$ to qualify as a Lyapunov function for the system, it should be positive definite and should be decreasing along every system trajectory. Thus, it is obvious that the latter property can be evaluated by computing the difference of the function in between two consecutive time steps.

There are multiple results and variations in literature (for just some of them see [9,10,43,44]) aiming to translate these very broad requirements to more compact mathematical conditions, but usually different assumptions are made, changing slightly the final preposition. In this section, to avoid confusion, while providing all the necessary tools for later analysis only a couple of them concerning **global** stability will be given without proof. In the next subsection, the case of local stability will also be examined in more detail.

For continuous Lyapunov function and continuous dynamics the following theorem concerning global asymptotic stability can be proposed:

Theorem 2.1 (Global asymptotic stability [10]). *Let x_e be an equilibrium of* (2-1) *and f be continuous in x. If there exists a continuous function $V\colon \mathbb{R}^n \to \mathbb{R}$ such that the following conditions are satisfied:*

- $V(x_e) = 0$ (2-2a)
- $V(x) > 0, \qquad \forall x \in \mathbb{R}^n \setminus \{x_e\}$ (2-2b)
- $||x|| \to \infty \implies V(x) \to \infty$ (2-2c)
- $\Delta V(x) = V(f(x)) - V(x) < 0, \quad \forall x \in \mathbb{R}^n \setminus \{x_e\}$ (2-2d)

then equilibrium x_e is globally asymptotically stable.

To summarize, if all four of these conditions are fulfilled for a known candidate function V, then the system is guaranteed to be asymptotically stable for every initial condition. Equivalently, a theorem proposing similar conditions for global exponential stability can be phrased as follows:

Theorem 2.2 (Global exponential stability [43, Th. 2.2.4]). *Let x_e be an equilibrium of* (2-1) *and f be possibly discontinuous in x. If there exist a possibly discontinuous function $V : \mathbb{R}^n \to \mathbb{R}$ and $\alpha_1, \alpha_2, \alpha_3, \eta \in \mathbb{R}_{>0}$ such that the following conditions are satisfied:*

- $V(x_e) = 0$ (2-3a)
- $\alpha_1 ||x||^\eta \leq V(x) \leq \alpha_2 ||x||^\eta, \qquad \forall x \in \mathbb{R}^n$ (2-3b)
- $\Delta V(x) = V(f(x)) - V(x) \leq -\alpha_3 ||x||^\eta, \quad \forall x \in \mathbb{R}^n$ (2-3c)

then equilibrium x_e is globally exponentially stable.

It is worth noting that this theorem includes an additional relaxation compared to Theorem 2.1, which required that the Lyapunov function should be continuous. Now the Lyapunov function can also be discontinuous in order to assess exponential stability and this relaxation will be proven important in later chapters.

2-2-4 Local stability

Although global stability, when possible to be proven, provides very strong information about the system, it requires a unique equilibrium, which is not always the case especially for more complex dynamical systems. Moreover, ensuring that the controlled dynamics will converge for any arbitrary initial condition to the equilibrium is too restrictive and sometimes even unnecessary. Therefore, it is necessary to provide additional conditions of a Lyapunov function for more "local" results.

To characterize and describe local stability in this study another important notion will be introduced: *Positively Invariant (PI) set.* Formally, a set $\mathcal{B} \subseteq \mathbb{R}^n$ is called PI under dynamics (2-1) if for every $x \in \mathcal{B}$, $f(x) \in \mathcal{B}$. Intuitively, this implies that if the trajectory enters set $\mathcal{B}$, then it is certain that it will remain in $\mathcal{B}$ for **all future** time steps.

Consider now the following definition of (local) stability in set $\mathcal{B}$.

Theorem 2.3 (Exponential stability in $\mathcal{B}$ [43, Th. 2.2.4])**.** *Let x_e be an equilibrium of* (2-1)*, f be continuous in x, and $\mathcal{B} \subseteq \mathbb{R}^n$ be a PI set. If there exist a possibly discontinuous function $V : \mathcal{B} \to \mathbb{R}$ and $\alpha_1, \alpha_2, \alpha_3, \eta \in \mathbb{R}_{>0}$ such that the following conditions are satisfied:*

- $V(x_e) = 0 \qquad \text{(2-4a)}$
- $\alpha_1 ||x||^\eta \leq V(x) \leq \alpha_2 ||x||^\eta, \quad \forall x \in \mathcal{B} \qquad \text{(2-4b)}$
- $\Delta V(x) = V(f(x)) - V(x) \leq -\alpha_3 ||x||^\eta, \quad \forall x \in \mathcal{B} \qquad \text{(2-4c)}$

then equilibrium x_e is exponentially stable *in $\mathcal{B}$.*

If the right part of inequality (2-4b) is true only in a subset of $\mathcal{B}$ (that contains x_e), then x_e is said to be *locally exponentially stable.* Although this definition seems very similar to the notion of stability in set $\mathcal{B}$, it does not provide any precise information about how far the trajectory can be from the equilibrium and still converge to it, since the subset of $\mathcal{B}$ can be arbitrarily small.

This interest to compute more concretely these special regions leads to the notion of *Region of Attraction (RoA)* (or *Domain of Attraction*) of an equilibrium. Formally, if $\phi(k,x_\text{init})$ is the solution of the first-order difference Equation (2-1) with x_init being the initial condition, then the RoA of x_e $\mathcal{R}$ can be defined as the following set:

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid \lim_{k\to\infty} |\phi(k,x)| = 0\}$$

Essentially, if the trajectory starts from a point inside this set, it is guaranteed that it will be attracted to the corresponding equilibrium. Computing the exact RoA of a particular equilibrium is usually a very difficult task, if not impossible and in most cases a close approximate of $\mathcal{R}$ is sufficient. For the case of Theorem 2.3, if conditions (2-4) are fulfilled, then $\mathcal{B}$ is a Region of Attraction of equilibrium x_e.

2-2-5 Conservativeness consideration

Lyapunov theory is generally not able to provide *necessary and sufficient* conditions for stability and the existence of a Lyapunov function in the three aforementioned theorems is a *sufficient* stability condition. This implies that if it is possible to find a function that has the properties predicted by the theorem, then it is proved that the corresponding equilibrium is indeed stable. On the other hand, if it is not possible to compute such a function, no conclusion can be drawn, while the equilibrium could be stable in reality.

It is, nevertheless, crucial to evaluate how the proposed conditions in each theorem can help to deduce stability properties with a level of confidence. To better explain this notion of *conservativeness*, consider the following simple, theoretic case study. Assume that one is given a set of precise (different) dynamical models, which are all known to be globally asymptotically stable. Then, two different stability results like the previous theorems are developed to assess whether the equilibria are globally asymptotically stable. The stability theorem that is able to find more stable models in the previous set can be thought as the **less** conservative, and thus, the more "useful". This, obviously, is not an absolute notion and depends on the dynamical models, composing the data set, but provides strong insight into the level of trust that should be put in a certain set of conditions.

It can be argued then that from a variety of stability results given in literature (for the same dynamics formulation), someone should always choose the one that is less conservative in the long run. Unfortunately, an important trade-off between conservativeness and computational cost usually exists, which will be examined in detail in this study. Moreover, as it will be shown in a number of cases, the assumptions on the candidate Lyapunov functions affect largely the quality of the result and a number of factors should be taken into account, while deciding on these assumptions. Therefore, one of the contributions of this study will be the "uncovering" of these factors for a number of useful dynamics representations.

2-3 Linear Matrix Inequalities (LMIs)

2-3-1 Preliminaries

According to [45], a Linear Matrix Inequality (LMI) has the following general form:

$$F(x) = F_0 + \sum_{i=1}^{p} x_i F_i > 0, \tag{2-5}$$

where $x \in \mathbb{R}^p$ is a vector full of variables and $F_i \in \mathbb{R}^{m\times m},\ i = 0,\ldots,p$ are symmetric matrices. For the LMI to be true, it is required for $F(x)$ to be a positive-definite matrix in the sense that $z^T F(x) z > 0$ for every $z \neq \mathbf{0_n}$

$F(x)$ is positive-definite if and only if all the principal minors of $F(x)$ are positive, and thus LMI (2-5) can be always transformed in a number of polynomial inequalities on vector x (see [46]).

More than one such LMIs $F^1(x) > 0, F^2(x) > 0, \ldots, F^M(x) > 0$ can be transformed to a single one the following way:

$$\operatorname{diag}\left(F^1(x), F^2(x), \ldots, F^M(x)\right) > 0$$

where diag is an operator that creates a square diagonal matrix from the elements given as inputs. Given an LMI $F(x) > 0$, the solution of the corresponding LMI problem is an x_0 such that $F(x_0) > 0$ (or determine that the problem is infeasible) and can be derived using well-known Convex Programming algorithms like interior-point methods [47].

2-3-2 Simple example for LTI systems

To illustrate how LMI problems are related with stability and more specifically with Lyapunov stability Theory, one very simple case related to a discrete-time Linear Time Invariant model is examined[2]. As known the dynamics of an autonomous LTI system are fully described as:

$$x(k+1) = Ax(k)$$

where $A \in \mathbb{R}^{n\times n}$ is a known matrix. If the Lyapunov function is parametrized as a quadratic function with respect to the state vector i.e. $V(x(k)) = x(k)^T P x(k)$, where P is an unknown matrix, then Theorem 2.1 indicates that the linear system is **global asymptotically stable** if conditions (2-2) are guaranteed.

To ensure condition (2-2b) it is required that $P > 0$ i.e. P is positive-definite and then conditions (2-2a), (2-2c) are true. Finally, condition (2-2d) can be transformed as:

$$A^T PA - P < 0$$

In order to see how these two conditions are LMIs in P i.e. P is the variable, then assume a basis $B^{11}, B^{21}, \dots, B^{ij}, \dots, B^{nn}$ for symmetric matrix P such that $B^{ij} \in \mathbb{R}^{n\times n}$ with $i \geq j$ has its $(i,j)^{\text{th}}$ and $(j,i)^{\text{th}}$ entries equal to 1 and all its other elements equal to 0. Then, matrix P can be written as:

$$P = \sum_{j=1}^{n}\sum_{i\geq j}^{n}(P)_{ij}B^{ij}$$

Now, it is obvious how condition $P > 0$ can be reformulated as in (2-5) using this basis. For condition $A^T PA - P < 0$ (sometimes called *discrete Lyapunov inequality*), it is noted that:

$$\begin{aligned} A^T PA - P &= A^T\Big(\sum_{j=1}^{n}\sum_{i\geq j}^{n}(P)_{ij}B^{ij}\Big)A - \sum_{j=1}^{n}\sum_{i\geq j}^{n}(P)_{ij}B^{ij} \\ &= \sum_{j=1}^{n}\sum_{i\geq j}^{n}(P)_{ij}\Big(A^T B^{ij} A - B^{ij}\Big) \end{aligned}$$

Therefore, it can be seen that the second LMI can be brought into the standard form by replacing x_i with $(P)_{ij}$ and F_i with $-A^T B^{ij} A + B^{ij}$. Finally, these two LMIs can be expressed as one the following way:

$$\operatorname{diag}(P, -A^T PA + P) > 0.$$

2-3-3 S-procedure

The *S-procedure* is a relaxation technique that allows to replace a specific inequality by a stronger one, imposing on the meanwhile an additional number of conditions. For this study, the version of *lossy S-procedure with strict inequalities* presented below will be mainly used, but the interested reader can find more on S-procedure on a number of sources (see e.g. [45, 48]).

Let $F^0(x), F^1(x), \dots, F^L(x)$ be some real-valued functions of x and $\tau_1, \dots, \tau_L$ some real numbers. Assume the following two conditions:

[2]Similar examples have been proposed in other fundamental studies on LMIs [45, 46], and thus most of the details will be left out to keep the analysis as short as possible, while providing important insight into LMIs.

(a) $F^0(x) > 0$ for all x such that $F^1(x) \geq 0, \ldots, F^L(x) \geq 0$

(b) There exist $\tau_1 \geq 0, \ldots, \tau_L \geq 0$ such that $F^0(x) - \sum_{l=1}^{L} \tau_l F^l(x) > 0$

It can be proven that condition (b) implies condition (a) and the S-procedure refers to this attempt to prove the first condition through the second one. The free variables τ_l are called *multipliers* and the condition can be also easily transformed to check when $F^0(x)$ is negative instead of positive.

In the context of this study the S-procedure will be used to reduce conservativeness of a quadratic function being positive (or negative) by restricting the region of interest in a specific subspace of the state space. Therefore, from now, $F^0(x), F^1(x), \ldots, F^L(x)$ are quadratic functions of vector x.

To illustrate how S-procedure will be used in the following chapter, consider the following simple LMI problem. Assume that it is desired to prove that $F^0(x) = x^T P x > 0$ for every $x \in \mathcal{D} \subset \mathbb{R}^n$. If it is possible to prove that P is positive definite, then obviously the previous condition is necessarily fulfilled. However, this requirement is more conservative and can be relaxed by considering only the necessary subspace $\mathcal{D}$.

To achieve that assume that it is known that $F^1(x) = x^T S x \geq 0$ for every $x \in \mathcal{D}$. Then, if there exists $\tau \geq 0$ such that:

$$P - \tau S > 0,$$

then:

$$x^T P x - x^T \tau S x > 0$$

and condition (b) from above is fulfilled. By implication then $F^0(x) = x^T P x > 0$ for all $x \in \mathcal{D}$. In conclusion, the condition $P - \tau S > 0$ is more easier to be fulfilled that $P > 0$ and with the cost of some additional computations (since we need to compute an appropriate value for the multiplier τ) less conservative conditions can be derived, rendering stability analysis more informative as will be seen in next chapters.

Chapter 3

Stability analysis of Deep Neural Networks

In this chapter, a class of representations used commonly to learn nonlinear dynamics will be examined with respect to stability: *ReLU Neural Networks*. In the first section, a different way to describe state dynamics represented with DNNs will be introduced in order to setup the stage for later stability analysis. The equivalence of DNNs with an other popular class of dynamical models, namely Piecewise Affine (PWA), will be formally proved in Section 3-2, a result that constitutes an important contribution of this chapter. Relying on the powerful theory of PWA systems, a complete framework for stability analysis of the learned dynamics is fully developed in Section 3-3, providing conditions for global exponential stability and a precise algorithm to compute estimates of Regions of Attraction (RoAs) for the case of multiple equilibria. In the latter sections of this chapter, the proposed stability analysis is thoroughly evaluated in numerical examples and real case studies.

3-1 Description of dynamics with Deep ReLU Neural Networks

It would not be an exaggeration to say that the general theory of NNs has been well-established and examined in numerous sources (see e.g. [49,50]), to which the interested reader is referred for an extensive insight on their basic characteristics. Due to this availability of material only the necessary aspects required for later analysis will be mentioned in this section, starting from short description of function approximation with Deep Neural Networks (DNNs) and continuing to ReLU NNs, which is by far the most commonly used class of DNNs. Nevertheless, some notation and tools introduced in parts of this section are novel and necessary due to the lack of similar, previous results for this class of dynamics representations.

3-1-1 Modeling dynamics with Deep Neural Networks

The general aim of Artificial Neural Networks is the derivation of an optimal mapping between inputs with dimensionality p i.e. $x \in \mathbb{R}^p$ and outputs with dimensionality q i.e. $y \in \mathbb{R}^q$ with respect to some measure of the function approximation error. Assume that a dataset of N observations i.e. $\{x_j, y_j\}_{j=1}^N$ is given to learn this mapping. NNs belong to the class of *parametric learning techniques*, where the existence of a fixed number of parameters is assumed before learning [51]. The number of parameters for NNs is explicitly dependent on the number of *neurons* of the network, which are essentially their building block.

Although some of the remarks for DNNs that will follow were made briefly back in Chapter 1 (see Subsection 1-1-4), a full mathematical description will be now provided such that an overall analysis is available. Consider that a number of n_1 separate affine transformations are applied to the input vector $x \in \mathbb{R}^p$, followed by a nonlinear vector-valued function $\sigma : \mathbb{R}^{n_1} \to \mathbb{R}^{n_1}$. This operation can be better formulated as:

$$y_{neu} = \sigma(W_1 x + B_1)$$

where matrix $W_1 \in \mathbb{R}^{n_1 \times p}$ is usually referred to as the *weight matrix* and $B_1 \in \mathbb{R}^{n_1}$ as the *bias vector*. The above relation explains the operations described by a *single hidden layer* of n_1 neurons and together with a commonly-used affine transformation on the outputs of this layer the *shallow* neural network is created:

$$y = W_2 \sigma(W_1 x + B_1) + B_2 \tag{3-1}$$

where $W_2 \in \mathbb{R}^{q \times n_1}$ and $B_2 \in \mathbb{R}^q$. The sizes of weight matrices W_1, W_2 and bias vectors B_1, B_2 are partly determined by the input and output dimensionalities and partly by the number of neurons, The training process attempts to compute values for them such that the error measure is minimized. The nonlinear function σ is also determined before the training and a number of different choices has been proposed in literature such as the *logistic* and the *hyperbolic tangent* function. This study will focus on an other activation function: the *Rectified Linear Unit (ReLU)*, but more details on that will be given in the next section.

Equation (3-1) describes a mapping with one hidden layer and one output layer, a very common and useful structure in learning applications, which, however, does not manage to integrate hierarchical and structural features of the desired mapping. The insertion of additional hidden layers that process the output of previous layers intends to do so and a more general description of these *deep* networks needs to be provided, generalizing previous notation of shallow networks.

Consider the number of hidden layers known and equal to L and the additional affine layer denoted indexed as $L+1$, where n_j $(j = 1, 2, ..., L)$ denotes the number of neurons in the j^{th} layer. With a slight abuse of notation, let $a_{j-1} \in \mathbb{R}^{n_{j-1}}$ be the input-vector for the j^{th} layer, and $W_j \in \mathbb{R}^{n_j \times n_{j-1}}$ and $B_j \in \mathbb{R}^{n_j}$ the corresponding weighting matrix and bias. The output of the j^{th} ReLU layer a_j is then given by:

$$a_j = \sigma(W_j a_{j-1} + B_j)$$

For the first hidden layer and its output, it is assumed that:

$$a_1 = \sigma(W_1 a_0 + B_1) = \sigma(W_1 x + B_1)$$

Let $\ell_j(a_{j-1}) = W_j a_{j-1} + B_j$ denote the affine transformation of a_{j-1}. Therefore, the input-output mapping expressed as a deep neural network with L hidden layers can be given as:

$$y = f_{\text{NN}}(x) = \ell_{L+1} \circ \sigma \circ \ldots \sigma \circ \ell_1(x)$$

Since the mapping to be approximated by the DNN is the discrete-time state-space dynamics, the following assumptions (similarly to Chapter 3) are required:

- The input vector is replaced by $x(k)$ i.e. the state vector at time k.
- The output vector is replaced by $x(k+1)$ i.e. the state vector at the next time step.
- Input and output dimensionality are equal to state dimensionality n.
- If function composition is denoted with $\circ$, then the dynamics of the learned system can be expressed as follows:

$$x(k+1) = f_{\mathrm{NN}}(x(k)) = \ell_{L+1} \circ \sigma \circ \ldots \sigma \circ \ell_1(x(k)) \tag{3-2}$$

  where $n_1 = n_{L+1} = n$.

3-1-2 ReLU Neural Networks

Equation (3-2) suffices to express state-space dynamics with a deep neural network for any selected activation function σ. However, a special activation function has been adopted by the deep learning community; namely *Rectified Linear Unit* [52]. The reason for this popularity lies in the fact that this activation function suffers less from the *vanishing gradient problem* and will be the only one examined in this study. A very simple shallow ReLU neural network with $n_1 = 3$ neurons is presented in Figure 3-1a and will be used throughout this section as an example.

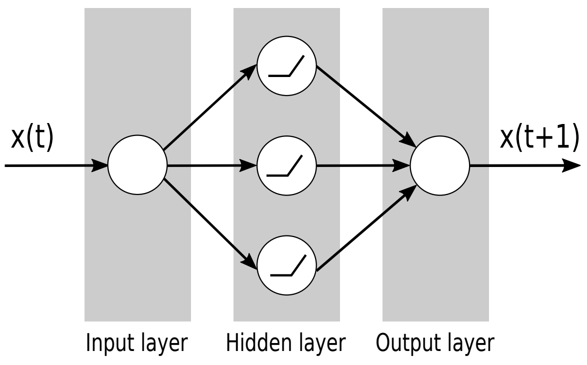
Formally, the ReLU activation function is introduced as the following vector-valued function:

$$\sigma(u) = [\max(0, (u)_1), \max(0, (u)_2), \ldots, \max(0, (u)_n)]^{\mathrm{T}}$$

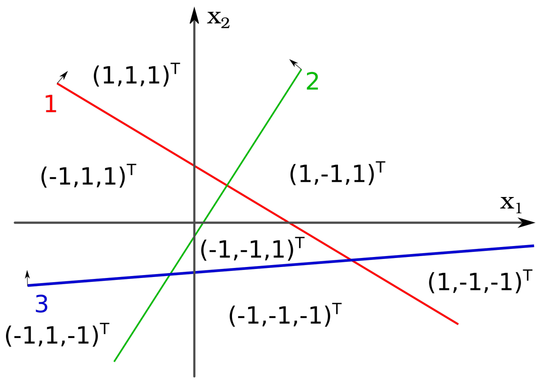
which is evaluated element-wise. The composition of the ReLU with an affine transformation ℓ_j will constitute the corresponding hidden layer, while the number of elements of vector u is equal to the number of neurons in it.

To better explain how the aforementioned activation function affects the dynamics, the well-established connection of a ReLU hidden layer with a hyperplane arrangement will be shortly described [54–56]. Each neuron in the j$^{\mathrm{th}}$ hidden layer corresponds to a hyperplane in $\mathbb{R}^{n_{j-1}}$, fully described by a row of matrix $W_j \in \mathbb{R}^{n_j \times n_{j-1}}$ and the corresponding entry in the vector $B_j \in \mathbb{R}^{n_j}$. For neural network in Figure 3-1a, given that the state-space is two dimensional, three lines in $\mathbb{R}^2$ can be defined for the three ReLU neurons as shown in Figure 3-1b.

Depending on the input a_{j-1}, different activation patterns will occur in layer j; that is, some neurons will be activated (input to neuron greater than zero), while others will not (less than zero). The hyperplane corresponding to a neuron is the boundary between the the half-space in which this particular neuron is activated and the half-space that it is not. A vector of markings is then introduced for each hidden layer, whose entries are $+1$ when the neurons on layer j are activated and -1 otherwise. In Figure 3-1b the 7 different activation patterns occurring for all the possible state combinations in $\mathbb{R}^2$ are defined and the corresponding vector markings are shown for each *linear region.*



(a) ReLU NN with one hidden layer.



(b) Hyperplane arrangement in $\mathbb{R}^2$ and corresponding marking vectors (each arrow shows the half-space in which the corresponding neuron is activated)[a].

[a]Inspired from [53]

Figure 3-1: Example of Neural Network Dynamics with 1 hidden layer and 3 Rectified Linear Units.

In order to keep up with all possible activation patterns in layer j, index $i_j \in \{1, \ldots, N_j\}$ with $N_j \leq 2^{n_j}$ is introduced. Therefore, for a given input $x(k)$ to the network (3-2), there is a specific combination of neurons activated in each layer, which is specified by $(i_1, i_2, \ldots, i_L)$. If the activation pattern i_j is observed in the j$^{\text{th}}$ layer, then the corresponding vector of markings is denoted as $z_j{}^{i_j} \in \{1, -1\}^{n_j}$. For the simple example of Figure 3-1 only one index i_1 is required, since there is only one hidden layer and its value will variate from 1 till 7. If $i_1 = 1$ denotes the linear region where all the neurons are activated, then $z_1^1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\text{T}}$. Obviously the vector of markings is implicitly depended on $x(k)$ and responsible for the desired nonlinear behavior of the network. Finally, the output of the j$^{\text{th}}$ ReLU layer for the i_j activation pattern will be given by:

$$a_j = W_j^{i_j} a_{j-1} + B_j^{i_j} \tag{3-3}$$

where $W_j^{i_j}$ and $B_j^{i_j}$ have rows of 0s when the corresponding neurons that are not activated (for the i_j activation pattern) and the rest of the rows are equal to the rows of W_j and B_j. By simple manipulations it can be seen that:

$$W_j^{i_j} = \left[\frac{1}{2}\operatorname{diag}(\mathbf{1}_{\mathbf{n_j}} + z_j^{i_j}) W_j\right] \qquad B_j^{i_j} = \left[\frac{1}{2}\operatorname{diag}(\mathbf{1}_{\mathbf{n_j}} + z_j^{i_j}) B_j\right]. \tag{3-4}$$

For a shallow (L= 1) ReLU NN, (3-2) can be simplified as follows:

$$x(k+1) = W_2\sigma(W_1 x(k) + B_1) + B_2 = W_2(W_1^{i_1} x(k) + B_1^{i_1}) + B_2 \tag{3-5}$$

The partition of the state space by hyperplanes corresponding to the neurons of Deep Neural Network has been thoroughly studied in a recent popular work [55]. These polyhedra were named *linear regions* and several important results on their maximum number, and thus their expressibility with respect to piecewise linear mappings, have been given in [54,55]. For shallow ReLU NNs, their exact number is even fixed. If they are in *general position* e.g. none of the corresponding hyperplanes are parallel to another, then the number of linear regions is [54]:

$$N_1 = \sum_{i=0}^{n} \binom{n_1}{i} = 1 + n_1 + \cdots + \binom{n_1}{n}$$

The resulting number of linear regions certainly gives an idea about the dynamics represented by the DNN, but the exact derivation of all activation patterns in terms of the markings vectors $z_j^{i_j}$ is more crucial for this study, since they describe how the state progression will proceed throughout the whole state space. This cell enumeration is not computationally trivial and two general approaches can be proposed for DNN.

The first approach has been reported on recent work and aim to reformulate the ReLU Deep Neural Network as a 0-1 Mixed-Integer Linear Program (MILP) [57,58]. Using state-of-the-art MILP solvers, the linear region enumeration can be done very quickly, but selecting some constants before the optimization is also necessary. These constants are depended on the considered range of the input space, require some prior knowledge and affect how quickly the optimization problem will be solved. In this study, the second approach will be mainly used.

The second approach for linear region enumeration is essentially a structured approach relying mainly on brute computational force, and thus probably less fast than the previous one for large problems. It resembles a tree search algorithm, where the nodes are visited in a *level-order* fashion. This search is usually called *Breadth-first* and starting from a node of the tree (called sometimes *tree root*) the neighboring nodes are visited before start searching on the next level. For the problem of DNN cell enumeration, the levels are the different hidden layers and the nodes in each level are all the possible combinations of neurons in that layer. This method does not require any "hyperparameter" tuning and knowledge around the range of the input space, while it is kept tractable for DNNs with relatively small number of neurons and layers.

3-2 Neural Networks with ReLUs are Piecewise Affine systems

The previous theory provide new aspects in the description of any deep (or shallow) NN, but does not bring a general stability result any closer. To that end, it is shown in this section that dynamical systems described by NNs with ReLUs are equivalent to Piecewise Affine (PWA) systems. For this, the Piecewise Affine (PWA) dynamical system representations are formally introduced at first and then their equivalence with ReLU NNs is proved.

3-2-1 Piecewise Affine systems

Firstly defined in [59], PWA systems are determined by a finite number of affine state update equations, where each one is associated with an activation region (typically a polyhedron) in the state space. More precisely, a discrete-time PWA system defined over a domain $\mathcal{D} \subseteq \mathbb{R}^n$,

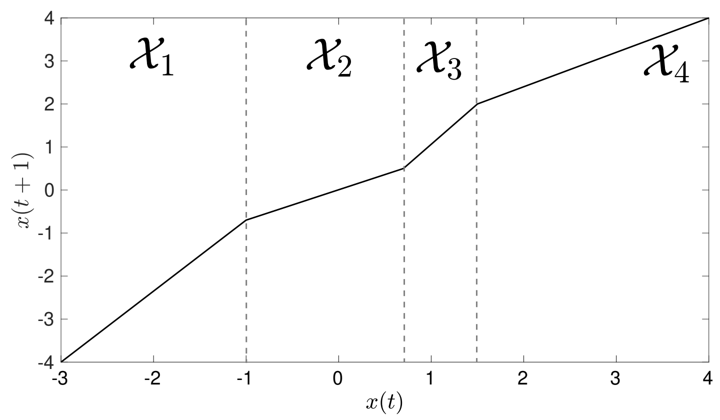


Figure 3-2: Example of a Piecewise Affine (PWA) system in $\mathbb{R}$ with 4 activation regions.

is given by one of the following two equivalent forms:

$$x(k+1) = A_i x(k) + b_i, \qquad x(k) \in \mathcal{X}_i \subset \mathcal{D} \tag{3-6}$$

$$\tilde{x}(t+1) = \begin{bmatrix} A_i & b_i \\ 0 & 1 \end{bmatrix} \tilde{x}(k) = \tilde{A}_i \tilde{x}(k), \qquad x(k) \in \mathcal{X}_i \subset \mathcal{D} \tag{3-7}$$

where $\tilde{x}(k) := \begin{bmatrix} x(k) & 1 \end{bmatrix}^{\mathrm{T}}$, $A_i \in \mathbb{R}^{n \times n}$, $b_i \in \mathbb{R}^n$, and $i \in \{1, 2, \dots, \mathcal{N}\}$ indexes the activation region $\mathcal{X}_i$.

While each individual system (A_i, b_i) is essentially linear, the overall PWA system (3-7) represents a switching between these subsystems, which essentially introduces the nonlinearity. PWA systems can represent complex nonlinear systems and have been subject of many studies in control by different authors [59–63]. A primary concern in design and analysis of PWA systems is stability. It should be emphasized that stability does *not* follow from stability of an individual system A_i, but the overall PWA dynamics and switching nature have to be considered in general [64].

In the vast majority of studies, the PWA systems are **continuous across the whole domain** $\mathcal{D}$, while the individual activation regions $\mathcal{X}_i$ are **convex polyhedra**. The latter property implies that each polyhedron can be described by a number of closed half-spaces in $\mathbb{R}^n$ (*H-representation*); that is, they can be given in the form of linear inequalities:

$$\mathcal{X}_i = \{x \in \mathbb{R}^n | E_i x + \epsilon_i \geq 0\} = \{x \in \mathbb{R}^n | \tilde{E}_i \tilde{x} \geq 0\} \tag{3-8}$$

with $\tilde{E}_i := \begin{bmatrix} E_i & \epsilon_i \end{bmatrix} \in \mathbb{R}^{r_i \times (n+1)}$. It is also common in PWA models to assume that the **intersection of the interiors of two polyhedra is the empty set**, i.e., $\text{int}(\mathcal{X}_i) \cap \text{int}(\mathcal{X}_j) = \emptyset$ for all $i, j \in \{1, \dots, \mathcal{N}\}$. A simple PWA system with 1-dimensional state space with separate local models is given in Figure 3-2.

