Dissertation ETH Zürich No. 18950

– Constrained Optimal Control –
Piecewise Affine and Linear Parameter-Varying Systems

A dissertation submitted to the
ETH Zurich

for the degree of
Doctor of Sciences

presented by

Thomas Besselmann

Dipl.-Ing. Mechatronik, Hamburg University of Technology
born 06.05.1980
citizen of Germany

accepted on the recommendation of

Prof. Dr. Manfred Morari, examiner
Prof. Dr. Herbert Werner, co-examiner
Dr. Johan Löfberg, co-examiner

2010

This thesis is dedicated to
Eckhard & Magda.

Acknowledgements

Pursuing my doctoral research was a fantastic experience accompanied by many ingenious people. Without their support this thesis would not exist in its current form.

Firstly, I would like to express my deep gratitude to my advisor Prof. Manfred Morari, for all the outstanding opportunities offered and for all the trust put into me. Always supportive and never constraining my academic freedoms, Prof. Morari presented a great encouragement during all phases of my PhD. I am fully aware of the fact that the working conditions at the Automatic Control Laboratory are outstanding. My gratitude and appreciation goes beyond professional matters, it was an honor to work for you. Professor Morari, Ihnen gebührt mein aufrichtiger Dank!

My gratitude extends to Johan Löfberg. Although Johan left the Automatic Control Laboratory far too early, our communication via Skype and Email turned out to be extremely productive and resulted in a large share of this thesis. His deep understanding of control and optimization is simply incredible, and I was fortunate to have arrived just early enough to get the opportunity to learn from him. Johan was of enormous help, and his optimization interface YALMIP served more than once as a shortcut to new research results.

Apart from Johan, I would like to thank my other co-authors Philipp Rostalski and Miroslav Barić. It was a pleasure to collaborate with you. I'm indebted to a number of other senior students at the lab, from whom I learned a lot. Especially I would like to thank Daniel Axehill, Sébastien Mariéthoz and Colin Jones for our extensive research discussions, which did not culminate in publications (yet). Furthermore I enjoyed being involved in the supervision of various student projects together with Sébastien Mariéthoz, Tobi Delbrück, Stefan Richter and Philipp Rostalski. A number of motivated students represented the core in these projects, namely Martin Zahnert, Felix Althaus, Kim Chung, Josef Cupić and Christian Brändli. I wish you all the best!

I owe a special thanks to my co-examiners Prof. Herbert Werner and Johan Löfberg, who were willing to read my thesis, and gave valuable feedback. At this point I would also like to thank Stefan Richter, Daniel Axehill, Helfried

Peyrl, Philipp Rostalski, Andreas Kominek and Marc Lawrence for proof-reading parts of my thesis.

The Automatic Control Laboratory is a terrific place full of amazing and interesting people, many of whom became friends. To you I want to say: It was a fantastic time indeed! I savoured to the fullest spending time with you inside and outside the lab and wish you all the best for your lives. This applies in particular to my past and present office mates. I was lucky to share my office with such great people. Here I also want to show my honest respect for everyone in the lab, who didn't lean back when there was some extra work to do. You make the difference!

I want to express my deepest gratitude to my parents and my sisters, who encouraged and supported me in every phase of my life. Everyone of them, in their very own way, were crucial to become the person I am. A special thanks goes to Doro and Alex, who made Zürich much more of a home.

Finally, I want to thank my wife Nadine for her love and care. Nadine supported me unconditionally in good times as well as in bad times. Words can not describe how much you mean to me. Nadine, you are wonderful!

Thomas Besselmann
Zürich 2010

Abstract

The scope of this thesis lies in the field of constrained optimal control. More precisely, it is concerned with the constrained finite-time optimal control of two system classes: piecewise affine systems and linear parameter-varying systems.

Firstly, constrained finite-time optimal control (CFTOC) of *piecewise affine* (PWA) systems is revisited. Since the actual computation of hybrid controllers as the explicit solution to parametric CFTOC problems for piecewise affine systems is already rather mature, this part deals with a post-processing algorithm and the actual application of explicit hybrid controllers.

If the cost function of the CFTOC problem is quadratic, the optimization problem at hand is a parametric mixed-integer quadratic program, which can be solved by decomposing it into a number of quadratic programs. Thereby redundant regions are computed, which increase the storage demand and the online evaluation time of the resulting controller. Consequently, we propose a post-processing algorithm for the removal of redundant regions.

