Student Thesis 204

Learning an Approximate Model Predictive Controller with Guarantees

Student Thesis

Michael Hertneck

March 12, 2018

Examiner: *Prof. Dr.-Ing. Frank Allgöwer*
Supervisor: *Johannes Köhler, M.Sc.*
Dr.sc. Sebastian Trimpe

Institute for Systems Theory and Automatic Control
University of Stuttgart
Prof. Dr.-Ing. Frank Allgöwer

Intelligent Control Systems Group
Max Planck Institute for Intelligent Systems Stuttgart

Abstract

In this thesis, a supervised learning framework to approximate a model predictive controller (MPC) with guarantees on stability and constraint satisfaction is proposed. The approximate controller has a reduced computational complexity in comparison to standard MPC which makes it possible to implement the resulting controller for systems with a high sampling rate on a cheap hardware. The framework can be used for a wide class of nonlinear systems.

In order to obtain closed-loop guarantees for the approximate MPC, a robust MPC (RMPC) with robustness to bounded input disturbances is used which guarantees stability and constraint satisfaction if the input is approximated with a bound on the approximation error.

The RMPC can be sampled offline and hence, any standard supervised learning technique can be used to approximate the MPC from samples. Neural networks (NN) are discussed in this thesis as one suitable approximation method.

To guarantee a bound on the approximation error, statistical learning bounds are used. A method based on Hoeffding's Inequality is proposed to validate that the approximate MPC satisfies these bounds with high confidence. This validation method is suited for any approximation method. The result is a closed-loop statistical guarantee on stability and constraint satisfaction for the approximated MPC.

Within this thesis, an algorithm to obtain automatically an approximate controller is proposed. The proposed learning-based MPC framework is illustrated on a nonlinear benchmark problem for which we learn a neural-network controller that guarantees stability and constraint satisfaction.

The combination of robust control and statistical validation can also be used for other learning based control methods to obtain guarantees on stability and constraint satisfaction.

Parts of this thesis have been submitted for publication at the IEEE Control Systems Letters. The title of the submitted paper is "Learning an Approximate Model Predictive Controller with Guarantees". The authors of the paper are Michael Hertneck, Johannes Köhler, Sebastian Trimpe and Frank Allgöwer.

Contents

1 Introduction

1.1 Problem description

Model predictive control (MPC) [1] is a modern control method which has been actively researched in the last years. It is based on repeatedly solving an optimization problem online. Applications in industry are widespread. An advantage of MPC is the guaranteed satisfaction of hard constraints and the optimality of the solution with respect to a certain cost function for nonlinear systems. One major drawback of MPC is the computational effort that arises when solving optimization problems online under real time requirements. This happens especially for settings with a large number of optimization variables, e.g if the prediction horizon is large, or if a high sampling rate is required.

For linear systems, the optimization problem can be solved offline i.e. before the runtime of the system under some mild assumptions [2]. Thus an explicit control law is obtained. The extension of [2] to nonlinear systems is not straightforward. Hence, the goal of this thesis is to develop a framework for approximating a nonlinear MPC through supervised learning with statistical guarantees on stability and constraint satisfaction.

1.2 Proposed approach

In this thesis, we propose a framework to learn a controller with *guaranteed* stability and constraint satisfaction. The key idea is to approximate a robust MPC (RMPC) with robustness to bounded input disturbances with a machine learning technique. Any3 regression method is admissible within the proposed framework, if the method is capable to satisfy the chosen bound on the approximation error. There are several machine learning techniques that can achieve an arbitrary small approximation error. In this thesis, we focus on neural networks which can approximate any nonlinear function with a finite number of nonlinearities arbitrary well [3]. However, in general, guaranteeing a bound on the approximation error for machine learning techniques can be challenging.

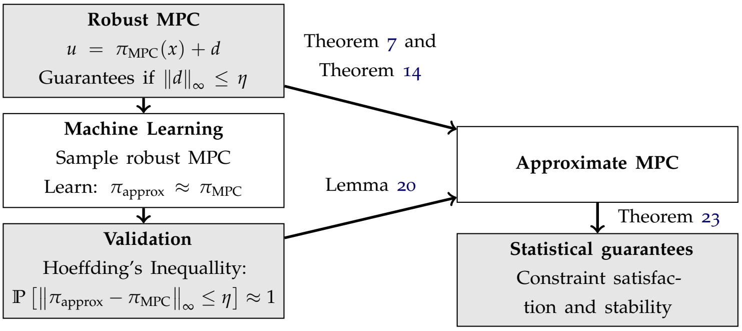
Robust MPC
$u = \pi_{\text{MPC}}(x) + d$
Guarantees if $\|d\|_\infty \leq \eta$
Machine Learning
Sample robust MPC
Learn: $\pi_{\text{approx}} \approx \pi_{\text{MPC}}$
Validation
Hoeffding's Inequallity:
$\mathbb{P}\left[\|\pi_{\text{approx}} - \pi_{\text{MPC}}\|_\infty \leq \eta\right] \approx 1$
Theorem 7 and
Theorem 14
Lemma 20
Approximate MPC
Theorem 23
Statistical guarantees
Constraint satisfac-
tion and stability

Figure 1.1: Diagram of the proposed framework. We design an MPC π_{MPC} with robustness to input disturbances d. The resulting feedback law is sampled offline and approximated (π_{approx}) via machine learning techniques. Hoeffding's Inequality is used for validation to provide a bound on the error between approximate controller and MPC in order to guarantee stability and constraint satisfaction. The result is a controller with statistical guarantees.

A probabilistic approach to provide a bound on the approximation error is to use Hoeffding's Inequality [4]. The resulting validation method is based on sampling the approximated RMPC and evaluate weather the approximation error satisfies a chosen bound. This probabilistic validation in combination with a learning method that is suitable to achieve any bound on the approximation error enables us to guarantee stability and the satisfaction of hard constraints with any desired probability. Therewith, a new method to obtain an approximate MPC can be established. A sketch of a complete framework for the controller synthesis based on the proposed methods is given in Figure 1.1.

Advantages are the guarantees on stability and constraint satisfaction and a cheap implementable controller. The framework is suited to design controller for a wide class of nonlinear systems in an automatic fashion.

Instead of MPC and NN, other learning based control approaches as e.g. [5, 6, 7] can be adapted similar to the proposed framework, if a robust control method is used in combination with the proposed validation method. Thus, the proposed framework is also relevant for other learning based control methods.

1.3 Contribution

This thesis makes contributions to the theory of learning based control and approximate MPC (AMPC). The main purpose is to propose a framework for the synthesis of an AMPC with guarantees on stability and satisfaction of hard constraints. The framework is applicable to a wide class of nonlinear systems. The controller synthesis using this framework works automated with few design parameters to choose.

One contribution of this thesis is to adjust results from tube based RMPC with guaranteed stability and constraint satisfaction for nonlinear systems [8] into a terminal cost/ terminal constraint setting with additive input disturbance. The resulting RMPC guarantees stability and satisfaction of hard constraints under disturbance.

Another contribution of this thesis is a validation method that delivers guarantees on the approximation error in a probabilistic fashion based on Hoeffding's Inequality [4]. Therefore, the learned control law is compared to the RMPC along trajectories. The key idea is, that stability and constraint satisfaction can be guaranteed for the closed loop system, if the approximation error of the learned control law is smaller than the admissible input disturbance for the robust MPC. The bound on the approximation error is guaranteed probabilistic due to the validation method.

The combination of this contributions delivers a framework for the automatic controller synthesis to obtain an approximate MPC with guarantees on stability and constraint satisfaction.

The last contribution of this thesis is the application of the proposed framework to a continuous stirred tank reactor from [9] as benchmark. We show the practicability of the framework and substantiate that this method can yield a controller which guarantees closed loop stability and constraint satisfaction with a low computational complexity.

Parts of this thesis have been submitted for publication at the IEEE Control Systems Letters. The title of the submitted paper is "Learning an Approximate Model Predictive Controller with Guarantees". The authors of the paper are Michael Hertneck, Johannes Köhler, Sebastian Trimpe and Frank Allgöwer.

1.4 Literature review

For linear systems, the optimization problem can be solved offline i.e. before the runtime of the system under some mild assumptions [2]. Thus an explicit control law is obtained. The extension of [2] to nonlinear systems is not straight forward and there are also computational complexity issues concerning the method from [2].

Hence, there exist several approaches to obtain an approximative solution for the MPC optimization. For linear systems, in [10] a learning algorithm is presented with additional constraints to guarantee stability and constraint satisfaction of the approximate MPC. This stands in contrast to our method, which uses standard learning methods and is applicable to nonlinear systems.

For nonlinear systems, there are several approaches. One approach to approximate MPC is convex multi parametric nonlinear programming [11, 12], where a suboptimal approximation of the MPC control law is computed. Another approach is to approximate an MPC with machine learning techniques. This is done by neural networks in [13, 14, 15]. These methods can in general not provide guarantees on stability or constraint satisfaction for the approximated MPC which is especially crucial if hard state constraints are considered.

In [5], a support vector machine informed method is used to approximate the MPC. Stability and constraint satisfaction can be guaranteed for arbitrary small approximation errors based on inherent robustness properties. In [16], an MPC with Lipschitz based constraint tightening is approximated, which ensures stability for non vanishing approximation errors. The approximation error deduced in [5, 16] is typically not achievable for practical application (compare example in Chapter 7) and is thus not suited for the proposed framework.

1.5 Outline

This thesis is structured as follows: In Chapter 2, we recap results from the theory of MPC. In Chapter 3, we formulate the problem and present our main ideas. The input robust MPC is presented in Chapter 4. In Chapter 5, we discuss supervised learning and present a validation method to obtain guarantees on the learned MPC. In Chapter 6, the proposed framework is summarized and an algorithm to compute the approximate

MPC offline is given. Chapter 7 contains a numerical benchmark example for the framework. Chapter 8 concludes the thesis.

1.6 Notation

In this thesis, the following notation is used: The positive real numbers are denote by $\mathbb{R}_{>0}$, $:= \{r \in \mathbb{R} | r > 0\}$, and respectively, $\mathbb{R}_{\geq 0} = \mathbb{R}_{>0} \cup \{0\}$. A function f is positive definite if $f(0) = 0$ and $f(x) > 0 \; \forall x \in \mathbb{R} \cup \{0\}$. The function $sign(x)$ is defined as

$$sign(x) = \left\{ \begin{matrix} 1 & if \; x > 0 \\ -1 & if \; x < 0 \\ 0 & \text{else} \end{matrix} \right\}.$$

Class $\mathcal{K}$ functions

A continuous function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}$ function if it is strictly increasing and $\alpha(0) = 0$.

Sets and set operations

A set $\mathcal{S}$ is a robust positive invariant (RPI) set under some dynamics $x(t+1) = f(x(t), w(t))$, if $x(t+1) \in \mathcal{S} \; \forall x(t) \in \mathcal{S}, \forall w(t) \in \mathcal{W}$. The Minkovski set addition is defined as $X \oplus Y := \{x + y : x \in X, y \in Y\}$. The Pontryagin set difference is defined by $X \ominus Y := \{z \in \mathbb{R}^n : z + y \in X, \forall y \in Y\}$.

Norms

The quadratic norm $\|x\|^2 = x^\top x$ is denoted by $\|\cdot\|$. The quadratic norm with respect to a positive definite matrix $Q = Q^\top$ is denoted by $\|x\|_Q^2 = x^\top Q x$.

Eigenvalues

The minimum and maximum eigenvalue of a symmetric matrix $Q = Q^\top$ are indicated by $\lambda_{\min}(Q)$ and $\lambda_{\max}(Q)$. $\lambda_{\max}(P/Q)$ and $\lambda_{\min}(P/Q)$ denote the largest and smallest generalized eigenvalue of $(P - \lambda Q)v = 0$. $P > 0$ implies that $\lambda_{\min}(P) > 0$ and $P > Q$ denotes that $\lambda_{\min}(P - Q) > 0$.

Probabilistic expressions

Let X be a random variable. $\mathbb{P}\left[X \geq x\right]$ denotes the probability for $X \geq x$.

2 Model Predictive Control - Theory

This chapter gives a brief overview over relevant MPC theory. First we recap some standard MPC methods to ensure stability and constraint satisfaction. Then, RMPC and AMPC are reviewed. A more detailed overview over well established results in MPC theory can be found in [1].

The basic idea of MPC is to use a dynamic model of a system to forecast the system behavior. Therewith an optimal input sequence can be computed and the first input of this sequence is used as feedback. Thus, contrary to most other control techniques, the control law is only implicit defined as the solution to an optimization problem. Satisfaction of constraints can be ensured by a suitable formulation of the optimization problem. An advantage of MPC is that complex nonlinear dynamics can be handled. We present a formulation for the MPC optimization problem that guarantees stability and constraint satisfaction in the next section.

2.1 MPC problem formulation

We consider the nonlinear discrete time dynamics $x(t+1) = f(x(t),u(t))$ with continuous f, satisfying $f(0,0) = 0$, where $x \in \mathbb{R}^n$ denotes the system state and $u \in \mathbb{R}^m$ are the system inputs. The MPC optimization problem for setpoint stabilization for discrete time systems, the setpoint $(x,u) = (0,0)$ and a positive definite stage cost $l(x,u)$ can be stated as follows: For each time step, solve

$$\begin{aligned} V_N(x) =& \min_{u(\cdot|t)} J_N(x(t),u(\cdot|t)) \\ =& \min_{u(\cdot|t)} \sum_{k=0}^{N-1} l(x(k+t|t),u(k+t|t)) + V_f(x(t+N|t)) \end{aligned}$$

subject to constraints

$$\begin{aligned} x(t|t) &= x(t), \\ x(k+t+1|t) &= f(x(k+t|t),u(k+t|t)), \\ x(k+t|t) &\in \mathcal{X}, k = 0,...,N-1, \\ u(k+t|t) &\in \mathcal{U}, k = 0,...,N-1, \\ x(t+N|t) &\in \mathcal{X}_f, \end{aligned}$$

and apply $u(t) = u^\star(t|t)$. The state and input constraints are denoted by $\mathbb{R}^n \supseteq \mathcal{X} \supseteq 0$ and $\mathbb{R}^m \supseteq \mathcal{U} \supseteq 0$, $V_f(x)$ the terminal cost and J_N the open loop cost. The prediction for $x(k+t)$ at time t is denoted by $x(k+t|t)$. The solutions of the MPC optimization problem is denoted with $u^\star(\cdot|t)$, $x^\star(\cdot|t)$ and the value function $V_N(x(t))$. For continuous time systems, a similar formulation is possible, for details see [1].

We call the MPC problem feasible at time t, if there exists at least one input $\tilde{u}(\cdot|t)$ that satisfies the constraints . The MPC problem is recursive feasible if feasibility at time t implies feasibility for all times $k+t, k \in \mathbb{N}$. We assume in the following chapters that the state can be measured. If the state can only be estimated, robust output feedback MPC is required, compare e.g. [17].

Stability results for MPC

The goals of model predictive control are to achieve stability of the origin $x = 0$ for the resulting closed loop system and guaranteed satisfaction of state and input constraints for all times. Therefore, recursive feasibility of the optimization problem and closed loop stability have to be established using the finite horizon MPC optimization problem. Suitably designed terminal constraints and terminal costs for the MPC problem can be used to obtain this guarantees [18]. We sketch this procedure in this section.

To guarantee stability and constraint satisfaction, we use a suitable terminal cost V_f and a suitable terminal set $\mathcal{X}_f$ [18]. Within the terminal set, an explicit control law $k_f(x) \in \mathcal{U}\ \forall x \in \mathcal{X}_f \subseteq \mathcal{X}$, with

$$f(x,k_f(x)) \in \mathcal{X}_f\ \forall x \in \mathcal{X}_f,$$

needs to exist. Furthermore,

$$V_f(f(x,k_f(x))) - V_f(x) \leq -l(x,k_f(x))$$

must hold $\forall x \in \mathcal{X}_f$. Then recursive feasibility of the MPC problem can be proven by considering the candidate solution

$$u(k+t|t+1) = \begin{cases} u(k+t|t) & k = 1,...,N-1 \\ k_f(x(N)|t+1) & k = N \end{cases},$$

which guarantees that a solution that satisfies all constraints will exist at $t+1$ if it exists at t due to the terminal constraints. To show stability of the MPC, the value function $V_N(x)$ can be used as Lyapunov function.

In [18], a procedure to compute terminal costs, a terminal set and a terminal controller for continuous time systems with stabilizable linearization at the origin and quadratic stage costs is provided. The method is termed Quasi Infinite Horizon MPC [18] as the cost function J_N is an approximation of the infinite horizon cost.

On possibility to satisfies the assumptions on the terminal ingredients is a zero terminal constraint with $V_f(x) = 0$, $\mathcal{X}_f = \{0\}$ and $k_f(x) = 0$. However, under disturbance, the satisfaction of this zero terminal constraint cannot be robustified using standard approaches.. Thus, for robust MPC, a zero terminal constraint is not suitable.

Further MPC schemes

There exist a lot of further MPC schemes with guarantees. In MPC without terminal constraints, stability of the closed loop system can be shown by choosing the prediction horizon large enough under some controllability assumptions [19, 20]. This can bring computational benefits in comparison to MPC with terminal constraints but a longer prediction horizon may be necessary and the required controllability condition can be hard to verify. MPC without terminal constraints is equally applicable to our setup.