Similarly to hybrid systems, the analysis of the possible transitions between regions $\mathcal{X}_i$ of a PWA system is necessary to fully characterize its dynamic behavior. The set Ω of all possible transitions between regions $\mathcal{X}_i$ and the corresponding transition matrix T are defined as:

$$\Omega := \{(i, j) \,|\, x(k) \in \mathcal{X}_i, x(k+1) \in \mathcal{X}_j\}. \tag{3-9}$$

$$(T)_{ij} = \begin{cases} 1 & \text{if } \exists\, x(k) \in \mathcal{X}_i : \; x(k+1) \in \mathcal{X}_j \\ 0 & \text{otherwise.} \end{cases}$$

The (one-step ahead) *reachability analysis* for PWA systems is relatively simple and it can be performed accurately by solving consecutive Linear Programs (LPs). Each Linear Program (LP) studies whether it is possible for the state to transit from one region $\mathcal{X}_i$ to an other region $\mathcal{X}_j$. Therefore, $\mathcal{N}^2$ LPs have to be solved in total for complete reachability analysis. These LPs may become more computationally intensive depending on the number of facets of the corresponding regions r_i, r_j. If there exist a solution to the following LP, derived using polyhedral description (3-8), then transition (i, j) belongs to Ω, while if not, the transition (i, j) is considered infeasible.

$$\begin{aligned} \min_{x} \quad & 0 \cdot x \\ \text{s.t.} \quad & E_i x + \epsilon_i \geq 0 \\ & E_j(A_i x + b_i) + \epsilon_j \geq 0 \end{aligned}$$

In order to increase computational efficiency, the previous LP is usually simplified and reduced in size using the bounding box outer approximation of region $\mathcal{X}_i$ [65]. Furthermore, several heuristic algorithms have been proposed in order to discard quickly impossible transitions. For more details, the interested reader can check the relative code, provided with Multi-Parametric Toolbox (MPT) [66] for MATLAB.

Now that the essential characteristics of PWA systems are presented, their connection and representational similarities with ReLU NNs can be more thoroughly studied.

3-2-2 Equivalence theorem

Although there have been previous studies showing the representational similarity of ReLU NNs with general piecewise linear functions, no connection with PWA systems has been made before and no compact equations to represent the polyhedra and input-output relations have been provided. In [67] it is argued that the input-output mapping represented by a ReLU DNN is a continuous piecewise linear function through the whole state-space and in [55] it is stated that each region composing the input space is an intersection of half-spaces, and thus a *convex polyhedron*. The following theorem, on the other hand, claims that the NN dynamics (3-2) can equivalently be represented as the PWA system (3-7), and presents exact relations to compute the polyhedral description of region $\mathcal{X}_i$ and the double (A_i, b_i). The theorem is a main insight of this thesis and will enable the stability analysis of the NN dynamics in the following sections.

Theorem 3.1. *Any dynamical system* (3-2) *represented by a NN with ReLU activations is* *equivalent to the discrete-time PWA system* (3-7) *with*

$$A_i = W_{L+1}\Big(\prod_{j=0}^{L-1} W_{L-j}^{i_{L-j}}\Big), \quad b_i = W_{L+1}\Big[\sum_{j=1}^{L}\Big(\prod_{l=0}^{L-j-1} W_{L-l}^{i_{L-l}}\Big)B_j^{i_j}\Big] + B_{L+1},$$

$$\tilde{E}_i = \begin{bmatrix} \operatorname{diag}(z_1^{i_1})W_1 & \operatorname{diag}(z_1^{i_1})B_1 \\ \operatorname{diag}(z_2^{i_2})W_2W_1^{i_1} & \operatorname{diag}(z_2^{i_2})(W_2B_1^{i_1}+B_2) \\ \vdots & \vdots \\ \operatorname{diag}(z_L^{i_L})W_L\Big(\prod_{j=1}^{L-1} W_{L-j}^{i_{L-j}}\Big) & \operatorname{diag}(z_L^{i_L})\Big(W_L\Big[\sum_{j=1}^{L-1}\Big(\prod_{l=1}^{L-j-1} W_{L-l}^{i_{L-l}}\Big)B_j^{i_j}\Big]+B_L\Big) \end{bmatrix}$$

and $W_j^{i_j} = [\frac{1}{2}\operatorname{diag}(\mathbf{1_{n_j}} + z_j^{i_j})W_j]$, $B_j^{i_j} = [\frac{1}{2}\operatorname{diag}(\mathbf{1_{n_j}} + z_j^{i_j})B_j]$, *where* $i \in \{1,\ldots,\mathcal{N}\}$ *indexes the affine models.*

Proof. The above result will be proved by *induction*. Firstly, equivalence of PWA systems and ReLU NN will be shown for the *base case* i.e. for a shallow NN ($L = 1$) and then the corresponding *inductive step* for a Deep Neural Network with $m > 1$ hidden layers will be presented.

For a shallow ReLU NN, only a single index i_1 is sufficient to characterize all the regions, while each region is defined by a unique vector of markings $z_1^{i_1} \in \{-1,1\}^{n_1}$, showing which neurons in that region are activated. Therefore, this linear region can be (non-uniquely[1]) described by the following matrix inequality:

$$\mathcal{X}_i := \{x \in \mathbb{R}^n |\operatorname{diag}(z_1^{i_1})(W_1x + B_1) \geq 0\}. \tag{3-10}$$

Thus, matrix $\tilde{E}_i$ will be:

$$\tilde{E}_i = \Big[\operatorname{diag}(z_1^{i_1})W_1 \quad \operatorname{diag}(z_1^{i_1})B_1\Big]$$

Using an equivalent expression for the σ vector-valued function based on the marking vector $z_1^{i_1}$, the state progression can now be given in the form:

$$\begin{aligned} x(k+1) =& W_2[\frac{1}{2}\operatorname{diag}(\mathbf{1_{n_1}} + z_1^{i_1})(W_1x(k) + B_1)] + B_2, \\ =& \frac{1}{2}W_2\operatorname{diag}(\mathbf{1_{n_1}} + z_1^{i_1})W_1x(k) + \frac{1}{2}W_2\operatorname{diag}(\mathbf{1_{n_1}} + z_1^{i_1})B_1 + B_2, \qquad x(k) \in \mathcal{X}_i \end{aligned} \tag{3-11}$$

Therefore, the state dynamics for shallow ReLU NNs can be expressed in the same form as (3-7) with:

$$A_i = \frac{1}{2}W_2\operatorname{diag}(\mathbf{1_{n_1}} + z_1^{i_1})W_1, \qquad b_i = \frac{1}{2}W_2\operatorname{diag}(\mathbf{1_{n_1}} + z_1^{i_1})B_1 + B_2.$$

It is proven that shallow ReLU NNs representing dynamics (3-2) are equivalent in every way to PWA systems.

[1]Non-uniquely in the sense that some linear inequalities of the following polyhedral description could be redundant.

For the *inductive step*, a DNN with m hidden layers will be considered. Assume a point in the state space and a combination of activation patterns for each layer denoted with the combination of indices $(i_1, \dots, i_m)$. It is implied from (3-10) that this point should satisfy the following inequalities corresponding the first hidden layer:

$$\operatorname{diag}(z_1^{i_1})(W_1 x(k) + B_1) \geq 0 \tag{3-12}$$

The output of the first hidden layer will then be according to (3-3):

$$a_1 = W_1^{i_1} x(k) + B_1^{i_1} \tag{3-13}$$

Accordingly, for the same point in the state space, a pattern of neurons indexed as i_2 will be activated in the second layer and a new vector $z_2^{i_2} \in \{1, -1\}^{n_2}$ will be assigned, For this point, except for the matrix inequality (3-12), another matrix inequality will be satisfied:

$$\operatorname{diag}(z_2^{i_2})(W_2 a_1 + B_2) \geq 0$$

Replacing a_1 from (3-13), it follows that:

$$\operatorname{diag}(z_2^{i_2})[W_2(W_1^{i_1} x(k) + B_1^{i_1}) + B_2] \geq 0$$

The output of the second hidden layer is given as:

$$a_2 = W_2^{i_2} a_1 + B_2^{i_2} = W_2^{i_2} W_1^{i_1} x(k) + W_2^{i_2} B_1^{i_1} + B_2^{i_2}.$$

Repeating the previous procedure till the m$^{\text{th}}$ layer ($m < L$) and defining the corresponding vector $z_m^{i_m}$ for the i_m activation pattern, the following inequalities are true:

$$\operatorname{diag}(z_m^{i_m})(W_m a_{m-1} + B_m) \geq 0$$

Or equivalently:

$$\operatorname{diag}(z_m^{i_m})\{W_m[\dots(W_1^{i_1} x(k) + B_1^{i_1}) + \dots] + B_m\} \geq 0 \tag{3-14}$$

After simple manipulations, inequality (3-14) can be expressed with respect to the state vector $x(k)$ the following way:

$$\operatorname{diag}(z_m^{i_m})\left\{W_m\left(\prod_{j=1}^{m-1} W_{m-j}^{i_{m-j}}\right)x(k) + W_m\left[\sum_{j=1}^{m-1}\left(\prod_{l=1}^{m-j-1} W_{m-l}^{i_{m-l}}\right)B_j^{i_j}\right] + B_m\right\} \geq 0 \tag{3-15}$$

Furthermore, the output of the m$^{\text{th}}$ hidden layer with respect to $x(k)$ can be computed by repeating the previous procedure:

$$x(k+1) = a_m = \left(\prod_{j=0}^{m} W_{m-j}^{i_{m-j}}\right)x(k) + \left[\sum_{j=1}^{m}\left(\prod_{l=0}^{m-j-1} W_{m-l}^{i_{m-l}}\right)B_j^{i_j}\right]$$

Therefore, for a general DNN with L hidden layers, it is proven by induction that a point $x(k)$ in $\mathbb{R}^n$ will fulfill a set of L matrix inequalities (given that the combination $(i_1, i_2, \dots, i_L)$

of neurons is activated):

$$\begin{aligned} \operatorname{diag}(z_1^{i_1})(W_1 x(k) + B_1) \geq 0 \\ \operatorname{diag}(z_2^{i_2})[W_2(W_1^{i_1} x(k) + B_1^{i_1}) + B_2] \geq 0 \\ \vdots \\ \operatorname{diag}(z_L^{i_L})\left\{W_L\left(\prod_{j=1}^{L-1} W_{L-j}^{i_{L-j}}\right)x(k) + W_L\left[\sum_{j=1}^{L-1}\left(\prod_{l=1}^{L-j-1} W_{L-l}^{i_{L-l}}\right)B_j^{i_j}\right] + B_L\right\} \geq 0. \end{aligned} \tag{3-16}$$

From (3-16), it can be noted that each linear region $\mathcal{X}_i$, can be (non-uniquely) defined by the previous $M = \sum_{j=1}^{L} n_j$ linear inequalities and it is a convex polyhedron. Furthermore, matrix $\tilde{E}_i$ describing the polyhedron can be determined from simple manipulations of (3-16).

For the $(i_1, i_2, \ldots, i_L)$ combination, corresponding to the current state vector $x(k)$, the state vector at the next time step can be computed also by induction with the following relation:

$$x(k+1) = W_{L+1}\left(\prod_{j=0}^{L-1} W_{L-j}^{i_{L-j}}\right)x(k) + W_{L+1}\left[\sum_{j=1}^{L}\left(\prod_{l=0}^{L-j-1} W_{L-l}^{i_{L-l}}\right)B_j^{i_j}\right] + B_{L+1}, \qquad x \in \mathcal{X}_i. \tag{3-17}$$

Besides its complexity this is an affine mapping with respect to the current state value for a fixed combination of indices $(i_1, i_2, \ldots, i_L)$ i.e. for a given region $\mathcal{X}_i$. The above state update matrices and vectors linked to a specific region $\mathcal{X}_i$ are equivalent to the double (A_i, b_i) of PWA systems and this concludes the proof. □

3-3 Stability Analysis for ReLU NN dynamics

The equivalence between ReLU DNNs and PWA systems, which was established in the previous section under mild assumptions, allows to leverage powerful analysis tools that have been developed for PWA systems and adjust them to perform stability analysis for the NN dynamics (3-2). These tools are typically based on *Linear Matrix Inequality (LMI)* formulations, which lead to convex optimization problems [45] with some readily available solvers. In this section, exponential stability is examined and a complete computational framework to analyze stability properties of the NN dynamics (3-2) is established.

3-3-1 Stability theorem

As described in Chapter 2, a common way to assess stability for nonlinear models is the pursuit of a Lyapunov function V that will fulfill a number of conditions. For a general model like (2-1), this is hard and no general method to systematically construct such a Lyapunov function for PWA systems exists. To that end, it is necessary to impose a specific parametrization on V, such that an optimal tradeoff between complexity and expressibility is achieved.

Following [65], where a number of such different parametrizations are compared, the Piecewise Quadratic Lyapunov Function (PQLF) was proven a good choice for PWA systems with respect to these two criteria. A PQLF postulates a quadratic function $V_i(x) = \tilde{x}^{\mathrm{T}}\tilde{P}_i\tilde{x}$ with $\tilde{P}_i$ a symmetric matrix corresponding to a single region $\mathcal{X}_i$. Combined with the PWA system dynamics (3-6), LMI conditions in the free variables $\tilde{P}_i$ can then be derived. If feasible $\tilde{P}_i$ that satisfy the LMI conditions are found, this proves exponential stability of the PWA system. Searching for feasible $\tilde{P}_i$ typically is a convex optimization problem, for which computational tools are available [68, 69]. Now these tools are proposed for stability analysis of the ReLU NN dynamics (3-2).

Assume that the ReLU NN (3-2) has a single known equilibrium x_{e}, which without loss of generality coincides with the origin, since an appropriate coordinate change $x - x_e$ can always be imposed. By Theorem 3.1 the equivalent representation of (3-2) as PWA system given by the dynamics (3-6), (3-7) is assured and $\mathcal{N}$ polyhedral regions (3-8) are considered. It is further assumed $x_{\mathrm{e}} \in \mathcal{X}_1$[2]. The set of all possible transitions is Ω as in (3-9), which can be computed using reachability analysis as described in Section 3-2-1. With this, the following theorem can be stated:

Theorem 3.2. *If there exist symmetric matrices $P_1 \in \mathbb{R}^{n\times n}, Y_1 \in \mathbb{R}^{r_1\times r_1}; \tilde{P}_i \in \mathbb{R}^{(n+1)\times(n+1)}, Y_i \in \mathbb{R}^{r_i\times r_i}$ for all $i \in \{2,\ldots,\mathcal{N}\}$; and $U_{ij} \in \mathbb{R}^{r_i\times r_i}$ for all $(i,j) \in \Omega$ such that Y_i, U_{ij} have non-negative entries and the following LMIs are fulfilled:*

$$\begin{cases} P_1 - E_1^T Y_1 E_1 > 0, \\ \tilde{P}_i - \tilde{E}_i^T Y_i \tilde{E}_i > 0, \quad i \in \{2,\ldots,\mathcal{N}\}, \end{cases} \tag{3-18}$$

$$\begin{cases} A_1^T P_1 A_1 - P_1 + E_1^T U_{11} E_1 < 0, \\ \tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i + \tilde{E}_i^T U_{ij} \tilde{E}_i < 0, \quad (i,j) \in \Omega \setminus \{(1,1)\}, \end{cases} \tag{3-19}$$

then the equilibrium x_{e} of (3-2) *is* globally exponentially stable.

Proof. In this proof it will be shown that if LMIs (3-18) and (3-19) are true, matrices $P_1, \tilde{P}_i, Y_1, Y_i, U_{11}, U_{ij}$ are symmetric and matrices Y_1, Y_i, U_{11}, U_{ij} have positive entries, then all the conditions of Theorem (2.2) are necessarily fulfilled for system (3-7).

The Lyapunov function is parametrized as a Piecewise Quadratic function with the origin being included in the $\mathcal{X}_1$ region and thus given by the following relation

$$V(x) = \begin{cases} x^T P_1 x, & x \in \mathcal{X}_1, \\ \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_i \begin{bmatrix} x \\ 1 \end{bmatrix} = \tilde{x}^T \tilde{P}_i \tilde{x}, & x \in \mathcal{X}_i, \; i \neq 1 \end{cases}$$

Two well-known Lemmas of linear algebra will be used to show condition (2-3b):

- $\lambda_{\min}(M)||x||_2^2 \leq x^T M x \leq \lambda_{\max}(M)||x||_2^2$ (3-20a)
- $M > 0 \iff \lambda_i(M) > 0, \forall i$ (3-20b)

[2]This is without loss of generality as long as the origin is not on the boundary of more than one region, which is a singular case that almost never occurs in practice.

Furthermore, the following proposition resulting from properties of norms will be useful:

$$||\tilde{x}||_2^2 = ||x||_2^2 + 1 \tag{3-21}$$

To ease analysis, the following simple manipulation is made:

$$M_1 = P_1 - E_1^T Y_1 E_1 > 0, \qquad \tilde{M}_l = \tilde{P}_l - \tilde{E}_l^T Y_l \tilde{E}_l > 0, \qquad \forall l \neq 1$$

Then, from (3-20a) and (3-21):

$$\begin{cases} x^T P_1 x \geq x^T M_1 x \geq \lambda_{\min}(M_1)||x||_2^2, & \forall x \in \mathcal{X}_1 \\ \tilde{x}^T \tilde{P}_l \tilde{x} \geq \tilde{x}^T \tilde{M}_l \tilde{x} \geq \lambda_{\min}(\tilde{M}_l)||\tilde{x}||_2^2 \geq \lambda_{\min}(\tilde{M}_l)||x||_2^2, & \forall x \in \mathcal{X}_l, l \neq 1 \end{cases}$$

Since every eigenvalue of matrices $M_1, \tilde{M}_l$ is positive, there exists a constant $\alpha_1 = \min_l(\lambda_{\min}(M_1), \lambda_{\min}(\tilde{M}_l))$ such that:

$$V(x) = \begin{cases} x^T P_1 x \geq \alpha_1 ||x||_2^2, & x \in \mathcal{X}_1 \\ \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_i \begin{bmatrix} x \\ 1 \end{bmatrix} \geq \alpha_1 ||x||_2^2, & x \in \mathcal{X}_l, \ l \neq 1 \end{cases} \tag{3-22}$$

For $l \neq 1$:

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)||\tilde{x}||_2^2, \qquad \forall x \in \mathcal{X}_l. \tag{3-23}$$

while:

$$V(x) \leq \lambda_{\max}(P_1)||x||_2^2, \qquad \forall x \in \mathcal{X}_1. \tag{3-24}$$

From (3-21), (3-23) becomes:

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)(||x||_2^2 + 1).$$

Therefore,

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)\left(||x||_2^2 + \frac{||x||_2^2}{c_l}\right)$$

where $c_l = \min_{x \in \mathcal{X}_l} ||x||_2^2$.

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)\left(\frac{c_l + 1}{c_l}\right)||x||_2^2. \tag{3-25}$$

From inequalities (3-24) and (3-25), it is then evident that there exists α_2 such that:

$$V(x) \leq \alpha_2 ||x||_2^2, \qquad \alpha_2 > 0.$$

Therefore, property (2-3c) is fulfilled for $\eta = 2$.

The difference of the candidate Lyapunov function between two time steps is now considered. If the state at time step k is $x(k) \in \mathcal{X}_l$ and at time step $k+1$ is $x(k+1) \in \mathcal{X}_j$, two possible cases need to be examined:

1. $(l, j) \in \Omega \backslash (1,1)$
2. $(l, j) = (1,1)$

Since every entry of $\tilde{E}_i \cdot \tilde{x}, E_1 \cdot x, U_{ij}$ and U_{11} is non-negative, then it is obvious that:

$$(\tilde{E}_i\tilde{x})^T U_{ij}(\tilde{E}_i\tilde{x}) \geq 0, \qquad (E_1 x)^T U_{11}(E_1 x) \geq 0$$

This proposition will be useful as the above two cases are examined.

For the first case, it follows that if $\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i + \tilde{E}_i^T U_{ij} \tilde{E}_i < 0$ (from (3-18)), then there exist $\rho > 0$ such that:

$$\begin{aligned}
\Delta V &= V(x(k+1)) - V(x(k)) = \tilde{x}^T(k)\tilde{A}_i^T \tilde{P}_j \tilde{A}_i \tilde{x}(k) - \tilde{x}^T(k)\tilde{P}_i\tilde{x}(k)\\
&= \tilde{x}^T(k)[\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i]\tilde{x}(k)\\
&\leq \tilde{x}^T(k)(-\rho I - \tilde{E}_i^T U_{ij}\tilde{E}_i)\tilde{x}(t)\\
&\leq \tilde{x}^T(k)(-\rho I)\tilde{x}(k)\\
&\leq -\rho||x(k)||_2^2
\end{aligned}$$

where $\tilde{P}_1 = \begin{bmatrix} P_1 & \mathbf{0_n} \\ \mathbf{0_n}^\mathbf{T} & 0 \end{bmatrix}$.

For the second case, it follows that if $A_1^T P_1 A_1 - P_1 + E_1^T U_{11} E_1 < 0$ (from (3-18)), then there exist $\rho > 0$ such that:

$$\begin{aligned}
\Delta V &= V(x(k+1)) - V(x(k)) = x^T(k)A_1^T P_j A_1 x(k) - x^T(k)\tilde{P}_i]x(k)\\
&= x^T(k)[A_1^T P_j A_1 - \tilde{P}_i]x(k)\\
&\leq x^T(k)(-\rho I - E_1^T U_{11} E_1)x(k)\\
&\leq x^T(k)(-\rho I)x(k)\\
&\leq -\rho||x(k)||_2^2
\end{aligned}$$

The above proves that there exists $\alpha_3 > 0$ such that:

$$\Delta V(x) \leq \alpha_3 ||x||_2^2$$

Therefore, property (2-3c) is fulfilled for $\eta = 2$. □

Remarks on Theorem 3.2

Theorem 3.2 provides LMI conditions given in terms of the NN parameters (cf. Theorem 2.2), which can be checked with computational tools. The problem of searching for feasible solution (i.e., P_i, $\tilde{P}_i$, Y_i, U_{ij}) is convex and if a solution is found, stability for the NN dynamics (3-2) is guaranteed.

The terms $E_1^T Y_1 E_1$, $\tilde{E}_i^T Y_i \tilde{E}_i$, $E_1^T U_{11} E_1$, and $\tilde{E}_i^T U_{ij} \tilde{E}_i$ appearing in LMIs (3-18) and (3-19) follow from the implementation of the S-procedure (see Chapter 2) for the corresponding polyhedra $\mathcal{X}_i$ given by (3-10) and aim to relax the stability conditions. The use of similar terms as a result of the S-procedure is something well-known in literature of PWA systems (see e.g. [62, 63, 70]).

Nevertheless, the conditions can be conservative in some cases; that is, the criterion may fail to determine stability despite the actual dynamics being stable. This is a typical characteristic of these stability tools and, in a sense, the price one has to pay to obtain tractable stability conditions. However, there are ways to reduce conservativeness at the expense of more involved formulations. One simple example for this model representations is to consider only the subset of region $\mathcal{X}_i$ from which the transition to region $\mathcal{X}_j$ is possible instead of the whole $\mathcal{X}_i$ and consequently replace matrix $\tilde{E}_i$ with matrix $\tilde{E}_{ij} \in \mathbb{R}^{(r_{ij})\times(n+1)}$ with $r_i \leq r_{ij} \leq r_i + r_j$ defined such that:

$$\mathcal{X}_{ij} := \{x(k)|x(k) \in \mathcal{X}_i, x(k+1) \in \mathcal{X}_j\} = \{x(k)|\tilde{E}_{ij}\tilde{x}(k) \geq 0\}.$$

This change will make the solution of the LMIs (3-18) and (3-19) more likely, but increase the size of optimization matrices U_{ij} and thus the computational cost. For complex partitions and high dimensional state-spaces, where due to the curse of dimensionality the polyhedra need exponentially more hyperplanes to be fully described, this technique may render the above optimization problem intractable. In later sections, this technique to reduce conservativeness is evaluated for both low-dimensional and higher-dimensional examples.

3-3-2 Complete framework for stability analysis

Theorem 3.2 is a global stability result; that is, it states conditions for a single equilibrium being globally attractive (cf. Definition 2.3). Most complex nonlinear systems (2-1) (and thus also their approximation as NN (3-2)) will not have this property. Furthermore, it was often noted during this study that when the real dynamics included a *single equilibrium*, even very accurate NN approximations had multiple of them throughout the state space, usually in regions that were not sampled at all. This last phenomenon should be expected, since there is no incentive during training to approximate well "unexplored" regions or this kind of dynamic features. Consequently, the existence of multiple equilibria is the common case, and there is an extreme need to resort to a local stability notion similar to Definition 2.3.

In this section, we leverage the result of Theorem 3.2 to propose a complete framework for analyzing stability properties of (3-2). This will include the computation of all equilibria, determination of their local stability properties, and computation of the region of attraction.

Computation of equilibria First, the exact coordinates of all equilibria of the ReLU NN dynamics (3-2) represented by (3-6), (3-7), and (3-8) should be computed. The mathematical description of the equilibrium of dynamics (3-2) will be $x_\text{e} = f_\text{NN}(x_\text{e})$. Since the state update equation is piecewise affine, all candidate equilibrium points are obtained by solving (3-6) with $x(k) = x(k+1) = x_\text{e}$ for every local model i, which yields $x_e^i = (I - A_i)^{-1}b_i$ (assuming the inverse exists). For the case of linear local model (i.e. $b_i = 0$), it is obvious that the origin is its corresponding candidate equilibrium. Then, by examining whether $x_e^i \in \mathcal{X}_i$ for each one of them, all equilibria of (3-2) can be identified.

Local stability Furthermore, each local update equation can be seen as an affine linearization of the dynamics and thus, use well-known theory for linearized dynamics (*see* eg. [71, Cha. 2]). In particular, stability for each equilibrium[3] $x_e^i \in \text{int}(\mathcal{X}_i)$ is characterized by the eigenvalues λ_j, $j = 1, \ldots, n$, of A_i:

(i) If $|\lambda_j| < 1$ for all $j \in \{1, 2, \ldots, n\}$, then x_e^i is an (exponentially) *stable equilibrium.*

(ii) If there exists $j \in \{1, 2, \ldots, n\}$ such that $|\lambda_j| > 1$, then x_e^i is an *unstable equilibrium.*

It is very common in literature to make the distinction between *unstable* and *saddle* equilibria. In the former case, all the eigenvalues of the corresponding matrix A_i have absolute value greater than 1, while in the latter there is at least one such eigenvalue, but not all of them. For this study, the distinction of these two cases does not affect at all the stability analysis and the corresponding framework, and consequently it is preferable to avoid getting in such details.

Region of Attraction (RoA) While the above tests (i) and (ii) characterize local stability of an equilibrium $x_e^i \in \text{int}(\mathcal{X}_i)$, they do not make any statement about the RoA $\mathcal{R}_i$ of that equilibrium. In fact, this RoA can be arbitrarily small (for small $\mathcal{X}_i$), or large and comprise several regions (the typical case). Therefore, it is of key interest to determine the RoA. The exact computation of the true RoA $\mathcal{R}_i$ is generally a very difficult task and for most cases computing a sufficiently large estimate $\mathcal{D}_i$ satisfactory. To ease the presentation, a single (locally) stable equilibrium x_e is considered in the following and its approximate RoA $\mathcal{D}$ is computed (the i index is dropped). The process can be repeated for all stable equilibria.

In the case of PWA systems, simply adjusting Theorem 3.2 by examining only a smaller number of regions around the equilibrium instead of the whole state space will not be sufficient to deduce the set $\mathcal{D}$. In addition to the LMIs (3-18) and (3-19), $\mathcal{D}$ also has to be a Positively Invariant (PI) set under dynamics (3-7) (see Chapter 2). By applying a coordinate change such that $x_e = 0$ is the new equilibrium, the following corollary then follows from Theorem 3.2. Again without loss of generality, $x_e \in \mathcal{X}_1$ and the number of regions that compose $\mathcal{D}$ is denoted $\mathcal{N}' \leq \mathcal{N}$.

Corollary 3.1. *Let $\mathcal{D} \subset \mathbb{R}^n$ be a PI under dynamics* (3-7)*, $\mathcal{X}_i$ with $i \in \{1, 2, \ldots, \mathcal{N}'\}$ be the polyhedral regions that compose $\mathcal{D}$, and Ω' denote the corresponding transitions. If there exist symmetric matrices $P_1 \in \mathbb{R}^{n \times n}, Y_1 \in \mathbb{R}^{r_1 \times r_1}, \tilde{P}_i \in \mathbb{R}^{(n+1)\times(n+1)}, Y_i \in \mathbb{R}^{r_i \times r_i}$, $i = 2, \ldots, \mathcal{N}'$, and $U_{ij} \in \mathbb{R}^{r_i \times r_i}, \forall (i,j) \in \Omega'$ such that Y_i, U_{ij} have nonnegative entries and the following LMIs are fulfilled:*

$$\begin{cases} P_1 - E_1^T Y_1 E_1 > 0, \\ \tilde{P}_i - \tilde{E}_i^T Y_i \tilde{E}_i > 0, \qquad i \in \{2, \ldots, \mathcal{N}'\} \end{cases} \tag{3-26}$$

$$\begin{cases} A_1^T P_1 A_1 - P_1 + E_1^T U_{11} E_1 < 0, \\ \tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i + \tilde{E}_i^T U_{ij} \tilde{E}_i < 0, \qquad (i,j) \in \Omega' \setminus \{(1,1)\}, \end{cases} \tag{3-27}$$

then x_e is (locally) exponentially stable and $\mathcal{D}$ an estimate of its RoA.

The corollary can be proved the same way as Theorem 3.2, except that now only the polyhedral partitions inside $\mathcal{D}$ must be examined instead of the whole state space $\mathbb{R}^n$.

[3]The equilibrium x_e on the boundary is again a singular case, typically not relevant in practice.

Computation of a PI set Corollary 3.1 can now be exploited to compute a RoA around each stable equilibrium. Obviously applying this Corollary to unstable equilibria its futile, since its already known that the conditions will never be fulfilled. However, the Corollary considers the PI set $\mathcal{D}$ already known and therefore, a procedure to compute it is necessary. For PWA and by implication ReLU NNs, the derivation of a PI can be achieved by Algorithm 4.1 in [72], where starting from an initial set Q_0, composed of possibly non-convex polyhedra, its Maximal Positively Invariant (MPI) subset Q_∞ is derived. The choice of the initial set Q_0 affects whether it will be possible to find an approximate RoA for the equilibrium, if selected small and the computational cost, if selected large. A good choice would be a hyperrectangle located around the examined equilibrium, such that includes most of the samples used for training of the network.

Although the algorithm is straightforward to implement, it may become intractable as the state dimensionality increases and deriving PI sets for larger state-spaces efficiently is an open question. Nevertheless, in [73] two sufficient conditions for the finite determination of a PI set included in the initial set are proposed:

- $I - A_i$ invertible, $\quad \forall i \in \{1, 2, \dots \mathcal{N}'\}$ (3-28a)
- $x_e^i \notin \mathcal{X}_i, \quad \forall i \in \{1, 2, \dots \mathcal{N}'\}$ (3-28b)

In order to ensure that the second condition is fulfilled, no unstable equilibria should be included in the final set Q_∞.