Furthermore, we examine the application of hybrid control methods to two systems. In the first application, a mechanical system with backlash is established as an experimental testbed for the evaluation of hybrid control techniques. The second application considers autonomous vehicle steering and involves a comparison of different MPC schemes in terms of control performance and online computation times.

The second part of this thesis is devoted to constrained optimal control of *linear parameter-varying* (LPV) systems. A sequence of dynamic programming procedures is proposed, in order to solve constrained finite-time optimal control problems explicitly for LPV-A systems and for general LPV systems. Both cases of an arbitrarily varying scheduling parameter and a bounded rate of parameter variation are considered. Likewise, a procedure to solve the constrained time-optimal control problems is proposed, enabling a low-complexity alternative to explicit LPV-MPC with guaranteed stability. With these developments, explicit MPC schemes and explicit minimum-time MPC schemes are enriched by the class of LPV systems, increasing the possibilities of their application.

Finally, we demonstrate such an application of explicit LPV-MPC to autonomous vehicle steering, in order to emphasize the potential benefits of considering scheduling parameters in the controller design.

Zusammenfassung

Der Schwerpunkt dieser Dissertation liegt im Gebiet der beschränkten optimalen Regelung. Die vorliegende Arbeit beschäftigt sich mit der Regelung von beschränkten stückweis affinen und linear parameter-variierenden Systemen basierend auf der Optimierung von Gütefunktionen mit endlichem Zeithorizont.

Im ersten Teil wird die beschränkte optimale Regelung (engl. constrained finite-time optimal control, CFTOC) von *stückweis affinen* (engl. piecewise affine, PWA) Systemen rekapituliert. Da die Entwicklung von Methoden zur Berechnung der hybriden Regelgesetze als Lösung der CFTOC Probleme für stückweis affine Systeme bereits ziemlich weit fortgeschritten ist, handelt dieser Teil von einem Algorithmus zur Effizienzsteigerung hybrider Regler und von der Anwendung hybrider Regler.
Wenn die Kostenfunktion des CFTOC Problems quadratisch ist, ist das resultierende Optimierungsproblem ein parametrisches gemischt-ganzzahliges quadratisches Programm, das gelöst werden kann, indem man es in eine Reihe von parametrischen quadratischen Programmen zerlegt. Bei dieser Vorgehensweise werden redundante Regionen berechnet, die sowohl den Speicherbedarf als auch die benötigte Auswertungszeit des Regelgesetzes erhöhen. Um dem entgegen zu wirken, wird ein Algorithmus zur Detektierung und Entfernung redundanter Regionen vorgestellt.
Zusätzlich testen wir die Anwendung hybrider Regelungsmethoden an zwei Systemen. In der ersten Anwendung wird ein mechanisches System mit Zahnspiel als experimenteller Versuchsstand für die Evaluierung hybrider Regelungsmethoden zur Verfügung gestellt. Die zweite Anwendung beschäftigt sich mit der autonomen Steuerung von Fahrzeugen und zieht einen Vergleich verschiedener MPC Entwürfe anhand der Regelgüte und des online Berechnungsaufwandes.

Der zweite Teil dieser Dissertation ist der beschränkten optimalen Regelung von *linear parameter-variierenden* (LPV) Systemen gewidmet. Eine Reihe von Varianten der dynamischen Programmierung wird vorgeschlagen, um Optimierungsprobleme für die optimale Regelung von LPV-A Systemen und allgemeinen LPV Systemen explizit zu lösen. Sowohl der Fall eines willkürlich variierenden Parameters als auch der Fall einer begrenzten Änderungsrate des Parameters werden betrachtet. Zudem wird ein Verfahren zur Lösung

von beschränkten zeit-optimalen Regelungsproblemen für LPV Systeme vorgeschlagen, welches eine Alternative zu explizitem LPV-MPC darstellt, die sich durch eine niedrige Reglerkomplexität und garantierte Stabilität auszeichnet. Mit diesen Entwicklungen werden die Methoden des expliziten MPC und des expliziten zeitminimalen MPC um die Klasse der LPV Systeme angereichert und dadurch die Anzahl der möglichen Anwendungen dieser Regelungsmethoden erhöht.
Abschließend wird solch eine Anwendung des expliziten LPV-MPC zur autonomen Steuerung von Fahrzeugen demonstriert, um den möglichen Nutzen der Berücksichtigung von Parametern wie die Fahrzeuggeschwindigkeit im Reglerentwurf hervor zu heben.