In economic MPC [21], the stage cost for the MPC optimization problem is not requested to be positive definite. As a result, the closed-loop system might not converge to an equilibrium, if other trajectories (e.g. periodic) exist, that lead to a better performance. The proposed framework is with modifications in the RMPC also applicable to economic MPC.

2.2 Robust MPC

In this section we review existing RMPC techniques. One of this methods will be extended in Chapter 4 to be robust with respect to input disturbances.

This will enable us to approximate the RMPC algorithm with a machine learning tool to obtain an explicit MPC control law.
RMPC has to guarantee stability and constraint satisfaction for an uncertain (nonlinear) system for all possible realizations of the uncertainty. This is typically achieved by a constraint tightening. Assume an additive, bounded disturbance. Then, the disturbed system can be formulated as

$$x(t+1) = f(x(t),u(t)) + w(t)$$

with $w(t) \in \mathcal{W} : \{w(t) \in \mathbb{R}^n | \; \|w(t)\|_\infty \leq w_{\max}, \}$.

Under some assumptions, inherent robustness properties of standard MPC formulations can be provided [22]. For inherent robustness, only arbitrary small disturbances may be admissible.

In Min-Max schemes the optimization problem is extended to find a control input that minimizes the predicted cost for the worst case disturbance, i.e. the disturbance that maximizes the cost. An overview over this approach is given in [23]. A disadvantage of Min-Max schemes is that the optimization may be computational intractable.

A compromise between inherent robustness and Min-Max schemes are tube based approaches. For linear systems, there are two main tube based approaches. In [24] an additional error feedback is used to keep the real system state in a tube around the nominal system state for the system dynamics without disturbance. The tube is an RPI set for the system with error feedback under disturbance. The optimal nominal state for a given real system state is used as an additional optimization variable in the optimization problem. To achieve recursive feasibility, the state constraints are tightened by the size of the tube. The input constraints must be tightened according to the error feedback.

In [25], a constraint tightening with growing tubes is used for linear systems. If the system is stabilizable, a stabilizing controller can be used to obtain a bounded representation of the reachable system states after each time step along the prediction horizon if a disturbance occurs at the actual time step. The constraints are tightened such that the untightened constraints are still satisfied if one of this reachable system states occur. Since the pre-stabilized system is stable, the necessary constraint tightening is bounded for a bounded disturbance.

For nonlinear systems, there exist several tube based approaches. In Lipschitz based methods (e.g. [26]), a tightening of the state constraints based on a Lipschitz constant of the system is used. For unstable nonlinear system dynamics, this approach leads to a constraint tightening which grows exponentially with the prediction horizon. Hence, the constraint tightening may be quite restrictive since the set of admissible states can decrease fast with growing prediction horizon. This can decrease the initial feasible region and the possible prediction horizon even for small disturbances. Even for small disturbances and a small prediction horizon, the tightened constraint set may be empty.

In [27], a RMPC with constraint tightening founded on an incremental input to state stability (δISS) using a δISS-Lyapunov function for uncertain systems is proposed. Based on the concept of local incremental stabilizability, [8] uses a constraint tightening similar, to the growing tubes approach for linear systems. With this approach a system with an additive state disturbance can be robustly stabilized. The constraint tightening grows along the prediction horizon but is bounded by a maximum value. This value depends on the maximum admissible disturbance. For small enough disturbances, this leads to a non empty constraint set for arbitrary large prediction horizons.

It may be perhaps premature to select a particular approach at the current stage of research [1]. Nevertheless, we present in Chapter 4 an adaption of [8] to bounded input disturbances and terminal constraints and costs since this method allows a good trade off between computational complexity and limited conservatism.

2.3 Approximate MPC

One major drawback of MPC is the required computational effort for the online solution of the MPC optimization problem. Especially for settings with a large number of optimization variables, e.g. if the prediction horizon is large, or if a high sampling rate is required, the online optimization can get intractable for a lot of applications.

The goal of approximate MPC is to decrease the online computational effort by approximating the solution to the MPC optimization problem offline. This subsection deals with the problem of solving the optimization for the MPC algorithm offline to obtain an approximate control law. We require, that the control law can be written as a function of the states and does not need any iteration to solve it. Clearly, fast online evaluation of the approximate MPC control law must be possible.

For linear systems with polytopic state constraints, this MPC optimization can be solved offline according to [2]. Therein, the MPC optimization is formulated as a multi parametric quadratic program. This can be used with the Karush-Kuhn-Tucker (KKT) conditions to obtain an explicit piece wise affine control law that guarantees the satisfaction of hard constraints.

Since the extension of [2] to nonlinear systems is not straight forward and since there are also complexity issues concerning the method from [2], there exist several approximate MPC approaches. For linear systems, [10] proposed to add constraints to the learning problem to ensure stability and constraint satisfaction of the suboptimal learned control law. An ansatz to approximate nonlinear MPC is convex multi parametric nonlinear programming [11, 12], where a suboptimal approximation of the MPC is computed. The approximation of an MPC by neural networks, as also done herein, has for example been proposed in [14, 15]. These methods can in general not provide guarantees on stability or constraint satisfaction with the approximated MPC. This is especially crucial if hard state constraints are considered.

In [5], a support vector machine informed method is used to approximate an MPC. Stability and constraint satisfaction can be guaranteed for arbitrary small approximation errors based on inherent robustness properties. In [16], an MPC with Lipschitz based constraint tightening is approximated, which ensures stability for non vanishing approximation errors. This method has the same drawbacks as Lipschitz based RMPC since the constraint tightening is too conservative which makes it impractical for the application to most nonlinear systems (compare e.g. the example in Chapter 7).

We will present a novel approximate MPC approach with guarantees on stability and constraint satisfaction based on approximation of a RMPC in combination with a validation method based on Hoeffding's Inequality in Chapter 6.

3 Main approach

In this chapter, we pose the control problem and present the proposed approach.

3.1 Problem formulation

In this section, we introduce the control problem. We consider the following nonlinear discrete time system

$$x(t+1) = f(x(t),u(t)), \tag{3.1}$$

with the state $x(t) \in \mathbb{R}^n$, the control input $u(t) \in \mathbb{R}^m$, the time step $t \in \mathbb{N}$, and continuous f, satisfying $f(0,0) = 0$. We consider compact polytopic constraints

$$\mathcal{X} = \left\{x \in \mathbb{R}^n | Hx \leq \mathbf{1}_p\right\}, \mathcal{U} = \left\{u \in \mathbb{R}^m | Lu \leq \mathbf{1}_p\right\},$$

and a quadratic stage cost

$$l(x,u) = \|x\|_Q^2 + \|u\|_R^2, \tag{3.2}$$

with a positive definite Q and a positive semidefinite R. The control objective is to ensure constraint satisfaction, i.e.

$$(x(t),u(t)) \in (\mathcal{X} \times \mathcal{U})\ \forall t \geq 0,$$

stability of the resulting system, which means

$$\lim_{t\to\infty} x(t) \to \mathcal{Z}_{\mathrm{RPI}}$$

for suitable initial conditions and a RPI set $\mathcal{Z}_{\mathrm{RPI}}$, and to minimize some cost function

$$\sum_{k=0}^{\infty} l(x(k),u(k)).$$

The resulting controller should be implementable on cheap hardware for systems with high sampling rates.

Our goal is to develop a framework for automatic controller synthesis. The framework has to be such that all computations can be done numerically with few design parameters.

3.2 General approach

We propose a framework based on approximating an MPC. The guarantees for stability and constraint satisfaction from MPC must be preserved under the approximation. This is achieved by modifying the MPC to be robust to bounded input disturbances.

The controller synthesis with the proposed framework works as follows: We set up an RMPC. The RMPC scheme guarantees stability and constraint satisfaction for $u(t) = \pi_{\mathrm{MPC}}(x(t)) + d(t)$ with $\|d(t)\|_\infty \leq \eta$ with a chosen bound on the input error η and the RMPC feedback law π_{MPC}. The RMPC is sampled offline over the set of feasible states $\mathcal{X}_{\mathrm{feas}}$. The RMPC feedback is approximated using supervised learning techniques based on these samples. In this thesis we use NN to approximate the RMPC, even though any other supervised learning technique can be used within the framework. The learning yields an AMPC

$$\pi_{\mathrm{approx}} : \mathcal{X}_{\mathrm{feas}} \to \mathcal{U} \ |u = \pi_{\mathrm{approx}}(x).$$

With this controller, the closed-loop system is given by

$$x(t+1) = f(x(t), \pi_{\mathrm{approx}}(x(t))). \tag{3.3}$$

Hence, the AMPC feedback $\pi_{\mathrm{approx}}(x)$ guarantees stability and constraint satisfaction if the approximation error $\left\|\pi_{\mathrm{MPC}} - \pi_{\mathrm{approx}}\right\|_\infty$ is bounded by the disturbance bound η, because then guarantees from the RMPC are preserved under approximation. We use a validation method based on Hoeffding's Inequality to guarantee a desired bound η on the approximation error. The overall framework will be summarized in Algorithm 1.

4 Input robust MPC

In this chapter, we present a formulation of an RMPC algorithm which preserves its guarantees on stability and constraint satisfaction under additive input disturbances. Hence, guarantees on stability and constraint satisfaction will be preserved if the RMPC is approximated with any approximation technique as long as the approximation error is below the bound on the admissible input disturbance of the RMPC. The chapter is organized as follows:

An RMPC formulation with robustness to additive input disturbances based on a constraint tightening is introduced in Section 4.1. The scheme is an adaption of [8], where a robust MPC scheme for systems under state disturbances and without terminal constraints is presented. In Section 4.2, we present the constraint tightening method based on a growing tube along the prediction horizon for open loop trajectories in the RMPC optimization problem. The tube is subtracted from the constraint set. In the last section, recursive feasibility and closed loop stability for the perturbed system with bounded input disturbance are proven.

4.1 RMPC formulation

To achieve the robustness of the RMPC, we use a quasi infinite horizon MPC formulation [18] with robust constraint tightening [8], which can be formulated as

$$\begin{aligned} V_N(x(t)) &= \min_{u(\cdot|t)} J_N((x(t)),u(\cdot|t)) \\ &= \min_{u(\cdot|t)} \sum_{k=0}^{N-1} l(x(k+t|t),u(k+t|t)) + V_f(x(N+t|t)) \end{aligned} \tag{4.1a}$$

subject to constraints

$$x(t|t) = x(t), \tag{4.1b}$$
$$x(k+t+1|t) = f(x(k+t|t),u(k+t|t)), \tag{4.1c}$$
$$x(k+t|t) \in \bar{\mathcal{X}}_k,\ k = 0,...,N-1, \tag{4.1d}$$
$$u(k+t|t) \in \bar{\mathcal{U}}_k,\ k = 0,...,N-1, \tag{4.1e}$$
$$x(N+t|t) \in \mathcal{X}_f. \tag{4.1f}$$

We denote the set of states where (4.1) is feasible by $\mathcal{X}_{\text{feas}}$. The solution of the RMPC optimization problem (4.1) is denoted by $u^\star(\cdot|t)$. The RMPC feedback at time t is $\pi_{\text{MPC}}(x(t)) := u^\star(t|t)$. The state and input constraints from standard MPC formulations are replaced by tightened constraints $\bar{\mathcal{X}}_k$ and $\bar{\mathcal{U}}_k$. The design of the terminal ingredients $\mathcal{X}_f$ and V_f will be made precise later. The closed loop of the RMPC under disturbance is given by

$$x(t+1) = f(x(t),\pi_{\text{MPC}}(x(t)) + d(t)), \tag{4.2}$$

with

$$d(t) \in \mathcal{W} = \{d \in \mathbb{R}^m : \|d\|_\infty \leq \eta\},\ \forall t \geq 0 \tag{4.3}$$

for some η. We present in the next subsection, how the constraint tightening for the RMPC is done for a chosen bound on the input disturbance.

4.2 Constraint tightening

In this section, we state the constraint tightening and necessary assumptions for the RMPC to guarantee stability and constraint satisfaction under approximation. The constraint tightening is based on a local incremental stabilizability condition and depends on an exponential contraction rate ρ. Hence, the following assumption is used in order to design the RMPC:

Assumption 1. *(Local incremental stabilizability [8, 28]) There exists a control law $\kappa : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^m$, a δ-Lyapunov function $V_\delta : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$, that is continuous in the first argument and satisfies $V_\delta(x,x,v) = 0\ \forall x \in \mathcal{X},\ \forall v \in \mathcal{U}$, and parameters $c_{\delta,l}$, $c_{\delta,u}$ δ_{loc}, $k_{\max} \in \mathbb{R}_{>0}, \rho \in (0,1)$, such that the following properties hold for all $(x,z,v) \in \mathcal{X} \times \mathcal{X} \times \mathcal{U}, (z^+,v^+) \in \mathcal{X} \times \mathcal{U}$ with $V_\delta(x,z,v) \leq \delta_{loc}$:*

$$c_{\delta,l}\|x-z\|^2 \leq V_\delta(x,z,v) \leq c_{\delta,u}\|x-z\|^2, \tag{4.4}$$
$$\|\kappa(x,z,v) - v\| \leq k_{\max}\|x-z\|, \tag{4.5}$$
$$V_\delta(x^+,z^+,v^+) \leq \rho V_\delta(x,z,v), \tag{4.6}$$

with

$$x^{+} = f(x, \kappa(x, z, v)),\ z^{+} = f(z, v).$$

Remark 2. *Assumption 1 is for example fulfilled, if the linearized dynamics at any point $(x, v) \in \mathcal{X} \times \mathcal{U}$ are stabilizable with a common quadratic Lyapunov function $V = x^\top P x$ and f is locally Lipschitz. Hence, Assumption 1 is not very restrictive.*

The concept of incremental stability [8] describes incremental robustness properties, and is thus suited for the analysis of perturbed trajectories in the RMPC design. Note that neither V_δ nor κ need to be known explicitly to design the RMPC. Only the exponential decay rate ρ and bounds $c_{\delta,l}$, $c_{\delta,u}$ and $k_{\max}$ are used within the RMPC design for the constraint tightening. These parameters are computed for an example system in Chapter 7.

To overestimate the influence from the system input on the system state, we use the following assumption:

Assumption 3. *(Local Lipschitz continuity of the input) There exists a $\lambda \in \mathbb{R}$, such that $\forall x \in \mathcal{X}$, $\forall u \in \mathcal{U}$, $\forall u + d \in \mathcal{U}$*

$$\|f(x, u + d) - f(x, u)\| \leq \lambda \|d\|_\infty. \tag{4.7}$$

With this assumption, we can introduce a bound on the admissible input disturbance which will be used in the proof of robust stability and recursive feasibility for the RMPC:

Assumption 4. *(Bound on the input disturbance) The input disturbance bound satisfies*
$\eta \leq \eta_1 := \frac{1}{\lambda}\sqrt{\frac{\delta_{loc}}{c_{\delta,u}}}$.

Note that the polytopic set $\mathcal{W}$ contains all admissible input disturbances. The set

$$\mathcal{U}_t = \mathcal{U} \ominus \mathcal{W}$$

ensures that $u + d \in \mathcal{U}\ \forall u \in \mathcal{U}_t$. Since $\mathcal{U}$ and $\mathcal{W}$ are polytopic, we can write $\mathcal{U}_t$ as

$$\mathcal{U}_t = \left\{ u \in \mathbb{R}^m | L_t u \leq \mathbf{1}_p \right\}.$$

We set

$$\epsilon_x := \eta\lambda\sqrt{\frac{c_{\delta,u}}{c_{\delta,l}}}\|H\|_\infty,\ \epsilon_u = \eta\lambda\sqrt{\frac{c_{\delta,u}}{c_{\delta,l}}}\|L_t\|_\infty k_{\max}. \tag{4.8}$$

Therewith, the constraint tightening is achieved with scalar tightening parameters

$$\epsilon_{k,x} := \epsilon_x \frac{1-\sqrt{\rho}^k}{1-\sqrt{\rho}},\ \epsilon_{k,u} := \epsilon_u \frac{1-\sqrt{\rho}^k}{1-\sqrt{\rho}},\ k \in \{0, ..., N\},$$

based on the exponential contraction rate ρ, ϵ_x and ϵ_u. The tightened constraint sets are given by

$$\bar{\mathcal{X}}_k := (1-\epsilon_{k,x})\mathcal{X} = \{x \in \mathbb{R}^n | Hx \leq (1-\epsilon_{k,x})\mathbf{1}_p\},$$

$$\bar{\mathcal{U}}_k := (1-\epsilon_{k,u})\mathcal{U}_t = \{u \in \mathbb{R}^m | L_t u \leq (1-\epsilon_{k,u})\mathbf{1}_q\}.$$

Note that, if k approaches infinity, we get

$$\mathcal{X}_\infty := (1-\epsilon_{\infty,x})\mathcal{X},\ \mathcal{U}_\infty := (1-\epsilon_{\infty,u})\mathcal{U}_t,\ \epsilon_{\infty,x} := \frac{\epsilon_x}{1-\sqrt{\rho}},\ \epsilon_{\infty,u} := \frac{\epsilon_u}{1-\sqrt{\rho}}.$$

Contrary to Lipschitz based tightening approaches as in [26], the constraint tightening is bounded even as k approaches infinity. If η is chosen small enough, the tightened constraint sets are not empty for all k.