If the computation of the MPI becomes intractable, while the above two conditions for finite determination are guaranteed, the PI set required for Corollary 3.1 will not be derived in the end. Instead, if the algorithm is stopped after r steps, a sufficiently close overapproximation Q_r will be computed, due to the nature of the algorithm for the computation of MPI. However, the set Ω', computed by the reachability analysis that will follow, should only include the transitions to the regions comprising the overapproximation. Then, if LMIs (3-26), (3-27) are solved for the corresponding transitions of set Ω', it is guaranteed that a subset of Q_r will be exponentially stable, even though computing the subset exactly is not be possible.

Complete algorithm All steps of the proposed stability analysis are summarized in Algorithm 1. First, it is necessary to identify all the linear regions in $\mathbb{R}^n$ (or a region of interest) and two different approaches have been proposed back in Subsection 3-1-2 along with few advantages and disadvantages to be considered. Given the markings of the feasible combinations of neurons, the corresponding polyhedral descriptions and update equations can be derived with Theorem 3.1.

The second step would be to locate all equilibria of the NN and characterize them as stable or unstable. For the stable ones, PI subsets are searched. If the computation of such a PI subset is tractable and gives a solution, LMIs (3-26), (3-27) are solved e.g., using the YALMIP toolbox of MATLAB [69] and the SeDuMi Solver [68]. For more information on these two toolboxes, see Appendix B. If the optimization problem is solved, then the PI subset is an estimate of the RoA. If the computations are intractable, an overapproximation can be derived and the LMIs are solved for this set. To assess whether the computations become intractable, it should be noted that as the number of steps of the algorithm increases, the time between consecutive steps also increases exponentially.

```
**Data:** ReLU Neural Network
**Result:** All equilibria, regional stability, estimates of RoA
Find all the linear regions, their $\tilde{E}_i$ and $\tilde{A}_i$ matrices;
Compute all local equilibria and their characterization;
for stable equilibria do
    Define initial subset $Q_0$ around equilibrium;
    if MPI computation finishes quickly then
        Compute transition matrix $T$;
        Solve LMIs (3-26), (3-27) for $Q_\infty$;
        Result: estimate of the RoA
    else
        Derive MPI overapproximation $Q_r$;
        Compute transtion matrix $T$;
        Solve LMIs (3-26), (3-27) for $Q_r$;
        Result: overapproximation of the RoA
    end
end
```

Algorithm 1: Summary of the proposed stability analysis for ReLU NN dynamics.

3-4 Illustrative numerical example

In order to illustrate the proposed framework for stability analysis of NN dynamics, a simple yet illustrative example will be studied in this section. Consider the dynamics of roll motion of ships in rough seas as studied in [74] and described by the second-order differential equation:

$$\ddot{\theta} + (2\mu_1\dot{\theta} + \mu_2\dot{\theta}^3) + (\omega_0^2\theta + a_1\dot{\theta}^3 + a_2\dot{\theta}^5) = 0,$$

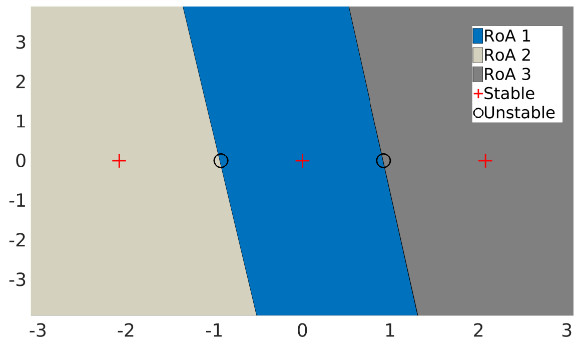
where $\omega_0, a_i, \mu_i,\ i = 1, 2$ are known coefficients. A discrete-time model (2-1) with state dimensionality $n = 2$ can be derived using standard discretization techniques [8]. The choice of these dynamics as an illustrative example was made due to the existence of multiple equilibria in the state space and the ease to compare visually the regions of attraction for both the real dynamics and their neural network approximation, derived by the proposed framework. More specifically, the dynamics above have three stable equilibria $(\theta, \dot{\theta}) = \{(0, 0), (0, 2.078), (0, -2.078)\}$ and two unstable ones $(\theta, \dot{\theta}) = \{(0, 0.924), (0, -0.924)\}$. The equilibria and the true RoA (computed via numerical simulation) are shown in Figure 3-3a.

For this synthetic example, a NN was trained from data randomly sampled from a rectangle in the two-dimensional state space. These obviously are the ideal sampling conditions, which are not realistic for actual dynamical systems, but feasible in simulation. Because of the relatively simple dynamics, a shallow NN with 8 ReLUs in the single hidden layer was sufficient. A ReLU NN (3-2) approximating the real dynamics sufficiently well (the Mean Squared Error (MSE) was approximately $3 \cdot 10^{-4}$) was relatively easily derived using MATLAB Neural Network toolbox with random initialization of the network weights.

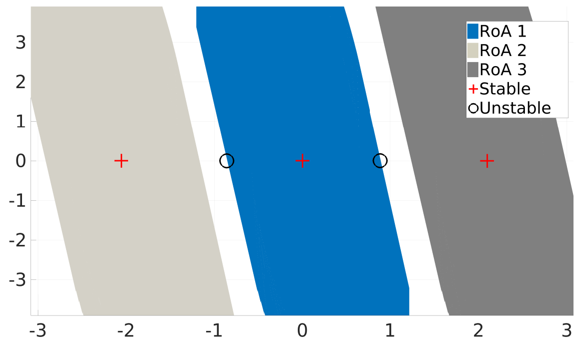
The number of linear regions $\mathcal{N}$ in the whole state space $\mathbb{R}^2$ was found to be 37 as predicted by the corresponding relation in Subsection 3-1-2 and the markings necessary to derive (A_i, b_i) doubles of (3-7) and the polyhedral description (3-8) were computed with brute computational force i.e. consecutive solution of LPs for different combination of activated neurons.

Table 3-1: Stability results computed for ReLU NN dynamics of the ship roll example using the proposed framework.

Neural network	Equilibrium	Local stability	Volume of RoA
Ship-NN	1: $(0, 0)$	stable	13.888
	2: $(0, 2.091)$	stable	13.525
	3: $(0, 0.866)$	unstable	0
	4: $(0, -2.061)$	stable	13.6848
	5: $(0, -0.88)$	unstable	0



(a) True dynamics.



(b) ReLU NN dynamics.

Figure 3-3: Equilibria and regions of attraction (RoA) for the numerical ship roll example.

This computation obviously remains computationally tractable for NNs with small number of neurons and do not suffer from the problems of the Mixed-Integer Programming formulation as presented in Subsection 3-1-2.

The framework from Sec. 3-3-2 was applied to compute equilibria, their stability characterization and their RoA. The results are presented in Table 3-1 (including the volume of the RoA) and Figure 3-3b. For this example, matrices $\tilde{E}_i$ of (3-27) were replaced with matrices $\tilde{E}_{ij}$ as instructed by the technique described at the end of Subsection 3-3-1 in order to account for smaller polyhedral regions. This choice has reduced conservativeness with a small computational cost due to the simplicity of the regions. As can be seen in Table 3-1, all five equilibria are identified and correctly characterized as stable/unstable. Moreover, the computed RoA are reasonable approximations of the true ones (cf. Figure 3-3a and 3-3b). For the unstable ones, the volume of RoA was assumed to be 0.

While there is a good match of the stability properties of the true dynamics and the NN approximation in this example, this obviously depends on how well one succeeds in training the NN. It should be emphasized that the contribution of this work is *not* in how to learn good NN dynamics models, but rather in analyzing a given or trained NN. In fact, the results of the stability analysis can help to evaluate different NN models, for example, when some stability properties are known.

3-5 Case Study of a cart-pole experiment

After demonstrating the efficacy of the proposed framework on a 2-dimensional synthetic example in the previous section, the framework was extended to higher-dimensional dynamics,

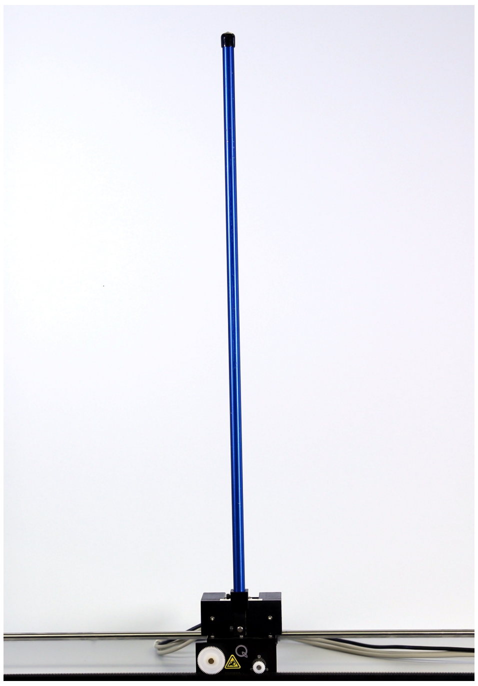


Figure 3-4: Cart-pole testbed used in experiments.

where the NNs were trained on **real-world** data from a hardware experiment. The standard cart-pole experiment was chosen [75] and the experimental setup is presented in Figure 3-4. The cart is able to travel along a line via a rack and pinion and has four states: position of the cart p, angle of the pole θ and the corresponding linear $\dot{p}$ and angular $\dot{\theta}$ velocities. It is controlled only by the current applied to the motor, attached to the cart. For the rest of this case study, the dynamics of the cart-pole stabilized about its upright equilibrium with a standard Linear Quadratic Regulator (LQR) [76] will be considered.

A data set was obtained by exciting the system with a suitable chirp signal (sinusoid with increasing frequency), in superposition to the control input of the LQR, in order to keep the angle of the pendulum close to the one in the upright position. Angle and position of the cart are measured with the available encoders, from which the velocities are computed through finite differences. Although the setup is able to provide a large sampling frequency up to 1000Hz, the trajectories comprising the data set were downsampled to 100 Hz. All signals were low-pass filtered and additionally a (non-causal) zero-phase digital filter was applied after sampling the trajectories.

With this data, multiple ReLU NNs (3-2) (with $L = 2$ hidden layers, and $n_1 = 8$, $n_2 = 6$ neurons) were trained. The number of neurons was set after consecutive trials and after noticing that additional neurons will not improve the performance with respect to the performance criterion i.e. Mean Squared Error. For each configuration, more than one training trials were performed to assure that the optimization will not be stuck in a suboptimal local equilibrium far away from the global one. For training the NNs, a dynamics model in the form of $x(k+1) = x(k) + f(x(k))$ was assumed instead of (2-1); that is, the NN represents incremental, rather than absolute dynamics. This is beneficial especially for small sample times (and thus small increments), and common practice when training NNs to approximate

Table 3-2: Stability results for the five top scoring NNs.

Neural network	Score	Equilibrium $(p, \theta, \dot{p}, \dot{\theta})$	Local stability	Volume of RoA
NN1	0.0223	$(0.13, 0.01, -8 \cdot 10^{-3}, 0.08)$	stable	0.032
NN2	0.0229	$(-0.55, 0.13, 0.98, -0.29)$	unstable	0
NN3	0.0232	$(-1.11, -0.02, 0.04, -0.29)$	stable	–
NN4	0.0265	$(-0.68, -0.09, 0.35, 0.14)$	stable	–
NN5	0.0279	$(1.37, 0.09, -0.21, 0.69)$	stable	0.021

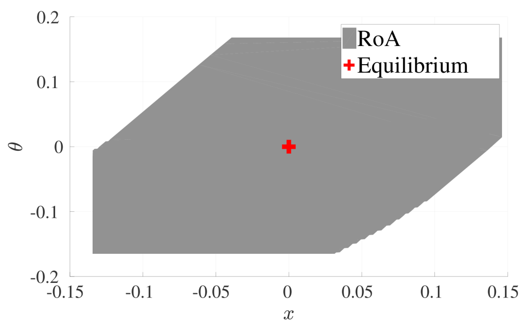
state dynamics. The stability analysis directly extends to this case, simply by adding the identity matrix to the A_i matrices computed by the cell enumeration algorithm.

A big challenge observed in these experiments was how to rank the different NNs that were obtained from different random initialization and training runs. In particular, the MSE on a validation set, which is usually the criterion used to evaluate NNs, was not a good indication of the performance of a NN in approximating well the true dynamics. For example, it was rather common that trained networks achieving a good MSE on the validation set were unstable for the same initial conditions that belonged in the RoA of the real dynamics. In order to capture long-term predictions well, the MSE after unrolling the dynamics for 500 steps was considered as a better criterion to rank the NNs ("Score"). For the top five NNs (with respect to the latter criterion), the stability analysis was performed as per Sec. 3-3-2 and its results are given in Table 3-2.

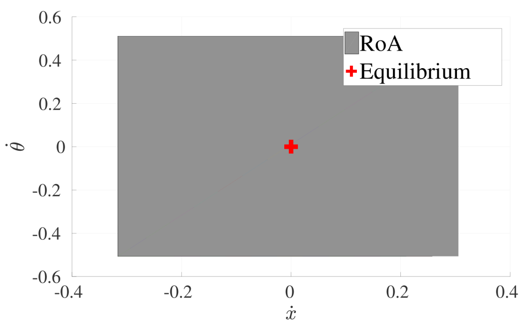
As expected, in each case one equilibrium was found. While most NNs have (locally) stable equilibria, in case of NN2, the NN dynamics were found to be unstable. As remarked in Sec. 3-4, a useful application of the proposed stability analysis is to select suitable NN models. In this case, where from physical considerations it is already known that there is one stable equilibrium, one can discard NN2.

Although the number of linear regions $\mathcal{X}_i$ varies significantly from training trial to training trial, it was not uncommon to get more than 1000 regions through the whole state-space. For NN3 and NN4, no volume of the RoA is computed, since the computations required to determine the Maximal Positively Invariant became intractable. Thus, an overapproximation was derived by stopping the MPI algorithm when the number of regions grew too large as explained in Subsection 3-3-2. The resulting polyhedra were composed of many inequalities due partly to the high state dimensionality, which made the solution of the LMI costly. Again an overapproximation of the complex polyhedra with bounding boxes solved the problem and stability could be determined. For NN1 and NN5, the computations could be performed without problem. For NN1, the computed RoA is illustrated in Figure 3-5.

Unfortunately, it is difficult to compute the volume of RoA for the 4-dimensional learned dynamics as it was done in Section 3-4 (through multiple tests with random initial conditions). However, an alternative way to assess the validity of the computed RoA could be to sample the initial conditions inside this complex polytope. More precisely, multiple rollouts were performed for the learned dynamics, where the initial conditions for the state were randomly sampled inside this closed-set. In every case, the state converged to the computed equilibrium.



(a) Projection to position/angle plane.



(b) Projection to linear/angular velocity plane.

Figure 3-5: Illustration of the computed estimate of RoA for NN1.

3-6 Discussion

The framework proposed in this chapter successfully determined a set of conditions able to deduce a number of important stability properties for the DNN dynamics. After proving that the later are fully equivalent to PWA systems, an algorithm with several steps was formulated and implemented for two different examples. As a result, it was possible to compute equilibria, Positively Invariant subsets and Regions of Attraction for the learned dynamics and gain a fundamental understanding on this complicated formulation. Nevertheless, it is essential to make some remarks on the efficiency of the framework and assess its conservativeness in order to give the full picture.

A big concern of the proposed method of analysis should be its scalability with the respect to the number of neurons. Although it is not clear from previous literature on DNN dynamics how many neurons and layers are enough for an accurate approximation, it is expected that a larger number will be required for more complex systems, which in turn will result in greater numerical complexity. Assessing the computational cost in the previous examples for "larger" DNNs, it was noted that the algorithm was tractable under some adjustments. The most important of them was the over-approximation of the linear regions in order to reduce their complexity i.e. their number of linear inequalities (see (3-8)). Given a PI set, the partitions had firstly every redundant inequality removed and then were replaced by a corresponding bounding box. This over-approximation was done mainly in the context of LMI problem of Corollary 3.1 and provided a significant help with a small cost in terms of conservativeness. For the reachability analysis, the corresponding algorithm of Multi-Parametric Toolbox (see Appendix B) has been optimized to be very time efficient. Therefore, it can be argued that the framework is numerically affordable and scalable even for far larger DNNs than the ones given in this study.

In contrast, scalability with respect to state dimensionality poses a greater challenge mainly due to the requirement to compute positively invariant sets and regions of attractions. For the cart-pole system, most of the time required for the framework was devoted to the computation of a Maximal Positively Invariant subset and proved that the corresponding algorithm can become difficult to solve or even intractable. Although higher state dimensionality naturally implies that more linear inequalities will be used to describe the partitions, the bounding box over-approximations can help significantly. More precisely, only 2 more inequalities are added in the H-representation as the state space is extended by 1 dimension. These over-approximations may be in cases far larger that the exact partitions as dimensionality grows, but it was noted in experiments and trials that it did not induce large conservativeness.

Despite the success in deducing important properties for most cases, it was noted, especially for the cart-pole system, that there were NNs that did not manage to approximate really well the underlying stable dynamics, probably because the training optimization was stuck in a local optimum of the weight space. Although this is not a concern of the framework, but rather of the learning process, this divergence between real and learned dynamics complicates the analysis and prevents an accurate evaluation of the conservativeness for high dimensional systems. For example, the **learned** dynamics of the cart-pole system were unstable for many trained DNNs and this led to select a different measure in order to assess their validity. But even amongst DNNs that performed better with respect to the new criterion, it was common for the learned dynamics to be unstable (see NN2 in the previous section). Since it was well-known that the closed-loop dynamics were stable, these DNNs could be discarded as very inaccurate. Consequently, if there is some very rough knowledge about some properties of the real dynamics, the framework can aid with model selection and model validation.

In conclusion, the framework for stability analysis is not significantly conservative, but there exist certain improvements that need to be made in order to extend it to more complex systems. It manages to give fundamental insight into the dynamics of the learned representations and could help evaluate whether a trained DNN is accurate enough. Finally, while scalability with respect to state dimensionality is an issue, it is worth mentioning that, before this work, there was no method for stability analysis even for low-dimensional ReLU NNs.

Chapter 4

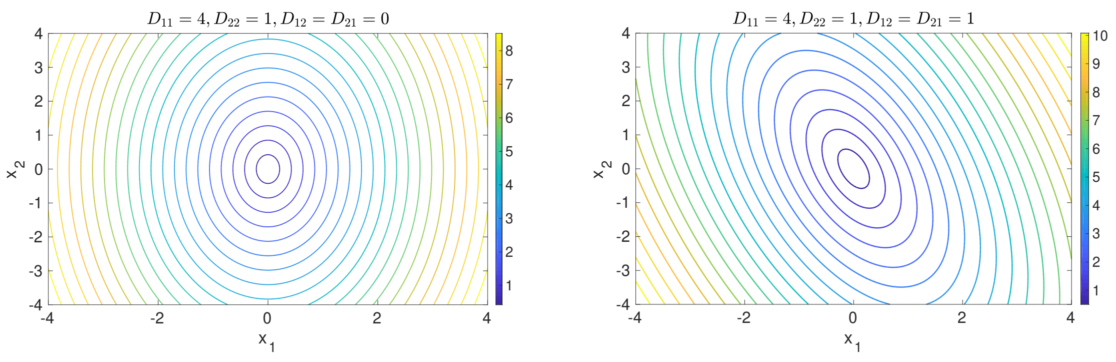
Stability analysis for Locally Weighted Learning

Although the previous chapter, described a stability analysis method for a very popular non-incremental method, its applicability in real control settings remains difficult, as also indicated in the Introduction. In this part, the stability analysis for the class of Locally Weighted Learning (LWL) methods will be presented, aiming to close that gap. The first section provides the formulation of state-space dynamics with an appropriate Locally Weighted Learning model as well as the motivation for the selection of this particular representational structure. In Sections 4-2 and 4-3, the general conditions of Lyapunov Stability Theory for asymptotic stability will be translated into sets of Linear Matrix Inequalities (LMIs) under the assumption of two different Lyapunov function parameterizations. A major contribution of this chapter will be the thorough examination of the resulting LMI conditions with respect to conservativeness and applicability in common LWL settings. Finally, their performance will be tested in a number of numerical examples in Section 4-4.

4-1 Description of dynamics with LWL methods

4-1-1 Local learning

As for every supervised learning task, a mapping between inputs $x \in \mathbb{R}^p$ and outputs $y \in \mathbb{R}^q$ is searched, given a training dataset $\mathbb{D} = \{x_j, y_j\}_{j=1}^{N}$, such that a measure of performance is minimized. Locally Weighted Learning presented in this chapter is a class of supervised learning methods, relying on the idea of *local learning*, where a number of models are fitted **independently** from one another, instead of updating a single global model. Furthermore, they approximate functions without assuming a limited number of parameters, which was the case for the Deep Neural Network presented in the previous chapter, and thus belong to the class of *nonparametric learning methods* [51]. Although in literature the class of methods that are based on this approach is regularly called Locally Weighted Regression (LWR), it should



(a) Diagonally weighted Euclidean distance. **(b)** Mahalanobis distance.

Figure 4-1: Distance for a query point $x = (0, 0)$ for two dimensional input space (x_1, x_2).

be mentioned that LWR is a specific variant that was developed first few decades ago giving the class its name.

An important premise of LWL is that data points x_i closer to the query c will be considered more important for the prediction than points with a greater distance d, contrary for example to least-squares regression that assigns equal important to all data points. This importance is mainly adjusted by selecting two critical factors, existing in every LWR method, the distance function $d(c, x_j)$ and the kernel $w(d)$ i.e. the weighting function.

Starting from the distance function, its structure and domain can vary between tasks. It is usually given in the form of weighted Euclidean distance:

$$d(c, x_j) = \sqrt{(x_j - c)^T D(x_j - c)}, \tag{4-1}$$

where D is referred as the *distance metric* or *distance matrix*. This positive-definite matrix is one of the parameters optimized with respect to the cost function during LWR algorithms and may be the identity matrix (*unweighted Euclidean distance*), a diagonal matrix (*diagonally weighted Euclidean distance*) or an arbitrary symmetric matrix (*Fully weighted Euclidean or Mahalanobis distance*). A diagonal distance matrix is considered to result in an optimal trade-off between expressibility and computational cost. In Figure 4-1, the distance functions for diagonal and symmetric distance metrics are shown for a query coinciding with the origin. The input is considered to be two-dimensional. Although a *global distance* function is by far the most common choice, there have been cases where *query-based local* or *point-based local* distance functions have been used [77]. The choice of a global distance function implies that for a given kernel the distance will not be depended on the query or the training point under consideration.

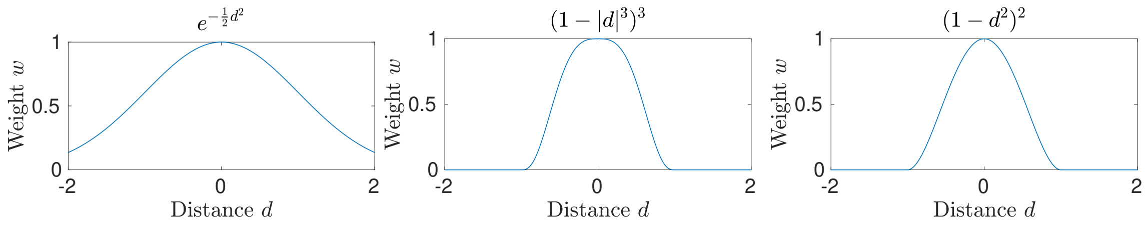
On the other hand, the kernel w intends to normalize the distance so that the maximum "importance" is assigned when the distance is 0, while is reduced as distance increases. Its value usually lies in $[0, 1]$, but its support may be finite or infinite. A common example of kernels with infinite support is the Gaussian [78]:

$$w(x) = e^{-\frac{1}{2}d^2},$$

while Bisquare kernels [79] have a finite support and are described by the following relation:

$$w(x) = \begin{cases} (1 - \frac{1}{2}d^2)^2 & \text{if} \quad |d| < \sqrt{2} \\ 0 & \text{if} \quad |d| > \sqrt{2} \end{cases}$$

These above weighting functions along with another popular kernel with finite support are given in Figure 4-2. Although kernels with infinite support have nonzero value for every point in the input space, in practice (especially during training) an activation threshold is defined to reduce the computational cost with very little effect on the prediction performance.



(a) Gaussian kernel. **(b)** Tricube kernel [80]. **(c)** Bisquare kernel.

Figure 4-2: Kernel functions used regularly for LWR.

LWL methods can now combine these distance and weighting functions with a linear in the parameters model computed by a weighted version of linear regression and provide a localized prediction for every query. Thus, every LWR algorithm aims to compute both the distance metric and the parameters of local linear models, always independently for different kernels.

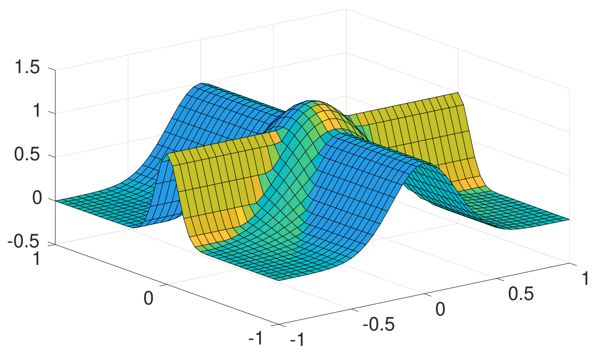
4-1-2 Choice of LWL method

The local nature of learning renders LWL very computationally efficient, whereas they are also known for being robust against negative interference and handling non-stationary input distributions. Nevertheless, LWR algorithms differ significantly in representation and numerical efficiency from one another, and deriving an overall stability result for every one of them is very difficult, if not impossible. Thus, an algorithm that manages to be computationally efficient, while keeping its model relatively simple will be sought after.

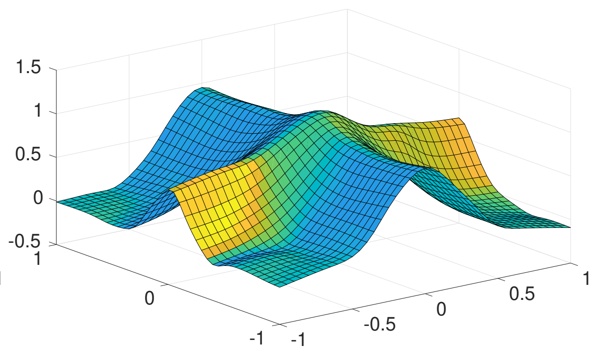
Starting from the LWL variant that was proposed first chronologically, Locally Weighted Regression, it can be noted that it belongs to the class of *lazy-learning* or *memory-based learning*, for which all the training points should be kept in memory. Given an initial guess for the distance metric, this method computes a unique local model for every query by repeatedly performing a weighted version of least squares regression. This results in an increase in memory and computation cost as the number of training points rises. Except the previous limitations, it is not clear how stability analysis for LWR could be accomplished, given that no explicit dynamics model exists, but the later depends on every considered query.

In contrast, a variant of LWR that does not require storing all the training points, while preserves a model from learning iteration to learning iteration is Receptive Field Weighted Regression (RFWR) [81, 82] and could be a valid candidate for stability analysis. Just like LWR, Receptive Field Weighted Regression updates its parameters, when a new training point is examined (*incremental method*), but does not discard the whole input/output mapping as the former. Being a non-parametric method, RFWR does not require fixing the number of parameters and allows addition and pruning of local models (also called Receptive Fields (RFs)) on an as-needed basis. This results in an optimal trade-off between flexibility and computational efficacy, while deriving a fixed representation at the end of training. The later is an important aid for the stability analysis.

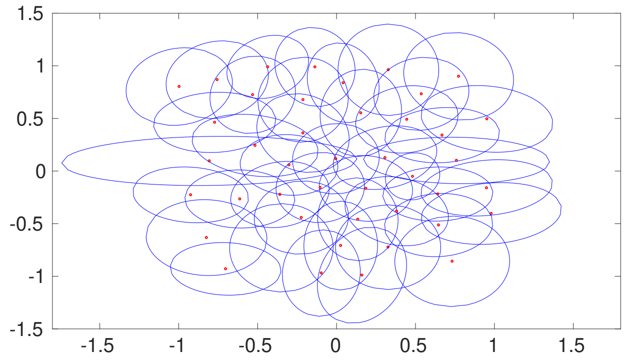
In Figure 4-3 a nonlinear function commonly used as an example in LWR studies is approximated by RFWR and a training set of 1700 data points randomly spread (without noise in the measurements) through the input space.



(a) True function.



(b) RFWR approximation.



(c) Local models of RFWR approximation in input space. The red points denote the centers of the RFs.

Figure 4-3: Approximation of a complex nonlinear function $\mathbb{R}^2 \to \mathbb{R}$ with 43 Bisquare kernels.

Probably the most important downside of RFWR is its susceptibility in the so called *curse of dimensionality*. As the number of input dimensions increases, **exponentially** more training points and local models are required to learn a sufficiently good approximation. Thus, an extension of RFWR able to handle high-dimensional inputs was proposed in [83]. This algorithm was named Locally Weighted Projection Regression (LWPR) and it is relied on an incremental locally weighted implementation of a well-known projection regression algorithm, Partial Least Squares (PLS) [84,85]. For every Receptive Field, a number of directions in the input space is incrementally updated for every training point, while univariate regression is performed along them. The update of the distance metric as well as the addition and pruning of kernels is done the same way as for RFWR. Thus, it is possible to identify redundant dimensions and project **locally** the input space in a reduced space.

LWPR improves significantly both training and prediction performance only when *redundant and irrelevant* input dimensions exist, but when these conditions do not apply, the gains are not significant compared to RFWR. On the other hand, the locality of projections performed individually for every kernel renders the description of dynamics more complicated than RFWR, which naturally will affect the ability to carry out the stability analysis. Therefore, dynamics representation corresponding to a RFWR model is considered a good compromise between complexity and numerical efficiency, especially under the assumption that dimensionality reduction is already performed and no redundancy is induced when the states are measured. The description of the state-space dynamics required for RFWR will be provided in the next subsection.