Contents

1 Introduction

'Everything starts somewhere, although many physicists disagree.'

Terry Pratchett

THE SYSTEMS which control engineers face in practice most often incorporate some sort of *nonlinear behaviour*. Moreover, real-life systems are typically accompanied by constraints on the state and/or the input. The choice of an appropriate controller design method in this situation is mainly driven by the following requirements: The control method should achieve 'good' control performance, and it should be inexpensive, i.e. easy to comprehend, effortless to apply and to tune to different kinds of systems, and economical to implement.

Model predictive control (MPC) is a control method which takes constraints directly into account, and it fulfils or excels in all of these requirements – except the implementation costs. At the same time the application of MPC is limited to systems with sufficiently slow dynamics. Around the millennium this limitation was tackled by the proposal to apply parametric programming and dynamic programming to the MPC optimization problem. The solution to the MPC optimization problem is computed a-priori as an explicit function of the state – the so-called *explicit MPC* was born. Online, the solution of the optimization problem reduces to the evaluation of a look-up table, which can be performed by a common microprocessor in the range of microseconds.

Unfortunately, explicit MPC is not as flexible as classical MPC. The computation of explicit control laws is only possible for systems of modest size and only for certain types of systems. In the general case of nonlinear systems, explicit solutions are only computable for tiny systems and difficult to store and to evaluate. Apart from approximate grid-and-interpolation approaches, this leaves the control engineer with the approximation of the system behaviour by a model of simpler structure, and to use this model to compute an explicit MPC controller. Having this purpose in mind, we examine constrained optimal control of two system classes in this thesis: piecewise affine systems and linear parameter-varying systems.

Outline and Contributions

In the following section we provide a brief outline of the thesis. At the same time we will summarize the aims of the presented work and the contributions to the field. It should be emphasized that the results were obtained in close collaboration with various researchers, as indicated below.

The thesis itself is subdivided into three parts. *Part I* contains background material which will be of use in the two subsequent parts of the thesis. In *Chapter 2* basic mathematical definitions, terminologies and system-theoretical concepts are provided. The prominent theorem of Pólya, which forms a cornerstone of some of the computational procedures in later chapters, is introduced.

In *Chapter 3* ideas from constrained optimization are recapitulated. The most common classes of mathematical programming and parametric programming problems are presented. References to established solvers are given, which will serve as our computational tools to solve constrained optimization problems. Afterwards the two optimization paradigms branch-and-bound and dynamic programming are presented.

In *Chapter 4* the basic concept of model predictive control is explained by means of general nonlinear systems. In model predictive control, an optimization problem is solved at each sampling instance using the optimization tools defined in the previous chapter. Different types of finite-time optimal control optimal control problems, which reflect the variation of model predictive control and the type of the system at hand, are presented, and possible solution approaches using mathematical programming, parametric programming or dynamic programming are indicated.

Part II is concerned with the post-processing and the application of optimal control of piecewise affine (PWA) systems. PWA systems can be used to model saturation effects, logics and switches. In *Chapter 5*, piecewise affine systems are introduced, and the constrained finite-time optimal control problem for piecewise affine systems employing a quadratic cost function is discussed.

In *Chapter 6* the first contribution of this thesis is covered. The Multi-Parametric Toolbox, version 2.6.2, contains an algorithm to compute optimal controllers for piecewise affine systems employing a quadratic cost function. During this computation however, many redundant regions are computed and stored. Those regions will never be applied to the plant, but increase the complexity of the resulting controllers and slow down the online evaluation of the look-up table. Consequently an algorithm was developed and implemented for the removal of redundant regions. Several measures were taken to improve the computational speed. This chapter provides an explanation of this algorithm and is based on the technical report

> *Removal of Regions for Quadratic Norm Optimal Control of Piecewise Affine Systems*. T. Besselmann and M. Morari. Technical Report, Automatic Control Laboratory, ETH Zurich, vol. AUT07-12, 2007.