Clearly, the size of the tightened constraints depends on ϵ_x and ϵ_u and thus on η. Hence, decreasing the approximation accuracy η increases the feasible region.

The maximum influence of a disturbance at the current time step on the predicted system state $x(N+t|t)$ is described by

$$\mathcal{W}_N = \left\{ x \in \mathbb{R}^n : \|x\| \leq \lambda\eta \sqrt{\rho^N \frac{c_{\delta,u}}{c_{\delta,l}}} \right\}. \tag{4.9}$$

We use the following Assumption on the terminal set to guarantee recursive feasibility and closed loop stability similar to [18, 29, 30]

Assumption 5. *(Terminal set) There exists a local control Lyapunov function $V_f(x) = \|x\|_P^2$, a terminal set $\mathcal{X}_f = \left\{x \in \mathbb{R}^n | V_f(x) \leq \alpha_f\right\}$ and a control law k_f such that $\forall x \in \mathcal{X}_f$*

$$f(x, k_f(x)) + w \in \mathcal{X}_f, \forall w \in \mathcal{W}_N, \tag{4.10}$$

$$V_f(f(x,k_f(x))) \leq V_f(x) - l(x,k_f(x)), \tag{4.11}$$

$$(x,k_f(x)) \subseteq (\bar{\mathcal{X}}_N \times \bar{\mathcal{U}}_N). \tag{4.12}$$

Remark 6. *For continuous time systems [18] provides the existence of a terminal set if the linearization of (3.1) is stabilizable and (0,0) lies in the interior of $\mathcal{X} \times \mathcal{U}$. Furthermore an approach to calculate the terminal set is stated. In [30], the results from [18] are adapted to discrete time systems. These approaches are both for systems without disturbances. For a disturbed system, a terminal set can be calculated by using the approach from [30] for $\bar{\mathcal{X}}_N \times \bar{\mathcal{U}}_N$ instead of $\mathcal{X} \times \mathcal{U}$. Then, there will always be a small enough η or a large enough N to ensure that (4.10) is satisfied. This has resemblances to [29]. We show, how this can be done in Section 6.2.3.*

4.3 Stability and constraint satisfaction

In this section we study the properties of the RMPC closed loop system (4.2) under disturbance. First, recursive feasibility of the RMPC under disturbance for an initial feasible point is proven in Theorem 7. Then, Lemma 11 and Propositions 12 and 13 deliver the prerequisites to prove the stability of the closed loop system as it is done in Theorem 14.

4.3.1 Recursive feasibility

First we show recursive feasibility of the MPC-algorithm with the following theorem:

Theorem 7. *Let Assumptions 1, 3, 4 and 5 hold. Assuming that there is a feasible input sequence $u^\star(\cdot|t)$ for problem (4.1) at time t, a feasible input sequence $\bar{u}(\cdot|t+1)$ for system (4.2) under input disturbance at time t+1 can be constructed as follows:*

$$\begin{aligned}&\bar{u}(k+t|t+1)\\ &= \begin{cases}\kappa(\bar{x}(k+t|t+1), x^\star(k+t|t), u^\star(k+t|t)), & 1 \leq k \leq N-1\\ \kappa(\bar{x}(k+t|t+1), x^\star(k+t|t), k_f(x^\star(k+t|t))), & k = N\end{cases},\end{aligned} \tag{4.13}$$

The corresponding state sequence is given by $\bar{x}(k+t|t+1)$, $k = 1, \dots, N$ with

$$\begin{aligned}\bar{x}(k+t+1|t+1) &= f(\bar{x}(k+t|t+1), \bar{u}(k+t|t+1)),\ k = 1,...,N,\\ \bar{x}(t+1|t+1) &= x(t+1) = f(x(t|t), u^\star(t|t) + d(t)), \|d(t)\|_\infty \leq \eta.\\ &= x^\star(t+1|t) + \triangle x, \quad \|\triangle x\| \leq \lambda\eta,\end{aligned}$$

.

Proof. The proof is composed of two parts. Part I shows the satisfaction of state constraints (4.1d) and input constraints (4.1e). Part II establishes the satisfaction of the terminal constraint (4.1f).
Part I: For $k \leq N$, it holds that $x^\star(k+t|t) \in \bar{\mathcal{X}}_k$ due to (4.1d) and (4.1f). By (4.1e), $u^\star(k+t|t) \in \bar{\mathcal{U}}_k$ for $k < N$. Furthermore, define $u^\star(N+t|t) = k_f(x^\star(t+N|t))$ with $k_f(x^\star(t+N|t)) \in \bar{\mathcal{U}}_N$ by (4.12) and (4.1f) as input at time $t+N$. By Assumption 3 and (3.3) it holds that

$$\|\bar{x}(t+1|t+1) - x^\star(t+1|t)\| \leq \lambda \|u(t) - u^\star(t|t)\|_\infty \leq \lambda\eta.$$

The bound $\eta \leq \eta_1$ from Assumption 4 implies $V_\delta(x(t+1), x^\star(t+1|t), u^\star(t+1|t)) \leq \delta_{\text{loc}}$. Using (4.6) recursively, we get

$$V_\delta(\bar{x}(k+t|t+1), x^\star(k+t|t), u^\star(k+t|t)) \leq \rho^{k-1} c_{\delta,u} \lambda^2 \eta^2 \leq \delta_{\text{loc}}. \tag{4.14}$$

With (4.4) and (4.5) this yields

$$\|\bar{x}(k+t|t+1) - x^\star(k+t|t)\| \leq \sqrt{\rho^{k-1} \frac{c_{\delta,u}}{c_{\delta,l}}} \lambda\eta, \tag{4.15}$$

$$\|\bar{u}(k+t|t+1) - u^\star(k+t|t)\| \leq \sqrt{\rho^{k-1} \frac{c_{\delta,u}}{c_{\delta,l}}} k_{\max} \lambda\eta. \tag{4.16}$$

Note that $\|.\|_\infty \leq \|.\|$ and

$$H\bar{x}(k+t|t+1) - Hx^\star(k+t|t) = H(\bar{x}(k+t|t+1) - x^\star(k+t|t)).$$

With the choice of ϵ_x from (4.8) we can overestimate

$$\begin{aligned} H\bar{x}(k+t|t+1) &\leq Hx^\star(k+t|t) + \|H\|_\infty \sqrt{\rho^{k-1} \frac{c_{\delta,u}}{c_{\delta,l}}} \lambda\eta \mathbf{1}_p \\ &\overset{(4.8)}{\leq} Hx^\star(k+t|t) + \sqrt{\rho}^{k-1} \epsilon_x \mathbf{1}_p \leq (1 - \epsilon_{k,x} + \sqrt{\rho}^{k-1} \epsilon_x) \mathbf{1}_p \\ &= (1 - \epsilon_{k-1,x}) \mathbf{1}_p. \end{aligned} \tag{4.17}$$

The same procedure for the input with

$$L_t \bar{u}(k+t|t+1) - L_t u^\star(k+t|t)) = L_t(\bar{u}(k+t|t+1) - u^\star(k+t|t))$$

delivers

$$\begin{aligned}
L_t\bar{u}(k+t|t+1) &\leq L_t u^\star(k+t|t) + \|L_t\|_\infty \sqrt{\rho^{k-1}c_{\delta,l}}k_{\max}\lambda\eta\mathbf{1}_q \\
&\overset{(4.8)}{\leq} L_t u^\star(k+t|t+1) + \sqrt{\rho}^{k-1}\epsilon_u\mathbf{1}_q \leq (1-\epsilon_{k,u}+\sqrt{\rho}^{k-1}\epsilon_u)\mathbf{1}_q \\
&= (1-\epsilon_{k-1,u})\mathbf{1}_q. \qquad (4.18)
\end{aligned}$$

From (4.17) and (4.18) it follows that

$$\bar{u}(k+t|t+1) \in \bar{\mathcal{U}}_{k-1}, \quad \bar{x}(k+t|t+1) \in \bar{\mathcal{X}}_{k-1}, \quad 1 \leq k \leq N.$$

The choice of U_t, i.e. $\mathcal{U}_0 = \mathcal{U}_t$ guarantees $u^\star(t|t) + d(t) \in \mathcal{U}$, $\forall d(t) \in \mathcal{W}$.

Part II: recursive satisfaction of the terminal constraints The recursive satisfaction of the terminal constraints is guaranteed by the application of the terminal controller for $\bar{u}(N+t|t+1)$. For the unperturbed system at time $t+N+1$ predicted at time t if k_f is applied at time $t+N$, it holds that

$$\begin{aligned}
x^\star(N+1+t|t) =& f(x^\star(N+t|t), u^\star(N+t|t)), \\
u^\star(N+t|t) =& k_f(x^\star(N+t|t)).
\end{aligned}$$

Due to (4.15) it holds that

$$\bar{x}(t+N+1|t+1) = x^\star(t+N+1|t) + w, \; w \in \mathcal{W}_N.$$

Therefore with (4.10) and $x^\star(t+N|t) \in \mathcal{X}_f$ from (4.1f), the recursive satisfaction of the terminal constraint can be shown with

$$\bar{x}(t+N+1|t+1) = x^\star(t+N+1|t) + w \in \mathcal{X}_f \; \forall \; w \in \mathcal{W}_N.$$

□

Remark 8. *Instead of choosing an approximation accuracy η, it is also possible to chose a maximum constraint tightening. This induces a bound on ϵ_x and ϵ_u. Then, η must be chosen such that it satisfies the following inequalities:*

$$\eta \leq \eta_1 \qquad (4.19)$$

$$\eta \leq \eta_2 := \sqrt{\frac{c_{\delta,l}}{c_{\delta,u}}}\frac{\epsilon_x}{\lambda\|H\|_\infty} \qquad (4.20)$$

$$\eta \leq \eta_3 := \sqrt{\frac{c_{\delta,l}}{c_{\delta,u}}}\frac{\epsilon_u}{\lambda\|L_t\|_\infty k_{\max}}. \qquad (4.21)$$

Corollary 9. *If* (4.5) *from Assumption 1 is modified to*

$$\|\kappa(x,z,v) - v\| \leq \tilde{k}_{\max}\sqrt{V_\delta(x,z,v)},$$

then, the less conservative bound

$$\epsilon_u = \eta\lambda\sqrt{c_{\delta,u}}\tilde{k}_{\max}\|L_t\|_\infty$$

can be used.

Proof. Replace (4.16) in the proof for Theorem 7 by

$$\begin{aligned}&\|\bar{u}(k+t|t+1) - u^\star(k+t|t)\|\\ =&\|\kappa(\bar{x}(k+t|t+1),x^\star(k+t|t),u^\star(k+t|t)) - u^\star(k+t|t)\|\\ \leq&\tilde{k}_{max}\sqrt{V_\delta(\bar{x}(k+t|t+1),x^\star(t+1|t),u^\star(t+1|t))}\\ \leq&\tilde{k}_{\max}\sqrt{\rho^{k-1}c_{\delta,u}}\lambda\eta.\end{aligned}$$

Then the remainder of the proof for Theorem 7 can be used. □

Corollary 10. *Suppose that Assumption 1 is satisfied with* $V_\delta(x,z,v) = \|x-z\|^2_{P_r}$ *and* $\kappa(x,z,v) = K(x-z) + v$ *with* K_r, P_r *matrices parameterized by the point* $r = (z,v)$. *Assume further, that*

$$\|x-z\|^2_{P_r} \leq 1 \Rightarrow \|H(x-z)\|_\infty \leq c_1 \tag{4.22}$$

and

$$\|x-z\|^2_{P_r} \leq 1 \Rightarrow \|L_tK_r(x-z)\|_\infty \leq c_2. \tag{4.23}$$

Then we can replace ϵ_x *and* ϵ_u *from* (4.8) *by the less conservative constraints*

$$\tilde{\epsilon}_x := \eta\lambda\sqrt{c_{\delta,u}}c_1,\ \tilde{\epsilon}_u = \eta\lambda\sqrt{c_{\delta,u}}c_2.$$

Proof. (4.14) delivers

$$V_\delta(x,z,v) = \|x-z\|^2_{P_r} \leq \rho^{k-1}c_{\delta,u}\lambda^2\eta^2$$

which implies due to (4.22)

$$\|H(\bar{x}(k+t|t+1) - x^\star(k+t|t))\|_\infty \leq \sqrt{\rho^{k-1}c_{\delta,u}}\lambda c_1\eta.$$

Hence, (4.17) can be rewritten as

$$\begin{aligned} H\bar{x}(k+t|t+1) \leq & Hx^\star(k+t|t) + \|H(\bar{x}(k+t|t+1) - x^\star(k+t|t))\|_\infty \mathbf{1}_p \\ \leq & Hx^\star(k+t|t) + \sqrt{\rho^{k-1} c_{\delta,u}} \lambda c_1 \eta \mathbf{1}_p \\ \leq & (1 - \tilde{\epsilon}_{k,x} + \sqrt{\rho}^{k-1} \tilde{\epsilon}_x) \mathbf{1}_p = (1 - \tilde{\epsilon}_{k-1,x}) \mathbf{1}_p. \end{aligned}$$

The proof for $\bar{u}(k+t|t+1)$ can be done in the same way. □

This may deliver a less conservative constraint tightening, but the verification of (4.22) and (4.23) may be more challenging than the computation of the bounds in Assumption 1.

4.3.2 Closed loop stability

Due to the input disturbance, asymptotic stability of a setpoint cannot be guaranteed. Instead, we show that an RPI set $\mathcal{Z}_{RPI}$ can be stabilized. Since we do not assume control invariance of the constraint set $\mathcal{X}$, convergence can only be established for all initial conditions in some region of attraction $\mathcal{X}_{\text{feas}}$. The shape of the set $\mathcal{Z}_{RPI}$ can be characterized with the following Lemma.

Lemma 11. *([8] Lemma 7) Let the value function V_N satisfy*

$$\|x(t)\|_Q^2 \leq V_N(x(t)) \leq \gamma \|x(t)\|_Q^2 \tag{4.24}$$

$$V_N(x(t+1)) - V_N(x(t)) \leq -\|x(t)\|_Q^2 + \bar{w} \tag{4.25}$$

for all $x(t) \in \mathcal{X}_{\text{ROA}} = \{x | V_N(x) \leq V_{\max}\}$, with constants $\gamma, \bar{w}, V_{\max} \in \mathbb{R}_{>0}$. For $V_{\max} \geq \gamma \bar{w} =: V_{\text{RPI}}$, the set $\mathcal{Z}_{RPI} := \{x | V_N(x) \leq V_{\text{RPI}}\}$ is robustly stabilized for all initial conditions $x(0) \in \mathcal{X}_{\text{ROA}}$.

In Proposition 12 a continuity like property of the value function $V_N(x)$ will be established. This helps us to apply Lemma 11 in the proof of convergence.

Proposition 12. *Let Assumptions 1, 3, 4 and 5 hold. Given a $V_{\max} \in \mathbb{R}_{>0}$. Then the value function V_N satisfies for all $x(t) \in \{x | V_N(x) \leq V_{\max}\}$*

$$V_N(x(t+1)) - V_N(x(t)) \leq -\|x(t)\|_Q^2 + \alpha_{\eta, V_{\max}}(\eta)$$

where $\alpha_{\eta,V_{\max}}$ is a class $\mathcal{K}$ function with

$$\begin{aligned}\alpha_{\eta,V_{\max}}(\eta) =& \frac{1-\rho^N}{1-\rho} c_{\max}\lambda^2\eta^2 + 2\sqrt{V_{\max}c_{\max}}\frac{1-\sqrt{\rho}^N}{1-\sqrt{\rho}}\lambda\eta \\ &+2\sqrt{\alpha_f d_{\max}\rho^N}\lambda\eta + d_{\max}\rho^N\lambda^2\eta^2. \end{aligned} \tag{4.26}$$

with $c_{\max} := (\lambda_{\max}(Q) + k_{\max}^2\lambda_{\max}(R))\frac{c_{\delta,u}}{c_{\delta,l}}$ *and* $d_{\max} := \lambda_{\max}(P)\frac{c_{\delta,u}}{c_{\delta,l}}$.