4-1-3 Description of dynamics with RFWR

After training is finished for the RFWR algorithm, a number of kernels are scattered throughout the input space as shown in Figure 4-3c. The prediction denoted as $\hat{y} \in \mathbb{R}^q$ for a given input x is computed as weighted sum of the individual predictions $\hat{y}_i \in \mathbb{R}^q$ of all local models:

$$\hat{y} = \frac{\sum_{i=1}^{M} w_i \hat{y}_i}{\sum_{i=1}^{M} w_i} \tag{4-2}$$

where M is the total number of RFs and $w_i \in \mathbb{R}$ is the weight corresponding to each RF, which depends on the selected kernel (see Subsection 4-1-1) and the distance of the input x from the center c_i of the receptive field. The total prediction should rely more on the prediction of the i$^{\text{th}}$ RF as the distance from its center tends to 0, and thus the query c as it was defined in (4-1) should be replaced by the center of the RF c_i:

$$d_i(x) = \sqrt{(x - c_i)^T D_i (x - c_i)}$$

The local prediction $\hat{y}_i$ is always given by a linear in the parameters model, which can be composed of any nonlinear terms with respect to the input vector. However, it has been noted [81] that linear terms are able to represent adequately well any nonlinearities without increasing the complexity. The local prediction for a given input is then computed in most cases as:

$$\hat{y}_i = \beta_i^T x + \beta_i^0 = \tilde{\beta}_i \tilde{x}$$

where $\tilde{x} = [x^T, 1]^T \in \mathbb{R}^{p+1}$ and $\tilde{\beta}_i = [\beta_i^T, \beta_i^0] \in \mathbb{R}^{q\times(p+1)}$ is the parameter matrix computed from regression corresponding to i$^{\text{th}}$ local model. Thus, (4-2) can be better expressed as:

$$\hat{y} = \frac{\sum_{i=1}^{M} w_i(x)(\beta_i^T x + \beta_i^0)}{\sum_{i=1}^{M} w_i(x)} = \sum_{i=1}^{M} h_i(x)\tilde{\beta}_i \tilde{x} \tag{4-3}$$

where $h_i(x) = \frac{w_i(x)}{\sum_{l=1}^{M} w_l(x)}$ is the normalized weight, for which it obvious that:

$$\sum_{i=1}^{M} h_i(x) = 1, \qquad 0 \leq h_i(x) \leq 1 \quad \forall i$$

Since it is desired for the RFWR to learn the (discrete-time) state-space model as this was broadly described by nonlinear relation (4-3), the following assumptions are implicit:

- The input vector is replaced by $x(k)$ i.e. the state vector at time k.
- The output vector is replaced by $x(k+1)$ i.e. the state vector at the next time step.
- Input and output dimensionality are equal to state dimensionality n.
- The dynamics of the learned system can be expressed as follows:

$$x(k+1) = f_{\text{lwl}}(x(k)) = \sum_{i=1}^{M} h_i(x(k))[\beta_i^T x(k) + \beta_i^0]$$

  where $\beta_i \in \mathbb{R}^{n\times n}$ and $\beta_i^0 \in \mathbb{R}^n$.

In order to keep up with notation frequently in similar control studies, A_i will denote the square parameter matrix β_i and b_i will denote the constant bias vector β_i^0. There are, then, two equivalent state-space representations for the RFWR model, which will be mainly used for the rest of this chapter.

$$x(k+1) = \sum_{i=1}^{M} h_i(x(k))[A_i x(k) + b_i] \tag{4-4a}$$

$$\tilde{x}(k+1) = \sum_{i=1}^{M} h_i(x(k)) \begin{bmatrix} A_i & b_i \\ 0 & 1 \end{bmatrix} \tilde{x}(k) = \sum_{i=1}^{M} h_i(x(k))\tilde{A}_i\tilde{x}(k) \tag{4-4b}$$

If a local model i contributes to the state progression in time k, $x(k)$ is included in its activation region $\mathcal{X}_i$, with the latter defined strictly as:

$$\mathcal{X}_i := \{x \in \mathbb{R}^n : h_i(x) > 0\}$$

In the case of Gaussian kernels or other kernels with infinite support, a boundary will always be set, such that $\mathcal{X}_i$ is finite. Because the parameters of the local models A_i and b_i are computed through regression, it is rather uncommon to be equal to 0, unless they are biased before initiating the learning algorithm. However, in the exceptional case where $b_i = 0$, the i$^{\text{th}}$ local model becomes linear instead of affine and its index is included in the set $\mathcal{I}_0$, while for the cases that $b_i \neq 0$ the corresponding index is included in the set $\mathcal{I}_1$.

- $\mathcal{I}_0 := \{i \in \mathcal{I} \;:\; b_i = 0\}$
- $\mathcal{I}_1 := \{i \in \mathcal{I} \;:\; b_i \neq 0\}$

4-1-4 Equilibrium and state transformation

Lyapunov stability theory as it was briefly introduced in Chapter 2 always examines the stability of a particular equilibrium in the state space. If a single equilibrium x_e is assumed for the learned system (4-4), it is necessary then to locate its coordinates. Although computing the equilibrium x_e^i of an individual local affine model is usually trivial:

$$x_e^i = (I - A_i)^{-1} b_i \tag{4-5}$$

the same does not apply for the "global" model, for which the equation defining the equilibrium $x_{\mathrm{e}} = f_{\mathrm{lwl}}(x_{\mathrm{e}})$ should be transformed in a system of n nonlinear equations:

$$\sum_{i=1}^{M} w_i(x_{\mathrm{e}})[(I - A_i)x_{\mathrm{e}} - b_i] = 0$$

Given some prior knowledge on the position of the equilibrium of the true dynamics, a non-linear iterative algorithm can compute very quickly in most cases the solution of the above system of equations starting from an approximate initial condition and then locate the equilibrium with arbitrary accuracy.

In order to facilitate analysis, the nonlinear system is usually transformed such that the equilibrium under examination coincides with the origin i.e. $x_{\mathrm{e}} = 0$. A simple change of coordinates $z(k) = x(k) - x_{\mathrm{e}}$ can impose the above requirement and the state equation (4-4a) will become:

$$z(k+1) = \sum_{i=1}^{M} h_i(z(k))[A_i z(k) + b_i + A_i x_{\mathrm{e}} - x_{\mathrm{e}}] = \sum_{i=1}^{M} h_i(z(k))[A_i z(k) + b_i']$$

where the distance affecting the normalized weight $h_i(z(k))$ now becomes:

$$d_i(z) = \sqrt{[z - (c_i - x_{\mathrm{e}})]^T D_i [z - (c_i - x_{\mathrm{e}})]}$$

For the rest of the analysis, the learned dynamics (4-4) will be assumed to be already transformed such that the equilibrium coincides with the origin.

4-1-5 Similarities with Takagi-Sugeno (T-S) fuzzy models

Describing nonlinear dynamics through convex combination of simpler local models has been the basis for different classes of nonlinear models and a large part of control research has focused on studying different notions of stability for them. One of them, Takagi-Sugeno (T-S) fuzzy models, shares a lot of representational similarities with LWR models and the large existing literature could help accelerate the stability analysis for the latter. Therefore, the T-S fuzzy model of discrete-time autonomous dynamics will be briefly described in this section, but the interested reader could check for example [86–88] for more comprehensive analysis of T-S fuzzy modeling in control.

The progression of the state in time using T-S fuzzy models could be given by a set of fuzzy rules $\{R_i\}_{i=1}^{M}$ in the following form:

$$\begin{aligned} R_i : \text{IF } (\xi)_1 \text{ is } F_{i1} \text{ and } (\xi)_2 \text{ is } F_{i2} \text{ and } \dots \text{ and } (\xi)_v \text{ is } F_{iv} \\ \text{THEN } x(k+1) = A_i x(k) \end{aligned}$$

where $\xi(k) = [(\xi)_1, (\xi)_2, ..., (\xi)_v] \in \mathbb{R}^v$ are the *premise variables*, v is their dimensionality (in general different than the state dimensionality) and F_{ij} are the *fuzzy variables*. The premise variables could be any measurable quantity e.g. output variables or states of the system, whose change could suggest a variation in dynamics. The consequent part in these rules is a linear state update equation and it is common to write the fuzzy model in the following Input-output form:

$$x(k+1) = \frac{\sum_{i=1}^{M} w_i A_i x(k)}{\sum_{i=1}^{M} w_i} \tag{4-6}$$

with the "activation" or weight given by:

$$w_i = \prod_{j=1}^{p} F_{ij}\Big((\xi)_j\Big)$$

It can be noted that state progression (4-6) is based on a set of smoothly overlapping local models, like the RFWR model describ ed in the previous sections. While both representations include a weighting function, the weight w_i on T-S fuzzy models is a nonlinear function of the premise variables ξ and the weight in RFWR is a nonlinear function of the state vector x. In general, the space along which the premise variables are spanned and the state space are different and only under appropriate assumptions, the weights can be equal. Furthermore, most research on stability of T-S systems has focused on linear (in the state) local models, while RFWR is mainly using affine ones. However, there have been few studies on affine T-S systems, which may seem a straightforward extension of linear ones, but it has been proven that the constant bias term actually complicates analysis. These studies [88–90] will be an important aid for the rest of the analysis for LWL methods.

4-2 Stability analysis with Common Quadratic Lyapunov Function

Now that the dynamics and the equilibrium are well established, the stability analysis can proceed. Starting from the most common and simple parametrization of Lyapunov functions for T-S fuzzy models, a number of conditions equivalent to exponential stability will be derived similarly to [89, 90]. These conditions will be then assessed with respect to their conservativeness for the LWL dynamics formulation using simple synthetic examples and then novel strict mathematical proofs will explain and justify the conclusion.

4-2-1 Conservative quadratic stability

Assume the nonlinear dynamics (4-4a) are fully known i.e. update matrices A_i, bias vectors b_i, distance metrics D_i and centers c_i have been computed by the learning algorithm with one of the standard kernels mentioned in the previous sections. The following simple parametrization of the candidate Lyapunov function is proposed:

$$V(x(k)) = x^T(k)Px(k) \tag{4-7}$$

In literature, this is referred to as *Common Quadratic Lyapunov Function (CQLF)* and the goal now is to find a matrix $P \in \mathbb{R}^{n\times n}$ such that the conditions for a notion of stability (see Chapter 2) is fulfilled. The transformation of the conditions in Theorem 2.1 into a convex optimization problem under the Common Lyapunov function parametrization can be summarized in Theorem 4.1 and its proof adapted from similar work on T-S fuzzy systems [89, 90] will be presented shortly below.

Theorem 4.1. *Let $x_\mathrm{e} = 0$ be an equilibrium of* (4-4)*. If there exists a positive definite matrix $P \in \mathbb{R}^{n\times n}$ such that the following LMIs are fulfilled:*

$$A_i^T P A_i - P < 0, \qquad i \in \mathcal{I}_0 \tag{4-8a}$$

$$\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} < 0, \qquad i \in \mathcal{I}_1 \tag{4-8b}$$

then x_e is (globally) asymptotically stable

The proof of this Theorem is adapted to representation (4-4) from similar works present in literature [89, 90], and in order to keep this work as compact as possible it is given in detail in Appendix A. Although the above result provides a first set of conditions to be examined for asymptotic stability, as soon as it is further explored it is noticed that these conditions can **never be fulfilled** for any RFWR model as long as at least one of the local models is affine i.e. $\mathcal{I}_1 \neq \emptyset$ and the following corollary is proposed with its proof being also included in Appendix A.

Corollary 4.1. *If there exists at least one local affine model, then the conditions of Theorem 4.1 can never be fulfilled.*

To provide some insight into the previous corollary, the requirement for constantly decreasing Lyapunov function results in a number of single terms, each one independent from the others and corresponding to a single local model and is independent. Each such term $(A_i x + b_i)^T P (A_i x + b_i) - x^T P x$ has to be negative for the same matrix P as the state tends to the global equilibrium x_e. However, it can be seen that for affine models the global equilibrium does not coincide with the equilibrium of the local model x_e^i given in (4-5) and the above term will be negative only as the state converges to that point. This large conservativeness will be partly handled in the next subsection by reducing the region of applicability of the conditions.

4-2-2 Reducing conservativeness with S-procedure

In the previous subsection, the proposed LMI conditions provided a very conservative result that requires for all the local models to be linear. Although the learning algorithm can certainly be adjusted, several tests have shown that the prediction performance deteriorates significantly when the bias terms were set to 0. There is, nevertheless, an alternative to reduce the conservativeness of Theorem 4.1 by considering only the activation region of each local model instead of the whole state space $\mathbb{R}^n$ with a relaxation process presented in Chapter 2; the *S-procedure.*

To reduce conservativeness with the S-procedure it is necessary to derive a matrix S_i for every affine model such that:

$$\tilde{x}^T S_i \tilde{x} \geq 0 \qquad \forall x \in \mathcal{X}_i, i \in \mathcal{I}_1 \tag{4-9}$$

This S_i matrix will depend on the mathematical description of the activation regions $\mathcal{X}_i$, but to keep the analysis as general as possible it will not be further constrained. Thus, the following less conservative version of Theorem 4.1 is proposed:

Theorem 4.2. *Let $x_{\mathrm{e}} = 0$ be an equilibrium of* (4-4) *and $S_i \in \mathbb{R}^{(n+1)\times(n+1)}$ such that* (4-9) *is satisfied for every $i \in \mathcal{I}_1$. If there exists a positive definite matrix $P \in \mathbb{R}^{n\times n}$ and multipliers $\tau_i \in \mathbb{R}_{\geq 0}$ such that the following LMIs are fulfilled:*

$$A_i^T P A_i - P < 0, \qquad i \in \mathcal{I}_0 \tag{4-10a}$$

$$\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} + \tau_i S_i < 0, \qquad i \in \mathcal{I}_1 \tag{4-10b}$$

then x_{e} is (globally) asymptotically stable

Proof. Assuming that the conditions of this theorem are true, it needs to be proven that all conditions (2-2) of Theorem 2.1 are necessarily fulfilled. The proof of this theorem lies heavily on the proof of Theorem 4.1 and the S-procedure as described above.

The first three conditions can be shown to be fulfilled the exact same way as for Theorem 4.1, and thus their proof is omitted.

Applying the S-procedure it is known that if there exists $\tau_i \geq 0$ such that:

$$\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} + \tau_i S_i < 0$$

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^T \left(-\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} - \tau_i S_i \right) \begin{bmatrix} x \\ 1 \end{bmatrix} > 0$$

then $\tilde{x}^T S_i \tilde{x} \geq 0, \ \forall x \in \mathcal{X}_i$ implies:

$$-\begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} > 0, \qquad \forall x \in \mathcal{X}_i$$

Therefore, it is proven that $(A_i x + b_i)^T P (A_i x + b_i) - x^T P x < 0$ for every x in the i^{th} activation region (where $h_i(x) \neq 0$). For the fourth condition, ΔV meets the following condition:

$$\Delta V(x) \leq \sum_{i}^{M} h_i^2(x) \left[(A_i x + b_i)^T P (A_i x + b_i) - x^T P x \right] + \sum_{i<j}^{M} h_i(x) h_j(x) \cdot$$
$$\cdot \left[(A_i x + b_i)^T P (A_i x + b_i) - x^T P x + (A_j x + b_j)^T P (A_j x + b_j) - x^T P x \right]$$

Thus, ΔV will be negative for every x different than 0 and asymptotic stability is proven. □

In order to provide some extra insight into matrices S_i, different descriptions of the activation regions will be considered. It is worth noting that (4-9) has to be fulfilled in a superset of the activation region, and not necessarily on its equal set. In Table 4-1, the description and the corresponding S_i matrix are given for every type of activation region. Although the bounding box activation region is not selected as kernel, it may be used as an overapproximation of a more complex region. Finally, since the Gaussian kernel has infinite support, a boundary should be set arbitrarily by setting the α variable (a number from 0 to 1) [91]. For a specific value of α, there is $1-\alpha$ probability that the x vector lies in the ellipsoid $\mathcal{E}$ defined as:

$$\mathcal{E} = \left\{x \in \mathbb{R}^n \mid (x-c_i)^T D_i (x-c_i) \leq \chi_n^2(\alpha)\right\}$$

where χ_n^2 is the chi-squared distribution with n degrees of freedom.

$\mathcal{X}_i$	Mathematical description	S_i
Bisquare	$\left\{x \in \mathbb{R}^n \mid (x-c_i)^T D_i (x-c_i) \leq 2\right\}$	$-\begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & c_i^T D_i c_i - 2 \end{bmatrix}$
Bounding box	$\left\{x \in \mathbb{R}^n \mid u_{min} \leq x \leq u_{max}\right\}$	[89]
Gaussian	$\left\{x \in \mathbb{R}^n \mid (x-c_i)^T D_i (x-c_i) \leq \chi_n^2(\alpha)\right\}$	$-\begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & c_i^T D_i c_i - \chi_n^2(\alpha) \end{bmatrix}$

Table 4-1: Computation of S_i for different kernels.

4-2-3 Evaluation with respect to conservativeness

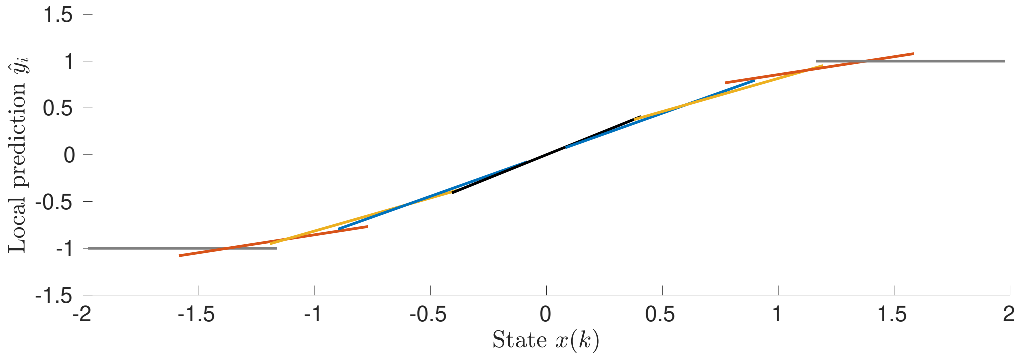
In order to assess how conservative is Theorem 4.2, a synthetic example was created in 1-dimensional state-space by approximating a simple nonlinear function in $(-\pi/2, \pi/2)$:

$$x(k+1) = \sin\Big(x(k)\Big)$$

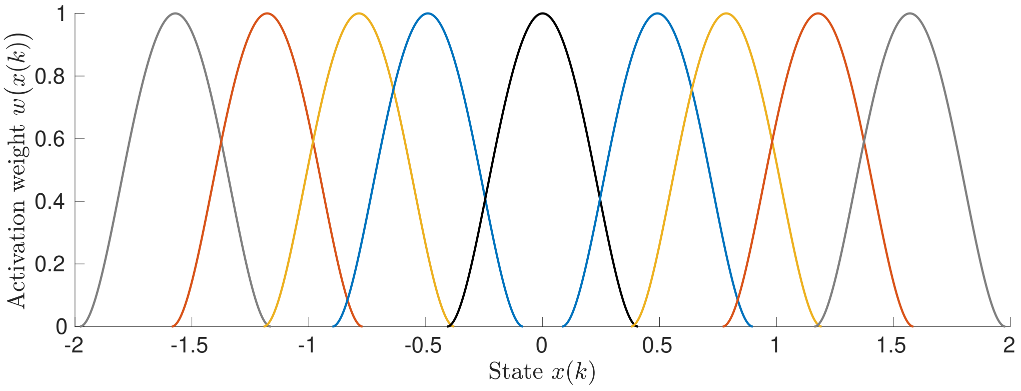
The parameters of local models and their kernels, usually given by the learning procedure, in this case were designed through iterations such that the function looks relatively close to the real one. The approximated dynamics were tested for random initial conditions inside the interval $(-\pi/2, \pi/2)$ and it was found that the state converges to 0 for all of them. The local affine models $\hat{y}_i = A_i x(k) + b_i$ for this case are given in Figure 4-4a, while the corresponding kernels distributed along the state space are shown in Figure 4-4b.

Since only one local model is covering the origin, then it was set arbitrarily that this local model will be linear instead of affine in order to ensure that the origin will be the equilibrium. The conditions mentioned in Theorem 4.2 were checked for the 8 affine models and the single linear model and indeed a solution was found through Semidefinite Programming (SDP) i.e. $P > 0$ matrix and $\tau_i \geq 0$ such that conditions (4-10) are fulfilled.

For the same dynamics, the local models and kernels were computed with Receptive Field Weighted Regression and the 13 kernels are shown in Figure 4-5. The stability of this approximation was evaluated initially by sampling random initial conditions in $(-\pi/2, \pi/2)$ and in every case the state converged to 0. Then, when a solution that would satisfy the LMIs of Theorem 4.2 was searched, it was impossible to find one. In conclusion, it was impossible to prove stability for a globally stable system.



(a) Prediction with affine local models $\hat{y}_i$.



(b) Bisquare kernels.

Figure 4-4: Approximation of $\sin(x)$ function with a synthetic RFWR model.

Searching for the reason of this conservativeness, it was noted that the local models covering the origin are affine instead of linear, an assumption that was made for the single kernel activated on the origin in Figure 4-4b. In studies where T-S affine models are considered, the later assumption is made by definition [89, 92], but usually in these cases local models do not overlap around the origin and therefore, it does not impose any issues on the LMI conditions proposed.

In control theory, it is commonly accepted that the dynamics around the equilibrium are "close" to linear and no large nonlinearities exist. Furthermore, in Figure 4-6 only the kernels near the origin are plotted along with their corresponding local equilibria x_e^i. The three kernels activated on the origin are very close to linear (their equilibria are near the origin) and this assumption seems valid in this example. Imposing local linear models around the origin before or after learning came with a small prediction cost in this case and the bias term β^0 was set to 0 for the 3 local models in the middle of Figure 4-6.

For this formulation, the LMIs (4-10) were posed again for an increased number of linear models, but nevertheless a solution was not found. The LMIs that were impossible to be satisfied in this case were the ones that corresponded to local (affine) models that had in their activation region $\mathcal{X}_i$ their own local equilibrium x_e^i. Except for the three models that are activated in the origin, four more have their local equilibrium in their activation region for the simple example of learning the sin dynamics as can be seen in Figure 4-6. The following corollary better describes this important observation and its proof can be found in Appendix

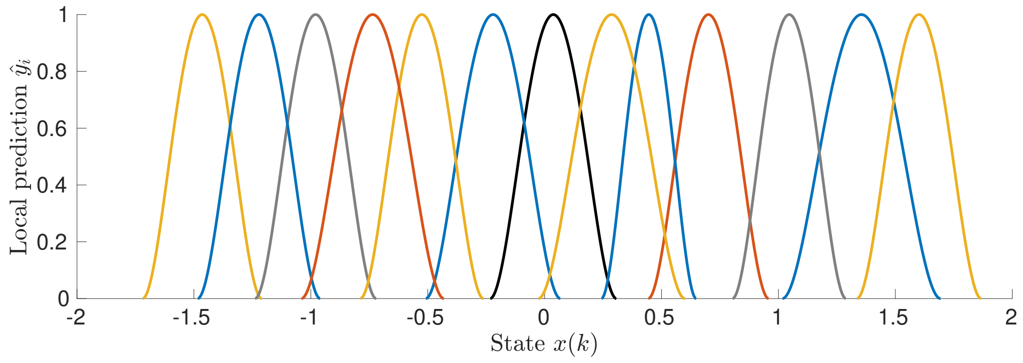


Figure 4-5: Kernels computed with RFWR for the sin function in the $(-\pi/2, \pi/2)$ interval.

A.

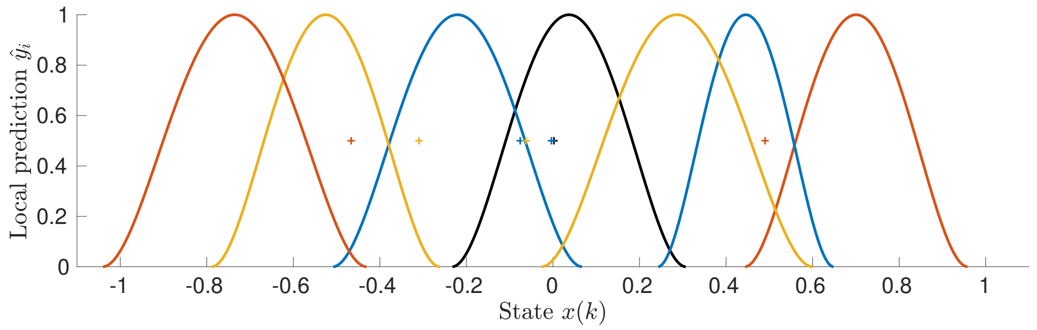


Figure 4-6: Kernels near the origin and their corresponding local equilibria shown with a cross (the y coordinate for the equilibria is irrelevant).

Corollary 4.2. *If there exists a local model, for which:*

$$x_e^i \in \mathcal{X}_i, \qquad\qquad i \in \mathcal{I}_1$$

then the conditions of Theorem 4.2 can never be fulfilled.

Proof. Consider a local model for which $x_e^i \in \mathcal{X}_i, i \in \mathcal{I}_1$ and the corresponding LMI of Theorem 4.2:

$$\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} + \tau_i S_i < 0$$

In this proof, a Bisquare kernel will be considered, but the same arguments can be adjusted also for any kernel and function S_i that represents exactly or outer-approximates the activation region. Therefore, from Table 4-1 the description of the activation region of the Bisquare kernel and the corresponding S_i matrix are:

$$\mathcal{E} = \left\{ x \in \mathbb{R}^n \mid (x - c_i)^T D_i (x - c_i) \leq 1 \right\} \qquad\qquad S_i = -\begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & c_i^T D_i c_i - 1 \end{bmatrix}$$

Thus, the previous LMI becomes:

$$\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} - \tau_i \begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & c_i^T D_i c_i - 1 \end{bmatrix} < 0$$

If multiplied on both sides with vector $\tilde{x}$ then:

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^T \left(\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} - \tau_i \begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & c_i^T D_i c_i - 1 \end{bmatrix} \right) \begin{bmatrix} x \\ 1 \end{bmatrix} < 0$$

If a change of coordinates is performed such that the local equilibrium x_e^i coincides with the origin i.e. $w = x - x_e^i$, it follows that:

$$\begin{bmatrix} w + x_e^i \\ 1 \end{bmatrix}^T \left(\begin{bmatrix} A_i^T P A_i - P & A_i^T P b_i \\ b_i^T P A_i & b_i^T P b_i \end{bmatrix} - \tau_i \begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & c_i^T D_i c_i - 1 \end{bmatrix} \right) \begin{bmatrix} w + x_e^i \\ 1 \end{bmatrix} < 0$$

After simple manipulations of the above relation and replacing b_i according to (4-5), the following inequality is derived:

$$\begin{bmatrix} w \\ 1 \end{bmatrix}^T \left(\begin{bmatrix} A_i^T P A_i - P & -P x_e^i + A_i^T P x_e^i \\ -(x_e^i)^T P + (x_e^i)^T P A_i & 0 \end{bmatrix} \right.$$
$$\left. - \tau_i \begin{bmatrix} D_i & -D_i c_i \\ -c_i^T D_i & (x_e^i - c_i)^T D_i (x_e^i - c_i) - 1 \end{bmatrix} \right) \begin{bmatrix} w \\ 1 \end{bmatrix} < 0$$

Finally, we get that:

$$\begin{bmatrix} A_i^T P A_i - P - \tau_i D_i & -P x_e^i + A_i^T P x_e^i + \tau_i D_i c_i \\ -(x_e^i)^T P + (x_e^i)^T P A_i + \tau_i c_i^T D_i & -\tau_i [(x_e^i - c_i)^T D_i (x_e^i - c_i) - 1] \end{bmatrix} < 0$$

Since $x_e^i \in \mathcal{X}_i$, then by definition of hyperellipsoid:

$$(x_e^i - c_i)^T D_i (x_e^i - c_i) \leq 1$$

Thus, the lower right entry of the above matrix is non-negative (τ_i always non-negative) and using Schur Complement, it can be proven that it can never be negative definite. □

The intuition behind this claim lies in the process of transforming the requirement for constantly decreasing CQLF into LMIs as it presented in the proof of Theorem 4.1. To be more specific, ΔV was shown to be explicitly dependent on the normalized weights for each model $h_i(x)$ as well as on multiple overlapping local models for the same state x. However, each single LMI examined in Theorems 4.1 and 4.2 does not include in any way the normalized weight. All the LMIs refers to independent models, without considering the interactions between them. If for each local model a single Lyapunov function $V_i(x)$ is considered, then $V_i(x)$ will only be decreasing as the state approaches the local equilibrium x_e^i. On the other hand, LMIs like (4-10b) can only be fulfilled, if $V_i(x)$ is always decreasing as the state converges to 0. Thus, if x_e^i is included in $\mathcal{X}_i$, these two conditions are in conflict and they cannot be satisfied on the same time.

The case where local equilibria are included in the corresponding activation region may seem rather specific, and thus of little interest, but it has been observed during this work that it occurs for many local affine models around the equilibrium and for most nonlinear dynamics. "Linearizing" all these models will have a large cost in prediction and computation, since affine local models have been noted in practice to be a far better parametrization for LWL models than linear ones. The above remarks ascertain the fact that the Common Quadratic Lyapunov Function is indeed a very conservative parametrization that *does not allow to consider any interactions between the local models.* LWL models, however, rely more on these interactions and overlaps between local models to approximate complex and nonlinear dynamics than T-S fuzzy systems and a less conservative parametrization should be considered.

4-3 Stability analysis with Piecewise Quadratic Lyapunov Function

The analysis for the Common Quadratic Lyapunov Function showed the significant limitations imposed by the nature of this parametrization with respect to Locally Weighted Learning models. The simplicity of the resulting LMI conditions presented in Theorem 4.2 for a CQLF comes at a cost of neglecting important characteristics of the dynamics such as the interactions between local models, and thus generates large conservativeness. In this section, another popular parametrization, *Piecewise Quadratic Lyapunov Function (PQLF)*, will be proposed as a less conservative, yet tractable solution. This parametrization was also proposed in Section 3-3-1 for Deep Neural Networks, but in that case the partition in non-overlapping regions was naturally available.

4-3-1 Partition of the state space

Since local models in LWL are activated only in bounded, restricted regions of the state-space, a Lyapunov function that changes its relation depending the region would be more suitable and in general a less conservative choice. Such an example is PQLF, which is determined by a number of non-overlapping quadratic relations with respect to $x(k)$ (or $\tilde{x}(k)$), each one corresponding to a single subregion $\mathcal{X}_l \subset \mathbb{R}^n$:

$$V(x(k)) = \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_l \begin{bmatrix} x \\ 1 \end{bmatrix} = \tilde{x}^T(k)\tilde{P}_l\tilde{x}(k), \qquad x \in \mathcal{X}_l \tag{4-11}$$

where $\tilde{P}_l \in \mathbb{R}^{(n+1)\times(n+1)}$ and the index l implies that the regions $\mathcal{X}_l$ are in general different than the activation regions of the local models $\mathcal{X}_i$.