In *Chapter 7* the establishment of an experimental benchmark system for control of hybrid systems is reported. Recent years saw a huge amount of literature concerning the identification, the state estimation and the controller design of hybrid systems, but relatively little literature on experimental applications and comparisons between competing methods. Following this finding, a mechanical system with backlash was constructed and made accessible over the internet. This benchmark system can serve researchers worldwide as a test bench to test their methods and provides experimental results. As a competing candidate, a hybrid approach including modelling, control and state estimation of the benchmark system was pursued using tools developed at the Automatic Control Laboratory. This chapter is based on

> *A Hybrid Approach to Modeling, Control and State Estimation of Mechanical Systems with Backlash*. P. Rostalski, T. Besselmann, M. Barić, F. von Belzen and M. Morari. International Journal of Control, vol. 80, no. 11, pp. 1729 – 1740, Special Issue on "Automotive Systems and Control", 2007.

In *Chapter 8* the development of controllers for autonomous vehicle steering is described. The lateral vehicle model of a car comprises nonlinearities as well as state and input constraints. Tight performance criteria advocate the use of advanced control techniques such as nonlinear model predictive control. Previous publications revealed the success of nonlinear model predictive control in meeting the performance criteria, but also the difficulties to cope with a limited amount of online computations. Responding to this situation we provided a comparison of different model predictive control schemes using approximative models ranging from a linear to the fully nonlinear model. This comparison, which forms the basis of Chapter 8, was published in

> *Hybrid Parameter-Varying MPC for Autonomous Vehicle Steering*. T. Besselmann and M. Morari. European Journal of Control, vol. 14, no. 5, pp. 418 – 431, 2008.

> *Hybrid Parameter-Varying Model Predictive Control for Lateral Vehicle Stabilization*. T. Besselmann, P. Rostalski and M. Morari. European Control Conference, Kos, Greece, 2007.

An important observation for the further course of this thesis was that explicit control schemes – while significantly reducing the required online computational effort – were not able to take parameter-varying system dynamics into account. This observation led to the development of explicit MPC schemes for linear parameter-varying systems, and thus to the scope of *Part III*.

In *Chapter 9* classical gain-scheduling is reviewed, leading to the concept of linear parameter-varying (LPV) systems and a brief overview on the controller design cycle for LPV gain scheduling. Finally, constraints on the inputs and states are taken into account, and possible control approaches for constrained LPV systems are listed. LPV systems can be used to model systems whose dynamics depend on an external parameter, or to embed nonlinearities in a quasi-linear framework.

In *Chapter 10* we restrict ourselves to the class of LPV-A systems, where the input matrix is independent of the scheduling parameter. It is shown that using an affine input parametrization and the uncontrolled successor state as parameter in the final dynamic programming step results in the optimal state-feedback control law. This chapter is partially based on

> *Explicit MPC for Systems with Linear Parameter-Varying State Transition Matrix*. T. Besselmann, J. Löfberg and M. Morari. IFAC World Congress, Seoul, Korea, 2008.

The extension of the ideas behind this publication to other kinds of discrete-time systems are straightforward, as long as the closed-loop system is a polytopic system. This extension was the subject of the publication

> *Max-Min Optimal Control of Constrained Discrete-Time Systems*. M. Barić, S.V. Rakovic, T. Besselmann and M. Morari. IFAC World Congress, Seoul, Korea, 2008.

In *Chapter 11* the general case of LPV systems with a parameter-varying input matrix is considered. Contrary to the previous chapter, the closed-loop system is not polytopic, requiring a more involved solution machinery. While the optimal solution to the considered problem is only an option for trivial problems, a suboptimal solution to the dynamic programming problems can be computed by making use of Pólya's relaxation. Different approaches to explicit model predictive control of LPV systems are proposed and compared in numerical examples. This chapter is partly based on

> *Explicit MPC for LPV Systems*. T. Besselmann, J. Löfberg and M. Morari. IEEE Conference on Decision and Control, Cancun, Mexico, 2008.

In *Chapter 12* the schemes from the previous two chapters are extended to the case, when a bound on the rate of parameter variation is known. Not taking such limits into account can give rise to conservative control performance. Hence a variation of the previous schemes is proposed which is able to take a bound on the rate of parameter-variation into account. This chapter is based on

> *Explicit LPV-MPC with Bounded Rate of Parameter Variation*. T. Besselmann, J. Löfberg and M. Morari. IFAC Symposium on Robust Control Design, Haifa, Israel, 2009.