Proof. Consider the candidate solution (4.13). Since this is a feasible solution, is holds that

$$\begin{aligned}V_N(x(t+1)) \leq & J_N((\bar{x}(t+1)),\bar{u}(\cdot|t+1)) \\ =& \sum_{k=1}^{N} l(\bar{x}(k+t|t+1),\bar{u}(k+t|t+1)) + V_f(\bar{x}(N+t+1|t+1)).\end{aligned}$$

The quadratic stage cost of the candidate solution (4.13) satisfies for $1 < k < N$

$$\begin{aligned}& l(\bar{x}(k+t|t+1),\bar{u}(k+t|t+1)) \\ =& \|\bar{x}(k+t|t+1)\|_Q^2 + \|\bar{u}(k+t|t+1)\|_R^2 \\ =& \|x^\star(k+t|t) + (\bar{x}(k+t|t+1) - x^\star(k+t|t))\|_Q^2 \\ &+ \|u^\star(k+t|t) + (\bar{u}(k+t|t+1) - u^\star(k+t|t))\|_R^2 \\ \leq& \|x^\star(k+t|t)\|_Q^2 + \|u^\star(k+t|t)\|_R^2 \\ &+ \|\bar{x}(k+t|t+1) - x^\star(k+t|t)\|_Q^2 \\ &+ \|\bar{u}(k+t|t+1) - u^\star(k+t|t)\|_R^2 \\ &+ 2\|x^\star(k+t|t)\|_Q\|\bar{x}(k+t|t+1) - x^\star(k+t|t)\|_Q \\ &+ 2\|u^\star(k+t|t)\|_R\|\bar{u}(k+t|t+1) - u^\star(k+t|t)\|_R. \end{aligned} \tag{4.27}$$

From (4.15) and (4.16) we can derive

$$\|\bar{x}(k+t|t+1) - x^\star(k+t|t)\|_Q^2 + \|\bar{u}(k+t|t+1) - u^\star(k+t|t)\|_R^2 \leq c_{\max}\rho^{k-1}\lambda^2\eta^2$$

with $c_{\max} := (\lambda_{\max}(Q) + k_{\max}^2\lambda_{\max}(R))\frac{c_{\delta,u}}{c_{\delta,l}}$. Given a $V_{\max} \in \mathbb{R}_{>0}$ with $V_N(x(t)) \leq V_{\max}$ it follows that

$$l(x^\star(k+t|t),u^\star(k+t|t)) \leq V_{\max}, \quad \forall k \in \{0,\ldots,N-1\}.$$

Hence, (4.27) implies

$$l(\bar{x}(k+t|t+1),\bar{u}(k+t|t+1)) \leq l(x^\star(k+t|t),u^\star(k+t|t))$$

$$+ c_{\max}\rho^{k-1}\lambda^2\eta^2 + 2\sqrt{V_{\max}c_{\max}}\sqrt{\rho}^{k-1}\lambda\eta. \tag{4.28}$$

Furthermore, with $x^\star(N+1+t|t) = f(x^\star(N+t|t),k_f(x^\star(N+t|t))$ the terminal costs satisfy

$$\begin{aligned}
&V_f(\bar{x}(t+N+1|t+1)) = \|\bar{x}(t+N+1|t+1)\|_P^2\\
=&\|x^\star(t+N+1|t) + (\bar{x}(t+N+1|t+1) - x^\star(t+N+1|t))\|_P^2\\
\leq&\|x^\star(t+N+1|t)\|_P^2 + 2\|x^\star(t+N+1|t)\|_P\|\bar{x}(t+N+1|t+1) - x^\star(t+N+1|t)\|_P\\
&+ \|\bar{x}(t+N+1|t+1) - x^\star(t+N+1|t)\|_P^2\\
\leq&V_f(x^\star(t+N+1|t)) + 2\sqrt{\alpha_f d_{\max}\rho^N}\lambda\eta + d_{\max}\rho^N\lambda^2\eta^2
\end{aligned} \tag{4.29}$$

with $d_{\max} := \lambda_{\max}(P)\frac{c_{\delta,u}}{c_{\delta,l}}$ and $V_f(x^\star(t+N+1|t+1)) \leq \alpha_f$. Consider

$$\begin{aligned}
&V_N(x(t+1)) - V_N(x(t))\\
\leq&\sum_{k=1}^{N} l(\bar{x}(k+t|t+1),\bar{u}(k+t|t+1)) + V_f(\bar{x}(N+1+t|t+1))\\
&-\sum_{k=0}^{N-1} l(x^\star(k+t|t),u^\star(k+t|t)) - V_f(x^\star(N+t|t))\\
\overset{(4.28)}{\leq}& l(\bar{x}(N+t|t+1),\bar{u}(N+t|t+1)) + V_f(\bar{x}(N+1+t|t+1))\\
&+\sum_{k=1}^{N-1}\left(c_{\max}\rho^{k-1}\lambda^2\eta^2 - 2\sqrt{V_{\max}c_{\max}}\sqrt{\rho}^{k-1}\lambda\eta\right)\\
&- l(x^\star(t|t),u^\star(t|t)) - V_f(x^\star(N+t|t)).
\end{aligned}$$

With (4.11), (4.28), (4.29) and $u^\star(t+N|t) = k_f(x^\star(t+N|t))$ where

$x^\star(t+N|t) \in \mathcal{X}_f$ this can be modified to

$$\begin{aligned}
&V_N(x(t+1)) - V_N(x(t)) \\
\overset{(4.28),(4.29)}{\leq}\ & l(x^\star(N+t|t),u^\star(N+t|t)) + V_f(x^\star(N+1+t|t+1)) \\
&- l(x^\star(t|t),u^\star(t|t)) - V_f(x^\star(N+t|t)) \\
&+ \sum_{k=1}^{N} \left(c_{\max}\rho^{k-1}\lambda^2\eta^2 - 2\sqrt{V_{\max}c_{\max}}\sqrt{\rho}^{k-1}\lambda\eta \right) \\
&+ 2\sqrt{\alpha_f d_{\max}\rho^N}\lambda\eta + d_{\max}\rho^N\lambda\eta \\
\leq & V_f(x^\star(N+1+t|t+1)) + l(x^\star(N+t|t),k_f(x^\star(N+t|t)) \\
&- V_f(x^\star(N+t|t)) - l(x^\star(t|t),u^\star(t|t)) + \alpha_{\eta,V_{\max}}(\eta) \\
\overset{(4.11)}{\leq} & - l(x^\star(t|t),u^\star(t|t)) + \alpha_{\eta,V_{\max}}(\eta).
\end{aligned}$$

Note that

$$l(x(t),u(t)) = \|x\|_Q^2 + \|u\|_R^2 \geq \|x\|_Q^2.$$

Therefore it holds that

$$V_N(x(t+1)) - V_N(x(t)) \leq -\|x\|_Q^2 + \alpha_{\eta,V_{\max}}(\eta). \tag{4.30}$$

□

Another prerequisite for the proof of stability of the closed loop system (3.3) is a bound on $V_N(x)$ in terms of $\|x\|_Q^2$. This bound is established in the following proposition:

Proposition 13. *Let Assumption 5 hold. Given $V_{\max} \in \mathbb{R}_{>0}$. Then for all $V_N(x) \leq V_{\max}$ it holds that*

$$\|x\|_Q^2 \leq V_N(x) \leq \gamma\|x\|_Q^2$$

with $\gamma := \lambda_{\max}(P/Q) max \left\{ \frac{V_{\max}}{\alpha_f}, 1 \right\}$.

Proof. For all $x \in \mathcal{X}_f$, it holds that

$$\|x\|_Q \leq V_N(x) \leq V_f(x) = \|x\|_P^2 \leq \lambda_{\max}(P/Q)\|x\|_Q^2.$$

With

$$\|x\|_Q^2 \lambda_{\max}(P/Q) \geq \|x\|_P^2$$

and Assumption 5, it follows for all $x \notin \mathcal{X}_f$,

$$\|x\|_Q^2 > \frac{\alpha_f}{\lambda_{\max}(P/Q)}.$$

Together with $V_N(x) \leq V_{\max}$, this implies

$$V_N(x) \leq \frac{V_{\max}}{\alpha_f} \lambda_{\max}(P/Q) \|x\|_Q^2.$$

As a result, it holds that

$$V_N(x) \leq \|x\|_Q^2 \underbrace{\lambda_{\max}(P/Q) \max\left\{\frac{V_{\max}}{\alpha_f}, 1\right\}}_{:=\gamma}. \tag{4.31}$$

The lower bound follows direct from the definition of $V_N(x)$. □

Now we are ready to state the theorem which guarantees closed loop convergence to $\mathcal{Z}_{RPI}$:

Theorem 14. *Let Assumptions 1, 3, 4 and 5 hold. Given $V_{\max} \in \mathbb{R}_{>0}$. Then for the closed loop system (3.3) the set $\mathcal{Z}_{RPI} := \{x | V_N(x) \leq V_{\text{RPI}}\}$ with $V_{\text{RPI}} := \alpha_{\eta,V_{\max}}(\eta)\gamma \leq V_{\max}$ is stabilized for all initial conditions $x(0) \in \mathcal{X}_{\text{ROA}} = \{x | V_N(x) \leq V_{\max}\} \subseteq \mathcal{X}_{\text{feas}}$ and all disturbances $d(t) \in \mathcal{W}$.*

Proof. By Proposition 12, (4.30) holds for all initial conditions $x(0) \in \mathcal{X}_{\text{ROA}} = \{x | V_N(x) \leq V_{\max}\}$. From the initial feasibility follows as proven in Theorem 7 the recursive feasibility. Take $\alpha_{\eta,V_{\max}}$ from Proposition 12 and γ form Proposition 13. Since $V_N(x(t+1)) - V_N(x(t)) \leq -\|x\|_Q^2 + \alpha_{\eta,V_{\max}}(\eta)$ and $\|x\|_Q^2 \leq V_N(x(t)) \leq \gamma \|x(t)\|_Q^2$, Lemma 11 can be used. Hence, the set $\mathcal{Z}_{\text{RPI}} := \{x | V_N(x) = V_{\text{RPI}}\}$ with $V_{\text{RPI}} := \alpha_{\eta,V_{\max}}(\eta)\gamma$ is stabilized for all initial conditions $x(0) \in \mathcal{X}_{\text{ROA}} = \{x | V_N(x) \leq V_{\max}\}$ and all disturbances $d(t) \in \mathcal{W}$. □

Corollary 15. *The size of $\mathcal{Z}_{\text{RPI}}$ from Theorem 14 depends on the input error bound η. If η is chosen small enough, $\mathcal{Z}_{\text{RPI}} \in \mathcal{X}_f$ and asymptotic stability of the origin can be guaranteed by applying the terminal controller once the system state enters the terminal set.*

Proof. Due to (4.31) it holds that

$$\mathcal{Z}_{\text{RPI}} \subseteq \tilde{\mathcal{Z}} := \left\{ x \in \mathbb{R}^n \middle| \; \|x\|^2 \leq \frac{\alpha_{\eta,V_{\max}}(\eta)\gamma}{\lambda_{\max}(Q)} \right\}.$$

Furthermore, it holds that

$$\mathcal{X}_f \supseteq \tilde{\mathcal{X}}_f := \left\{ x \in \mathbb{R}^n \middle| \; \|x\|^2 \leq \frac{\alpha_f}{\lambda_{\min}(P)} \right\}.$$

The condition

$$\alpha_{\eta,V_{\max}}(\eta) < \frac{\lambda_{\max}(Q)\alpha_f}{\lambda_{\min}(P)\gamma} \tag{4.32}$$

implies

$$\mathcal{Z}_{\text{RPI}} \subseteq \tilde{\mathcal{Z}} \subseteq \tilde{\mathcal{X}}_f \subseteq \mathcal{X}_f.$$

Since $\alpha_{\eta,V_{\max}}$ is a class $\mathcal{K}$ function in η, $\eta > 0$ can be chosen such that $\alpha_{\eta,V_{\max}}$ can get arbitrary small and hence (4.32) holds. Thus, if the disturbance is small enough, $\mathcal{Z}_{RPI}$ lies in $\mathcal{X}_f$ and the system converges to the interior of $\mathcal{X}_f$. Inside $\mathcal{X}_f$, the terminal controller can be applied to achieve asymptotic stability. $\square$

5 Approximation of the RMPC

In this chapter, we state an approach to learn a control law using supervised learning in detail. First we recap useful results from machine learning theory to approximate the RMPC. Then, we introduce a probabilistic validation method for the resulting controller.

The chapter is structured as follows: In Section 5.1 we present machine learning in general. In Section 5.2, we recap important results from the theory of NN and discuss how a NN can be used to learn the RMPC. Since it is challenging to verify a bound on the approximation error for a learned control law, in Section 5.3 we present a validation method based on Hoeffding's Inequality. With this validation method, it is possible to deliver a probabilistic bound on the approximation error of the learned control law and hence to deliver guarantees on stability and constraint satisfaction for the AMPC what will be the subject of the Chapter 6.

5.1 Machine learning

This section covers the topic of machine learning in general. According to [31], machine learning can be described as the process of how a machine can learn specific tasks by an algorithm to obtain a general rule. This rule has to work on examples which were used to learn and has to be generalizable to previously unseen, new examples. According to [32] this is based on data and not on modeling the rule. [32] distinguishes between three types of learning. For supervised learning, learning data contains explicit example inputs and corresponding outputs for the desired task. Reinforcement learning is based on input data without the target output. Instead, it contains some possible output together with a measure how good that output is. Unsupervised learning does not depend on output information for the learning data. Therefore it is rather a clustering of inputs which may belong together. In the following, we focus on the supervised learning problem since we can compute a unique output for an arbitrary number of inputs $x \in \mathcal{X}_{\text{feas}}$ for the RMPC algorithm (4.1).

The supervised learning problem

We define the supervised learning problem similar to [32] as follows:

Definition 16. *(Supervised learning) There is an input $x \in \mathcal{X}_{\text{feas}}$, an output $u \in \mathcal{U}$ and an unknown target function $\pi_{\text{MPC}} : \mathcal{X}_{\text{feas}} \to \mathcal{U}$ where $\mathcal{X}_{\text{feas}}$ is the input space and $\mathcal{U}$ is the output space. There is a data set $\mathcal{D}$ of Π samples of inputs and corresponding outputs (x_i,u_i) with $u_i = \pi_{\text{MPC}}(x_i)$, $i = 1,...,\Pi$. Furthermore, there is a learning algorithm that uses the data set $\mathcal{D}$ to pick a function $\pi_{\text{approx}} : \mathcal{X}_{\text{feas}} \to \mathcal{U}$ that approximates π_{MPC} from a hypothesis set $\mathcal{H}$. The process of choosing π_{approx} is called supervised learning.*

Clearly, we can produce an arbitrary number of samples $(x,\pi_{\text{MPC}}(x))$ to use them for the learning step.

According to [31], we distinguish further between regression and classification. For classification problems, the output space contains a finite number of elements, the classes. Regression problems have an output space with infinite many elements. The problem of learning an MPC is a regression problem since the desired output π_{MPC} is continuous valued. We focus in this thesis on NN which can be used for both, regression and classification.

Other function approximator that can learn essentially any nonlinear function arbitrary well are for example Gaussian processes [33] and support vector machines [34]. They could be used in our AMPC framework as well instead of NN.

5.2 Neural networks

In this section we recap NN for regression according to [35, 36]. NN can be learned to deliver a function π_{approx} which can approximate any nonlinear function with finitely many discontinuities arbitrary well [3] without a priori knowledge of the approximated function. Hence NN are suited for the learning task. We recap some basic results for NN in the next subsection.

5.2.1 Neural network basics

NN are networks of simple computation units called neurons. The outputs of the neurons are connected to the inputs of other neurons which enables the network to perform complex computations. This principle is inspired from the human brain. A NN can be described as graph $G = (V,E)$ with neurons as nodes V and links between the neurons as edges E. We focus on

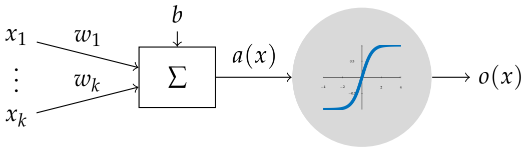

Figure 5.1: Neuron model .

feedforward NNs which do not contain cycles in the underlying graph. The restriction on feedforward NN simplifies the learning process. Furthermore, this architecture is sufficient for an arbitrary small approximation accuracy of the NN.

The neuron model

Each neuron is modeled with a simple scalar transfer function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$, called the activation function. Appropriate choices for the activation function are e.g.

$$\sigma(a) = sign(a), \sigma(a) = \begin{cases} a & a \geq 0 \\ 0 & \text{else} \end{cases}, \sigma(a) = \frac{1}{1 + exp(-a)}.$$

The last one is a smooth approximation of the threshold function $\sigma(a) = \frac{1}{2}(1 + sign(a))$. The input of a neuron consist of a weighted sum of the outputs of all neurons connected to it. Furthermore, the input of each neuron can have additional bias b that is added to the sum of weighted inputs. The output o of the neuron is computed with the activation function as follows:

$$o = \sigma(\sum_{i=1}^{k}(w_i x_i) + b)$$

where w_i are the edge weights and x_i are the outputs of the previous neurons. w_i and b are the optimization variables in the learning task. A sketch of this neuron model is given in Figure 5.1.

The network structure

We assume that the network is organized in Λ layers. Since we consider only feedforward NN, the outputs of the nodes of layer i are only connected to the inputs of the nodes of layer $i+1$. For simplicity, we consider fully connected feedforward NN. Fully connected means that each neuron of a layer has a connection to each neuron of the next layer. This simplifies the description of the network. The set of nodes can hence be decomposed into Λ disjoint subsets $V_i,\ i = 1,...,\Lambda$. The layer which produces the NN output is called the output layer. All other layers are called hidden layers. In [35] the inputs are referred to as input layer and modeled as neuron. We omit this here as done in [36].