PQLFs have been found to be largely successful for hybrid systems i.e. systems that combine continuous dynamics and discrete events [62, 63, 93]. For these systems, the changes of dynamics are caused by discrete-time events, which are well defined and can provide an easy way to partition the state-space. For LWL models, as they were described in previous sections, this partition is not so trivial, due to overlapping local dynamics. Taking advantage previous studies on T-S fuzzy models two different approaches on space-space partition can be proposed:

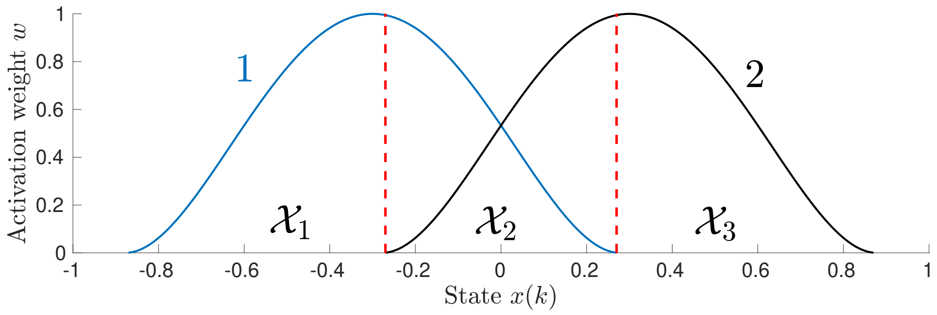


Figure 4-7: Partition of 1-dimensional state-space in operating regions ($\mathcal{X}_1$, $\mathcal{X}_3$) and interpolation regions ($\mathcal{X}_2$) for simple example of two overlapping kernels.

1. The state-space is divided in *operating regimes.* where only one local model is activated and *interpolation regimes*, where a unique combination of local models is considered [88, 94].
2. The state-space is divided in M regions, each one defined by the following relation [95–97]:

$$\mathcal{X}_l = \Big\{ x \in \mathbb{R}^n \mid h_l(x) > h_j(x),\ j \in \mathcal{I}_0 \cup \mathcal{I}_1 \setminus \{l\} \Big\}$$

In [88] these two partition approaches and the CQLF were compared (for T-S fuzzy systems) with respect to conservativeness in numerical examples and it was concluded that the first approach i.e. the division of state-space in interpolation and operating regimes is less conservative, since the second one requires the introduction of uncertainty. Thus, from the rest of this section only partition 1. will be analyzed, but its increased computational cost should always be kept in mind.

In order to proceed with the analysis, one important new feature is required for the full dynamics description. For each region $\{\mathcal{X}_l\}_{l=1}^{\mathcal{N}}$, a set $\mathcal{J}(l)$ with all the indices of the local models activated within its boundaries should be computed. For the operating regimes, $\mathcal{J}(l)$ should contain one element, while for the interpolation regimes at least two elements. To explain better how the division in interpolation and operating regimes happens, in Figure 4-7 a simple example of two overlapping kernels in one dimension is presented, along with their partition in three separate subregions $\{\mathcal{X}_l\}_{l=1}^{3}$. In this simple case (where local model 1 is in blue color and model 2 in black), it is easy to see that $\mathcal{J}(1) = \{1\}$, $\mathcal{J}(2) = \{1, 2\}$, $\mathcal{J}(3) = \{2\}$. The state-dynamics are formulated for this partition the following way:

$$x(k+1) = \textstyle\sum_{j \in \mathcal{J}(l)} h_j(x(k))[A_j x(k) + b_j], \qquad x(k) \in \mathcal{X}_l \tag{4-12a}$$

$$\tilde{x}(k+1) = \textstyle\sum_{j \in \mathcal{J}(l)} h_j(x(k)) \tilde{A}_j \tilde{x}(k), \qquad x(k) \in \mathcal{X}_l \tag{4-12b}$$

Contrary to the CQLF approach implemented in the previous section, where the state-space was analyzed as a whole, its partition for PQLF renders the study of transitions between regions $\mathcal{X}_l$ necessary. In discrete-time dynamics, these transitions do not occur necessarily between adjacent regions as it is for continuous-time dynamics, but the examination of all possible transitions between defined regions is required in a process usually called for hybrid systems *(one step) reachability analysis* [65]. Since the local dynamics are time-invariant (A_i, b_i are constant in time), it suffices to consider the transition for one time step for each region in order to characterize all possible transitions. The set of all possible transitions and the corresponding transition matrix are defined the same way as for Chapter 3:

$$\Omega := \{(l,j) \mid x(k) \in \mathcal{X}_l, x(k+1) \in \mathcal{X}_j\}, \tag{4-13}$$

$$(T)_{lj} = \begin{cases} 1 & \text{if } \exists\, x(k) \in \mathcal{X}_l:\ x(k+1) \in \mathcal{X}_j \\ 0 & \text{otherwise} \end{cases}.$$

To perform the reachability analysis for this case, the reachable set $\mathcal{U}_l = \{x(k{+}1) \mid x(k) \in \mathcal{X}_l\}$ will be outer approximated computing the maximum reachable set as it was done in [98] and then consecutive Linear Programs (LPs) will be solved to find which regions $\mathcal{X}_l$ intersect with the resulting polyhedron.

4-3-2 Description of regimes

For LWL, the boundaries and the activation regions of the local models are usually described by quadratic forms with respect to the state vector $x(k)$. More specifically, both Gaussian and Bisquare kernels as described in Table 4-1 are equivalent to hyperellipsoids in $\mathbb{R}^n$. A general hyperellipsoid in state space centered at c is given by

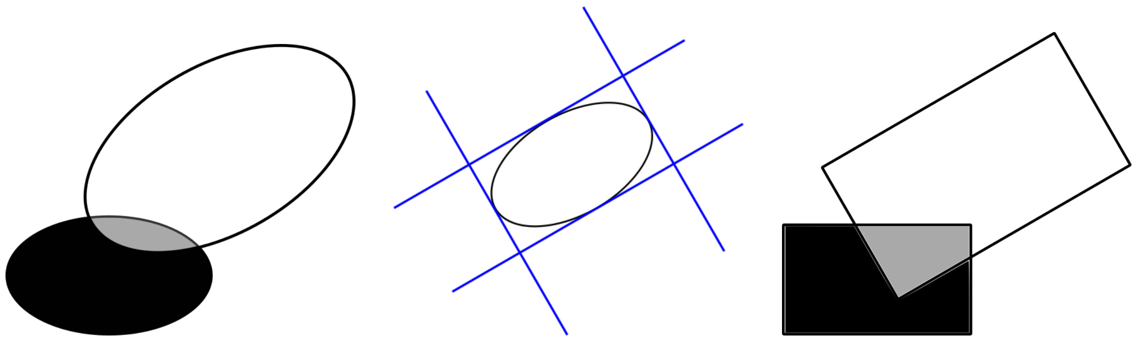
$$\mathcal{E} := \{x \in \mathbb{R}^n \mid (x-c)^T A (x-c) \leq 1\}, \tag{4-14}$$

where A is a (symmetric) positive definite matrix. Let a region $\mathcal{X}_l$ be an interpolation regime, where at least two kernels intersect. It is difficult (in general) to describe $\mathcal{X}_l$ precisely with a compact relation as it was done for the original kernels. For example, in Figure 4-8a two ellipses representing two overlapping kernels in $\mathbb{R}^2$ are presented and their intersection has not a geometric shape easy to describe with a standard relation. On the other hand, a description of $\mathcal{X}_l$ is required to study the transitions from and to that region, as well as reduce conservativeness by applying the stability conditions only in the region of interest and not the whole state space.

To progress with analysis, overapproximations of the hyperellipsoids with polytopes and hyperellipsoids will be briefly proposed. However, these outer approximations will obviously increase the conservativeness of the stability analysis, since more transitions than the ones actually taking place will be considered and the stability conditions have to be fulfilled in larger regions than necessary. The outer approximation of region $\mathcal{X}_l$ will be denoted as $\bar{\mathcal{X}}_l$ and the set of possible transitions for $\bar{\mathcal{X}}_l$ as $\bar{\Omega} \supseteq \Omega$.

Overapproximation with polytopes Given that a specific number of hyperellipsoids are intersecting in $\mathcal{X}_l$, then each one of them can be outer approximated by a rotated Bounding Box (BB) (or sometimes called Object Oriented Bounding Box). These Bounding Boxes, used commonly in Multi-Parametric Programming [99], are the hyperrectangles with the minimum volume containing all the points of the hyperellipsoid, while their hyperplanes are aligned with the semi-axes of the later. To compute one for an ellipsoid (4-14), it is necessary to apply a Singular Value Decomposition (SVD) for A. Since A is a positive definite matrix:

$$A = U \Sigma U^T,$$



(a) Intersection of two ellipsoids (gray area). **(b)** Lines in $\mathbb{R}^2$ constructing a bounding box around an ellipsis. **(c)** Intersection of two bounding boxes (gray area).

Figure 4-8: Approximation of intersection of two ellipsoids using bounding boxes.

where the columns of U are the unit vectors directing towards the principal exes of the ellipsoid and the entries of diagonal matrix Σ are equal to the inverse of the length of the corresponding principal semi-axis [100]. The bounding box will have the same center as the ellipsoid and matrices U and Σ are necessary to compute the hyperplanes enclosing the ellipsoid as shown Figure 4-8b and then derive the *H-representation* of the polytope:

$$G_l := \{x \mid \tilde{E}_i \tilde{x} \geq 0\},$$

where $\tilde{E}_i \in \mathbb{R}^{2n \times (n+1)}$, since each bounding box has $2n$ facets.

This process is repeated for all the ellipsoids activated in the interpolation regime and all the G_l descriptions are derived. As shown in Figure 4-8c for the simple two dimensional state-space, the intersection of the bounding boxes will be an outer approximation of the intersection of the ellipses:

$$\bar{\mathcal{X}}_l = \{x \in \mathbb{R}^n \mid \tilde{E}_i \tilde{x} \geq 0, \quad \forall i \in \mathcal{J}(l)\}. \tag{4-15}$$

It is shown that $\bar{\mathcal{X}}_l$ is a convex polyhedron of $|\mathcal{J}(l)| \cdot 2n$ hyperplanes[1], which should be reduced in case of redundant constraints. The above is a very simple and fast method to outer approximate regions $\mathcal{X}_j$, but the volume of the outer approximation can be significantly larger. However, for reasons that will be further explained in the next two sections the polytopic/polyhedral approximation facilitates reachability analysis and could under some circumstances be proven less conservative than the corresponding approximation with ellipses. Furthermore, variations that approximate the hyperellipsoid with more complex polytopes could be proposed and depending on the number of hyperplanes of the approximation the computation cost may be increased significantly.

Overapproximation with hyperellipsoids The problem of approximating the intersection of hyperellipsoids with a minimum volume hyperellipsoid can only be solved suboptimally with Convex Programming and such approaches can be found in [45] and references therein. One popular method for outer approximation uses the *S-procedure* to ascertain that only the intersection of the hyperellipsoids will be considered and minimizes the volume of the

[1]If $\mathcal{A}$ is a finite set, then $|\mathcal{A}|$ is its cardinality.

ellipsoids that can fulfill the above *sufficient* condition. Although this method may give a more accurate approximation of the intersection than the corresponding methods with polytopes, it may still be required to derive a polytopic approximation of this ellipsoid in order to reduce conservativeness in later analysis.

4-3-3 Piecewise Quadratic stability theorem

Once the transition matrix $\bar{\Omega}$ and outer approximations of all the regions comprising the state space are derived, the translation of the general stability conditions given in Chapter 2 into Semidefinite Programs (SDPs) under the Piecewise Quadratic Lyapunov Function parametrization can be performed.

To facilitate analysis, the assumption of linear local models for the region that includes the origin/equilibrium will be imposed with low prediction cost, while for the rest of the regions all the models will be affine, a valid hypothesis in probably all of the cases, since the local models are computed by a learning procedure. The index of the region including the origin will be set to 1 i.e. $0 \in \mathcal{X}_1$. The Piecewise Quadratic Lyapunov Function (4-11) can then be simplified as:

$$V(x) = \begin{cases} x^T P_1 x, & x \in \mathcal{X}_1, \\ \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_l \begin{bmatrix} x \\ 1 \end{bmatrix} = \tilde{x}^T \tilde{P}_l \tilde{x}, & x \in \mathcal{X}_l,\ l \neq 1 \end{cases}$$

where $\tilde{P}_l \in \mathbb{R}^{(n+1)\times(n+1)}$ and $P_1 \in \mathbb{R}^{n\times n}$. Furthermore, define $\tilde{P}_1$ as: $\tilde{P}_1 := \begin{bmatrix} P_1 & \mathbf{0_n} \\ \mathbf{0_n^T} & 0 \end{bmatrix}$.

For the next theorem, which proposes a set of LMIs as stability conditions, the polyhedral overapproximations of regions $\mathcal{X}_l$ are used to study the stability analysis. Furthermore, each such overapproximation after removing redundant constraints from description (4-15) is given as:

$$\bar{\mathcal{X}}_l = \{x \in \mathbb{R}^n \mid \tilde{E}_l \tilde{x} \geq 0\}.$$

For the region including the origin, $\epsilon_1 = 0$ and

$$\bar{\mathcal{X}}_1 = \{x \in \mathbb{R}^n \mid E_1 x \geq 0\}.$$

Furthermore, the next Lemma, whose Proof can be found in Appendix A, will be useful for the Piecewise Quadratic stability theorem:

Lemma 1. *Let every x in a region $\mathcal{D}$ satisfying:*

$$\begin{aligned} x^T P_1 x &> 0, \\ x^T Q x &> 0, \\ x^T (A^T P_1 A - P_2 + Q) x &< 0, \\ x^T (B^T P_1 B - P_2 + Q) x &< 0 \end{aligned}$$

where A, B are matrices with appropriate dimensions, then:

$$x^T (A^T P_1 B + B^T P_1 A - 2P_2 + Q) x < 0, \quad \forall x \in \mathcal{D}$$

Note in the previous lemma that it is not required for the matrices in the premise to be positive-definite and negative definite, but the positiveness is necessary only for $x \in \mathcal{D}$. Now everything is set up for the theorem of Piecewise Quadratic Stability.

Theorem 4.3. *Let $x_\mathrm{e} = 0$ be an equilibrium of (4-4), regions $\mathcal{X}_l$, their outer approximations $\bar{\mathcal{X}}_l$ and $\bar{\Omega}$ the set of possible transitions between the later. If there exist symmetric matrices $P_1, \tilde{P}_l, Y_1, Y_l, W_{11}, W_{lj}, U_{11i}, U_{jli}$ and possibly full matrices Q_{11}, Q_{lj} of appropriate dimensions such that matrices $Y_1, Y_l, W_{11}, W_{lj}, U_{11i}, U_{lji}$ have non-negative entries and the following LMIs are fulfilled:*

$$P_1 - E_1^T Y_1 E_1 > 0, \tag{4-16a}$$
$$\tilde{P}_l - \tilde{E}_l^T Y_l \tilde{E}_l > 0, \qquad l \neq 1 \tag{4-16b}$$
$$Q_{11} - E_1^T W_{11} E_1 > 0, \tag{4-16c}$$
$$Q_{lj} - \tilde{E}_l^T W_{lj} \tilde{E}_l > 0, \qquad (l,j) \in \bar{\Omega} \setminus \{(1,1)\} \tag{4-16d}$$
$$A_i^T P_1 A_i - P_1 + Q_{11} + E_1^T U_{11i} E_1 < 0, \qquad i \in \mathcal{J}(1) \tag{4-16e}$$
$$\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_l + Q_{lj} + \tilde{E}_l^T U_{lji} \tilde{E}_l < 0, \qquad (l,j) \in \bar{\Omega} \setminus \{(1,1)\}, i \in \mathcal{J}(l) \tag{4-16f}$$

then x_e is (globally) exponentially stable

Proof. It will be proved that if LMIs (4-16) are true and matrices $Y_1, Y_l, W_{11}, W_{lj}, U_{11i}, U_{jli}$ have only non-negative entries, then conditions (2-3) of Theorem 2.2 are true. It is obvious from the parametrization of PQLF that $V(x_\mathrm{e}) = 0$ and the first condition is fulfilled. Proof of condition (2-3b) is quite easy, but rather long and less important compared to the rest of the proof. Thus, this part is moved to Appendix A.

For condition (2-3c), ΔV will be evaluated for all x using (4-12a). If a transition from region $\mathcal{X}_l$ to $\mathcal{X}_j$ with $(l,j) \in \bar{\Omega}$ is examined at time step k, then two distinct cases should be considered:

1. $l \neq 1$ **or** $j \neq 1$
2. $l = 1$ **and** $j = 1$

For the first case: (x implies $x(k)$)

$$\begin{aligned}
\Delta V(x) &= f_\mathrm{lwl}^T(x) f_\mathrm{lwl}(x) - x^T P x \\
&= \Big(\sum_{i \in \mathcal{J}(l)}^{\mathcal{N}} h_i(x) \tilde{A}_i \tilde{x} \Big)^T \tilde{P}_j \Big(\sum_{i \in \mathcal{J}(l)}^{\mathcal{N}} h_i(x) \tilde{A}_i \tilde{x} \Big) - \tilde{x}^T \tilde{P}_l \tilde{x} \\
&= \tilde{x}^T \left(\sum_{i \in \mathcal{J}(l)} h_i^2(x) \Big[\tilde{A}_i{}^T \tilde{P}_j \tilde{A}_i - \tilde{P}_l \Big] \right) \tilde{x} \\
&+ \tilde{x}^T \left(\sum_{i<m, i,m \in \mathcal{J}(l)} 2 h_i(x) h_m(x) \Big[\tilde{A}_i^T \tilde{P}_j \tilde{A}_m + \tilde{A}_m^T \tilde{P}_j \tilde{A}_i - 2\tilde{P}_l \Big] \right) \tilde{x}
\end{aligned}$$

From properties of normalized weighting function the following identity is true:

$$\sum_i h_i^2(x) + \sum_{i<m} 2 h_i(x) h_m(x) = 1 \qquad \forall x \in \mathcal{X}_l, i,m \in \mathcal{J}(l)$$

If Q_{lj} is a matrix corresponding to the (l, j) transition:

$$\begin{aligned}\Delta V(x) &= \tilde{x}^T \left(\sum_{i \in \mathcal{J}(l)} h_i^2(x) \left[\tilde{A}_i^{\,T} \tilde{P}_j \tilde{A}_i - \tilde{P}_l + Q_{lj} \right] \right) \tilde{x} - \tilde{x}^T Q_{lj} \tilde{x} \\ &+ \tilde{x}^T \left(\sum_{i<m, i,m \in \mathcal{J}(l)} 2h_i(x) h_m(x) \left[\tilde{A}_i^T \tilde{P}_j \tilde{A}_m + \tilde{A}_m^T \tilde{P}_j \tilde{A}_i - 2\tilde{P}_l + Q_{lj} \right] \right) \tilde{x}\end{aligned} \tag{4-17}$$

Since LMIs (4-16b), (4-16d) and (4-16f) are fulfilled, then:

$$\begin{aligned} &\tilde{x}^T \tilde{P}_l \tilde{x} > 0, \quad \forall x \in \bar{\mathcal{X}}_l, \qquad \tilde{x}^T (\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_l + Q_{lj}) \tilde{x}^T, \quad \forall x \in \bar{\mathcal{X}}_l, i \in \bar{\Omega}, \\ &\tilde{x}^T Q_{lj} \tilde{x} > 0, \quad \forall x \in \bar{\mathcal{X}}_l, \qquad \tilde{x}^T (\tilde{A}_m^T \tilde{P}_j \tilde{A}_m - \tilde{P}_l + Q_{lj}) \tilde{x}^T, \quad \forall x \in \bar{\mathcal{X}}_l, m \in \bar{\Omega}. \end{aligned}$$

and from Lemma 1 for $\mathcal{D} = \bar{\mathcal{X}}_l$ it follows that:

$$\tilde{x}^T \left[\tilde{A}_i^T \tilde{P}_j \tilde{A}_m + \tilde{A}_m^T \tilde{P}_j \tilde{A}_i - 2\tilde{P}_l + Q_{lj} \right] \tilde{x} < 0 \qquad i, m \in \mathcal{J}(l)$$

From (4-17) we get then that:

$$\Delta V(x) \leq \tilde{x}^T \left(\sum_{i \in \mathcal{J}(l)} h_i^2(x) \left[\tilde{A}_i^{\,T} \tilde{P}_j \tilde{A}_i - \tilde{P}_l + Q_{lj} \right] \right) \tilde{x}$$

Moreover, from (4-16f) it is obvious that there exists a constant $c_{lji} > 0$ such that:

$$\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_l + \tilde{E}_l^T U_{lji} \tilde{E}_l + c_{lji} I < 0$$

and thus there exists a a constant α_3 such that:

$$\Delta V(x) \leq -\alpha_3 ||x||_2^2$$

If we follow the exact same procedure now for the transition $(1, 1) \in \bar{\Omega}$ and use LMIs (4-16a), (4-16c) and (4-16e), the same result for $\Delta V(x)$ will be reached and therefore condition (2-3c) is fulfilled and the equilibrium is (globally) exponentially stable □

Remarks on Theorem 4.3

Theorem 4.3 provides a set of LMI conditions in terms of the parameters of the LWL model, which can be checked efficiently using computational tools of Semidefinite Programming. If a feasible solution can be computed, then stability is guaranteed, but due to the nature of Lyapunov stability theory these conditions are only sufficient and nothing can be said if no solution can be found.

To derive the same stability conditions only using this time ellipsoidal outer approximations of the regions $\mathcal{X}_l$ instead of polytopic, it is enough to replace the last parts of each LMI with the product of a non-negative variable and a matrix S defining the hyperellipsoid as:

$$\mathcal{E} = \{ x \in \mathbb{R}^n \mid \tilde{x}^T S \tilde{x} \geq 0 \}$$

For example, the terms $-\tilde{E}_l^T Y_l \tilde{E}_l$ in 4-16b should be replaced by $-\tau_l S_l$, where $\tau_l \in \mathbb{R}_{\geq 0}$ is an optimization variable and S_l can be computed using Table 4-1.

Although the ellipsoidal outer approximation of the regions may be naturally available or far tighter than the polytopic for the studied LWL models as discussed in Subsection 4-3-2, the latter is considered a better solution in this case, due to the freedom offered by the large number of search variables [70]. If the matrix $\tilde{E}_l$ has r rows, then the corresponding polyhedral relaxation has $(r-1)(r-2)/2$ free parameters, while for the polytopic ones only one free parameter is imposed. This flexibility, however, comes at a computational cost and will make the optimization problem more laborious for a larger state dimensionality and number of local models.

Different approaches and results with the above partition have been proposed also for different classes of models such as Piecewise Affine (PWA) systems [62, 65, 93] and T-S fuzzy systems [88, 94, 101]. In the later studies for T-S fuzzy systems, some similarities can be noted with the above LMIs, with some very crucial differences: LMIs (4-16c), (4-16d) are missing along with the inclusion of Q_{lj}.

In the previous section, it was shown that the main obstacle to prove stability with CQLF was the existence of local models with their local equilibrium in their activation region. It can be proven in a similar manner as in Corollary 4.2 that LMIs based on PQLF like the ones proposed in [88, 94, 101] will face the same problem in these cases and although the parametrization of the Lyapunov function became more complex, no significant gain was noted with respect to that issue.

Nevertheless, in Theorem 4.3 the additional LMIs are attempting to counteract this problem by increasing the flexibility. Assume that the ith local model is activated in region $\mathcal{X}_l$ and its equilibrium x_e^i is included in this region. If Q_{lj} is removed from LMIs (4-16f) along with the whole (4-16c), then it will be noted that the former LMI cannot be fulfilled for any $\tilde{P}_l$ satisfying (4-16b). Insertion of Q_{lj} intends to "alter" $\tilde{A}_i$ in these LMIs such that its equilibrium is moved away from region $\mathcal{X}_l$.

However, many local models may have already their equilibrium outside their region, and addition of Q_{lj} terms and LMIs (4-16d) will be redundant. Since Q_{lj} matrices are composed of $(n+1)n/2$ free parameters and refer each to a single transition, their removal could have significant computational gains in case of many models and larger state dimensionality. It is suggested, then, that if there does not exist $i \in \mathcal{J}(l)$ such $x_e^i \in \mathcal{X}_i$, one should consider for that region $\mathcal{X}_l$ the following LMIs instead of (4-16f):

$$\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_l + \tilde{E}_l^T U_{lji} \tilde{E}_l < 0, \qquad (l,j) \in \bar{\Omega} \setminus \{(1,1)\}, i \in \mathcal{J}(l) \tag{4-18}$$

while removing entirely LMIs (4-16d) from the optimization problem.

4-4 Experimental evaluation

To evaluate the performance of Theorem 4.3, especially with respect to conservativeness and computational cost, two simple examples will be evaluated in simulation in the next couple of subsections. The first example is the 1-dimensional sinusoid function, which has been used already throughout this chapter as an illustration tool for multiple concepts relative to conservativeness. The second example, usually going under the name of *inverted pendulum*, operates in a 2-dimensional state space and will pose a greater challenge for the approach. The final optimization problems that resulted from the application of the previous theory were solved using YALMIP toolbox, SeDuMi and MOSEK solvers. For more information, the reader is referred to Appendix B.

4-4-1 Sinusoid

As shown in Subsection 4-2-3, the very simple example of a sinusoid function posed a great challenge for the stability analysis of RFWR representations due to the continuous overlaps of their local models. The same example is now assessed with the new set of conditions proposed above, by training once with random initial conditions in the interval $(-\pi/2, \pi/2)$. Although the number of local models is $M = 13$, the number of separate regions was $\mathcal{N} = 25$. Therefore, there are 25 unique combinations of local models covering the 1-dimensional state-space.

After the reachability analysis, 39 possible transitions between the regions were found using the algorithm proposed in [98]. When the state-space was sampled randomly thousands of times for the same interval, the same 38 transitions were identified (out of the 39) and only one was not possible to be computed. Obviously in this case where 1-dimensional state-space is considered, brute force is numerically a good alternative, but this does not apply for systems with larger dimensionality. Nevertheless, no significant conservativeness was induced by the reachability analysis algorithm.

Firstly, the LMI problem as described in Theorem 4.3 was posed for the 25 regions by considering for every possible transition $(l, j) \in \Omega$, a corresponding matrix Q_{lj}. As a result, 285 LMIs were formed and the optimization was solved relatively fast[2]. It was proved (for the first time) that the learned dynamics are globally exponentially stable.

Then, some of the LMIs (4-16c), (4-16d) were removed and LMIs (4-16f) were replaced accordingly by LMIs (4-18) as described in the previous section. Consequently, the "reduced" optimization problem was formed with 218 constraints and the same stability result was derived, proving the exponential stability of the sinusoid dynamics.

4-4-2 Inverted pendulum

Theorem 4.3 already proved for the previous example that it can provide a less conservative result, which takes into consideration the high degree of overlapping local models present in the dynamical representations of LWL methods. However, it is essential to evaluate the performance of the approach in more examples. Such an example is the inverted pendulum [102], whose state-space is 2-dimensional and can be seen as a simplified case of the cart-pole system studied back in Section 3-5. Instead of stabilizing the mass at the end of the link through actuation of the cart, the pendulum is controlled by a torque directly applied at the fixed end of the link.

Since the upright equilibrium is by definition unstable, a Linear Quadratic Regulator was designed for the linearized dynamics around this point. Now it is possible to study the closed-loop dynamics using the proposed stability conditions. Inspired by relevant applications of

[2]The optimization problem required less than 0.1 seconds to be solved.

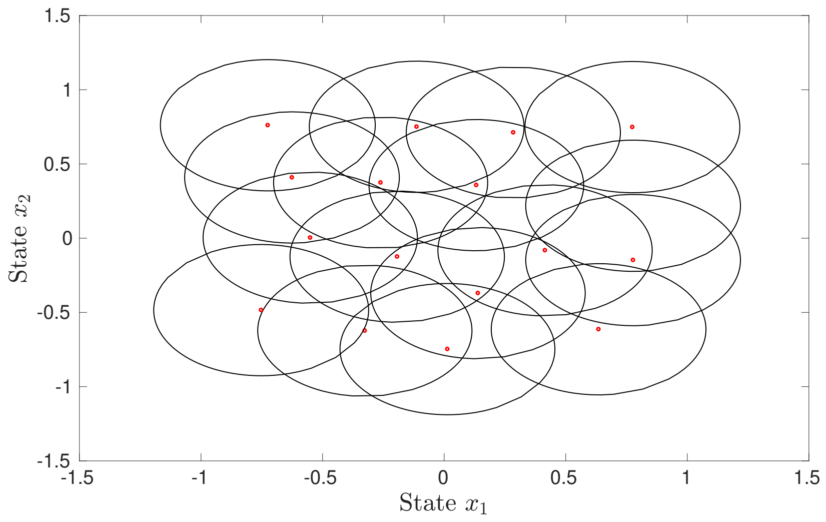


Figure 4-9: Kernels for the approximation of the inverted pendulum dynamics.

reinforcement learning for this numerical example, two simple scenarios were devised to assess the conservativeness:

1. Control with unlimited torque,

2. Control with limited torque.

For the first scenario, there are no input control restrictions and the closed-loop dynamics are approximated in a relatively small subset of the state-space. This subset is selected such that it is contained in the Region of Attraction of the LQR for the **learned dynamics**. Thus, it is expected that the stability analysis based on Theorem 4.3 should be able to prove that the system is globally stable. In Figure 4-9, the 17 Bisquare kernels derived from the learning procedure are shown. The number of regions rises significantly from the previous example i.e. $\mathcal{N} = 160$ and the number of LMIs composing this optimization problem was variating around 5000 (for the less conservative, but more computationally demanding case). Finally, it was possible to find a solution that was guaranteeing that all LMIs in (4-16) are fulfilled.

To demonstrate that the given conditions are not always proving stability by definition, the second scenario forces the learning method to approximate some globally unstable dynamics in the sense that there exist points that do not converge to the upward equilibrium, but nevertheless activate some kernels. To accomplish that, a limit on the applied torque is imposed and the subset, in which the dynamics are randomly sampled, is significantly enlarged. For this scenario, approximately 60 kernels and 750 regions were assigned (depending the trial), but it was impossible to compute a solution to the corresponding optimization problem. Nevertheless, this result agrees with the expectations for the limited torque scenario, for which the RoA is a relatively small subset of the considered state-space.

4-5 Discussion

The analysis performed in this chapter intended to determine a less conservative set of conditions in the form of LMIs that can decide reliably whether the learned dynamics are globally stable. Starting from the most common Lyapunov function parametrization i.e. Common Quadratic Lyapunov Function, it was firstly noted that the affine nature of local models requires special treatment and the *S-procedure* needs always to be employed, even for a single affine local model. The S-procedure managed to reduce conservativeness, but the new conditions proved to be very conservative in the case of local equilibria located inside the corresponding activation regions, a case very commonly encountered in practice.

To the best of the author's knowledge, this is the first time that such a thorough identification of factors affecting the conservativeness has been attempted for LMIs and no similar remarks exist in literature. Although the CQLF proved to be a poor choice for LWL, it has been by far the most popular choice for similar nonlinear representations, where overlap between local models occurs like T-S fuzzy systems. It can be easily deduced that the same conclusions for this parametrization apply also for these systems.