In *Chapter 13* an explicit minimum-time MPC is developed for LPV systems. Instead of optimizing a cost function, the number of time steps to reach a target set are minimized. A comparison with explicit LPV-MPC reveals that this scheme often results in significantly less complex controllers. This chapter is based on

> *Constrained Time Optimal Control of Linear Parameter-Varying Systems*. T. Besselmann, J. Löfberg and M. Morari. IEEE Conference on Decision and Control, Shanghai, China, 2009.

Finally, in *Chapter 14* the explicit LPV-MPC schemes from Chapter 10 and 12 are applied to the autonomous vehicle steering from Chapter 8. A significant decrease in the online computational time can be observed, while the controller is able to take the current values of the vehicle speed into account. This chapter is based on

> *Autonomous Vehicle Steering Using Explicit LPV-MPC*. T. Besselmann and M. Morari. European Control Conference, Budapest, Hungary, 2009.

An overview of the document structure is provided in Figure 1.1. It may guide the reader through the single chapters of this thesis.

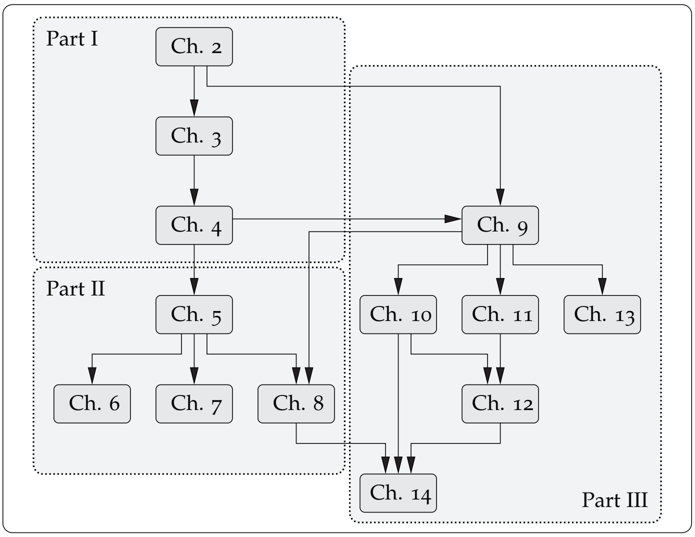


Figure 1.1: Block diagram showing the interdependence of the chapters in this thesis.

Part I
Background

2 Preliminaries

'In most sciences one generation tears down what another has built and what one has established another undoes. In mathematics alone each generations adds a new story to the old structure.'

Hermann Hankel

Mathematical and system-theoretical concepts form the basis of the topics treated in this thesis. Subsequently, this chapter on preliminaries is added for the sake of completeness. At first, some system-theoretical concepts such as Lyapunov stability or set invariance are recalled. Then we introduce Pólya's relaxation, a conservative criterium for the positivity of polynomials over the standard simplex. In later chapters of this thesis, Pólya's relaxation will play an important role for the computation of explicit MPC controllers for LPV systems. The used notation is summarized in Appendix A. For a clarification of mathematical definitions and terminology, see Appendix B.

2.1 System Theory

In this section discrete-time systems are introduced. A brief summary of stability properties based on Lyapunov theory and set invariance is provided. More details can be found e.g. in the textbooks [LaS86, BM08b, Kha00].

Discrete-time systems

Definition 2.1 (Discrete-time systems) *Let $k \in \mathbb{N}$ denote discrete time. We define the following* discrete-time systems*:*

$$\begin{aligned}
&x_{k+1} = f(x_k) && \textit{(Autonomous system)}\\
&x_{k+1} = f(x_k, u_k) && \textit{(Nonautonomous system)}\\
&x_{k+1} = f(x_k, w_k),\ x_{k+1} = f(x_k, u_k, w_k) && \textit{(Uncertain system)}\\
&x_{k+1} = f(x_k, \theta_k),\ x_{k+1} = f(x_k, u_k, \theta_k) && \textit{(Parameter-varying system)}
\end{aligned}$$

with the system state $x_k \in \mathbb{R}^{n_x}$, the control input $u_k \in \mathbb{R}^{n_u}$, the uncertainty $w_k \in \mathbb{R}^{n_w}$ and the scheduling parameter $\theta_k \in \mathbb{R}^{n_\theta}$.