Each layer V_i has a weight matrix W_i that connects the outputs from V_{i-1} to the inputs of V_i, a bias vector $\bar{b}_i$ and an output vector $\bar{o}_i$. Let the i^{th} layer consist of j neurons. Therewith we can define the output vector

$$\bar{o}_i = \bar{\sigma}_i(W_i \bar{o}_{i-1} + \bar{b}_i),$$

of layer i recursively, where

$$\bar{\sigma}_i(x) = \Big[\sigma_{i,1}(x_1), ..., \sigma_{i,j}(x_j)\Big]^\top$$

is the vector of the activation functions $\sigma_{i,j}$ of the i^{th} layer and $\bar{o}_0$ is the NN input. The NN output is $\bar{o}_\Lambda$. The activation of V_i is $A_t = (W_i \bar{o}_{i-1} + \bar{b}_i)$. As a result, we get the network representation $(V,E,\bar{\sigma}_1,...,\bar{\sigma}_\Lambda,W_1,...,W_\Lambda,\bar{b}_1,...,\bar{b}_\Lambda)$.

Universal approximation

The following theorem guarantees that a NN satisfying any desired approximation accuracy exists, if the RMPC output has only finitely many discontinuities. For the theorem, we need the following definition:

Definition 17. *A function σ is sigmoidal if*

$$\sigma(a) \to \begin{Bmatrix} 1 & as\ t \to +\infty \\ 0 & as\ t \to -\infty \end{Bmatrix}. \tag{5.1}$$

Therewith, we can state the theorem as follows:

Theorem 18. *[3] A feedforward network with only one hidden layer with sigmoidal activation functions and linear activation functions at the output layer can approximate any nonlinear function with finitely many discontinuities arbitrary well.*

This is called the universal approximator theorem that yields a universal approximation property. Hence, we can find a NN that satisfies

$$\left\|\pi_{\text{approx}}(x) - \pi_{\text{MPC}}(x)\right\|_{\infty} \leq \eta, \forall x \in \mathcal{X}_{\text{feas}}, \tag{5.2}$$

and thus guarantees stability and constraint satisfaction in combination with the RMPC for any choice of η. However, to achieve a certain quality of approximation, a large amount of neurons and sampled input data may be necessary. Also, the optimal weights must be found what can be challenging in the optimization step.

5.2.2 Learning neural networks

In this subsection, we recap results for the learning of neural networks with supervised learning. This process is also often referred to as training.

The learning for NN has to be such, that it can be done in an automated fashion. Precomputed learning samples $(x,\pi_{\text{MPC}}(x))$ can be used. The widespread learning methods for NN are based on optimization using the gradient of a performance criterion with respect to parameters of the NN for some learning data. We recap in this section how the gradient can be computed for the mean squared error as performance function and how the gradient can be used for the optimization to achieve a NN with good performance.

To keep the learning effort tractable, we define the hypothesis set $\mathcal{H}$ by fixing V, E, and $\bar{\sigma}_1,...,\bar{\sigma}_\Lambda$ before we start the gradient computation. Therefore, $(V, E, \bar{\sigma}_1, ..., \bar{\sigma}_\Lambda)$ is called the architecture of the network [35]. We assume that the the RMPC algorithm has only finitely many discontinuities. Then an architecture with one sigmoidal hidden layer and a linear output layer can be sufficient to satisfy Assumption 4 for any choice of η. Nevertheless, an other network structure could be easier to learn. Often, multilayer networks can be learned faster and need less optimization parameters and hence less operations to evaluate them. This can also make it possible to use the resulting controller on cheap hardware. There is no general theory how to choose the optimal network structure. One possibility to derive the structure is Bayesian optimization (see e.g. [37] and [38]). Alternatively a line search can be used, if offline computational time requirements play a subordinate role. An example for a possible network structure is given in Figure 5.2. The remaining degrees of freedom if the network architecture is fixed are the weight matrices and bias $W_1,...,W_\Lambda, \bar{b}_1, ..., \bar{b}_\Lambda$. A certain choice of the weight

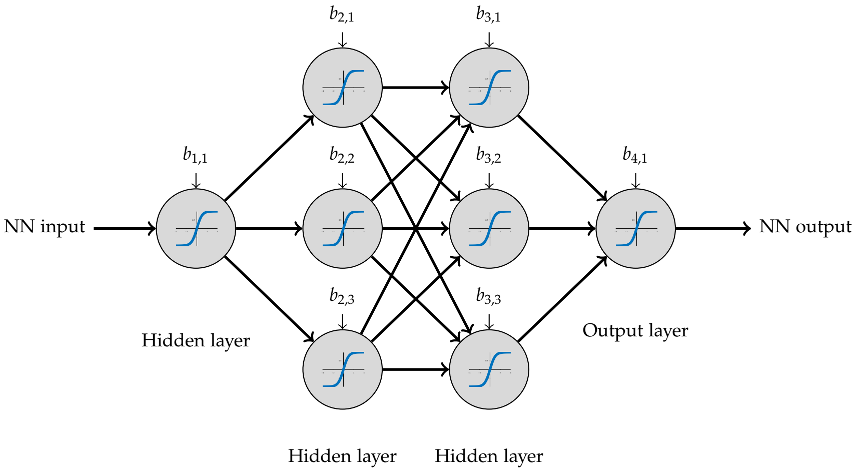

Figure 5.2: Example for a NN with three hidden layers.

matrices delivers one hypothesis from the hypothesis set $\mathcal{H}$. A performance measure for the network is given by the mean squared error (*mse*), which is defined as

$$mse(\pi_{\text{approx}}, \mathcal{D}) = \frac{1}{\Lambda} \sum_{i=1}^{\Lambda} (\pi_{\text{MPC}}(x_i) - \pi_{\text{approx}}(x_i))^2, \ (x_i, \pi_{\text{MPC}}(x_i)) \in \mathcal{D}.$$

The parameters $W_1, ..., W_\Lambda, \bar{b}_1, ..., \bar{b}_\Lambda$ are initialized randomly. Then the network is learned by optimizing the values of weight matrices and bias to minimizing the *mse* what enhances the network performance. To find good parameters $W_1, ..., W_\Lambda, \bar{b}_1, ..., \bar{b}_\Lambda$, gradient based optimization can be used.

The gradient or respectively the Jacobi matrix of a NN can be computed with backpropagation [35, pp. 278-281]. With the resulting gradient, any standard numerical optimization can be used to learn the NN. Indeed there are some methods that have shown excellent performance for neural networks. A list of possible algorithms and benchmarks for algorithms are given in [36].

The optimal learning algorithm depends heavily on the learning task.

In [36] this issue is investigated by experiments on sample recursion and classification problems for different learning algorithms. Their result is that the Marquardt-Levenberg algorithm [39] delivers the smallest approximation error for regression problems and is hence especially suitable, if very accurate learning is required. This algorithm approximates Newton's Method. Newton's Method is a second order method and needs hence the Hessian matrix. The Marquardt-Levenberg algorithm approximates the Hessian matrix by the squared gradient of the NN. However, for networks with a large number of weights the Marquardt-Levenberg algorithm may be computational inefficient since its computational effort increases geometrically. Then other algorithms like the scaled conjugate gradient algorithm [40] may be better.

We can spend some computational effort in learning the neural network and need an approximation error that satisfies a predefined bound. Furthermore, the RPI set $\mathcal{Z}_{RPI}$ of the RMPC algorithm gets smaller if the approximation error of the NN decreases. Hence it can be advantageous to spend more offline computation time for the learning to derive a better learned controller. Therefore we use the Marquardt-Levenberg algorithm. Now we discuss some details of how to facilitate the learning and to improve the approximation accuracy.

The learning of the NN is iteratively repeated until the *mse* of the NN for a set of validation data, that is independent from the learning data set, begins to increase or until the difference of the performance between two iterations is below a user defined bound. Using the validation data ensures the avoidance of overfitting of the NN to the set of learning data.

Remark 19. *For systems with m multiple inputs, i.e. if $u \in \mathbb{R}^m$, a NN can be computed for each input separately. This can have benefits in the learning. Furthermore, the computation of the AMPC feedback can then be parallelized by using m separate computation units.*

Incremental learning and batch learning

There are different ways to learn NN [36]. One possibility is incremental learning, which updates the weights and biases at each time an input/output pair is presented to the network only for this pair. Another learning method is batch learning where the learning is performed after all Π inputs and outputs for the network that shall be used for the learning are present at once. We focus in this thesis on batch learning. Incremental training may be necessary if the learning data exceeds the memory limits of the computer. Furthermore, depending on the used learning procedure, Mini Batch learning, where only a part of the Π samples is used, may improve the learning performance. For details see e.g. [41].

Initializing the optimization parameters

Another important issue concerning the learning of NN is the initialization of the weights and biases. It can happen that the weights and biases of the network converge to a local but not global minimum [36]. In order to avoid such local minima and ensure a smaller approximation error, the learning procedure is started multiple times using random initial weights. This process is repeated until a NN with sufficiently small approximation error is found. The performance is validated with samples which are independent from the learning samples. If no such network can be found, the network architecture must be changed, e.g. using Bayesian optimization, or by using more learning samples (e.g. created by a more dense sampling).

Preprocessing

The performance of the NN learning can be enhanced by preprocessing and postprocessing steps [36]. If sigmoidal transfer functions are used, the activation of a neuron becomes saturated for a large input. Then, the gradient gets small, which causes a slow learning. To avoid this, a preprocessing step can be performed to normalize the input and output of the supervised learning problem. We normalize the neural NN inputs as well as the NN outputs such that the learning data falls in the range of $[-1,1]$. For details on preprocessing and more complex methods see e.g. [36].

Sampling the RMPC algorithm

To learn a NN, we need learning data that contains enough information about the control law π_{MPC} on the whole feasible set of the MPC. This learning data can be obtained by sampling the RMPC.

One simple possibility is sampling with a uniform grid. The question that arises is how dense this grid needs to be. To keep the computational effort tractable, it is important to sample not with too high resolution. In general the question for the optimal density is hard to answer and depends heavily on the system to control. The density can be estimated empirically by

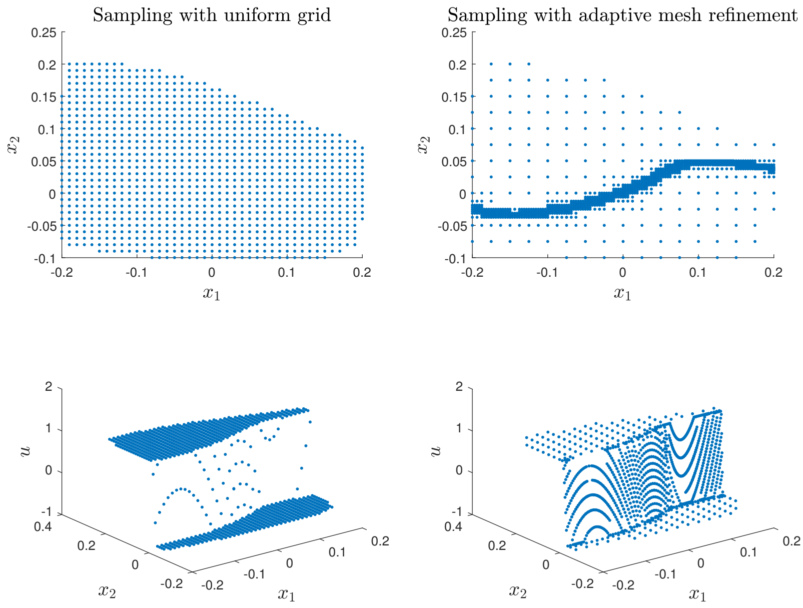

Figure 5.3: Comparison of different sampling strategies for the numerical example from Chapter 7.

trying to learn the MPC algorithm for a small region with several different densities and to choose the sparsest density for which a NN that satisfies Assumption 4 can be found.

Another sampling strategy is adaptive mesh refinement. For this strategy, the set $\mathcal{X}$ is divided into a user defined number of polytopes of the same size. The RMPC is sampled for all corner points of the polytope. If the output error between two corner points exceeds a user defined bound, the polytope is partitioned into n^2 smaller polytopes of equal size. The aim of this strategy is to decrease the number of required samples. It has resemblances to [42]. A comparison for adaptive mesh refinement and a uniform grid is given in Figure 5.3, in which both strategies are used with the same number of total samples for an example system. Note that infeasible samples are discarded for both methods.

MPC tailored learning

To enhance the learning, it can be helpful to classify active input constraints and learn the NN only where no input constraint is active. In case of multidimensional inputs, this may only be simple if the input constraints are not coupled. Then, this method is done for each input separately. In a first step, a sampling with a uniform grid is used. With the resulting samples, a classifier can be trained to classify the samples as infeasible samples, feasible samples with an active input constraint (one class for each active constraint) and feasible samples where no constraint is active. For this purpose, a NN with softmax activation functions in the output layer and one output neuron for each class can be used [36]. It is important that this classification NN generalizes well, i.e. there is no overfitting to the learning data. Then, each output of the classification NN represents the probability for the affiliation of its input to one certain class.

Using this classification, the learning is only performed for learning samples that belong with a certain, user defined probability, e.g. over 0.1%, to the feasible samples without active constraint. This increases the learning speed since only fewer learning samples are used and a smaller region has to be covered by the NN. The AMPC can then be computed as follows: First, the state is classified with the classifier. If the probability for the state to belong to the feasible class without active constraint is high (e.g. 1%, if previously 0.1% was chosen), then the output must be computed with the regression NN. Otherwise, the output is computed based on the classification result by choosing the constrained output for which the classifier delivers the highest probability. Details on the performance of this technique are still open to investigation.

5.3 Hoeffding's Inequality for validation

In this Section, we present a method to validate π_{approx} based on Hoeffding's Inequality. Equation (5.2) implies

$$\pi_{\text{approx}}(x) = \pi_{\text{MPC}}(x) + d$$

with $\|d\|_\infty \leq \eta$. Thus, stability and constraint satisfaction are guaranteed by Theorem 7 and Theorem 14 if (5.2) holds. To validate (5.2), we consider trajectories of the system (3.3), which is controlled by the approximate MPC.

We introduce

$$X_i := \{x(t), t \in \{0, \dots, T_i\} : x(0) = x_i \in \mathcal{X}_{\text{feas}}, x(T_i) \in \mathcal{X}_f \\ \text{and } x(t+1) = f(x(t), \pi_{\text{approx}}(x(t)))\} \tag{5.3}$$

to denote a trajectory of (3.3) starting at $x(0) = x_i \in \mathcal{X}_{\text{feas}}$ and ending in $\mathcal{X}_f$, where we can guarantee stability and constraint satisfaction with the terminal controller.

Then, let

$$I(X_i) := \begin{cases} 1 & \text{if } \|\pi_{\text{MPC}}(x) - \pi_{\text{approx}}(x)\|_\infty \leq \eta,\ \forall x \in X_i \\ 0 & \text{otherwise} \end{cases}$$

be an indicator function, which indicates weather a learned control law π_{approx} satisfies the posed accuracy η along a trajectory until the terminal set is reached.

For the validation, we consider p trajectories X_j, $j = 1,...,p$ with initial conditions $x(0)$ independently sampled from some distribution Ω over $\mathcal{X}_{\text{feas}}$. Because the initial conditions are independent, identically distributed (iid), also X_j and thus $I(X_j)$ are iid. Next, we state a statistical bound for the approximation accuracy of π_{approx} along iid trajectories (5.3). Define the empirical risk as

$$\tilde{\mu} := \frac{1}{p} \sum_{j=1}^{p} I(X_j). \tag{5.4}$$

The RMPC guarantees stability and constraint satisfaction if

$$I(X_i) = 1, \forall X_i \text{ with } x(0) = x_i \in \mathcal{X}_{\text{feas}} \tag{5.5}$$

holds. The probability for $I(X_i) = 1$ is $\mu := \mathbb{P}[I(X_i) = 1]$ for X_i with iid initial condition $x_j(0) \in \mathcal{X}_{\text{feas}}$ from the distribution Ω. Thus, μ is the probability hat the approximation error satisfies (5.2) along a trajectory and hence a lower bound for the probability of stability and constraint satisfaction for a randomly chosen trajectory. We can use Hoeffding's Inequality to estimate μ from the empirical risk $\tilde{\mu}$:

Lemma 20. *(Hoeffding's Inequality [4],[31, pp. 667-669])*
Let $I(X_j)$ $j = 1,...,p$ be p iid random variables with $0 \leq I(X_j) \leq 1$. Then,

$$\mathbb{P}\left[|\tilde{\mu} - \mu| \geq \epsilon_h\right] \leq 2\exp\left(-2p\epsilon_h^2\right). \tag{5.6}$$

Denote $\delta_h := 2\exp\left(-2p\epsilon_h^2\right)$ as the confidence level. Then (5.6) implies that with confidence of at least $1-\delta_h$,

$$\mathbb{P}[I(X_i)=1] = \mu \geq \tilde{\mu} - \epsilon_h. \tag{5.7}$$

Hence, with confidence $1-\delta_h$, the probability that the approximation error is below the chosen bound η along a trajectory with a random initial condition from Ω is larger than $\tilde{\mu}-\epsilon_h$.