The high degree of overlapping local models has led to consider more complex Lyapunov functions. The PQLF, also studied in other works, was then selected and a novel set of LMI conditions was determined in a principled manner based on this parametrization. These conditions impose greater computational cost for the corresponding optimization problem, since many more regions need to be accounted along with all the transitions in between them. In the numerical examples, this significant increase was noted, but the optimization was solved, nevertheless, relatively fast.

The proposed approach based on PQLF required over-approximations of the exact regions, inducing this way conservativeness in both the reachability analysis and the convex program (4-16). The degree of added conservativeness depends of course on the roughness of the approximation. However, it did not played a noticeable role for the numerical examples and it was still possible to derive a reliable result.

The stability conditions proposed in Theorem 4.3 performed well for the low-dimensional numerical examples examined in this work, but it is certain that they will suffer from the *curse of dimensionality*, as many other similar approaches. As mentioned also in Subsection 4-1-2, the chosen underlying learning algorithm i.e. Receptive Field Weighted Regression is already known to be susceptible on these kind of problems and the number of local models will increase exponentially as the dimension of the state-space increases. This will lead to more regions, more LMIs and larger computational cost and perhaps intractability for very high-dimensional systems. The number of LMIs will grow exponentially, but the numerical complexity will not increase accordingly due to the bounding box over-approximations. More precisely, for every dimension added to the state-space, only two linear inequalities are added to the corresponding H-representation of the bounding-box. This cost seems to be affordable taking into account the numerical efficiency of state-of-the-art solvers. However, the difference of the real partitions and their bounding boxes will increase significantly as dimensionality grows. Consequently, far more conservative optimization problems will have to be solved.

Finally, it should be remarked that the stability conditions proposed were always global in the sense that global properties were examined. The RFWR is composed of a finite number of kernels, which always have finite support (remember that for Gaussian kernels parts far away from the center are cut off in training and prediction). Consequently, there is an open subset of the state-space that is not covered by a kernel (see Figure 4-9), in contrast with DNNs of the previous chapter, where dynamical models were assigned for the entire state-space. Therefore, the global conditions are in a sense local, since the representation does not spread to $\mathbb{R}^n$.

Chapter 5

Conclusion

The main contribution of this study was the establishment of novel frameworks to perform stability analysis for two challenging learning control methods. In the following, the outcome of this work is summarized and a wide outlook for further research directions is presented.

5-1 Contributions

In Chapter 3, a complete framework for stability analysis of the most common class of Deep Neural Networks (DNNs) was developed. Due to their ability to approximate a very large variety of complex and high-dimensional mappings, ReLU DNNs can be of great use for many control applications. This is the first result in literature for stability of these dynamical models and offers a principled numerical approach to determine many of their important properties. Therefore, fundamental insight into the approximated dynamics can be gained for this complicated class of learning representations.

The proposed framework for stability analysis relies on a fundamental hypothesis; DNN dynamic models are fully equivalent to Piecewise Affine (PWA) systems. This connection was proven for the first time and allowed to adjust results from the powerful theory of PWA systems and determine stability characteristics such as global and local stability, equilibria , regions of attraction for DNNs of arbitrary size. It was further shown that given a relatively accurate approximation, the dynamical properties of ReLU DNNs were deduced reliably without conservativeness. Furthermore, this approach could be employed to assess the accuracy of the learning representations, if some rough knowledge of the corresponding real properties already exists.

Stability analysis for Locally Weighted Learning (LWL) methods was examined closely in Chapter 4, where a set of stability conditions was pursued to ensure a satisfactory trade-off between reduced conservativeness and reasonable computational cost. Because of their incremental nature and low computation cost, LWL methods enable online learning, which is an important quality for learning control. Despite their important advantages, there have been no other studies on stability analysis for any method of this class or even for methods that are fully equivalent. Therefore, careful analysis tailored to these dynamical representations was required in order to derive a set of non-conservative conditions.

Starting from the most common parametrizations of Lyapunov functions and increasing gradually the complexity, an investigation with respect to the conservativeness was performed. In every state of this analysis, it is thoroughly explained why the corresponding conditions are insufficient for a successful stability analysis, providing novel theorems on the nature of LMI conditions. These results could be used for a variety of LMI-based methods such as T-S fuzzy models in order to find the set of conditions with the optimal trade-off between conservativeness and computation cost. Despite the increased complexity of the representation of the LWL method, it was possible to derive such a set of LMIs that guarantees to reliably deduce stability, taking into consideration the interaction between overlapping local models, while remaining numerically tractable.

5-2 Future work

The frameworks for stability analysis of DNNs and LWL methods presented in the previous chapters provided fundamental results, upon which several future works can be built. For example, due to the connection between PWA systems and ReLU DNN dynamics proved in Chapter 3, several results of the PWA theory could be leveraged to study controllability or observability for ReLU DNNs. In what follows, the confidence that this study could potentially be the starting point for many interesting directions is justified by both high-level and precise recommendations for future research.

Extension to higher dimensions As remarked for both learning methods, the curse of dimensionality poses several challenges for the corresponding frameworks. Computing positively invariant subsets and regions of attraction for high dimensional state-spaces is an open problem for many known control representations, as well as PWA systems. Since stability analysis of ReLU DNNs relies on the theory of the latter, it was expected they will suffer from the same issues. Indeed, it was shown that when systems with higher dimensions were examined, the main computational burden was devoted to the derivation of a maximal positively invariant subset. Nevertheless, there are relevant studies trying to address this topic for high dimensional controlled positively invariant sets of linear dynamics with constraints on the input and the states [103].

The stability conditions proposed for the LWL representation under investigation, on the other hand, are heavily affected by the number of local models, since RFWR is suffering from the curse of dimensionality. Therefore, the number of kernels increases exponentially as the dimensionality grows and as a result the computation cost to solve the semidefinite program may become very large. A possible answer could be the adjustment of the LMIs proposed to the need of the Locally Weighted Projection Regression method. The latter is popular for its dimensionality reduction properties and performs better especially when redundant or irrelevant dimensions exist. Nevertheless, addressing scalability to higher-dimensional problems should be the main future research direction for both of these learning control methods in order to find more practical applications for challenging and high-dimensional systems like humainoid robots.

Robust stability analysis Since the proposed frameworks were focused on the stability analysis of the **learned** representations, questions about the implications on the **real** dynamics naturally arise. The exact conditions of this study can be directly applied for the real dynamics only under the assumption that they are exactly approximated, which typically is not the case. Unfortunately, for real applications uncertainty or a small prediction error are always present during learning. Stability analysis of the learned models is the important first step, but the error margins naturally available from training could be used via robust analysis to study the same stability properties for the real dynamics.

For many representations similar to the ones studied in this work like PWA and T-S fuzzy systems, it is very common to simply extend the conditions for nominal stability to robust stability (see e.g. [88, 104]). Furthermore, in most cases these conditions for robust stability are given in the form of LMIs, keeping the corresponding optimization problem tractable with slightly increased computational cost. Consequently, it seems feasible to expand the two proposed frameworks in order to consider stability of real dynamics. These expansions could rely to a large extend on the results presented in this study.

Controller synthesis Although knowing the exact dynamic properties of the underlying representations could be beneficial for many control problems, design of appropriate control policies to ensure stability, and thus the safety of the system, remains a very important goal for learning control methods. In this study, the dynamics that were investigated were always autonomous in the sense that no control input was considered.

Nevertheless, there are numerous studies in literature for PWA systems and T-S fuzzy systems that could be provide insight into controller synthesis for ReLU DNNs and LWL methods respectively (see e.g. [88, 90, 105]). In some of them, controller synthesis is performed through LMI problems, but the case of non-convex Bilinear Matrix Inequality problems is also very common, especially for affine local models. Future research on controller design with stability guarantees for these two learning methods should provide structured methods in the form of optimization problems, while maintaining the favorable attributes of convex programming.

Computational efficiency It was remarked in several points of this work that there is a strong correlation between the number of local models and the number of LMIs, which obviously affects the numerical efficiency of the approaches. Although the convex programs remained still easy to solve for the examples presented (even with a relatively old solver), it is expected that more complex dynamics will require more time for analysis, especially if it is desired to study stability in an online fashion with LWL. Many recommendations on how to improve computational efficiency through adjustment of the LMIs have been already made in the last sections of the corresponding chapters, but more high-level ideas should be explored.

Except for an algorithm for faster computation of PI sets, the over-approximations applied in several stages of the frameworks affect largely the computational cost for systems with low and high dimensionality. It is always possible to determine a bounding box as an over-approximation, but this will certainly have a negative effect in conservativeness. In order to reduce complexity of the polyhedra i.e. their number of linear inequalities, an algorithm should be developed such that polytopes with minimum difference from the exact partition are computed, while maintaining a maximum complexity limit as indicated in [106]. Finally, additional insights on the LMIs and what they represent as done in Chapter 4 can facilitate the construction of an appropriate convex program with minimum numerical cost.

Appendix A

Proofs

In an attempt to keep the Thesis as compact as possible all the proofs or parts of the proofs that were omitted in the main part are included in this Appendix.

Theorem 4.1

Proof. Assuming that the requirements of the Theorem are true, it will be proven that Lyapunov function (4-7) fulfills all four conditions (2-2) of Theorem 2.1.

Starting from the first condition, the value of Lyapunov function (4-7) should be equal to 0 in the equilibrium. Obviously, this is true since $x_{\mathrm{e}} = 0$:

$$V(x_{\mathrm{e}}) = x_e^T P x_{\mathrm{e}} = 0$$

If matrix P is positive definite, then:

$$P > 0 \iff x^T(k)Px(k) > 0, \quad \forall x \neq 0$$

and the condition $V(x(k)) > 0$, $\forall x \in \mathbb{R}^n \setminus x_{\mathrm{e}}$ is fulfilled. Furthermore, it is obvious due to the parametrization as a CQLF and (2-2b) that $V(x(k))$ is also radially unbounded i.e. $V(x) \to \infty$ as $||x|| \to \infty$ and condition (2-2c) is guaranteed.

To check whether the condition (2-2d) is true some manipulation on $\Delta V(x(k))$ is required: (to ease up notation the time index k will be dropped)

$$\begin{aligned}\Delta V(x) &= f_{lwl}^T(x) P f_{lwl}(x) - x^T P x = \\ &= \left\{ \sum_{i=1}^{M} h_i(x(k)) \Big[A_i x(k) + b_i \Big] \right\}^T P \left\{ \sum_{i=1}^{M} h_i(x(k)) \Big[A_i x(k) + b_i \Big] \right\} - x^T P x\end{aligned}$$

From well-known relation $\sum_{i=1}^{M} h_i^2(x) + \sum_{i<j}^{M} 2h_i(x)h_j(x) = 1$, $\Delta V(x)$ becomes:

$$\begin{aligned}\Delta V(x) = \sum_{i}^{M} h_i^2(x) &\Big[(A_ix+b_i)^T P(A_ix+b_i) - x^T Px\Big] + \\ &+ \sum_{i<j}^{M} h_i(x)h_j(x) \Big[(A_ix+b_i)^T P(A_jx+b_j) - x^T Px + \\ &+ (A_jx+b_j)^T P(A_ix+b_i) - x^T Px\Big]\end{aligned}$$

The large term in the bracket of the second sum can be further analyzed as follows:

$$\begin{aligned}&(A_ix+b_i)^T P(A_jx+b_j) - x^T Px + (A_jx+b_j)^T P(A_ix+b_i) - x^T Px = \\ &= -\Big[(A_ix+b_i) - (A_jx+b_j)\Big]^T P\Big[(A_ix+b_i) - (A_jx+b_j)\Big] + \\ &+ (A_ix+b_i)^T P(A_ix+b_i) - x^T Px + (A_jx+b_j)^T P(A_jx+b_j) - x^T Px\end{aligned}$$

Since $(A_ix+b_i)^T P(A_jx+b_j) - x^T Px + (A_jx+b_j)^T P(A_ix+b_i) - x^T Px \geq 0$ for every x, then:

$$\begin{aligned}\Delta V(x) \leq \sum_{i}^{M} h_i^2(x) &\Big[(A_ix+b_i)^T P(A_ix+b_i) - x^T Px\Big] + \sum_{i<j}^{M} h_i(x)h_j(x) \cdot \\ &\cdot \Big[(A_ix+b_i)^T P(A_ix+b_i) - x^T Px + (A_jx+b_j)^T P(A_jx+b_j) - x^T Px\Big]\end{aligned}$$

If $b_i = 0$ for some local models, then some terms in the sums simplify. Finally, if the LMIs (4-8) are fulfilled then:

- $A_i^T PA_i - P < 0 \iff x^T A_i^T PA_ix - x^T Px < 0 \qquad i \in \mathcal{I}_0$
- $\begin{bmatrix} A_i^T PA_i - P & A_i^T Pb_i \\ b_i^T PA_i & b_i^T Pb_i \end{bmatrix} < 0 \iff \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} A_i^T PA_i - P & A_i^T Pb_i \\ b_i^T PA_i & b_i^T Pb_i \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} < 0 \iff$
  $(A_ix+b_i)^T P(A_ix+b_i) - x^T Px < 0 \qquad i \in \mathcal{I}_1$

Thus, it is proven that $\Delta V(x) < 0$ for every x different than 0 and this concludes the proof. □

Corollary 4.1

Proof. It will be shown that LMI (4-8b) can never be fulfilled for any A_i and b_i. It can be noted that if the Schur Complement is applied on a matrix M:

$$M = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

M is negative definite if and only if $C < 0$ **and** $A - BC^{-1}B^T < 0$. Accordingly, for (4-8b)

$$\begin{bmatrix} A_i^T PA_i - P & A_i^T Pb_i \\ b_i^T PA_i & b_i^T Pb_i \end{bmatrix} < 0$$

if and only if

$$b_i^T P b_i < 0 \quad \text{and} \quad A_i^T P A_i - P - A_i^T P b_i (b_i^T P b_i)^{-1} b_i^T P A_i < 0$$

But since P is positive definite, the first part can never true and there is a conflict between the conditions of the Theorem □

Lemma 1

Proof. Starting from the expression to prove, it is shown that for every $x \in \mathcal{D}$

$$\begin{aligned} x^T(A^T P_1 B + B^T P_1 A - 2P_2 + Q)x =& x^T[-(A-B)^T P_1 (A-B)]x + \\ & x^T(A^T P_1 A + B^T P_1 B - 2P_2 + Q)x \end{aligned}$$

Since $x^T P_1 x > 0$, it follows:

$$x^T(A^T P_1 B + B^T P_1 A - 2P_2 + Q)x \leq x^T(A^T P_1 A + B^T P_1 B - 2P_2 + Q)x$$

After simple manipulations:

$$x^T(A^T P_1 B + B^T P_1 A - 2P_2 + Q)x \leq x^T(A^T P_1 A - P_2 + Q)x + x^T(B^T P_1 B - P_2 + Q)x - x^T Q x$$

Therefore, every term on the right part of the inequality is negative for $x \in \mathcal{D}$ and the proof is concluded. □

Theorem 4.3 for Piecewise Quadratic Stability

Proof. To show condition (2-3b) two well-known Lemmas of linear algebra will be used:

- $\lambda_{\min}(M)||x||_2^2 \leq x^T M x \leq \lambda_{\max}(M)||x||_2^2$ (A-1a)
- $M > 0 \iff \lambda_i(M) > 0, \forall i$ (A-1b)

Furthermore, the following proposition resulting from properties of norms will be useful:

$$||\tilde{x}||_2^2 = ||x||_2^2 + 1 \tag{A-2}$$

Firstly, to ease analysis we set:

$$M_1 = P_1 - E_1^T Y_1 E_1 > 0, \qquad \tilde{M}_l = \tilde{P}_l - \tilde{E}_l^T Y_l \tilde{E}_l > 0, \quad l \neq 1$$

Then, from (A-1a) and (A-2):

$$\begin{cases} x^T P_1 x \geq x^T M_1 x \geq \lambda_{\min}(M_1)||x||_2^2, \\ \tilde{x}^T \tilde{P}_l \tilde{x} \geq \tilde{x}^T \tilde{M}_l \tilde{x} \geq \lambda_{\min}(\tilde{M}_l)||\tilde{x}||_2^2 \geq \lambda_{\min}(\tilde{M}_l)||x||_2^2, \qquad \forall l \neq 1, \forall x \in \mathcal{X}_l \end{cases}$$

Since every eigenvalue of matrices $M_l, \tilde{M}_l$ is positive, there exists a constant $\alpha_1 = \min_l(\lambda_{\min}(M_l), \lambda_{\min}(\tilde{M}_l))$ such that:

$$V(x) = \begin{cases} x^T P_1 x \geq \alpha_1 ||x||_2^2, & x \in \mathcal{X}_1 \\ \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_i \begin{bmatrix} x \\ 1 \end{bmatrix} \geq \alpha_1 ||x||_2^2, & x \in \mathcal{X}_l,\ l \neq 1 \end{cases} \tag{A-3}$$

For $l \neq 1$:

$$V(x) \leq \lambda_{\max}(\tilde{P}_l) ||\tilde{x}||_2^2, \qquad \forall x \in \mathcal{X}_l. \tag{A-4}$$

while:

$$V(x) \leq \lambda_{\max}(P_1) ||x||_2^2, \qquad \forall x \in \mathcal{X}_1. \tag{A-5}$$

From (A-2), (A-4) becomes:

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)(||x||_2^2 + 1).$$

Therefore,

$$V(x) \leq \lambda_{\max}(\tilde{P}_l) \left(||x||_2^2 + \frac{||x||_2^2}{c_l} \right)$$

where $c_l = \min_{x \in \mathcal{X}_l} ||x||_2^2$.

$$V(x) \leq \lambda_{\max}(\tilde{P}_l) \left(\frac{c_l + 1}{c_l} \right) ||x||_2^2. \tag{A-6}$$

From inequalities (A-5) and (A-6), it is then evident that there exists α_2 such that:

$$V(x) \leq \alpha_2 ||x||_2^2, \qquad \alpha_2 > 0.$$

Therefore, condition (2-3b) is fulfilled for $\eta = 2$.

The rest of the proof can be found in the main part. $\square$

Appendix B

Software and toolboxes

Although sort mentions were made throughout the main text, the software used for experiments will be presented more thoroughly in this Appendix. Besides referring to tutorials and studies (no web-page links are provided), the precise function of all these toolboxes in the context of this work will be also stated.

YALMIP YALMIP (Yet Another LMI Parser) [69] is a free toolbox developed for MATLAB that essentially creates an interface between the user and the corresponding solver in order for the latter to solve an optimization problem. The user is able to write a number of commands in a form very familiar and compact to him (see for example the LMIs in the previous chapters) and the toolbox will rapidly identify the kind of optimization problem and select the appropriate solver.

It was by far the most used toolbox and allowed to perform rapid prototyping for most algorithms and ideas presented in this work. Due to this simplicity, it was not used only for Semidefinite Programming (SDP), which was its initial goal, but also for Linear Program and Quadratic Programming instead of searching for dedicated solutions. The Github page currently available includes a number of tutorials on how to install and familiarize with the toolbox.

Commands to check: *sdpvar, sdpsettings, optimizer*

SeDuMi solver The standard solver for SDP, coming together with the latest versions of YALMIP, is SeDuMi (Self-Dual-Minimization) [68]. It can be used only in MATLAB and Octave for a large variety of optimization problems, it is free and during this work YALMIP was always the intermediator between the user and this solver. For medium-sized LMI problems (a couple of thousands of conditions), SeDuMi performed well. However, it is not supported as closely as other solvers and for large-sized problems there are other alternatives that can solve the program faster. For information on how to install it and use in parallel with YALMIP, see the previous Github page of YALMIP and for more information about distributions and stand-alone use, there is a dedicated page.

MOSEK The next solver that was used especially for larger and more computationally intensive problems is MOSEK. In contrast with SeDuMi, it is not freely available and a full trial version for academic reasons was mainly used in this study. Nevertheless, a significant difference in optimization time was noted and it was preferred from SeDuMi as the number of LMIs and their complexity grew. This solver provides interface to a number of programming languages such as C, Java, Python and R and for MATLAB and YALMIP, a small change in the *optimizer* command was sufficient. Finally, the help of MOSEK toolbox in this work is gratefully recognized.

MPT The Multi-Parametric Toolbox (MPT) [66] is an open source, MATLAB-based toolbox that was used in multiple occasions for both Deep Neural Networks and Locally Weighted Learning. Although it has also other uses, in this study it was mainly used for computational geometry and perform numerous operations between convex polyhedra. While the MPT has been updated to a 3.0 version, only the 2.0 one have been used, since it contained some functions that could not be found in the latest version. The latest version comes together with YALMIP and there is a software dependency in between these two toolboxes. Version 2.0 has significant differences and need special procedure to be installed.

Operations between convex, closed or open polyhedra like intersections, unions, projections and Minkowski additions were easily computed numerically with the help of this toolbox. Finally, very important was the contribution of two other functions; *mpt_infsetPWA*, which performed the computation of Maximal Positively Invariant subset and *mpt_reachSets*, which performed the reachability analysis.

Commands to check: *polyhedron, polytope, mldivide, plus, union*

MATLAB Neural Network Toolbox The training of ReLU Neural Networks was accomplished in nearly all of the cases using the corresponding toolbox of MATLAB. Although the level of customization for the training procedure is far lower compared to other NN software packages, the requirements for function approximation in this study were not very demanding and it was considered more time efficient to stay with one software tool. However, for future work, where more complex ideas will be explored, several Python packages will be proven a better choice.

Commands to check: *fitnet, patternnet, poslin*

Locally Weighted Statistical Learning Software Although this piece of software does not come in a toolbox, it has also been a great aid for this work. It includes a number of functions and numerical examples for MATLAB that allow to derive LWL representations. This software is freely available in the page of Computational Learning and Motor Control (CLMC) lab of University of South California (USC) and refers to multiple LWL methods such as LWR, RFWR and LWPR. Although some parts of the code for the RFWR were taken directly, a very large part was adjusted to the needs of this study, since obviously no great customization can be offered.

Appendix C

Submitted conference paper

Parts of the work presented in the previous chapters have been submitted to Conference on Robot Learning (CoRL) 2018 in close cooperation with Dr. Sebastian Trimpe of Max-Planck Institute for Intelligent Systems in Stuttgart, Germany. More precisely, most of the results and analysis performed in Chapter 3 have been included in this submission, but numerous parts and details given in Chapter 3 had to be left out due to space limitations. There is also a *Supplementary material* section added to the end, where some important proofs of the main text were added, as well as the usual *Reference* section[1]. Since the submission of this paper to the conference, additional work on the topic has been done and Chapter 3 should be considered more thorough. No results from other chapters of the thesis are included to the paper.

[1]The page number given at the bottom of each page is referring to the submitted paper and not the thesis itself, whose page number is always written at the right or left upper corner.

Stability Analysis for Deep Neural Network Dynamics Models

Konstantinos Kokkalis [1,2]
kkokkalis@tuebingen.mpg.de

Sebastian Trimpe [1]
trimpe@is.mpg.de

[1] Intelligent Control Systems Group,
Max-Planck Institute for Intelligent Systems,
Stuttgart, Germany

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

Abstract: Neural networks (NNs) are a popular tool for learning dynamic system models from data. While such models have been shown to yield good prediction performance, one typically has no insight into the dynamic system properties that they represent, for example, stability. For an important class of NNs (those with ReLU activations), we develop a theoretical and computational framework to analyze stability properties of a given NN dynamics model. First, we establish equivalence between ReLU NN models and piecewise affine (PWA) systems. This allows us to leverage well-known tools for PWA system analysis for developing a framework to compute all equilibria of a ReLU NN model and characterize their stability property. Synthetic and real-world examples show the efficacy of the framework.

1 Introduction

Learning dynamic system representations from data is a key task in robot learning, [1, 2]. In this work, we study nonlinear dynamics models represented by (deep) neural networks (NNs). Specifically, we consider nonlinear discrete-time dynamics models

$$x(t+1) = f_{\text{NN}}(x(t)) \tag{1}$$

where t is the time index, $x(t) \in \mathbb{R}^n$ the state, and f_{NN} a NN representing the state transition dynamics. The objective of this work is to develop a theoretical and computational framework for analyzing the stability properties of (1).

We consider NNs with *rectified linear units* (ReLUs) as activations and an arbitrary number of layers. This class of NNs has become very popular, mainly due to their use for deep learning [3]. In robotics, NN dynamics (1) with ReLU activation have been used, for example, in [4]. Despite their popularity, there is a lack of understanding of what they actually represent and a number studies have been devoted the recent years to explain the success of this particular learning architecture [5, 6].

Probably the most important property for understanding and characterizing dynamic systems is *stability*. Loosely speaking, stability determines whether a dynamic system "blows up" as time progresses, or remains within some bounds. Thus, stability is critical for performance and safety. However, proving stability for general nonlinear dynamics is a difficult task. The key idea of this work is to take advantage the structure of NNs with ReLU activation. In particular, we will establish their equivalence to *piecewise affine* (PWA) systems, which is a well-known class of dynamics models and has been studied extensively in control literature [7, 8]. By leveraging powerful analysis tools for PWA systems, we develop a framework that will allow us to compute all equilibria of (1), characterize their stability, and compute approximate regions of attraction.

Contributions In detail, this paper makes the following main contributions: *(i)* establishing the equivalence between ReLU NN and PWA dynamics models; *(ii)* presenting a stability proof for ReLU NN dynamics models; *(iii)* developing a framework to compute equilibria, their characterization, and region of attraction; and *(iv)* illustration in numerical and real-world dynamic system.

Submitted to 2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland (under review).

Related Work Although NNs with one hidden layer and logistic or sigmoid activation functions have been studied extensively for modeling and control of dynamics [9, 10], the use of deep neural networks in this area has been very limited. In [11], deep neural networks with ReLUs were trained to approximate the solution to the Hamilton-Jacobi-Bellman equations, while in [12] a deep NN provided policies equivalent to PID controllers. In [13], a deep NN was trained to provide both state and control input for a whole trajectory.

Stability is an extensively studied property for recurrent NNs, which typically involve an internal state. Literature on this topic has been focused on different structures like cellular NNs [14], delayed cellular NNs [15, 16], and Hopfield NNs [17, 18]. On the other hand, literature on stability of feedforward NNs with piecewise linear activation functions has been narrow and, to the best of the authors' knowledge, only a couple of studies have been concerned with this issue. In [19, 20], stability conditions based on a piecewise affine activation function (piecewise affine perceptron (PAP)), were proposed using a piecewise quadratic Lyapunov function. Although the activation function mentioned is similar to the ReLU, more linear regimes are considered for PAP and its response tries to resemble the one of sigmoid activation functions, and thus also has a vanishing gradient as the latter.

For Gaussian processes (GPs), a different and similarly popular method for model learning, stability analysis tools have recently been published in [21, 22]. While ultimately stability analysis for nonlinear systems resorts to some Lyapunov-type argument, the concrete problem and tools are different from the ones herein.

Notation Given two vectors $u, v_i \in \mathbb{R}^n$, their j$^{\text{th}}$ entries ($j \leq n$) are denoted by u_j and v_{ij}. Given two matrices $U, V_i \in \mathbb{R}^{n \times m}$, their j$^{\text{th}}$ rows ($j \leq n$) are denoted by U_j and V_{ij}, while their $j \times k$ entries by $U_{j,k}$ and $V_{ij,k}$. $\text{diag}(z)$ is an operator that creates a square diagonal matrix with the elements of vector z on the main diagonal. The sets of strictly positive and non-negative real numbers are denoted by $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$ respectively.

2 Neural Networks with ReLUs are Piecewise Affine Systems

In this section, we show that dynamical systems described by NNs with ReLUs are piecewise affine (PWA) systems. For this, we first formally introduce both NN and PWA dynamical system representations, and then prove their equivalence.

2.1 ReLU Neural Networks

We consider nonlinear dynamics (1), where the dynamics function f_{NN} is represented by an NN with L hidden layers and ReLU activation functions $\max(0, \cdot)$. We assume an n-dimensional state $x(t) \in \mathbb{R}^n$ defined on a domain $\mathcal{D} \subseteq \mathbb{R}^n$.

Formally, we introduce $\sigma(y) = [\max(0, y_1), \max(0, y_2), \ldots, \max(0, y_n)]^{\text{T}}$ as the vector-valued ReLU function, which is evaluated for each element y_i of the vector y. Furthermore, let n_j ($j = 1, 2, ..., L$) denote the number of neurons in the j^{th} layer, $x_{j-1} \in \mathbb{R}^{n_j}$ the input-vector, and $W_j \in \mathbb{R}^{n_j \times n_{j-1}}$ and $B_j \in \mathbb{R}^{n_j}$ the corresponding weighting matrix and bias. The output of the j^{th} ReLU layer is then given by $x_j = \sigma(W_j x_{j-1} + B_j)$. In addition to L hidden layers, an affine transformation is commonly used as the output layer (indexed by $L+1$). Let $\ell_j(x_{j-1}) = W_j x_{j-1} + B_j$ denote the affine transformation of x_{j-1}. We can then formally state the considered NN as $f_{\text{NN}} = \ell_{L+1} \circ \sigma \circ \ldots \sigma \circ \ell_1$; that is, for (1),

$$x(t+1) = (\ell_{L+1} \circ \sigma \circ \ldots \sigma \circ \ell_1)(x(t)) \tag{2}$$

where $\circ$ denotes function composition, and $n_1 = n_{L+1} = n$.

Depending on the input x_{j-1}, different activation patterns will occur in layer j; that is, some neurons will be activated (input to neuron greater than zero), others will not (less than zero). In order to index all possible activation patterns in layer j, we use $i_j \in \{1, \ldots, N_j\}$ with $N_j \leq 2^{n_j}$. Therefore, for a given input $x(t)$ to the network (2), there is a specific combination of neurons activated in each layer, which is specified by $(i_1, i_2, \ldots, i_L)$.

For a shallow ($L = 1$) ReLU NN, (2) can be simplified as follows:

$$x(t+1) = W_2 \sigma(W_1 x(t) + B_1) + B_2. \tag{3}$$

2.2 Piecewise Affine Systems

Firstly defined in [7], PWA systems are determined by a finite number of affine state update equations, where each one is associated with an activation region (typically a polyhedron) in the state space. More precisely, a discrete-time PWA system is given by one of the following two equivalent forms:

$$x(t+1) = A_i x(t) + b_i \qquad \forall x(t) \in \mathcal{X}_i \subset \mathcal{D} \tag{4}$$

$$\tilde{x}(t+1) = \begin{bmatrix} A_i & b_i \\ 0 & 1 \end{bmatrix} \tilde{x}(t) = \tilde{A}_i \tilde{x}(t) \qquad \forall x(t) \in \mathcal{X}_i \subset \mathcal{D} \tag{5}$$

where $\tilde{x}(t) := \begin{bmatrix} x(t) & 1 \end{bmatrix}^{\mathrm{T}}$, $A_i \in \mathbb{R}^{n \times n}$, $b_i \in \mathbb{R}^n$, and $i \in \{1, 2, \ldots, \mathcal{N}\}$ indexes the polyhedron $\mathcal{X}_i \subset \mathcal{D}$. While each individual system (A_i, b_i) is essentially linear, the overall PWA system (5) represents a switching and thus nonlinear dynamical system. PWA can thus represent complex nonlinear systems and have been subject of many studies in control by different authors [7, 23, 24, 25, 26]. A primary concern in design and analysis of PWA systems is on stability. We emphasize that stability does *not* follow from stability of an individual system A_i, but the overall PWA dynamics and switching nature have to be considered in general [27].