A system without inputs is called autonomous system, otherwise nonautonomous system. The control input u_k, the uncertainty w_k and the scheduling parameter θ_k all denote inputs to a system. While the control input is determined by a controller, the uncertainty and the scheduling parameter denote the remaining external inputs such as disturbances, process noise or model variations. The distinction between uncertainty and scheduling parameter is based upon the existence of knowledge of the signal. While the scheduling parameter is known to the controller, the uncertainty can only be accessed indirectly via its influence on the state of the system.

Remark 2.1 *Often the state of a system is not entirely accessible, but only a subset of it, given by the output equation*

$$y_k = g(x_k, u_k, w_k, \theta_k)\,.$$

Then $y_k \in \mathbb{R}^{n_y}$ denotes the output *of the dynamic system.*

Definition 2.2 (Constrained system) *A discrete-time system is called a* constrained system *if state and/or control input are constrained to a set,*

$$\begin{aligned}
x_k &\in \mathcal{X}_k \subset \mathbb{R}^{n_x}\,,\\
u_k &\in \mathcal{U}_k \subset \mathbb{R}^{n_u}\,.
\end{aligned}$$

Note that typically uncertainties and scheduling parameters are also bounded,

$$\begin{aligned} w_k &\in \mathcal{W}_k \subset \mathbb{R}^{n_w}\,, \\ \theta_k &\in \Theta_k \subset \mathbb{R}^{n_\theta}\,, \end{aligned}$$

but the notion *constrained system* is only associated with constraints on the control input and/or the state of a system.

Stability properties

Definition 2.3 (Equilibrium point) *A point $\bar{x} \in \mathbb{R}^{n_x}$ is called* equilibrium point *of an autonomous discrete-time system $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, if*

$$\bar{x} = f(\bar{x})$$

holds.

A question one often encounters in system theory is if an equilibrium is stable, i.e. if after small deviations from the equilibrium point the state returns to the equilibrium point. The stability of equilibrium points of discrete-time systems is an important system-theoretic topic, see e.g. [LaS86].

In this work, we will often implicitly assume that the equilibrium point of interest lies at the origin. This assumption poses no restriction since it is always possible to shift an equilibrium point $\bar{x}$ to the origin by considering the state relative to this equilibrium point $\tilde{x} = x - \bar{x}$. Indeed this assumption is so common in the control literature that stability of a system is often used synonymously with stability of an equilibrium point at the origin. This however is only valid if the autonomous system at hand, as e.g. an autonomous linear system, possesses only a single equilibrium point.

Definition 2.4 (Lyapunov stability) *An equilibrium point of an autonomous discrete-time system $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ at the origin is called* (Lyapunov) stable, *if for each $\varepsilon > 0$ there exists a $\delta > 0$, such that*

$$\|x_0\| < \delta \quad \Rightarrow \quad \|x_k\| < \varepsilon \quad \forall k \in \mathbb{N} \tag{2.1}$$

holds.

Definition 2.5 (Asymptotic stability) *An equilibrium point of an autonomous discrete-time system $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ at the origin is called* asymptotically stable, *if it is stable and if there exists a $\delta > 0$ such that*

$$\|x_0\| < \delta \quad \Rightarrow \quad \lim_{k\to\infty} \|x_k\| = 0 \tag{2.2}$$

holds.

Definition 2.6 (Exponential stability) *An equilibrium point of an autonomous discrete-time system $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ at the origin is called* exponentially stable*, if there exists a $\delta > 0$, $\lambda \in [0, 1)$ and a $c \in \mathbb{R}$, such that*

$$\|x_0\| < \delta \quad \Rightarrow \quad \|x_k\| \leq c\|x_0\|\lambda^k \tag{2.3}$$

holds.

Definition 2.7 (Quadratic stability) *An equilibrium point of an autonomous discrete-time system $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ at the origin is called* quadratically stable*, if the system admits a quadratic Lyapunov function $V(x) = x^T P x$, $P = P^T \succ 0$. P is also called a* Lyapunov matrix.

Definition 2.8 (Lyapunov function) *Let $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be a continuous function denoting an autonomous discrete-time system with an equilibrium point at the origin. A positive definite function $V\colon \mathbb{R}^{n_x} \to \mathbb{R}_+$ is called a* Lyapunov function *for the system if the following conditions hold:*

(i) $V(0) = 0$,

(ii) $V(x) > 0 \quad \forall x \neq 0$,

(iii) $V(f(x)) \leq V(x) \quad \forall x \in \mathbb{R}^{n_x}$.