We can use this to establish a validation method to guarantee a chosen bound $\mu_{\text{crit}} \overset{!}{\leq} \mathbb{P}\left[I(X_i)=1\right]$ and a chosen confidence δ_h. If, for a number of samples p, the empirical risk $\tilde{\mu}$ satisfies

$$\mu_{\text{crit}} \leq \tilde{\mu} - \epsilon_h = \tilde{\mu} - \sqrt{-\frac{\ln\left(\frac{\delta_h}{2}\right)}{2p}}, \tag{5.8}$$

we can rewrite (5.7) as $\mathbb{P}\left[I(X_i)=1\right] \geq \mu_{\text{crit}}$, which holds at least with confidence level $1-\delta_h$. We use this for validation as follows: for chosen desired confidence δ_h and μ_{crit}, we compute $\tilde{\mu}$ and ϵ_h for a given number p of samples. If (5.8) holds for this p, the validation is successful. If (5.8) does not hold for this p, the number of samples for the validation p is increased, which decreases ϵ_h. The validation is then repeated iteratively while increasing p. We say the validation is failed, if p exceeds a maximum number of samples

$$p_{\max} \geq -\frac{ln(\frac{\delta_h}{2})}{2(1-\mu_{\text{crit}})^2}$$

and stop the validation. Then the learning has to be repeated and improved (to increase $\tilde{\mu}$). The proposed validation method is independent of the chosen learning method.

Given this validation method, a procedure to compute an approximate MPC will be presented in the next Chapter.

Remark 21. *Instead of validating Assumption 4, a different indicator function can be used to directly verify stability, constraint satisfaction or certain performance bounds. This can facilitate the learning process.*

Remark 22. *This way of validating properties based on samples with stochastic guarantees has resemblances to the ideas in [43]. The validation is independent of the approximation method. Hence, it can also be applied to other learning methods within the proposed framework and can be adapted for the use within other* *learning based control approaches to achieve guarantees on stability and constraint satisfaction.*

6 Automatic AMPC synthesis with guarantees

In this chapter, we summarize the controller synthesis and give a step by step algorithm to approximate an MPC. Section 6.1 contains Algorithm 1 to compute the AMPC and Theorem 23, which provides guarantees on stability and constraint satisfaction for the learned AMPC. In Section 6.2, the steps of the algorithm are discussed in detail.

6.1 AMPC synthesis

Suppose that the stage cost, the constraint and the prediction horizon are given based on a nominal MPC design. Then, Algorithm 1 can be used with design parameters μ_{crit}, δ_h, $p_{\max}$ and η to obtain an AMPC.

Using Algorithm 1, the following theorem guarantees stability and constraint satisfaction for a learned RMPC:

Theorem 23. *Let Assumptions 1, 3, 4 and 5 hold. Suppose that Algorithm 1 is used with suitable chosen $\eta, \mu_{\text{crit}}, \delta_h$ and $p_{\max}$ and with an approximation method that can achieve an approximation error smaller than the required bound η. Then Algorithm 1 terminates. With a confidence level of $1 - \delta_h$, the resulting AMPC ensures closed-loop stability and constraint satisfaction for a fraction of μ_{crit} of the random initial conditions distributed according to Ω.*

Proof. Algorithm 1 terminates, because we use an approximation method, that can achieve any desired bound on the approximation error, and suitable chosen design parameters for the validation. Hence, (5.8) holds and thus, Lemma 20 implies (5.7). We thus have $\mathbb{P}\left[I(X_i) = 1\right] \geq \mu_{\text{crit}}$ with confidence at least $1 - \delta_h$. That is, with confidence $1 - \delta_h$, for at least a fraction of μ_{crit} trajectories X_i, we have $I(X_i) = 1$, which implies stability and constraint satisfaction by Theorem 7 and Theorem 14. $\square$

Algorithm 1 Learn approximate model predictive controller

```
1. Show incremental stabilizability. Compute $\lambda$ (Assumptions 1 and 3).
2. Choose an accuracy $\eta$ (Assumption 4).
3. Design the RMPC:
    a) Set $\epsilon$ according to (4.8).
    b) Compute terminal ingredients $k_f$, $V_f$ and $\mathcal{X}_f$ (Assumption 5).
    c) Check weather (4.10) from Assumption 5 holds. If not, decrease
       $\eta$ (or increase $N$).
4. Learn $\pi_{\text{approx}} \approx \pi_{\text{MPC}}$, e.g. with a NN (Chapter 5).
5. Validate $\pi_{\text{approx}}$ using Hoeffding's Inequality according to Lemma 20.
6. If the validation fails, repeat the learning from step 4.
```

6.2 Automatic AMPC synthesis - step by step

Now we go step by step through Algorithm 1 and show how each step can be done numerically.

6.2.1 Step 1: Show incremental stabilizability and Lipschitz constant

In this subsection, we present how the local incremental stabilizability condition (Assumption 1) can be established numerically for compact $\mathcal{X} \times \mathcal{U}$ and how the Lipschitz constant from Assumption 3 can be computed for nonlinear systems.

Satisfaction of the local incremental stabilizability condition

To show that Assumption 1 holds and to compute $\rho, k_{\max}, c_{\delta,u}, c_{\delta,l}$ and δ_{loc} we use an approach similar to [28]. We first recap some results which deliver sufficient conditions for Assumption 1. Then we show how these conditions can be verified within the setup of the proposed framework and how the parameters for the RMPC design ρ, $c_{\delta,u}$, $c_{\delta,l}$, δ_{loc} and $k_{\max}$ can be obtained.

Assuming that f is at least twice continuously differentiable, we can write the first order Taylor-approximation of f around a point $r = (z,v) \in \mathcal{Z} = (\mathcal{X} \times \mathcal{U})$ for compact $\mathcal{Z}$ as

$$f(z + \triangle x, v + \triangle u) = f(z,v) + \underbrace{\left[\frac{\partial f}{\partial x}\right]_{(z,v)}}_{=:A_r} \triangle x + \underbrace{\left[\frac{\partial f}{\partial u}\right]_{(z,v)}}_{:=B_r} \triangle u + \Phi_r(\triangle x, \triangle u), \tag{6.1}$$

with

$$\|\Phi_r(\triangle x, \triangle u)\| \leq T(\|\triangle x\|^2 + \|\triangle u\|^2). \tag{6.2}$$

This helps us together with the following Assumption to show that a system is local incremental stabilizable:

Assumption 24. *[28, Assumption 2] For any point $r = (z,v) \in \mathcal{Z}$, there exists a matrix $K_r \in \mathbb{R}^{m \times n}$ and positive definite matrices $P_r, Q_r \in R^{n \times n}$ continuous in r such that for any point $r^+ = (z^+, v^+) \in \mathcal{Z}$ with $z^+ = f(z,v)$*

$$P_r - (A_r + B_r K_r)^\top P_{r+}(A_r + B_r K_r) \geq Q_r, \tag{6.3}$$

is satisfied.

With this assumption, we can state the following proposition:

Proposition 25. *[28, Proposition 1] Let $\mathcal{Z}$ be compact and let Assumption 24 hold. Assume that f is twice continuous differentiable on $\mathcal{Z}$. Then Assumption 1 is satisfied with*

$$V_\delta(x,z,v) = \|x - z\|^2_{P_r}, \quad \kappa(x,z,v) = v + K_r(x - z), \tag{6.4}$$

with K_r, P_r matrices based on the point $r = (z,v)$. The corresponding parameters from Assumption 1 are given by

$$\epsilon_l = \min_{r \in \mathcal{Z}} \lambda_{\min}(Q_r),\ c_{\delta,u} := \max_{r \in \mathcal{Z}} \lambda_{\max}(P_r),\ c_{\delta,l} := \min_{r \in \mathcal{Z}} \lambda_{\min}(P_r),$$

$$\rho := 1 - \frac{\epsilon_l}{2c_{\delta,u}} \in (0,1),\ k_{\max} := \max_{r \in \mathcal{Z}} \|K_r\|$$

$$\text{and } \delta_{\text{loc}} := c_{\delta,l} \min \left\{ \left(\frac{\epsilon_l}{(c_{\delta,u}(T(1 + k_{\max^2}))^2)} \right)^2, \frac{\epsilon_l}{4c_{\delta,u}(T(1 + k_{\max^2}))} \right\},$$

with Q_r from (6.3).

Thus, we need to compute matrices P_r and K_r satisfying Assumption 24 for any point within the constraints to show that Assumption 1 holds. The condition (6.3) is equivalent to

$$\begin{pmatrix} Y_r & Y_r A_r^\top + X_r^\top B_r^\top & Y_r Q^{1/2} & X_r^\top R^{1/2} \\ A_r Y_r + B_r X_r & Y_{r+} & 0 & 0 \\ Q^{1/2} Y_r & 0 & I & 0 \\ R^{1/2} X_r & 0 & 0 & I \end{pmatrix} \geq 0. \tag{6.5}$$

with $Y_r = P_r^{-1}$ and $X_r = K_r Y_r$ and can be used to verify Assumption 1 due to the following lemma:

Lemma 26. *[44, Lemma 10],[45, p. 21] Suppose that there exists matrices X_r, Y_r continuous in r, that satisfy the constraints (6.5) for all $r \in \mathcal{Z}$ and all $r^+ = (z^+, v^+) \in \mathcal{Z}$, $z^+ = f(z,v)$. Then $P_r = Y_r^{-1}$ and $K_r = X_r Y_r^{-1}$ satisfy Assumption 1.*

To obtain P_r and K_r numerically, we use a griding on the constraint set $\mathcal{X} \times \mathcal{U}$ and minimize

$$\min_{Y_r, X_r} - \log(\det(Y_{\max})) \tag{6.6}$$

subject to a constraint according to (6.5) and a constraint $Y_{\max} \geq Y_r$ for each grid point. Hence, we can verify Assumption 1 by solving multiple LMI's.

Remark 27. *Solving the LMI's only for the points of the grid is under some mild assumptions equivalent to solving it for all points, if the grid is dense enough. For details see [46, p. 260].*

In [28] constant matrices P_r and K_r are used for a numerical example. Depending on the system dynamics (3.1) and the constraints $\mathcal{X}$ and $\mathcal{U}$, parameter varying P_r and K_r with parameters continuous in r may be necessary. Such a parameter varying approach may also deliver a smaller ρ.

Remark 28. *A parameterization can be derived by replacing all nonlinearities in f by parameters $t_i(r)$ continuous in r. Then, the linearization (6.1) can be written as a linear combination of the form*

$$A_r = \sum_i A_i t_i(r),\ B_r = \sum_i B_i t_i(r)$$

and matrices A_i, B_i and the parameter varying ansatz

$$Y_r = Y_0 + \sum_i Y_i t_i(r),\ X_r = X_0 + \sum_i X_i t_i(r), \tag{6.7}$$

can be used. Thus, P_{r^+} has to be computed with the parameters $t_i(r^+)$, $r^+ = (z^+,v^+)$ with $z^+ = f(z,v)$ and for all v^+ which can occur at z^+. The coverage of all v^+ can be done by a further griding of the constraint sets. Alternatively, one parametrizes Y_r only with parameters t_i which do not depend v to avoid this further griding.

The computation of δ_{loc} and ρ with the Lipschitz bound T from (6.2) is conservative. To overcome this, it is possible to choose one (small) fixed value for δ_{loc} and compute ρ such that Assumption 1 holds. This can be done by a further griding of the set. We set

$$\rho = \max_{x,z,v,x^+,z^+,v^+} \frac{V_\delta(x^+,z^+,v^+)}{V_\delta(x,z,v)}$$

s. t.

$$\begin{aligned} x^+ =& f(x,\kappa(x,z,v)),\, z^+ = f(z,v) \\ (x,z,v) \in& \mathcal{X} \times \mathcal{Z},\, (x^+,z^+,v^+) \in \mathcal{X} \times \mathcal{Z}, \\ V_\delta(x,z,v) \leq& \delta_{\text{loc}}, \end{aligned}$$

The computation of $c_{\delta,l}, c_{\delta,u}$ and $k_{\max}$ can then be done by the same griding of the constraint set using Proposition 25. Hence, we have presented a procedure to verify Assumption 1 and to compute the necessary parameters for the RMPC design numerically.

Remark 29. *To use the less conservative bound on ϵ_u according to Remark 9, $\tilde{k}_{\max}$ can be computed as*

$$\tilde{k}_{\max} = \max_{r \in \mathcal{Z}} \left\| P_r^{-\frac{1}{2}} K_r \right\|.$$

Therewith, $\tilde{k}_{\max}$ can be determined based on the same griding of the constraint set that is used to compute $k_{\max}$.

Computation of the Lipschitz constant λ

First, we cover the widespread case of input affine systems with

$$f(x,u) = g(x) + h(x)u,$$

with a scalar u. Therefore, it holds that

$$\begin{aligned}&\|f(x,u+d)-f(x,u)\|\\ =&\|g(x)+h(x)(u+d)-g(x)-h(x)u\| = \|h(x)d\|\\ \leq&\max_{x\in\mathcal{X}}\|h(x)\|\|d\|_\infty = \lambda\|d\|_\infty \;\forall x\in\mathcal{X}.\end{aligned}$$

This implies

$$\lambda = \max_{x\in\mathcal{X}}\|h(x)\|.$$

For systems that are not input affine, the assumption can be verified by directly optimizing

$$\lambda = \max_{x_1,x_2\in\mathcal{X},u\in\mathcal{U},u+d\in\mathcal{U}} \frac{\|f(x,u+d)-f(x,u)\|}{\|d\|_\infty}.$$

In this case, the computation can be made less conservative by choosing $\|d\|_\infty \leq d_1$ for some d_1 and introducing an additional bound $\eta \leq d_1$.

6.2.2 Step 2: Set accuracy

In this step, an important design parameter is chosen. The accuracy η is chosen subject to the constraint $\eta \leq \eta_1$. A too small approximation accuracy η makes the learning more difficult. On the other hand, a too large approximation accuracy η decreases $\mathcal{X}_{\text{feas}}$. It is up to the user of the framework to decide on the choice of η.

6.2.3 Step 3: Design of the RMPC

The purpose of this subsection is to design the RMPC, and in particular to design the terminal ingredients. The constants ϵ_x and ϵ_u can be computed directly. Now, we first present a method to compute terminal ingredients V_f, k_f and $\mathcal{X}_f$ which satisfy (4.11) and (4.12). Then, we propose a method to validate that (4.10) holds for a chosen approximation accuracy η.

Terminal ingredients

In this section, we present a method to chose the terminal ingredients. Let

$$\Phi = \left[\frac{\partial f}{\partial x}\right]\Bigg|_{(0,0)} \qquad \Gamma = \left[\frac{\partial f}{\partial u}\right]\Bigg|_{(0,0)}$$

denote the linearization of f at the origin.The following Lemma guarantees the existence of terminal ingredients to satisfy (4.11) and (4.12) for stabilizable (Φ,Γ) :

Lemma 30. *[30, Lemma 1] Assume* (Φ,Γ) *is stabilizable and* $(0,0)$ *lies in the interior of* $\bar{\mathcal{X}}_N \times \bar{\mathcal{U}}_N$. *Due to the stabilizability of* (Ω,Γ), *there exists a linear controller* $u(x) = -K_{\text{term}}x$ *such that* $\Phi_c = A - \Gamma K_{\text{term}}$ *has all eigenvalues strictly inside the unit disk. Furthermore, there exists a matrix* P, *which solves the Lyapunov equation* $\Phi_c^\top P \Phi_c - P = -(Q^* + \triangle Q)$, $\triangle Q > 0$, $Q^* = Q + K_{\text{term}}^\top R K_{\text{term}}$, *such that the terminal controller* $K_f(x) = -K_{\text{term}}x$, *the terminal cost* $V_f(x) = \|x\|_P^2$ *and the terminal set* $\mathcal{X}_f = \left\{x \in \mathbb{R}^n | \|x\|_P^2 \leq \alpha_f\right\}$ *satisfy the conditions* (4.11) *and* (4.12) *in Assumption 5.*

Typically, the discrete time linear quadratic regulator (LQR) is considered as terminal controller $k_f(x) = -K_{\text{term}}x$. Now, we discuss how α_f can be chosen. First we demonstrate, how to choose α_f such that (4.12) holds. For this purpose, we use the following lemma:

Lemma 31. *[44, 45] The maximum admissible value* $\alpha_{1,f}$ *such that the terminal set* $\mathcal{X}_f := \left\{x \in \mathbb{R}^n | x^\top P x \leq \alpha_{1,f}\right\}$ *satisfies* (4.12), *can be computed with the linear program*

$$\begin{aligned} \alpha_{1,f} &= \max_{\alpha_f} \alpha_f, \\ \text{s.t.} \left\|P^{-1/2}K_{\text{term}}L_{t,i}\right\|^2 \alpha_f &\leq (1-\epsilon_{N,u})^2 \; i = 1,\ldots,q \\ \left\|P^{-1/2}H_j\right\|^2 \alpha_f &\leq (1-\epsilon_{N,x})^2, \; j = 1,\ldots,p, \end{aligned}$$

where $L_{t,i}$ *is the i-th row of* L_t *and* H_j *is the j-th row of* H.