In the vast majority of studies, the PWA systems are continuous across the whole domain $\mathcal{D}$, while the individual polyhedra $\mathcal{X}_i$ are convex. The latter property implies that each polyhedron can be described by a number of closed half-spaces in $\mathbb{R}^n$ (*H-representation*); that is, they can be given in the form of linear inequalities

$$\mathcal{X}_i = \{x \in \mathbb{R}^n | E_i x + \epsilon_i \geq 0\} = \{x \in \mathbb{R}^n | \tilde{E}_i \tilde{x} \geq 0\} \tag{6}$$

with $\tilde{E}_i := \begin{bmatrix} E_i & \epsilon_i \end{bmatrix}$. It is also common in PWA models to assume that the intersection of the interiors of two polyhedra is the empty set, i.e., $\operatorname{int}(\mathcal{X}_i) \cap \operatorname{int}(\mathcal{X}_j) = \emptyset$ for all $i, j \in \{1, \ldots, \mathcal{N}\}$. To characterize the dynamic behavior of the PWA system, the analysis of the possible transitions between their local regions is often useful. Since the local dynamics are time-invariant (A_i, b_i are constant), it suffices to consider the transition for one time step for each region in order to characterize all the possible transitions. This is called (one-step ahead) *reachability analysis* [8]. The set that represents all the possible transitions between regions $\mathcal{X}_i$ is given by

$$\Omega := \{(i, j) \,|\, x(t) \in \mathcal{X}_i, x(t+1) \in \mathcal{X}_j\}. \tag{7}$$

The reachability analysis (i.e., computing the set (7)) can be accomplished by solving a number of linear programs (LPs) [8]. The result of this analysis is usually represented by a square matrix $T \in \{0,1\}^{\mathcal{N} \times \mathcal{N}}$ (sometimes called *transtition matrix*), whose entries are defined as follows: $T_{i,j} = 1$ if $\exists\, x(t) \in \mathcal{X}_i,\ x(t+1) \in \mathcal{X}_j$, and $T_{i,j} = 0$ otherwise.

2.3 Equivalence Theorem

The following theorem states that the NN dynamics (2) can equivalently be represented as the PWA system (5). The theorem is a main insight of this paper and will enable the stability analysis of the NN dynamics in the following section.

Theorem 2.1. *Any dynamical system* (2) *represented by an NN with ReLU activations is equivalent to the discrete-time PWA system* (5) *with*

$$A_i = W_{L+1}\Big(\prod_{j=0}^{L-1} W_{L-j}^{i_{L-j}}\Big), \quad b_i = W_{L+1}\Big[\sum_{j=1}^{L}\Big(\prod_{k=0}^{L-j-1} W_{L-k}^{i_{L-k}}\Big)B_j^{i_j}\Big] + B_{L+1},$$

$$\tilde{E}_i = \begin{bmatrix} \operatorname{diag}(z_{1,i_1})W_1 & \operatorname{diag}(z_{1,i_1})B_1 \\ \operatorname{diag}(z_{2,i_2})W_2 W_1^{i_1} & \operatorname{diag}(z_{2,i_2})(W_2 B_1^{i_1} + B_2) \\ \vdots & \vdots \\ \operatorname{diag}(z_{L,i_L})W_L\Big(\prod_{j=1}^{L-1} W_{L-j}^{i_{L-j}}\Big) & \operatorname{diag}(z_{L,i_L})\bigg(W_L\Big[\sum_{j=1}^{L-1}\Big(\prod_{k=1}^{L-j-1} W_{L-k}^{i_{L-k}}\Big)B_j^{i_j}\Big] + B_L\bigg) \end{bmatrix}$$

and $W_j^{i_j} = [\frac{1}{2}\operatorname{diag}(\mathbf{1_{n_j}} + z_{j,i_j})W_j]$, $B_j^{i_j} = [\frac{1}{2}\operatorname{diag}(\mathbf{1_{n_j}} + z_{j,i_j})B_j$, *where* $i \in \{1, \ldots, \mathcal{N}\}$ *indexes the affine models, whose number* $\mathcal{N}$ *is upper bounded by* $\prod_{j=1}^{L} N_j$, *while its domain and image are continuous in the whole state space.*

Proof. To focus on the main ideas, we present the proof for $L = 1$ here, i.e., system (3). The general case is given in the supplementary material. For ease of notation, we replace i_1 with i.

Firstly, we show that the input space is decomposed into a finite number of convex polyhedra $\mathcal{X}_i$. It is well known from recent literature [28] that the use of piecewise affine σ activation functions in neural networks divides the original input space $\mathbb{R}^n$ in a number of linear regions, separated by hyperplanes. For the NN (3), this hyperplane arrangement is determined by the hyperplanes $H_j = \{x : W_{1j}x + B_{1j} = 0\}$, where W_{1j} and B_{1j} are the j^{th} rows of W_1 and B_1 respectively, with $j \in \{1, 2, \dots, n_1\}$. According to [29], the maximum number of regions created by an arrangement is finite.

Each linear region in the input/state space, denoted as $\mathcal{X}_i$, can be described by a unique vector of markings $z_i \in \{-1, 1\}^{n_1}$, whose entries are 1 when the corresponding neurons in that region are activated and -1 when they are not. Therefore, this linear region can be non-uniquely described by the following matrix inequality: $\mathcal{X}_i := \{x \in \mathbb{R}^n | \operatorname{diag}(z_i)(W_1 x + B_1) \geq 0\}$. This description effectively proves that each linear region is an intersection of a finite number of closed half-spaces and thus, by definition a convex polyhedron.

The next properties that needs to be proved is the existence of a unique affine update equation for each region as well as the continuity of its image, an assumption usually made for PWA systems. It can be noted that when the j^{th} neuron is not activated in a region, then the following prepositions are true: $W_{1j}x + B_{1j} \leq 0, \sigma(W_{1j}x + B_{1j}) = 0, z_{ij} = -1$. Using an alternative expression for the σ vector-valued function based on the marking vector z_i the state progression can now be given in the form: $x(t+1) = W_2[\frac{1}{2}\operatorname{diag}(\mathbf{1_{n_1}} + z_i)(W_1 x(t) + B_1)] + B_2$, where $\mathbf{1_{n_1}}$ is a vector with n_1 ones. Therefore, the state dynamics using shallow ReLU NNs can be expressed in the same form as (5) with: $A_i = \frac{1}{2}W_2 \operatorname{diag}(\mathbf{1_{n_1}} + z_i)W_1$ and $b_i = \frac{1}{2}W_2 \operatorname{diag}(\mathbf{1_{n_1}} + z_i)B_1 + B_2$. □

3 Stability Analysis for ReLU NN Dynamics

The equivalence between ReLU NNs and PWA systems, which has been established in the previous section under mild assumptions, allows for leveraging powerful analysis tools that have been developed for PWA systems and bringing them to bear for the analysis of the NN dynamics (1). These tools are typically based on *linear matrix inequality* (LMI) formulations, which lead to convex optimization problems [30] with some readily available solvers. In this work, we focus on stability, and establish a computational framework to analyze stability properties of the NN dynamics (1).

3.1 Notions of Stability

We start by introducing standard stability concepts for discrete-time dynamic systems

$$x(t+1) = f(x(t)), \quad x(0) = x_{\text{init}} \tag{8}$$

where x_{init} is the initial condition. We assume that (8) has an equilibrium, or fixed point, x_{e} (not necessarily unique), i.e., $x_{\text{e}} = f(x_{\text{e}})$, whose stability properties we seek to analyze. A common notion of stability is that of *exponential stability*:

Definition 3.1 (Global exponential stability, [31])**.** *Consider the autonomous nonlinear dynamics* (8)*. If for all $x(0) \in \mathbb{R}^n$ there exists some $\theta \in \mathbb{R}_{\geq 0}$, and $\rho \in [0, 1)$ such that: $||x(k) - x_{\text{e}}|| \leq \theta\rho^k ||x(0) - x_{\text{e}}||, \forall k \in \mathbb{N}$, then equilibrium x_{e} is* globally exponentially stable.

Exponential stability ensures that trajectories starting from an initial condition will converge exponentially quickly to an equilibrium. In Definition 3.1, stability is defined *globally*; that is, convergence must hold for every initial condition in the state space. Since global stability to a unique equilibrium is typically not present for complex nonlinear systems, we also consider the notion of *local* stability:

Definition 3.2 (Local Exponential stability, [31])**.** *Consider the autonomous nonlinear dynamics* (8)*. If for all $x(0) \in \mathcal{R} \subset \mathbb{R}^n$ there exists some $\theta \in \mathbb{R}_{\geq 0}$, and $\rho \in [0, 1)$ such that: $||x(k) - x_{\text{e}}|| \leq \theta\rho^k ||x(0) - x_{\text{e}}||, \forall k \in \mathbb{N}$, then equilibrium x_{e} is* exponentially stable *in $\mathcal{R}$.*

The set $\mathcal{R}$ is called the *region of attraction* (RoA). Obviously, the larger the RoA, the "stronger" the stability property in the sense that more initial conditions will converge to the equilibrium at hand. In the following, we shall be concerned with analyzing and computing global stability, local stability, and the region of attraction for the NN dynamics (1).

3.2 Stability Theorem

One of the most common ways to examine the stability of a nonlinear system (8) is the direct method of Lyapunov (see e.g., [32, 33]). It is based on finding a positive value function V constantly decreasing across all the possible trajectories $x(t)$ of the system.

Theorem 3.1 ([34])**.** *Let x_e be an equilibrium of (8), f be continuous in x, and denote by $V : \mathbb{R}^n \to \mathbb{R}$ a (possibly discontinuous) function of the state x. If there exists such a function V and $\alpha_1, \alpha_2, \alpha_3, \eta \in \mathbb{R}_{>0}$ such that the following conditions are satisfied: (i) $V(x_e) = 0$; (ii) $\alpha_1 ||x||^\eta \leq V(x) \leq \alpha_2 ||x||^\eta, \forall x \in \mathbb{R}^n \; \{x_e\}$; (iii) $\Delta V(x) = V(f(x)) - V(x) \leq \alpha_3 ||x||^\eta$, $\forall x \in \mathbb{R}^n \setminus \{x_e\}$, then the equilibrium x_e is* globally exponentially stable.

Finding a Lyapunov function V for a general system (8) is hard and no general method to compute it exists. However, for certain classes of dynamic systems and, in particular, for PWA systems, such methods do exist (e.g., [26, 8]). We briefly sketch the approach here and refer the interested reader to the mentioned references.

To systematically construct a Lyapunov function for PWA systems, it is necessary to impose a certain parametrization on V, such that an optimal tradeoff between complexity and expressibility is achieved. Following [8], the parametrization as piecewise quadratic Lyapunov function (PQLF) is a good choice for a number of PWA systems. A PQLF postulates a quadratic function $V_i(x) = x^\mathrm{T} \tilde{P}_i x$ with $\tilde{P}_i$ a symmetric matrix corresponding to a single region $\mathcal{X}_i$. Combined with the PWA system dynamics (4), LMI conditions in the free variables P_i can then be derived. If feasible P_i are found that satisfy the LMI conditions, this proves exponential stability of the PWA system. Searching for feasible $\tilde{P}_i$ typically is a convex optimization problem, for which computational tools are available [35, 36]. We propose to use these tools for stability analysis of the ReLU NN dynamics (2).

Assume that the ReLU NN (2) has a single equilibrium x_e. We can assume $x_\mathrm{e} = 0$ without loss of generality, because an appropriate coordinate change $x - x_e$ can always be imposed to move the equilibrium to the origin. We consider the equivalent representation of (2) as PWA system given by the dynamics (4), (5), and $\mathcal{N}$ polyhedral regions (6), which is assured by Theorem 2.1. We further assume $0 \in \mathcal{X}_1$.[1] The set of all possible transitions is Ω as in (7), which can be computed using reachability analysis as details in Sec. 2.2. With this, we can state the following theorem:

Theorem 3.2. *If there exist symmetric matrices $P_1 \in \mathbb{R}^{n \times n}, Y_1 \in \mathbb{R}^{r_1 \times r_1}; \tilde{P}_i \in \mathbb{R}^{(n+1)\times(n+1)}, Y_i \in \mathbb{R}^{r_i \times r_i}$ for all $i \in \{2, \ldots, \mathcal{N}\}$; and $U_{ij} \in \mathbb{R}^{r_i \times r_i}$ for all $(i,j) \in \Omega$ such that Y_i, U_{ij} have nonnegative entries and the following LMIs are fulfilled:*

$$P_1 - E_1^T Y_1 E_1 > 0, \qquad \tilde{P}_i - \tilde{E}_i^T Y_i \tilde{E}_i > 0 \quad \forall i \in \{2, \ldots, \mathcal{N}\} \tag{9}$$

$$A_1^T P_1 A_1 - P_1 + E_1^T U_{11} E_1 < 0, \quad \tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i + \tilde{E}_i^T U_{ij} \tilde{E}_i < 0 \quad \forall (i,j) \in \Omega \setminus \{(1,1)\} \tag{10}$$

then the equilibrium x_e of (2) is globally exponentially stable.

Proof. The proof follows along the same lines as other stability proofs for PWA system with PQLFs (e.g., [37, 25]). It is based on showing that the conditions of this theorem imply the conditions of Theorem 3.1; in particular, (9) and (10) ensure the second and third conditions in Theorem 3.1, while the first condition is satisfied by construction $V(x_\mathrm{e} = 0) = x_\mathrm{e}^\mathrm{T} P_1 x_\mathrm{e}$. A detailed proof is included in the supplementary material. □

Theorem (3.2) provides LMI conditions given in terms of the NN parameters (cf. Theorem 2.1), which can be checked with computational tools. The problem of searching for feasible solution (i.e., $P_i, \tilde{P}_i, Y_i, U_{ij}$) is convex. If a solution is found, stability for the NN dynamics (2) is guaranteed.

We remark that Theorem 3.2 provides a sufficient conditions for stability. Hence, the conditions can be conservative in some cases; that is, the criterion may fail to determine stability despite the actual dynamics being stable. This is a typical characteristic of these stability tools and, in a sense, the price one has to pay to obtain tractable stability conditions. Nonetheless, there are ways to reduce conservativeness at the expense of more involved formulations. One simple example is to consider only the subset of region $\mathcal{X}_i$ from which the transition to region $\mathcal{X}_j$ is possible instead of

[1]This is without loss of generality as long as the origin is not on the boundary of more than one region, which is a singular case that almost never occurs in practice.

the whole $\mathcal{X}_i$ and consequently replace matrix $\tilde{E}_i$ with matrix $\tilde{E}_{ij} \in \mathbb{R}^{(r_{ij})\times(n+1)}$ with $r_i \leq r_{ij} \leq r_i + r_j$ defined such that: $\mathcal{X}_{ij} := \{x(t)|x(t) \in \mathcal{X}_i, x(t+1) \in \mathcal{X}_j\} = \{x(t)|\tilde{E}_{ij}\tilde{x}(t) \geq 0\}$. This change will make the solution of the LMIs (9) and (10) more likely, but increase drastically the size of optimization matrices U_{ij} and thus the computational cost. For complex partitions and high dimensional state-spaces, where the polyhedra need exponentially more hyperplanes to be fully described, this technique may render the above optimization problem intractable. In Section 4, where the partition is simple, this procedure is preferred than the alternative, while in Section 5 the NN dynamics are far more complex it should be avoided.

3.3 Complete Framework for Stability Analysis

Theorem 3.2 is a global stability result; that is, it states conditions for a single equilibrium being globally attractive (cf. Definition 3.1). Most complex nonlinear systems (8) (and thus also their approximation as NN (1)) will not have this property. In general, there will be multiple equilibria, and we need to resort to local stability according to Definition 3.2. In this section, we leverage the result of Theorem 3.2 to propose a complete framework for analyzing stability properties of (1). This will include the computation of all equilibria, determination of their local stability properties, and computation of the region of attraction.

Computation of equilibria First, we compute all equilibria of the ReLU NN dynamics (2) represented by (4), (5), and (6). Since for x_e being an equilibrium of (1), we have $x_\mathrm{e} = f_\mathrm{NN}(x_\mathrm{e})$, all candidate equilibrium points are obtained by solving (4) with $x(t) = x(t+1) = x_\mathrm{e}$, which yields $x_e^i = (I - A_i)^{-1}b_i$ (assuming the inverse exists). Then, by examining whether $x_e^i \in \mathcal{X}_i$ for each one of them, we can identify all equilibria of (2).

Local stability Furthermore, each local update equation can be seen as an affine linearization of the dynamics and thus, use well-known theory for linearized dynamics (see e.g., [38, Cha. 2]). In particular, stability for each equilibrium[2] $x_e^i \in \mathrm{int}(\mathcal{X}_i)$ is characterized by the eigenvalues λ_j, $j = 1, \ldots, n$, of A_i:

(i) If $|\lambda_j| < 1$ for all $j \in \{1, 2, \ldots, n\}$, then x_e^i is a (locally) exponentially stable equilibrium.

(ii) If there exists $j \in \{1, 2, \ldots, n\}$ such that $|\lambda_j| > 1$, then x_e^i is an unstable equilibrium.

Region of attraction While the above tests (i) and (ii) characterize local stability of an equilibrium $x_e^i \in \mathrm{int}(\mathcal{X}_i)$, they do not make any statement about the RoA $\mathcal{R}_i$ of that equilibrium. In fact, this RoA can be arbitrarily small (for small $\mathcal{X}_i$), or large and comprise several regions (the typical case). Therefore, it is of key interest to determined the RoA. The exact computation of the true RoA $\mathcal{R}_i$ is generally a very difficult task and for most cases computing a sufficiently large estimate $\mathcal{D}_i$ satisfactory. To ease the presentation, we consider a single (locally) stable equilibrium x_e in the following and compute its approximate RoA $\mathcal{D}$. The process can be repeated for all stable equilibria.

In the case of PWA systems, simply adjusting Theorem 3.2 by examining only a smaller number of regions around the equilibrium instead of the whole state space will not be sufficient to deduce the set $\mathcal{D}$. In addition to the LMIs (9) and (10), $\mathcal{D}$ also has to be a *Positively Invariant* (PI) set under the dynamics (5). Formally, for $\mathcal{D}$ being PI this implies that if $x(k) \in \mathcal{D}$, then $f(x(k)) \in \mathcal{D}$, and this condition ensures that the state will never leave the boundaries of the set. By applying a coordinate change such that $x_e = 0$ is the new equilibrium, the following Corollary then follows from Theorem 3.2. Again without loss of generality, $x_e \in \mathcal{X}_1$, and the number of regions that compose $\mathcal{D}$ is denoted $\mathcal{N}' \leq \mathcal{N}$.

Corollary 3.1. *Let $\mathcal{D} \subset \mathbb{R}^n$ be PI, $\mathcal{X}_i$ with $i \in \{1, 2, \ldots, \mathcal{N}'\}$ be the polyhedral regions that compose $\mathcal{D}$, and Ω' denote the corresponding transitions. If there exist symmetric matrices $P_1 \in \mathbb{R}^{n\times n}, Y_1 \in \mathbb{R}^{r_1\times r_1}, \tilde{P}_i \in \mathbb{R}^{(n+1)\times(n+1)}, Y_i \in \mathbb{R}^{r_i\times r_i}$, $i = 2, \ldots, \mathcal{N}'$, and $U_{ij} \in \mathbb{R}^{r_i\times r_i}, \forall(i,j) \in \Omega'$ such that Y_i, U_{ij} have nonnegative entries and the following LMIs are fulfilled:*

$$P_1 - E_1^T Y_1 E_1 > 0, \qquad \tilde{P}_i - \tilde{E}_i^T Y_i \tilde{E}_i > 0 \quad \forall i \in \{2, \ldots, \mathcal{N}'\} \tag{11}$$

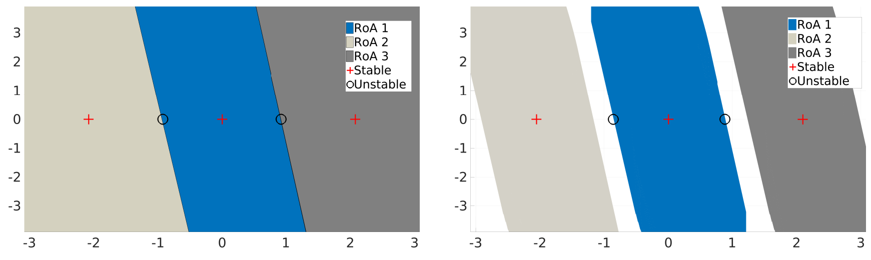
$$A_1^T P_1 A_1 - P_1 + E_1^T U_{11} E_1 < 0, \quad \tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i + \tilde{E}_i^T U_{ij} \tilde{E}_i < 0 \ \forall (i,j) \in \Omega' \setminus \{(1,1)\} \tag{12}$$

then x_e is (locally) exponentially stable, and $\mathcal{D}$ an estimate of its RoA.

[2]The equilibrium x_e on the boundary is again a singular case, typically not relevant in practice.

Table 1: Stability results computed for ReLU NN dynamics of the ship roll example.

Neural network	Equilibrium	Local stability	Volume of RoA
Ship-NN	1: $(0, 0)$	stable	13.888
	2: $(0, 2.091)$	stable	13.525
	3: $(0, 0.866)$	unstable	0
	4: $(0, -2.061)$	stable	13.6848
	5: $(0, -0.88)$	unstable	0



(a) True dynamics (b) ReLU NN dynamics

Figure 1: Equilibria and regions of attraction (RoA) for the numerical ship roll example.

The corollary can be proved the same way as Theorem 3.2, except that now only the polyhedral partitions inside $\mathcal{D}$ must be examined. Applying the corollary requires knowledge of the PI set $\mathcal{D}$, which can be computed using existing algorithms. See supplementary material for details.

4 Illustrative Numerical Example

In order to illustrate the proposed framework for stability analysis of NN dynamics, we consider the dynamics of roll motion of ships in rough seas as studied in [39]. The dynamics are described by the second-order differential equation, [39], $\ddot{\theta} + (2\mu_1\dot{\theta} + \mu_2\dot{\theta}^3) + (\omega_0^2\theta + a_1\dot{\theta}^3 + a_2\dot{\theta}^5) = 0$, where $\omega_0, a_i, \mu_i,\ i = 1, 2$ are known coefficients. A discrete-time model (8) with $n = 2$ states can be derived using standard discretization techniques [33]. This example is illustrative as it involves five equilibria; three stable ones $(\theta, \dot{\theta}) = (0, 0), (0, 2.078), (0, -2.078)$ and two unstable ones $(\theta, \dot{\theta}) = (0, 0.924), (0, -0.924)$. The equilibria and the true RoA (computed via numerical simulation) are shown in Fig. 1a. Matlab code for this example is available in the supplementary material.

For this synthetic example, we trained a NN from data randomly sampled from a rectangle in the two-dimensional state space. Because of the relatively simple dynamics, a shallow NN with 8 ReLUs in the single hidden layer was sufficient. A ReLU NN (2) approximating the real dynamics sufficiently well (by RMS test error $3 \cdot 10^{-4}$) was trained using MATLAB Neural Network toolbox with random initialization of the network weights.

We applied the framework from Sec. 3.3 to compute equilibria, their local stability property, and their RoA. The results are presented in Table 1 (including the volume of the RoA) and Fig. 1b. As can be seen, all five equilibria are identified and correctly characterized as stable/unstable. Moreover, the computed RoA are reasonable approximations of the true ones (cf. Fig. 1a and 1b).

While there is a good match of the stability properties of the true dynamics and the NN approximation in this example, this obviously depends on how well one succeeds in training the NN. We emphasize that the contribution of this work is *not* in how to learn good NN dynamics models, but rather in analyzing a given or trained NN. In fact, the results of the stability analysis can help to evaluate different NN models, for example, when some stability properties are known.

5 Case Study of a Cart-pole Experiment

After demonstrating the efficacy of the proposed framework on a 2D synthetic example in the previous section, we now present results for NN dynamics trained on real-world data from a hardware experiment. We use a standard cart-pole experiment for this study. The cart-pole system has four

Table 2: Stability results for the five top scoring NNs.

Neural network	Score	Equilibrium $(p, \theta, \dot{p}, \dot{\theta})$	Local stability	Volume of RoA
NN1	0.0223	$(0.13, 0.01, -8 \cdot 10^{-3}, 0.08)$	stable	0.032
NN2	0.0229	$(-0.55, 0.13, 0.98, -0.29)$	unstable	0
NN3	0.0232	$(-1.11, -0.02, 0.04, -0.29)$	stable	–
NN4	0.0265	$(-0.68, -0.09, 0.35, 0.14)$	stable	–
NN5	0.0279	$(1.37, 0.09, -0.21, 0.69)$	stable	0.021

states: position of the cart p, angle of the pole θ, and the corresponding linear and angular velocities. We consider the dynamics of the cart-pole stabilized around its upright equilibrium with a standard linear quadratic regulator (LQR) [40].

A data set was obtained on the cart-pole experiment by exciting the system with a suitable chirp signal (sinusoid with increasing frequency), which was applied in superposition to the control input of the LQR. Angle and position of the cart are measured with encoders, from which the velocities are computed through finite differences. All signals were low-pass filtered (noncausal) and downsampled to 100 Hz. With this data, multiple ReLU NNs (2) (with $L = 2$ hidden layers, and $n_1 = 8$, $n_2 = 6$ neurons) were trained. For training the NNs, a dynamics model in the form of $x(t+1) = x(t) + f(x(t))$ was assumed instead of (8); that is, the NN represents incremental, rather than absolute dynamics. This is beneficial especially for small sample times (and thus small increments), and common practice when training NN dynamics. The stability analysis directly extends to this case, which is a simple transformation.

A big challenge observed in these experiments was how to rank the different NNs that were obtained from different random initialization and training runs. In particular, the mean squared error (MSE) on a validation set, which is usually the criterion used to evaluate NNs, was not a good indication of the performance of a NN in approximating well the true dynamics. In order to capture long-term predictions well, we instead took the MSE after unrolling the NN dynamics for 500 steps as the criterion to rank the NNs ("Score"). For the top five NNs, we performed the stability analysis as per Sec. 3.3, whose results are given in Table 2.

As expected, in each case one equilibrium was found. While most NNs have (locally) stable equilibria, in case of NN2, the NN dynamics were found to be unstable. As remarked in Sec. 4, a useful application of the proposed stability analysis is to select suitable NN models. In this case, where from physical considerations we know that there is one stable equilibrium, one can discard NN2.

Although the number of regions $\mathcal{X}_i$ varies significantly from training trial to training trial, it was not uncommon to get more than 1000 regions through the whole state-space. For NN3 and NN4, no volume of the RoA is computed, since the computation required by the MPI algorithm became intractable. Thus, an overapproximation was derived by stopping the MPI algorithm when the number of regions grew too large as explained in Section 3. The resulting polyhedra were composed of many inequalities due partly to the high state dimensionality, which made the solution of the LMIs very costly. Again an overapproximation of the complex polyhedra with bounding boxes solved the problem and stability could be determined. For NN1 and NN5, the computations could be performed without problem. For NN1, the computed RoA is illustrated in Fig. 1 of the Supplementary material.

6 Conclusions

We presented a computation framework to analyze stability properties of (deep) neural network representing dynamic systems (1). By establishing the equivalence between ReLU NN and piecewise affine (PWA) dynamics models, we could leverage existing tools for PWA stability analysis. The resulting framework allows one to compute all equilibria, their local stability (stable/unstable), and approximate regions of attraction for a ReLU NN dynamics model. While the focus of this work is not on how to best train NN dynamics models, but rather on providing an analysis tool for a given NN, the proposed framework can be beneficial for model selection, as discussed for the cart-pole example. The presented synthetic and real-world examples demonstrate the efficacy in characterizing stability properties of NN dynamics, and thus interpreting learning results in terms of important system-theoretic properties. Addressing computational and scalability issues as partially observed in the real-world example is an important topic for future work.

References

[1] S. Schaal and C. Atkeson. Learning control in robotics. *IEEE Robotics Automation Magazine*, 17(2):20–29, June 2010.

[2] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[3] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[4] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

[5] M. Telgarsky. benefits of depth in neural networks. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[6] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

[7] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, Apr 1981. ISSN 0018-9286.

[8] P. Biswas, P. Grieder, J. Lfberg, and M. Morari. A survey on stability analysis of discrete-time piecewise affine systems. *IFAC Proceedings Volumes*, 38(1):283 – 294, 2005. ISSN 1474-6670. 16th IFAC World Congress.

[9] M. Norgaard, O. Ravn, N. Poulsen, and L. Hansen. *Neural networks for modelling and control of dynamic systems: a practitioners handbook. Advanced textbooks in control and signal processing*. Springer, Berlin, 2000.

[10] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27, 1990.

[11] C. Sánchez and D. Izzo. Real-time optimal control via deep neural networks: study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41(5):1122–1135, 2018.

[12] K. Cheon, J. Kim, M. Hamadache, and D. Lee. On replacing pid controller with deep learning controller for dc motor system. *Journal of Automation and Control Engineering Vol*, 3(6), 2015.

[13] M. Berniker and K. P. Kording. Deep networks for motor control functions. *Frontiers in computational neuroscience*, 9:32, 2015.

[14] X. Liao, Z. Wu, and J. Yu. Stability analyses of cellular neural networks with continuous time delay. *Journal of Computational and Applied Mathematics*, 143(1):29 – 47, 2002.

[15] S. Arik. An analysis of global asymptotic stability of delayed cellular neural networks. *IEEE Transactions on Neural Networks*, 13(5):1239–1242, Sep 2002. ISSN 1045-9227.

[16] H. Zhang and Z. Wang. Global asymptotic stability of delayed cellular neural networks. *IEEE Transactions on Neural Networks*, 18(3):947–950, May 2007. ISSN 1045-9227.

[17] T. Chen. Global exponential stability of delayed hopfield neural networks. *Neural Networks*, 14(8):977 – 980, 2001. ISSN 0893-6080.

[18] J. Cao. Global exponential stability of hopfield neural networks. *International Journal of Systems Science*, 32(2):233–236, 2001.

[19] C.-A. Lehalle and R. Azencott. Piecewise affine neural networks and nonlinear control: stability results. 1999.

[20] C.-A. Lehalle and R. Azencott. Basis of nonlinear control with piecewise affine neural networks. *Nonlinear theory and its applications. NOLTA*, 1998.

[21] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters. Stability of controllers for gaussian process dynamics. *Journal of Machine Learning Research*, 18(100):1–37, 2017.

[22] F. Berkenkamp and A. P. Schoellig. Safe and robust learning control with gaussian processes. In *Control Conference (ECC), 2015 European*, pages 2496–2501. IEEE, 2015.