Theorem 2.1 (Lyapunov stability criteria) *Let $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be a continuous function denoting an autonomous discrete-time system with an equilibrium point at the origin. If there exists a Lyapunov function $V\colon \mathbb{R}^{n_x} \to \mathbb{R}_+$, the origin is* stable. *If in addition*

$$V(f(x)) < V(x) \quad \forall x \neq 0\,,$$

the origin is asymptotically stable. *Moreover, if in addition V is* radially unbounded, *i.e.*

$$V(x) \to \infty \quad \text{as} \quad \|x\| \to \infty\,,$$

the origin is globally asymptotically stable, *implying that it is the unique equilibrium point of the system.*

A Lyapunov function can also be defined locally in a neighbourhood $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ of the equilibrium point, $f\colon \mathcal{D} \to \mathbb{R}_+$, in order to determine *local stability* in a *domain of attraction*.

Definition 2.9 (Domain of attraction) *Let $f\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ be an autonomous discrete-time system with an equilibrium point at the origin. The* domain of attraction *of the equilibrium point is defined as*

$$\mathcal{D} := \left\{ x_0 \in \mathbb{R}^{n_x} \;\middle|\; \lim_{k\to\infty} \|x_k\| = 0 \right\}.$$

The concepts of stability can be extended to uncertain and parameter-varying systems, requiring that the conditions are fulfilled for all $w_k \in \mathcal{W}_k$ or for all $\theta_k \in \Theta_k$, respectively. Since under the influence of external inputs stability as defined above typically can not be achieved, one instead requires a nonautonomous system to be *uniformly ultimately bounded*, [Kha00, BM08b].

Definition 2.10 (Boundedness) *Let $f\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$ be an nonautonomous uncertain discrete-time system. The solutions to the uncertain system are called*

- uniformly bounded *if, independent of $k \in \mathbb{N}$, for each $\delta \in (0, \bar{\delta})$ there exists an $\varepsilon > 0$ such that*

$$\|x_k\| < \delta \quad \Rightarrow \quad \|x_{k+i}\| < \varepsilon \quad \forall i \in \mathbb{N}\,, \tag{2.4}$$

- globally uniformly bounded *if it is uniformly bounded for arbitrarily large $\bar{\delta}$,*
- uniformly ultimately bounded *if, independent of $k \in \mathbb{N}$, for each $\delta \in (0, \bar{\delta})$ there exist an $\varepsilon > 0$ and a $T \geq 0$ such that*

$$\|x_k\| < \delta \quad \Rightarrow \quad \|x_{k+i+T}\| < \varepsilon \quad \forall i \in \mathbb{N}\,, \tag{2.5}$$

- globally uniformly ultimately bounded *if it is uniformly ultimately bounded for arbitrarily large $\bar{\delta}$.*

Set Invariance

The concept of invariant sets is a simple and basic one, nevertheless (or hence) it acquired some attention due to its universal nature. In the area of control, set invariance appears mainly in the context of constrained systems, for the analysis of regions of attraction and of controllability properties, [BM08b]. Intuitively speaking, a set is called invariant with regard to a dynamic system if, under the evolution of the dynamic system, the system state, once it enters the set, stays in the set forever.

Set invariance is a system property and has to be considered with regard to a specific system and with regard to present constraints on the system. Depending on the structure of the system at hand, different variations of set invariance are defined.

Definition 2.11 ((Positive) invariant sets) *Let f denote a constrained discrete-time system with the state $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and optionally a control input $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, an unknown but bounded uncertainty $w_k \in \mathcal{W} \subset \mathbb{R}^{n_w}$ and/or a measurable scheduling parameter $\theta_k \in \Theta \subset \mathbb{R}^{n_\theta}$. Depending on the type of the system at hand, we define the following invariance properties: A set $\mathcal{Z} \subseteq \mathcal{X}$ is called*

- (positive) invariant *w.r.t. the system* $x_{k+1} = f(x_k)$ *if*

$$\forall x_k \in \mathcal{Z} \quad x_{k+1} = f(x_k) \in \mathcal{Z}\,.$$

- robust invariant *w.r.t. the system* $x_{k+1} = f(x_k, w_k)$ *if*

$$\forall x_k \in \mathcal{Z} \quad \forall w_k \in \mathcal{W} \quad x_{k+1} = f(x_k, w_k) \in \mathcal{Z}\,.$$