Hence, we use this linear programming and chose $\alpha_f \leq \alpha_{1,f}$ to satisfy (4.12). An admissible value for α_f to satisfy (4.11) can be obtained by seeking a region for which

$$L_\Psi := \sup\left\{\frac{\|\Psi(x)\|}{\|x\|} | x \in \mathcal{X}_f\right\},$$

with $\Psi(x) = f(x,k_f(x)) - \Phi_c(x)$, satisfies

$$L_\Psi \leq \frac{\sqrt{\|P\Phi_c\|^2 + \lambda_{\min}(\triangle Q)\|P\|} - \|P\Phi_c\|}{\|P\|}. \tag{6.8}$$

Algorithm 2 Compute the size of the terminal set

```
1. Set $\alpha_f = \alpha_{1,f}$.
2. Compute $L_\Psi = \sup\left\{\frac{\|\Psi(x)\|}{\|x\|} | x \in \mathbb{R}^n | x^\top P x \leq \alpha_f\right\}$ e.g. using nonlinear programming.
3. If $L_\Psi > \frac{\sqrt{\|P\Phi_c\|^2 + \lambda_{\min}(\triangle Q)\|P\|} - \|P\Phi_c\|}{\|P\|}$, set $\alpha_f = \frac{\alpha_f}{2}$ and goto 2.
4. Return $\alpha_f$.
```

Using this, α_f can be computed iteratively with Algorithm 2. The detailed procedure can be reviewed in [30]. There this procedure is stated as *Method 1*. Another approach [30, *Method 2*] is to characterize the terminal set by seeking the largest possible $\alpha_f \leq \alpha_{1,f}$ such that

$$arg \max_x (-x^\top \triangle Q x + 2\Psi(x)^\top P\Phi_c x + \Psi(x)^\top P \Psi(x))$$

remains negative in $\mathcal{X}_f$. *Method 2* can be less conservative than *Method 1*, but is only relevant if $\alpha_f < \alpha_{1,f}$ after *Method 1* was used. Both methods are adoptions of the methods for continuous time systems from [18]. Now, it remains to show that (4.10) from Assumption 5 holds.

Invariance of the terminal set to input disturbances

Now, we provide a method to show that (4.10) holds. To achieve this, we use the following proposition:

Proposition 32. *Consider the disturbance $\mathcal{W}_N = \{x \in \mathcal{R}^n | \; \|x\| \leq w_N\}$. The terminal ingredients satisfy (4.10), if*

$$w_N^2 \lambda_{\max}(P) + 2w_N\sqrt{\lambda_{\max}(P)\alpha_f} \leq \frac{\alpha_f}{\gamma_{\text{term}}} \tag{6.9}$$

holds with $\gamma_{\text{term}} = \lambda_{\max}(P/Q^*)$.

Proof. Obviously,

$$\|x\|_{Q^\star}^2 \leq \|x\|_P^2 \leq \gamma_{\text{term}} \|x\|_{Q^\star}^2$$

holds. For the disturbed system, we have with $V_f(x) = \|x\|_P^2$:

$$\begin{aligned} V_f(f(x, -K_{\text{term}}x) + w) =& \|f(x, -K_{\text{term}}x) + w\|_P^2 \\ \leq& V_f(f(x, -K_{\text{term}}x)) + \|w\|_P^2 \\ &+ 2\|f(x, -K_{\text{term}}x\|_P \|w\|_P \end{aligned} \tag{6.10}$$

with $w \in \mathcal{W}_N$. Due to (4.11), it holds that

$$V_f(f(x, -K_{\text{term}}x)) \leq V_f(x) - \|x\|_{Q^\star}^2. \tag{6.11}$$

Furthermore, define

$$\|w\|_P^2 + 2\|f(x, -K_{\text{term}}x\|_P \|w\|_P \leq \|w\|_P^2 + 2\sqrt{\alpha_f}\|w\|_P := \bar{w}. \tag{6.12}$$

Plugging (6.11) and (6.12) into (6.10), we obtain

$$V_f(f(x, -K_{\text{term}}x) + w) - V_f(x) \leq -\|x\|_{Q^\star}^2 + \bar{w}.$$

With this result, we can apply Lemma 11 to show the robust stability of the terminal set $\mathcal{X}_f := \left\{x \in \mathbb{R}^n | x^\top P x \leq \alpha_f = V_{\max}\right\}$ with $V_{\max} \geq \gamma_{\text{term}}\bar{w} =: V_{\text{RPI}}$. Hence, we require $\bar{w} \leq \frac{\alpha_f}{\gamma_{\text{term}}}$. From the definition of $\mathcal{W}_N$, we know that $\|w\| \leq w_N$ which implies $\bar{w} \leq w_N^2 \lambda_{\max}(P) + 2w_N\sqrt{\lambda_{\max}(P)\alpha_f}$. Thus, if

$$w_N^2 \lambda_{\max}(P) + 2w_N\sqrt{\lambda_{\max}(P)\alpha_f} \leq \frac{\alpha_f}{\gamma_{\text{term}}} \tag{6.13}$$

is satisfied, (4.10) holds $\forall w \in \mathcal{W}_N$. □

For the RMPC, we have $w_N = \lambda\eta\sqrt{\rho^N \frac{c_{\delta,u}}{c_{\delta,l}}}$. If (6.13) does not hold, η can be decreased or N can be increased to achieve that (4.10) holds. Hence, we have demonstrated, how terminal ingredients for the RMPC can be designed to satisfy Assumption 5.

6.2.4 Step 4, 5 and 6: Learning and validation

Now we summarize the remaining steps of Algorithm 1.

Step 4: Learning

The learning step can be done by any learning method that can achieve an approximation error below η. According to Chapter 5, NN are suited. The difficulty can be to find a suitable NN architecture and to achieve convergence. It can be chosen by starting with a small network and enlarging it until the validation is successful, or with Bayesian optimization. We show how this can be done for an example system in Chapter 7.

Remark 33. *Any required bound on the approximation error can be achieved with a sufficient large NN. Choosing a suitable network structure and achieving convergence in the learning are a separate problem (compare Chapter 5).*

Step 5 Validation

The validation is straight forward. The choice of the design parameters depends on the use-cases. If μ_{crit} is close to μ, which is especially the case if μ_{crit} is close to 1, and if δ_h is close to 0, a large $p_{\max}$ may be necessary. $\tilde{\mu}$ can be computed directly by comparing the output of the learned controller with the output of the RMPC along a trajectory. Clearly, the probability distribution for the validation must be the same as the probability distribution Ω of initial conditions under which we run the AMPC later. Thus we need knowledge of Ω. Note, that the validation can be parallelized on several computers, which decreases the runtime of Algorithm 1.

7 Numerical example

In this section, we demonstrate the AMPC framework with a numerical benchmark example. We go step by step through Algorithm 1 and show how the steps of Algorithm 1 can be executed in an automated fashion. We perform simulations with the resulting controller and compare it to an LQR. The comparison emphasizes that the AMPC has performance advantages for nonlinear systems in comparison to the LQR, which is easy to compute, but does not include nonlinearities in the controller synthesis.

7.1 System modeling

We demonstrate the AMPC scheme with a continuous stirred tank reactor with

$$\dot{\tilde{x}} = f_c(\tilde{x},\tilde{u}) = \begin{pmatrix} \frac{(1-\tilde{x}_1)}{\theta} - k\tilde{x}_1 e^{-\frac{M}{\tilde{x}_2}} \\ \frac{x_f-\tilde{x}_2}{\theta} + k\tilde{x}_1 e^{-\frac{M}{\tilde{x}_2}} - \alpha\tilde{u}(\tilde{x}_2 - xc) \end{pmatrix}$$

where x_1 is the temperature, x_2 is the concentration and u is the coolant flow. This system is a common benchmark taken from[1] [9]. Since the system is input affine, it can be written in the form

$$f_c(\tilde{x},\tilde{u}) = g_c(\tilde{x}) + h_c(\tilde{x})\tilde{u}. \tag{7.1}$$

The system is discretized with a simple Euler approach. The discrete time system is given by

$$\tilde{x}^+ = f(\tilde{x},\tilde{u}) = \tilde{x} + hf_c(\tilde{x},\tilde{u}) = \begin{pmatrix} \tilde{x}_1 + h\left(\frac{(1-\tilde{x}_1)}{\theta} - k\tilde{x}_1 e^{-\frac{M}{\tilde{x}_2}}\right) \\ \tilde{x}_2 + h\left(\frac{x_f-\tilde{x}_2}{\theta} + k\tilde{x}_1 e^{-\frac{M}{\tilde{x}_2}} - \alpha\tilde{u}(\tilde{x}_2 - x_c)\right) \end{pmatrix}, \tag{7.2}$$

with sampling time $h = 0.1\ s$. We consider the unstable steady state $x_e = (0.2632, 0.6519)$ with steady state input $u_e = 0.7853$. To achieve

[1]The parameters are $\theta = 20$, $k = 300$, $M = 5$, $x_f = 0.3947$, $x_c = 0.3816$, $\alpha = 0.117$.

Table 7.1: State and input constraints

Variable	minimum Value	maximum Value
$u + u_e$	0	2
x_1	-0.2	0.2
x_2	-0.2	0.2

$f(0,0) = 0$, we use the transformed coordinates $x = \tilde{x} - x_e$ and $u = \tilde{u} - u_e$ The constraints that are considered are summarized in Table 7.1. Furthermore, we consider the stage cost $Q = I, R = 10^{-4}$ and use $N = 180$ for the prediction horizon. Now we can go step by step through Algorithm 1.

7.2 Computation of the AMPC using Algorithm 1

In this section, we demonstrate the AMPC synthesis using the proposed framework.

Step 1: Incremental stabilizability and Lipschitz constant

The set $\mathcal{Z} = \mathcal{X} \times \mathcal{U}$ is compact. Hence, the verification of the local incremental stability assumption is done according to Section 6.2.1. We use $V_\delta(x,z) = \|x - z\|^2_{P_r}$ and $\kappa(x,z,v) = v + K_r(x - z)$ with matrices[2] P_r and K_r continuous in $r = (x,u) \in \mathcal{X} \times \mathcal{U}$. To obtain P_r and K_r, we use a griding of the constraint set and an approach according to (6.7). We construct multiple LMI's containing one LMI according to (6.5) for each grid point. The LMI's can be solved with Yalmip [47]. As result, we obtain

$$c_{\delta_l} = 12.33, c_{\delta_u} = 199.03, k_{\max} = 45.72, \tilde{k}_{\max} = 3.6455, \rho = 0.9913, \delta_{\text{loc}} = 0.01.$$

The Lipschitz bound is $\lambda = 5.5 \cdot 10^{-3}$ and was computed with the method for input affine systems.

Step 2: Set accuracy

For the approximation accuracy η, we choose $\eta = 5.1 \cdot 10^{-3} \leq \eta_1 = 1.29$.

[2]For numerical values of P_r and K_r, see Appendix A.1

Step 3: Design of the RMPC

Using $\tilde{k}_{\max}$ according to Remark 9, we obtain[3] $\epsilon_x = \epsilon_u = 2.2 \cdot 10^{-3}$.

The terminal cost and controller are computed as detailed in Subsection 6.2.3, yielding $k_f(x) = -K_{\text{term}} x$ and $V_f(x) = x^\top P x$, with

$$K_{\text{term}} = \begin{pmatrix} -46.01 & 101.74 \end{pmatrix}, P = \begin{pmatrix} 33.21 & -3.61 \\ -3.61 & 6.65 \end{pmatrix}.$$

Due to the input constraints, we obtain $\alpha_{f,1} = 9.2 \cdot 10^{-5}$. Since $L_\Psi = 3.8 \cdot 10^{-3}$ for this α and $\frac{\sqrt{\|P\Phi_c\|^2 + \lambda_{\min}(\triangle Q)\|P\|} - \|P\Phi_c\|}{\|P\|} = 1.53 \cdot 10^{-2}$, Algorithm 2 terminates in one step and we obtain $\alpha_f = 9.2 \cdot 10^{-5}$. Since the input constraint is active for α_f, using *Method 2* would bring no advantage. To verify that (4.10) holds, we obtain $\gamma_{\text{term}} = 28.97$ which yields

$$\begin{aligned} \bar{w} &\leq \left(\lambda\eta\sqrt{\rho^N \frac{c_{\delta,u}}{c_{\delta,l}}}\right)^2 \lambda_{\max}(P) + 2\lambda\eta\sqrt{\rho^N \frac{c_{\delta,u}}{c_{\delta,l}} \lambda_{\max}(P)\alpha_f} \\ &= 2.938 \cdot 10^{-6} < 3.185 \cdot 10^{-6} = \frac{\alpha_f}{\gamma_{\text{term}}}. \end{aligned}$$

Hence, (4.10) holds for the chosen η and N. As a result, Assumption 5 holds. In Figure 7.1, the constraint tightening is visualized along an example trajectory.

Step 4: Learn RMPC

A fully connected feed forward NN with 2 neurons in the first hidden layer and two further hidden layers with 50 neurons in each is trained as the AMPC. The activations in the hidden layers are hyperbolic tangent sigmoid functions. In the output layer, linear activations are used. To create the learning samples, the RMPC is sampled over $\mathcal{X}$ with a uniform grid using Casadi 2.4.2 [48] on Python 2.7.2. We use a grid size of $2.5 \cdot 10^{-4}$. Therewith, $1.6 \cdot 10^6$ feasible samples of data points $(x, \pi_{\text{MPC}}(x))$ are generated for learning. The neural network is trained with the Marquardt-Levenberg algorithm using Matlabs neural network toolbox on R2017a. To

[3] For simplicity, we use $\epsilon_x = \epsilon_u = \eta\lambda\sqrt{c_{\delta,u}} \max\left\{\tilde{k}_{\max}\|L_t\|_\infty, \frac{\|H\|_\infty}{\sqrt{c_{\delta,l}}}\right\}$, which leads to a constraint tightening of $\bar{\mathcal{X}}_\infty$ and $\bar{\mathcal{U}}_\infty$ to 51.7% of the size of $\bar{\mathcal{X}}_0$ and $\bar{\mathcal{U}}_0$.

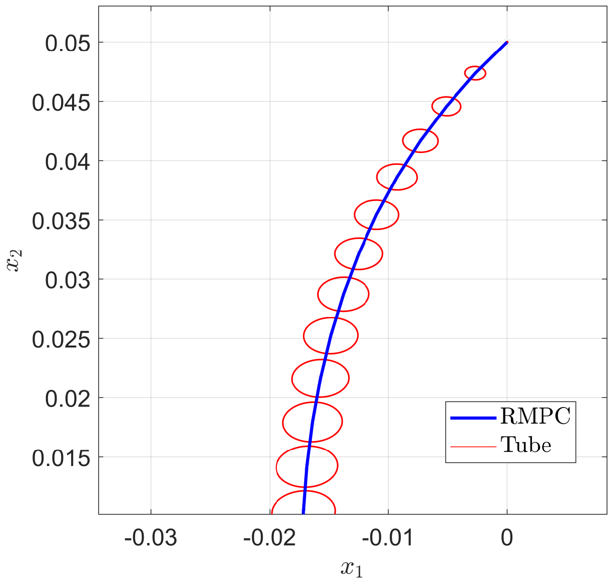

Figure 7.1: Concentration vs. temperature: Trajectory of AMPC (blue line) and growing tube (red ellipsoids). The constraint tightening is done by subtracting the ellipsoids from the untightened constraints.

enhance the learning speed, we start the learning with a reduced number of samples and enhance the number of used samples as the *mse* decreases. The resulting AMPC over the $\mathcal{X}_{\text{feas}}$ is given in Figure 7.2.

Step 5 Validation results

We choose $\delta_h = 0.01$ and $\mu_{\text{crit}} = 0.99$. As validation data, we sample initial conditions uniformly from $\mathcal{X}_{\text{feas}}$. Algorithm 1 terminates, with $\tilde{\mu} = 0.9987$ for $p = 34980$ trajectories. This implies $\mu_{\text{crit}} < \tilde{\mu} - \epsilon_h$ and hence, we achieve the desired guarantees. We thus conclude from Theorem 23 that, with a confidence of 99%, the closed-loop system (3.3) is stable and satisfies constraints for a ratio of at last 99% of random initial conditions

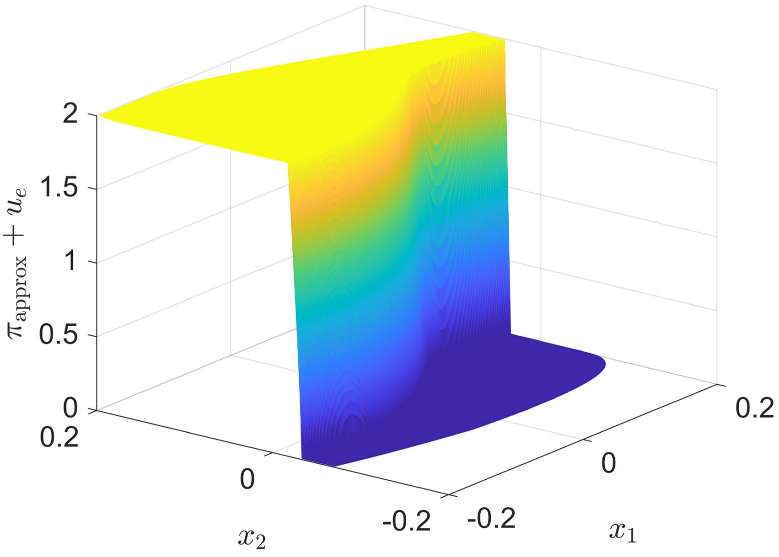

Figure 7.2: Approximate MPC π_{approx} over the feasible set $\mathcal{X}_{\text{feas}}$.

with distribution Ω. A histogram of the approximation error for 81000 of the learning samples is given in Figure 7.3. As we can see, the in-sample error is below 10^{-4} for most points, whereas $5.1 \cdot 10^{-3}$ is the bound.