[23] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, Oct 2000. ISSN 0018-9286.

[24] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35 – 65, 1995. ISSN 0304-3975. Hybrid Systems.

[25] M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, Apr 1998. ISSN 0018-9286.

[26] D. Mignone, G. Ferrari-Trecate, and M. Morari. Stability and stabilization of piecewise affine and hybrid systems: an lmi approach. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 1, pages 504–509 vol.1, 2000.

[27] M. S. Branicky. Stability of switched and hybrid systems. In *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, volume 4, pages 3498–3503, Dec 1994.

[28] 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*, pages 2924–2932, 2014.

[29] R. P. Stanley et al. An introduction to hyperplane arrangements. *Geometric combinatorics*, 13: 389–496, 2004.

[30] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

[31] M. Lazar, W. P. M. H. Heemels, and A. R. Teel. Lyapunov functions, stability and input-to-state stability subtleties for discrete-time discontinuous systems. *IEEE Transactions on Automatic Control*, 54(10):2421–2425, Oct 2009. ISSN 0018-9286.

[32] H. K. Khalil and J. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[33] K. J. Åström and B. Wittenmark. *Computer-controlled systems: theory and design*. Prentice-Hall, 1990.

[34] S. Sastry. *Nonlinear systems: analysis, stability, and control*, volume 10. Springer Science & Business Media, 2013.

[35] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[36] J. Lofberg. Yalmip : a toolbox for modeling and optimization in matlab. In *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, pages 284–289, Sept 2004.

[37] G. Feng. Stability analysis of piecewise discrete-time linear systems. *IEEE Transactions on Automatic Control*, 47(7):1108–1112, 2002.

[38] A. C. Luo. *Regularity and complexity in dynamical systems*. Springer, 2012.

[39] A. A. Zaher. Nonlinear control of systems with multiple equilibria and unknown sinusoidal disturbance. *Communications in Nonlinear Science and Numerical Simulation*, 12(8):1518 – 1533, 2007. ISSN 1007-5704.

[40] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Dover Publications, Mineola, New York, 2007.

Supplementary material

6.1 Full Proof for Theorem 2.1

Proof. The proof for the deep case ($L > 1$) will proceed the same way as for the shallow case by showing the partition of the input/state space in a finite number of convex polyhedra as well as the correspondence of each such polyhedron with an affine state update equation.

Given a point in the state space, only a number $m_1 \leq n_1$ of neurons in the first layer will be activated. As for the shallow neural network, a vector $z_{1,i_1} \in \{1, -1\}^{n_1}$ can describe which neurons in the first hidden layer are activated, where the index i_1 denotes one of the possible combinations of 1s and 1s for the first hidden layer. All the points in $\mathbb{R}^n$ that activate the same neurons constitute a region non-uniquely defined by either of the following inequalities:

$$\begin{aligned} \operatorname{diag}(z_{1,i_1})(W_1 x_0 + B_1) \geq 0 \\ \operatorname{diag}(z_{1,i_1})(W_1 x(t) + B_1) \geq 0. \end{aligned} \tag{13}$$

Accordingly, for the same point in the state space, only a number $m_2 \leq n_2$ of neurons in the second layer will be activated and a new vector $z_{2,i_2} \in \{1, -1\}^{n_2}$ can be assigned, denoting the activated neurons in the second hidden layer. For this case, except for the matrix inequality (13), another matrix inequality is true for the points that activate the same neurons in layers 1 and 2:

$$\begin{aligned} \operatorname{diag}(z_{2,i_2})(W_2 x_1 + B_2) \geq 0 \\ \operatorname{diag}(z_{2,i_2})[W_2(W_1^{i_1} x(t) + B_1^{i_1}) + B_2] \geq 0 \end{aligned} \tag{14}$$

where

$$W_1^{i_1} = [\frac{1}{2} \operatorname{diag}(\mathbf{1_{n_1}} + z_{1,i_1}) W_1] \qquad B_1^{i_1} = [\frac{1}{2} \operatorname{diag}(\mathbf{1_{n_1}} + z_{1,i_1}) B_1].$$

Generalizing the previous procedure for the m^th layer and defining the corresponding and vector z_{m,i_m} the following inequalities are true for that layer:

$$\begin{aligned} \operatorname{diag}(z_{m,i_m})(W_m x_{m-1} + B_m) \geq 0 \\ \operatorname{diag}(z_{m,i_m})\{W_m[\ldots(W_1^{i_1} x(t) + B_1^{i_1}) + \ldots] + B_m\} \geq 0 \end{aligned} \tag{15}$$

The last inequality can be written with respect to the state vector $x(t)$ the following way:

$$\operatorname{diag}(z_{m,i_m})\Big\{W_m\Big(\prod_{j=1}^{m-1} W_{m-j}^{i_{m-j}}\Big)x(t) + W_m\Big[\sum_{j=1}^{m-1}\Big(\prod_{k=1}^{m-j-1} W_{m-k}^{i_{m-k}}\Big)B_j^{i_j}\Big] + B_m\Big\} \geq 0 \tag{16}$$

In conclusion, a point in the state-space will always fulfill a set of L matrix inequalities:

$$\begin{aligned} \operatorname{diag}(z_{1,i_1})(W_1 x(t) + B_1) \geq 0 \\ \operatorname{diag}(z_{2,i_2})[W_2(W_1^{i_1} x(t) + B_1^{i_1}) + B_2] \geq 0 \\ \vdots \\ \operatorname{diag}(z_{L,i_L})\Big\{W_L\Big(\prod_{j=1}^{L-1} W_{L-j}^{i_{L-j}}\Big)x(t) + W_L\Big[\sum_{j=1}^{L-1}\Big(\prod_{k=1}^{L-j-1} W_{L-k}^{i_{L-k}}\Big)B_j^{i_j}\Big] + B_L\Big\} \geq 0. \end{aligned} \tag{17}$$

Matrices $W_j^{i_j}$ and vectors $B_j^{i_j}$ are given by the following relations:

$$W_j^{i_j} = [\frac{1}{2} \operatorname{diag}(\mathbf{1_{n_j}} + z_{j,i_j}) W_j] \qquad B_j^{i_j} = [\frac{1}{2} \operatorname{diag}(\mathbf{1_{n_j}} + z_{j,i_j}) B_j]. \tag{18}$$

Therefore, it can be seen that each linear region $\mathcal{X}_i$, can be non-uniquely defined by a number of $M = \sum_{j=1}^{L} n_j$ linear (affine) inequalities and uniquely by the combination of indices $(i_1, i_2, \ldots, i_L)$ (or a large marking vector $Z \in \{1, -1\}^M$). Consequently, each region is an intersection of M closed half-spaces and a convex polyhedron.

If the $(i_1, i_2, \dots, i_L)$ combination correspond to the current state vector $x(t)$, the state vector at the next time step can be formulated using relations (18) as:

$$x(t+1) = W_{L+1}\Big(\prod_{j=0}^{L-1} W_{L-j}^{i_{L-j}}\Big)x(t) + W_{L+1}\Big[\sum_{j=1}^{L}\Big(\prod_{k=0}^{L-j-1} W_{L-k}^{i_{L-k}}\Big)B_j^{i_j}\Big] + B_{L+1} \qquad \forall x \in \mathcal{X}_i. \tag{19}$$

Besides its complexity this is an affine mapping with respect to the current state value for a given set of indices $(i_1, i_2, \dots, i_L)$ i.e. for a given region $\mathcal{X}_i$. The above state update matrices and vectors linked to a specific region $\mathcal{X}_i$ will be denoted with A_i and b_i respectively as it was done for PWA systems and shallow NNs.

The continuity proof as well as the proof that the intersection of the interior of two linear regions is the empty set can be done the exact same way as it was done for the shallow case and thus can be omitted. □

6.2 Proof for Theorem 3.2

Proof. Given that LMIs (9) and (10) are true, matrices $P_1, P_i, Y_1, Y_i, U_{11}, U_{ij}$ are symmetric and matrices Y_1, Y_i, U_{11}, U_{ij} have positive entries, prove that all the conditions of Theorem (3.1) are fulfilled.

The Lyapunov function is parametrized as a Piecewise Quadratic function with the origin being included in the $\mathcal{X}_1$ region and thus given by the following relation

$$V(x) = \begin{cases} x^T P_1 x, & x \in \mathcal{X}_1, \\ \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_i \begin{bmatrix} x \\ 1 \end{bmatrix} = \tilde{x}^T \tilde{P}_i \tilde{x}, & x \in \mathcal{X}_i,\ i \neq 1 \end{cases}$$

Two well-known Lemmas of linear algebra will be used in the following proof:

1. $$\lambda_{\min}(M)||x||_2^2 \leq x^T M x \leq \lambda_{\max}(M)||x||_2^2 \tag{20}$$

2. $$M > 0 \iff \lambda_i(M) > 0, \forall i \tag{21}$$

Furthermore, the following proposition resulting from properties of norms will be useful:

$$||\tilde{x}||_2^2 = ||x||_2^2 + 1 \tag{22}$$

Firstly, LMIs (9) yield:

$$\begin{cases} M_1 = P_1 - E_1^T Y_1 E_1 > 0 \\ \tilde{M}_l = \tilde{P}_l - \tilde{E}_l^T Y_l \tilde{E}_l > 0, & \forall l \neq 1 \end{cases}$$

From (20) and (22):

$$\begin{cases} x^T P_1 x \geq x^T M_1 x \geq \lambda_{\min}(M_1)||x||_2^2, & \forall x \in \mathcal{X}_1 \\ \tilde{x}^T \tilde{P}_l \tilde{x} \geq \tilde{x}^T \tilde{M}_l \tilde{x} \geq \lambda_{\min}(\tilde{M}_l)||\tilde{x}||_2^2 \geq \lambda_{\min}(\tilde{M}_l)||x||_2^2, & \forall l \neq 1, \forall x \in \mathcal{X}_l \end{cases}$$

Since every eigenvalue of matrices $M_1, \tilde{M}_l$ is positive, there exists a constant $\alpha_1 = \min_l(\lambda_{\min}(M_1), \lambda_{\min}(\tilde{M}_l))$ such that:

$$V(x) = \begin{cases} x^T P_1 x \geq \alpha_1 ||x||_2^2 \\ \begin{bmatrix} x \\ 1 \end{bmatrix}^T \tilde{P}_i \begin{bmatrix} x \\ 1 \end{bmatrix} \geq \alpha_1 ||x||_2^2 \end{cases} \tag{23}$$

Consequently, $V(x) \geq \alpha_1 ||x||_2^2$ with $\alpha_1 > 0$. If $V(x) = x^T P_1 x,\ x \in \mathcal{X}_1$:

$$V(x) \leq \lambda_{\max}(P_1)||x||_2^2, \qquad \forall x \in \mathcal{X}_1. \tag{24}$$

If $V(x) = \tilde{x}^T \tilde{P}_1 \tilde{x},\ x \in \mathcal{X}_l$:

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)||\tilde{x}||_2^2, \qquad \forall x \in \mathcal{X}_l. \tag{25}$$

From (22), (26) becomes:

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)(||x||_2^2 + 1).$$

Therefore,

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)\Big(||x||_2^2 + \frac{||x||_2^2}{c_l}\Big)$$

where $c_l = \min_{x \in \mathcal{X}_l} ||x||_2^2$.

$$V(x) \leq \lambda_{\max}(\tilde{P}_l)\Big(\frac{c_l + 1}{c_l}\Big)||x||_2^2. \tag{26}$$

From inequalities (24) and (26), it is then evident that there exists α_2 such that:

$$V(x) \leq \alpha_2||x||_2^2, \qquad \alpha_2 > 0.$$

Therefore, property 2) is fulfilled for $\eta = 2$.

The difference of the candidate Lyapunov function between two time steps is now considered. If the state at time step t is $x(t) \in \mathcal{X}_l$ and at time step $t+1$ is $x(t+1) \in \mathcal{X}_j$, two possible cases need to be examined:

1. $(l, j) \in \Omega \backslash (1, 1)$
2. $(l, j) = (1, 1)$

For case 1):

It follows that if $\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i + \tilde{E}_i^T U_{ij} \tilde{E}_i < 0$, then there exist $\rho > 0$ such that:

$$\begin{aligned}
\Delta V &= V(x(t+1)) - V(x(t)) = \tilde{x}^T(t)[\tilde{A}_i^T \tilde{P}_j \tilde{A}_i - \tilde{P}_i]\tilde{x}(t) \\
&\leq \tilde{x}^T(t)(-\rho I - \tilde{E}_i^T U_{ij} \tilde{E}_i)\tilde{x}(t) \\
&\leq \tilde{x}^T(t)(-\rho I)\tilde{x}(t) \\
&\leq -\rho||x(t)||^2
\end{aligned}$$

For case 2):

It follows that if $A_1^T P_1 A_1 - P_1 + E_1^T U_{11} E_1 < 0$, then there exist $\rho > 0$ such that:

$$\begin{aligned}
\Delta V &= V(x(t+1)) - V(x(t)) = x^T(t)[A_1^T P_j A_1 - \tilde{P}_i]x(t) \\
&\leq x^T(t)(-\rho I - E_1^T U_{11} E_1)x(t) \\
&\leq x^T(t)(-\rho I)x(t) \\
&\leq -\rho||x(t)||^2
\end{aligned}$$

The above proves that there exists $\alpha_3 > 0$ such that:

$$\Delta V(x) \leq \alpha_3||x||^\eta$$

Therefore, property 3) is fulfilled for $\eta = 2$.

$\square$

Bibliography

[1] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[3] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[4] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 161–168, 2006.

[5] K. Fu, "Learning control systems and intelligent control systems: An intersection of artifical intelligence and automatic control," *IEEE Transactions on Automatic Control*, vol. 16, no. 1, pp. 70–72, 1971.

[6] P. Antsaklis, "Defining intelligent control," *IEEE Control Systems Magazine*, vol. 14, no. 3, pp. 1–31, 1994. Report of the Task Force on Intelligent Control.

[7] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers.* Princeton University press, 2010.

[8] K. J. Åström and B. Wittenmark, *Computer-controlled systems: theory and design.* Prentice Hall, 1990.

[9] S. Sastry, *Nonlinear systems: analysis, stability, and control.* Springer Science & Business Media, 2013.

[10] H. K. Khalil and J. Grizzle, *Nonlinear systems.* Prentice Hall, 2002.

[11] J.-J. E. Slotine and W. Li, *Applied nonlinear control.* Prentice Hall, 1991.

[12] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[13] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 2015.

[14] M. Telgarsky, “Benefits of depth in neural networks,” in *Conference on Learning Theory*, vol. 49, pp. 1517–1539, 2016.

[15] R. Vidal, J. Bruna, R. Giryes, and S. Soatto, “Mathematics of deep learning,” *CoRR*, vol. abs/1712.04741, 2017.

[16] A. M. Saxe, J. L. McClelland, and S. Ganguli, “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks,” *CoRR*, vol. abs/1312.6120, 2013.

[17] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, pp. 249–256, 2010.

[18] S. S. Haykin, *Neural networks and learning machines.* Pearson Education, 2009.

[19] L. O. Chua and L. Yang, “Cellular neural networks: applications,” *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1273–1290, 1988.

[20] X. Liao, Z. Wu, and J. Yu, “Stability analyses of cellular neural networks with continuous time delay,” *Journal of Computational and Applied Mathematics*, vol. 143, no. 1, pp. 29 – 47, 2002.

[21] S. Arik, “An analysis of global asymptotic stability of delayed cellular neural networks,” *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1239–1242, 2002.

[22] H. Zhang and Z. Wang, “Global asymptotic stability of delayed cellular neural networks,” *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 947–950, 2007.

[23] A. N. Michel, J. A. Farrell, and W. Porod, “Qualitative analysis of neural networks,” *IEEE Transactions on Circuits and Systems*, vol. 36, no. 2, pp. 229–243, 1989.

[24] T. Chen, “Global exponential stability of delayed hopfield neural networks,” *Neural Networks*, vol. 14, no. 8, pp. 977 – 980, 2001.

[25] J. Cao, “Global exponential stability of hopfield neural networks,” *International Journal of Systems Science*, vol. 32, no. 2, pp. 233–236, 2001.

[26] C. A. Lehalle and R. Azencott, “Piecewise affine neural networks and nonlinear control: stability results,” in *International Conference on Artificial Neural Networks*, vol. 2, pp. 608–612, 1999.

[27] C.-A. Lehalle and R. Azencott, “Basis of nonlinear control with piecewise affine neural networks,” *Nonlinear theory and its applications*, 1998.

[28] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters, “Stability of controllers for gaussian process dynamics,” *Journal of Machine Learning Research (JMLR)*, vol. 18, no. 1, pp. 3483–3519, 2017.

[29] F. Berkenkamp and A. P. Schoellig, “Safe and robust learning control with gaussian processes,” in *European Control Conference (ECC)*, pp. 2496–2501, 2015.

[30] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, “Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes,” in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 4661–4666, 2016.

[31] M. Norgaard, O. Ravn, N. Poulsen, and L. Hansen, *Neural networks for modelling and control of dynamic systems: a practitioner’s handbook. Advanced textbooks in control and signal processing.* Springer Science & Business Media, 2000.

[32] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical systems using neural networks,” *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[33] C. Sánchez and D. Izzo, “Real-time optimal control via deep neural networks: study on landing problems,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018.

[34] K. Cheon, J. Kim, M. Hamadache, and D. Lee, “On replacing PID controller with deep learning controller for dc motor system,” *Journal of Automation and Control Engineering*, vol. 3, no. 6, pp. 452–456, 2015.

[35] M. Berniker and K. P. Kording, “Deep networks for motor control functions,” *Frontiers in computational neuroscience*, vol. 9, pp. 32–42, 2015.

[36] Y. Li, “Deep reinforcement learning: An overview,” *CoRR*, vol. abs/1701.07274, 2017.

[37] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning for control,” in *Artificial Intelligence Review - Special issue on lazy learning*, pp. 75–113, Springer, 1997.

[38] D. Mitrovic, S. Klanke, and S. Vijayakumar, “Adaptive optimal feedback control with learned internal dynamics models,” in *From Motor Learning to Interaction Learning in Robots*, pp. 65–84, Springer, 2010.

[39] E. Todorov and W. Li, “A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems,” in *Proceedings of the American Control Conference (ACC)*, vol. 1, pp. 300–306, 2005.

[40] J. Nakanishi, J. A. Farrell, and S. Schaal, “Composite adaptive control with locally weighted statistical learning,” *Neural Networks*, vol. 18, no. 1, pp. 71–90, 2005.

[41] J.-A. Ting, S. Vijayakumar, and S. Schaal, “Locally weighted regression for control,” in *Encyclopedia of Machine Learning*, pp. 613–624, Springer, 2011.

[42] K. B. Petersen and M. S. Pedersen, “The matrix cookbook,” *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008.

[43] M. Lazar, “Model predictive control of hybrid systems: Stability and robustness,” 2006. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

[44] F. Blanchini and S. Miani, *Set-theoretic methods in control.* Springer Science & Business Media, 2008.

[45] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory.* SIAM, 1994.

[46] J. G. VanAntwerp and R. D. Braatz, "A tutorial on linear and bilinear matrix inequalities," *Journal of process control*, vol. 10, no. 4, pp. 363–385, 2000.

[47] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming.* SIAM, 1994.

[48] U. T. Jönsson, "A lecture on the S-procedure," 2001. Lecture Note at the Royal Institute of Technology, Sweden.

[49] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural network design.* Martin Hagan, 1996.

[50] S. Haykin, *Neural networks: a comprehensive foundation.* Prentice Hall, 1994.

[51] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Pearson Education, 2016.

[52] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017.

[53] T. Gerstner and M. Holtz, "Algorithms for the cell enumeration and orthant decomposition of hyperplane arrangements," 2006.

[54] R. Pascanu, G. Montúfar, and Y. Bengio, "On the number of inference regions of deep feed forward networks with piece-wise linear activations," *CoRR*, vol. abs/1312.6098, 2013.

[55] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances on Neural Information Processing Systems (NIPS)*, vol. 2, pp. 2924–2932, 2014.

[56] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 70, pp. 2847–2854, 2017.

[57] T. Serra, C. Tjandraatmadja, and S. Ramalingam, "Bounding and counting linear regions of deep neural networks," *CoRR*, vol. abs/1711.02114, 2017.

[58] M. Fischetti and J. Jo, "Deep neural networks and mixed integer linear optimization," *Constraints*, vol. 23, no. 3, pp. 296–309, 2018.

[59] E. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 346–358, 1981.

[60] A. Bemporad, G. Ferrari-Trecate, and M. Morari, "Observability and controllability of piecewise affine and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1864–1876, 2000.

[61] E. Asarin, O. Maler, and A. Pnueli, "Reachability analysis of dynamical systems having piecewise-constant derivatives," *Theoretical Computer Science*, vol. 138, no. 1, pp. 35 – 65, 1995.

[62] M. Johansson and A. Rantzer, "Computation of piecewise quadratic lyapunov functions for hybrid systems," in *European Control Conference (ECC)*, vol. 43, pp. 2005–2010, 1997.

[63] D. Mignone, G. Ferrari-Trecate, and M. Morari, "Stability and stabilization of piecewise affine and hybrid systems: An lmi approach," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 1, pp. 504–509, 2000.

[64] M. S. Branicky, "Stability of switched and hybrid systems," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 4, pp. 3498–3503, Dec 1994.

[65] P. Biswas, P. Grieder, J. Löfberg, and M. Morari, "A survey on stability analysis of discrete-time piecewise affine systems," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 283 – 294, 2005.

[66] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari, "Multi-parametric toolbox (mpt)," in *Hybrid Systems: Computation and Control*, pp. 448–462, 2004.

[67] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," in *International Conference on Learning Representations (ICLR)*, 2018.

[68] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1-4, pp. 625–653, 1999.

[69] J. Löfberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 284–289, 2004.

[70] M. Johansson, A. Ghulchak, and A. Rantzer, "Improving efficiency in the computation of piecewise quadratic lyapunov functions," in *Proceedings of the Mediterranean Conference on Control and Automation*, 1999.

[71] A. C. Luo, *Regularity and complexity in dynamical systems*. Springer Science & Business Media, 2012.

[72] S. V. Rakovic, P. Grieder, M. Kvasnica, D. Q. Mayne, and M. Morari, "Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 2, pp. 1418–1423, 2004.

[73] H. Benlaoukli, M. Hovd, S. Olaru, and P. Boucher, "On the construction of invariant sets for piecewise affine systems using the transition graph," in *Proceedings of the IEEE International Conference on Control and Automation (ICRA)*, pp. 122–127, 2009.

[74] A. A. Zaher, "Nonlinear control of systems with multiple equilibria and unknown sinusoidal disturbance," *Communications in Nonlinear Science and Numerical Simulation*, vol. 12, no. 8, pp. 1518 – 1533, 2007.

[75] S. Geva and J. Sitte, “A cartpole experiment benchmark for trainable controllers,” *IEEE Control Systems*, vol. 13, no. 5, pp. 40–51, 1993.

[76] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Dover Publications, 2007.

[77] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning,” *Artificial Intelligence Review - Special issue on lazy learning*, vol. 11, no. 1-5, pp. 11–73, 1997.

[78] S. Schaal, C. G. Atkeson, and S. Vijayakumar, “Scalable techniques from nonparametric statistics for real time robot learning,” *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[79] W. S. Cleveland, “Robust locally weighted regression and smoothing scatterplots,” *Journal of the American statistical association*, vol. 74, no. 368, pp. 829–836, 1979.

[80] W. S. Cleveland and C. Loader, “Smoothing by local regression: Principles and methods,” in *Statistical theory and computational aspects of smoothing*, pp. 10–49, Springer, 1996.

[81] S. Schaal and C. G. Atkeson, “Constructive incremental learning from only local information,” *Neural computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[82] S. Schaal and C. G. Atkeson, “Receptive field weighted regression,” 1997. ATR Human Information Processing Laboratories, Technical Report TR-H-209.

[83] S. Vijayakumar, A. D’souza, and S. Schaal, “Incremental online learning in high dimensions,” *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[84] S. Wold, M. Sjöström, and L. Eriksson, “PLS-regression: a basic tool of chemometrics,” *Chemometrics and intelligent laboratory systems*, vol. 58, no. 2, pp. 109–130, 2001.

[85] P. Geladi and B. R. Kowalski, “Partial least-squares regression: a tutorial,” *Analytica chimica acta*, vol. 185, pp. 1–17, 1986.

[86] R. Babuška, *Fuzzy modeling for control*. Springer Science & Business Media, 2012.

[87] H. Zhang and D. Liu, *Fuzzy modeling and fuzzy control*. Springer Science & Business Media, 2006.

[88] G. Feng, *Analysis and synthesis of fuzzy control systems: A model-based approach*. CRC Press, 2010.

[89] E. Kim and D. Kim, “Stability analysis and synthesis for an affine fuzzy system via LMI and ILMI: Discrete case,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 1, pp. 132–140, 2001.

[90] E. Kim and S. Kim, “Stability analysis and synthesis for an affine fuzzy control system via LMI and ILMI: a continuous case,” *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 3, pp. 391–400, 2002.

[91] Y. L. Tong, *The multivariate normal distribution*. Springer Science & Business Media, 2012.

[92] E. Kim, C.-H. Lee, and Y.-W. Cho, "Analysis and design of an affine fuzzy system via bilinear matrix inequality," *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 1, pp. 115–123, 2005.

[93] A. Hassibi and S. Boyd, "Quadratic stabilization and control of piecewise-linear systems," in *Proceedings of the American Control Conference (ACC)*, vol. 6, pp. 3659–3664, 1998.

[94] M. Johansson, A. Rantzer, and K.-E. Arzen, "Piecewise quadratic stability of fuzzy systems," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 713–722, 1999.

[95] S. Cao, N. Rees, and G. Feng, "Stability analysis and design for a class of continuous-time fuzzy control systems," *International Journal of Control*, vol. 64, no. 6, pp. 1069–1087, 1996.

[96] S. Cao, N. Rees, and G. Feng, "Analysis and design for a class of complex control systems part ii: Fuzzy controller design," *Automatica*, vol. 33, no. 6, pp. 1029 – 1039, 1997.

[97] G. Feng, "Stability analysis of discrete-time fuzzy dynamic systems based on piecewise lyapunov functions," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 1, pp. 22–28, 2004.

[98] M. Herceg, M. Kvasnica, and M. Fikar, "Transformation of fuzzy takagi-sugeno models into piecewise affine models," in *International Conference on Rough Sets and Intelligent Systems Paradigms*, pp. 211–220, 2007.

[99] R. Suard, J. Löfberg, P. Grieder, M. Kvasnica, and M. Morari, "Efficient computation of controller partitions in multi-parametric programming," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, vol. 4, pp. 3643–3648, 2004.

[100] S. B. Pope, "Algorithms for ellipsoids." Report: FDA-08-01, https://tcg.mae.cornell.edu/pubs/Pope_FDA_08.pdf, accessed: 01.07.2018.

[101] L. Wang and G. Feng, "Piecewise H infinity: controller design of discrete time fuzzy systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 682–686, 2004.

[102] K. Doya, "Reinforcement learning in continuous time and space," *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[103] M. Fiacchini and M. Alamir, "Computing control invariant sets is easy," *CoRR*, vol. abs/1708.04797, 2017.

[104] G. Ferrari-Trecate, F. A. Cuzzola, D. Mignone, and M. Morari, "Analysis and control with performance of piecewise affine and hybrid systems," in *Proceedings of the American Control Conference (ACC)*, vol. 1, pp. 200–205, 2001.

[105] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 938–948, 2006.

[106] N. Athanasopoulos, G. Bitsoris, and M. Lazar, “Construction of invariant polytopic sets with specified complexity,” *International Journal of Control*, vol. 87, no. 8, pp. 1681–1693, 2014.

Glossary

List of Acronyms

BB	Bounding Box
BMI	Bilinear Matrix Inequality
CP	Convex Programming
CQLF	Common Quadratic Lyapunov Function
DNN	Deep Neural Network
GP	Gaussian Process
LMI	Linear Matrix Inequality
LMIs	Linear Matrix Inequalities
LP	Linear Program
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
LWL	Locally Weighted Learning
LWPR	Locally Weighted Projection Regression
LWR	Locally Weighted Regression
MILP	Mixed-Integer Linear Program
MPI	Maximal Positively Invariant
MPT	Multi-Parametric Toolbox
MSE	Mean Squared Error
NN	Neural Network

PI	Positively Invariant
PLS	Partial Least Squares
PQLF	Piecewise Quadratic Lyapunov Function
PWA	Piecewise Affine
ReLU	Rectified Linear Unit
RF	Receptive Field
RFWR	Receptive Field Weighted Regression
RoA	Region of Attraction
SDP	Semidefinite Programming
SVD	Singular Value Decomposition
T-S	Takagi-Sugeno

List of Symbols

λ	Eigenvalue
Ω	Set of possible transitions between activation regions
σ	Nonlinear vector-valued function
$\tilde{\beta}$	Parameter vector
ξ	Premise variable
$\bar{\mathcal{X}}_l$	Outer-approximation of the l^{th} activation region
diag	Operator that creates a square diagonal matrix with the elements of a vector
$\hat{y}$	Prediction for a query point
$\hat{y}_i$	Local prediction of the i^{th} receptive field
int	Interior of a set
$\mathbb{D}$	Training dataset
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{>0}$	Set of positive real numbers
$\mathbb{R}_{\geq 0}$	Set of non-negative real numbers
$\mathcal{E}$	Hyperellipsoid
$\mathcal{I}_0$	Set of indices of linear local models
$\mathcal{I}_1$	Set of indices of affine local models
$\mathcal{J}$	Set of indices of local models activated in a region
$\mathcal{N}$	Number of activation regions

$\mathcal{R}$	Region of Attraction
$\mathcal{X}_i$	Activation region of the ith local model
$\tilde{x}$	Augmented state vector
x_{init}	Initial condition
A_i	State update matrix of the ith local model
a_j	Input-vector for the jth layer
b_i	State update bias vector of the ith local model
B_j	Bias vector jth hidden layer
c	Query point
c_i	Center of the ith receptive field
D	Distance metric
d	Distance function
F_{ij}	Fuzzy variable corresponding to the ith rule and the jth input
G_l	Bounding box corresponding to the lth activation region
h	Normalized weight
i_j	Index of the activation pattern for the jth hidden layer
L	Number of hidden layers
M	Number of local models
N	Number of training points
n	State dimensionality
N_j	Number of activation patterns for the jth hidden layer
n_j	Number of neurons in the jth hidden layer
p	Input dimensionality
q	Output dimensionality
R_i	Fuzzy rule i
T	Transition matrix
V	Lyapunov function
v	Dimensionality of premise variables
w	Weighting function
W_j	Weight matrix of the jth hidden layer
x	Input or state vector
x_e	Equilibrium of the state-space model
x_e^i	Equilibrium of the ith local model
x_j	Input vector for the jth training point
y	Output vector
y_j	Output vector for the jth training point
z_j	Vector marking for the jth hidden layer