The overall time to execute Algorithm 1 was roughly 500 hours on a Quad-Core PC. It is possible to significantly reduce this time, e.g., by parallelizing the sampling and the validation.

As result, we obtain an AMPC with the proposed framework in an automated fashion. In the next section, we investigate this controller applied to the benchmark system (7.2).

7.3 Simulation results

In this section, we evaluate the AMPC from Section 7.2. We compare it to an LQR controller and measure the time to evaluate the AMPC in comparison

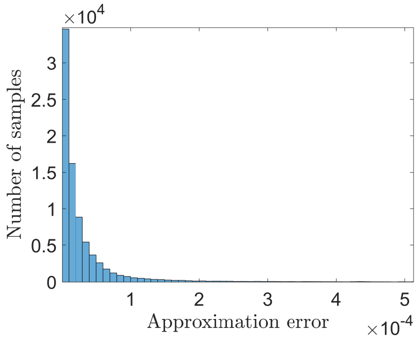

Figure 7.3: Histogram of the absolute in-sample error for a part of the learning samples

to the online optimization of the RMPC.

7.3.1 Comparison to LQR

In this subsection, we compare the performance of the AMPC to the saturated LQR that results from the same stage costs, but saturated with $\mathcal{U}$. Contrary to the AMPC, the LQR does not guarantee stability and constraint satisfaction for general nonlinear constrained systems. The convergence of both controllers applied on the system is illustrated in Figure 7.4 for some exemplary initial conditions.

While for some initial conditions, the AMPC and LQR trajectories are close (when nonlinearities play a subordinate role), there are significant differences for others. For example, in Figure 7.4 (bottom), the input flow is shown for the marked trajectory (top). Initially, the opposite input constraint is active for the LQR in comparison to the AMPC. This causes an initial divergence of the LQR trajectory leading to over three times higher costs. Due to the small approximation error, the trajectories of the AMPC are virtually indistinguishable from the original RMPC.

Remark 34. *In principle, the RMPC method from [16] could also be used within the framework. However, the admissible approximation error for this method would be at most $\eta = 10^{-41}$, which makes the learning intractable.*

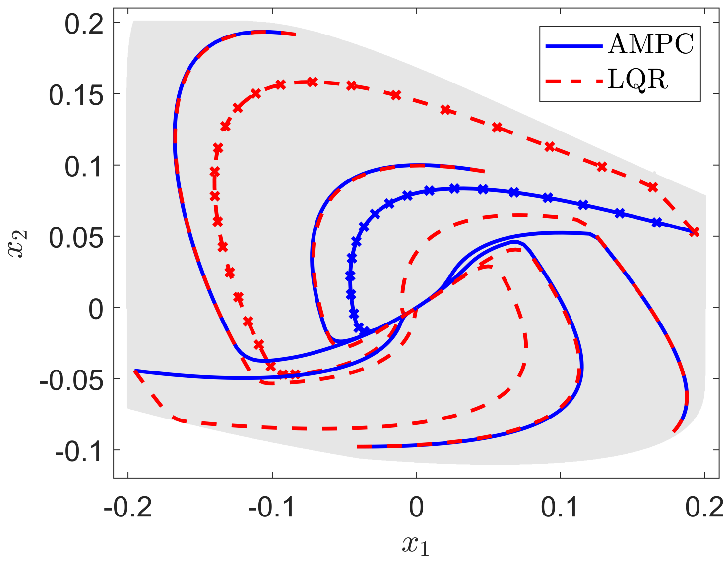

Figure 7.4: Concentration vs. temperature: Trajectories of AMPC (solid blue line) and LQR (dashed red line) and $\mathcal{X}_{\text{feas}}$ (grey area)

7.3.2 Online computational demand

To investigate the achievable sampling time of the AMPC, we evaluated the AMPC and the online solution of the RMPC optimization problem for 100 random points over $\mathcal{X}_{\text{feas}}$. For the evaluation of the RMPC, Casadi was used. The AMPC was computed with Matlab. Evaluating the AMPC was possible in an average time of $3ms$, whereas solving the optimal control problem for this initial conditions needed an average of $0.71s$. This is over 200 times faster and further speed up may be obtained by alternative NN implementations. Detailed results are given in Table 7.2.

Thus, the AMPC delivers a cost optimized, stabilizing feedback law for a nonlinear constrained example which can be evaluated online on a cheap hardware with a high sampling rate.

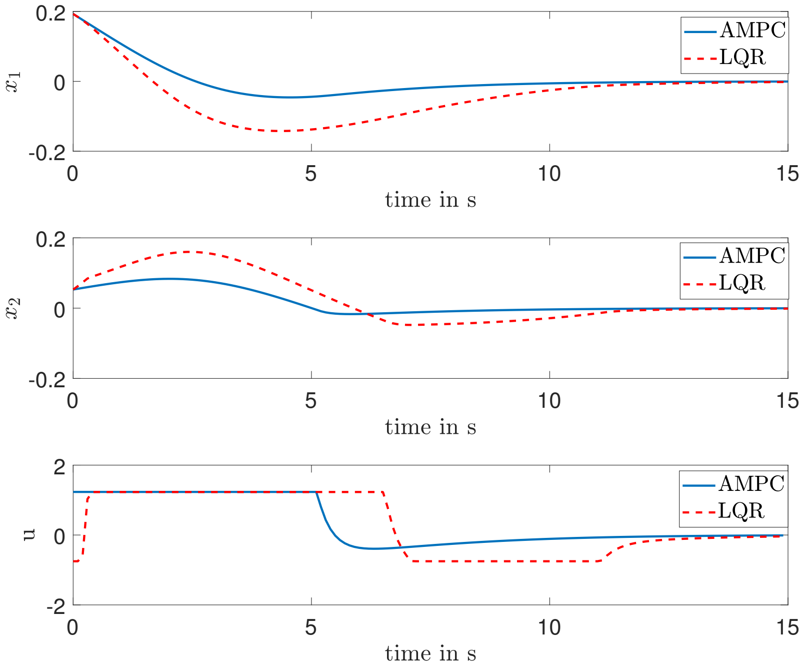

Figure 7.5: Comparison of trajectories for concentration (top), temperature (middle) and coolant flow (bottom) for AMPC (solid blue line) and LQR (dashed red line)

Table 7.2: Time to evaluate RMPC and AMPC for random initial conditions

Controller	minimum time	average time	maximum time
AMPC	$2.8ms$	$3ms$	$5.6ms$
RMPC	$0.55s$	$0.71s$	$1.22s$

8 Conclusion

This chapter summarizes the results of this thesis and gives an outlook on future work building on the results of this thesis.

8.1 Summary

The overall goal of this thesis was to develop a framework for the synthesis of an AMPC with guarantees on stability and constraint satisfaction.

Within the framework, we presented first an RMPC scheme with robustness to bounded input disturbances. A proof for robust stability and constraint satisfaction for this RMPC was given. This proof is an extension of [8].

The RMPC can be learned with any machine learning procedure. We discussed NN as a suitable method to learn the RMPC since NN have the property that they can approximate any nonlinear function with finitely many discontinuities arbitrary well [3]. The difficulty using NN was to guarantee a specific bound on the approximation error for a learned NN.

Hence, we proposed a probabilistic method for validation of the AMPC to give guarantees on stability and constraint satisfaction. This method is based on Hoeffding's Inequality [4] and is independent from the approximation method.

The combination of the RMPC with NN and the validation method yields a framework for the synthesis of an AMPC with guarantees on stability and constraint satisfaction for a wide class of nonlinear systems. An algorithm was given to use the proposed framework in an automatic fashion. We have shown step by step how this algorithm can be executed automatic with the choice of a small number of design parameters.

The practicability of the framework was demonstrated with a numerical example. Guarantees on stability and constraint satisfaction were given for the example and the AMPC could be evaluated in a short deterministic time. Hence, the proposed framework is suitable for systems with real-time requirements on cheap hardware.

The proposed methods can also be used within other learning based control approaches. Thus, the results of this thesis are also relevant for other approaches than MPC.

8.2 Future work

Future work could cover the application of robust control methods in combination with the proposed validation method to other learning based control approaches. Deriving a controller for systems of higher order with the proposed framework is still open to investigate. Designing learning techniques tailored to MPC, e.g. by exploiting the active constraints with a suitable classification as proposed in Chapter 5, is also part of future work.

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich, Michael Hertneck, die vorliegende Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäß entlehnte Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen. Weiterhin bestätige ich, dass das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

Ort, Datum	Unterschrift

Appendix

A.1 Numerical values for the verification of the local incremental stabilizability assumption

The continuous stirred tank reactor (7.2) can be written as quasi linear parameter varying system

$$x^+ = A_r x + B_r u$$

with

$$A_r = \begin{pmatrix} 1 - kht_1(r) - \frac{h}{\theta} & -kht_4(r) \\ -\frac{h}{\theta} + kht_1(r) & kht_4(r) - \alpha ht_2(r) - \frac{h}{\theta} + 1 \end{pmatrix}, B_r = \begin{pmatrix} 0 \\ -\left(\alpha h \left(t_3(r) - x_c\right)\right) \end{pmatrix}$$

using the parametrization according to Table A.1. Therewith, the solution to the multiple LMI's are

$$P_r = Y_r^{-1}, K_r = X_r Y_r^{-}1$$

with

$$Y_r = Y_0 + \sum_{i=1}^{4} t_i(r) Y_{r_i},\ X_r = X_0 + \sum_{i=1}^{4} t_i(r) X_{r_i},$$

with Y_0,Y_{r_i},X_0,X_{r_i} according to Table A.2.

Tabelle A.1: Parametrization

$t_1(r)$	$= e^{(-M/x2)}$
$t_2(r)$	$= u$
$t_3(r)$	$= x_2$
$t_4(r)$	$= \frac{x_1 M e^{\left(\frac{-M}{x_2}\right)}}{x_2^2}$

Tabelle A.2: Solution for the Multiple LMI's

$$Y_0 = \begin{pmatrix} 2.744 \cdot 10^{-3} & 5.997 \cdot 10^{-3} \\ 5.997 \cdot 10^{-3} & 3.589 \cdot 10^{-3} \end{pmatrix}$$
$$Y_{r_1} = \begin{pmatrix} 7.782 & -6.919 \\ -6.919 & 9.687 \end{pmatrix}$$
$$Y_{r_2} = \begin{pmatrix} -1.762 \cdot 10^{-5} & 3.154 \cdot 10^{-5} \\ 3.154 \cdot 10^{-5} & -5.130 \cdot 10^{-6} \end{pmatrix}$$
$$Y_{r_3} = \begin{pmatrix} -2.700 \cdot 10^{-2} & -3.589 \cdot 10^{-3} \\ -3.589 \cdot 10-3 & 7.713 \cdot 10^{-3} \end{pmatrix}$$
$$Y_{r_4} = \begin{pmatrix} 5.863 & 0.6192 \\ 0.6192 & -2.554 \end{pmatrix}$$
$$X_0 = \begin{pmatrix} -5.667 \cdot 10^{-3} & -0.4525 \end{pmatrix}$$
$$X_{r_1} = \begin{pmatrix} 3.450 & -12.967 \end{pmatrix}$$
$$X_{r_2} = \begin{pmatrix} -8.154 \cdot 10^{-4} & 8.919 \cdot 10^{-4} \end{pmatrix}$$
$$X_{r_3} = \begin{pmatrix} 1.062 \cdot 10^{-2} & 1.193 \end{pmatrix}$$
$$X_{r_4} = \begin{pmatrix} 8.196 & -10.36 \end{pmatrix}$$

Literaturverzeichnis

[1] James Blake Rawlings and David Q Mayne. *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[2] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[3] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[4] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[5] Ankush Chakrabarty, Vu Dinh, Martin J Corless, Ann E Rundell, Stanisław H Żak, and Gregery T Buzzard. Support vector machine informed explicit nonlinear model predictive control using low-discrepancy sequences. *IEEE Transactions on Automatic Control*, 62(1):135–148, 2017.

[6] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 528–535, 2016.

[7] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[8] Johannes Köhler, Matthias A. Müller, and Frank Allgöwer. A novel constraint tightening approach for nonlinear robust model predictive control. In *American Control Conference (ACC)*, 2018. to appear.

[9] David Q Mayne, Erric C Kerrigan, EJ Van Wyk, and P Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353, 2011.

[10] Alexander Domahidi, Melanie N Zeilinger, Manfred Morari, and Colin N Jones. Learning a feasible and stabilizing explicit model predictive control law by robust optimization. In *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 513–519, 2011.

[11] Tor A Johansen. Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 40(2):293–300, 2004.

[12] Alexandra Grancharova and Tor A Johansen. Computation, approximation and stability of explicit feedback min–max nonlinear model predictive control. *Automatica*, 45(5):1134–1143, 2009.

[13] Thomas Parisini and Riccardo Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10): 1443–1451, 1995.

[14] T Parisini, M Sanguineti, and R Zoppoli. Nonlinear stabilization by receding-horizon neural regulators. *International Journal of Control*, 70 (3):341–362, 1998.

[15] Bernt M Åkesson and Hannu T Toivonen. A neural network model predictive controller. *Journal of Process Control*, 16(9):937–946, 2006.

[16] Gilberto Pin, Marco Filippo, Felice Andrea Pellegrino, Gianfranco Fenu, and Thomas Parisini. Approximate model predictive control laws for constrained nonlinear discrete-time systems: analysis and offline design. *International Journal of Control*, 86(5):804–820, 2013.

[17] David Q Mayne, SV Raković, Rolf Findeisen, and Frank Allgöwer. Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42(7):1217–1222, 2006.

[18] Hong Chen and Frank Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[19] Daniel Limón, Teodoro Alamo, Francisco Salas, and Eduardo F Camacho. On the stability of constrained mpc without terminal constraint. *IEEE Transactions on automatic control*, 51(5):832–836, 2006.

[20] Lars Grüne. NMPC without terminal constraints. In *4th IFAC Nonlinear Model Predictive Control Conference (NMPC 2012)*, volume 4, pages 1–13, 2012.

[21] Matthias A Müller and Frank Allgöwer. Economic and distributed model predictive control: Recent developments in optimization-based control. *SICE Journal of Control, Measurement, and System Integration*, 10 (2):39–52, 2017.

[22] Shuyou Yu, Marcus Reble, Hong Chen, and Frank Allgöwer. Inherent robustness properties of quasi-infinite horizon MPC. In *18th World Congress of the International Federation of Automatic Control*, pages 179–184, 2011.

[23] Davide Martino Raimondo, Daniel Limon, Mircea Lazar, Lalo Magni, and Eduardo Fernandez Camacho. Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1):5–21, 2009.

[24] David Q Mayne, Maria M Seron, and SV Rakovic. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[25] Luigi Chisci, J Anthony Rossiter, and Giovanni Zappa. Systems with persistent disturbances: predictive control with restricted constraints. *Automatica*, 37(7):1019–1028, 2001.

[26] D Limon Marruedo, T Alamo, and EF Camacho. Input-to-state stable mpc for constrained discrete-time nonlinear systems with bounded additive uncertainties. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pages 4619–4624. IEEE, 2002.

[27] Florian Bayer, Mathias Burger, and Frank Allgower. Discrete-time incremental ISS: A framework for robust nmpc. In *European Control Conference (ECC)*, pages 2068–2073. IEEE, 2013.

[28] Johannes Köhler, Matthias A. Müller, and Frank Allgöwer. Nonlinear reference tracking: An economic model predictive control perspective. *IEEE Transaction on Automatic Control*, 2018. to appear.

[29] I Alvarado, D Limon, D Munoz de la Pena, T Alamo, and EF Camacho. Enhanced ISS nominal MPC based on constraint tightening for constrained linear systems. In *Proc. UKACC International Conf. on Control (CONTROL)*, pages 67–72, 2010.

[30] Chinmay Rajhans, Sachin C. Patwardhan, and Harish Pillai. Discrete time formulation of quasi infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Proceedings of the 20th IFAC World Congress*, pages 7181–7186, 2017.

[31] Ulrike Von Luxburg and Bernhard Schölkopf. Statistical learning theory: Models, concepts, and results. In *Handbook of the History of Logic*, volume 10, pages 651–706. Elsevier, 2011.

[32] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012.

[33] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Advances in neural information processing systems*, pages 514–520, 1996.

[34] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[35] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[36] Mark Hudson Beale, Martin T Hagan, and Howard B Demuth. Neural network toolbox ^{TM}user's guide. *The Mathworks Inc*, 1992.

[37] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Depz. Comput. Sci., University of British Columbia, Vancouver, Canada, Tech. Rep. UBC TR-2009-23*, 2009.

[38] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[39] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.

[40] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[41] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[42] Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[43] Roberto Tempo, Er-Wei Bai, and Fabrizio Dabbene. Probabilistic robustness analysis: Explicit bounds for the minimum number of samples. *Systems & Control Letters*, 30(5):237–242, 1997.

[44] Christian Conte, Colin N Jones, Manfred Morari, and Melanie N Zeilinger. Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. *Automatica*, 69:117–125, 2016.

[45] Johannes Köhler. Distributed economic model predictive control under inexact minimization with application to power systems. Master's thesis, Univerity Stuttgart, 2017.

[46] Carsten Scherer and Siep Weiland. Linear matrix inequalities in control. *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands*, 3, 2000.

[47] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[48] Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.