- parameter invariant *w.r.t. the system* $x_{k+1} = f(x_k, \theta_k)$ *if*

$$\forall x_k \in \mathcal{Z} \quad \forall \theta_k \in \Theta \quad x_{k+1} = f(x_k, \theta_k) \in \mathcal{Z}\,.$$

- control invariant *w.r.t. the system* $x_{k+1} = f(x_k, u_k)$ *if*

$$\forall x_k \in \mathcal{Z} \quad \exists u_k \in \mathcal{U} \,:\, x_{k+1} = f(x_k, u_k) \in \mathcal{Z}\,.$$

- robust control invariant *w.r.t. the system* $x_{k+1} = f(x_k, u_k, w_k)$ *if*

$$\forall x_k \in \mathcal{Z} \quad \exists u_k \in \mathcal{U} \,:\, \forall w_k \in \mathcal{W} \quad x_{k+1} = f(x_k, u_k, w_k) \in \mathcal{Z}\,.$$

- parameter control invariant *w.r.t. the system* $x_{k+1} = f(x_k, u_k, \theta_k)$ *if*

$$\forall x_k \in \mathcal{Z} \quad \forall \theta_k \in \Theta \quad \exists u_k \in \mathcal{U} \,:\, x_{k+1} = f(x_k, u_k, \theta_k) \in \mathcal{Z}\,.$$

Remark 2.2 *Note that the distinction between unknown uncertainty w_k and measurable scheduling parameter θ_k is only interesting for controlled systems, where knowledge of the uncertainty/parameter could be exploited for the applied control u_k. Consequently, there is no difference between the robust invariant and the parameter invariant set, while the parameter control invariant set can be significantly larger than its robust control invariant counterpart.*

A property stronger than set invariance is the contractiveness of a set. Analogue to set invariance, the set contractiveness is a property of the system at hand.

Definition 2.12 (Contractive sets) *Let f denote a constrained discrete-time system with the state $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and optionally a control input $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, an unknown but bounded uncertainty $w_k \in \mathcal{W} \subset \mathbb{R}^{n_w}$ and/or a measurable parameter $\theta_k \in \Theta \subset \mathbb{R}^{n_\theta}$. A set $\mathcal{Z}$ is λ-*contractive *with regard to the system f, for some* contraction ratio $\lambda \in [0, 1)$, *if and only if it is invariant with regard to the system f/λ.*

Many authors use the term λ-contractive set independently of the type of the system, though the meaning of contractiveness differs depending on the system type (compare Definition 2.11). For the computation of the region of attraction one is typically interested in the largest invariant set or the largest λ-contractive set.

Definition 2.13 (Maximum invariant sets) *Let f denote a constrained discrete-time system with the state $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and optionally a control input $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, an unknown but bounded uncertainty $w_k \in \mathcal{W} \subset \mathbb{R}^{n_w}$ and/or a measurable scheduling parameter $\theta_k \in \Theta \subset \mathbb{R}^{n_\theta}$. A set $\mathcal{Z}_\infty \subseteq \mathcal{X}$ is called the* maximum invariant set *of a discrete-time system $x_{k+1} = f(x_k)$, if*

(i) $\mathcal{Z}_\infty$ is invariant,

(ii) $\mathcal{Z}_\infty$ contains the origin and

(iii) $\mathcal{Z}_\infty$ contains all invariant sets in $\mathcal{X}$ that contain the origin.

The maximum robust invariant set, *the* maximum parameter invariant set, *the* maximum control invariant set, *the* maximum robust control invariant set *and the* maximum parameter invariant set *are defined by applying the variations of invariance accordingly, see Definition 2.11.*

The maximum invariant set of a discrete-time system can be computed by making repeatedly use of presets.

Definition 2.14 (Presets $\Omega(\mathcal{Z})$) *Let f denote a constrained discrete-time system with the state $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and optionally a control input $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, an unknown but bounded uncertainty $w_k \in \mathcal{W} \subset \mathbb{R}^{n_w}$ and/or a measurable parameter $\theta_k \in \Theta \subset \mathbb{R}^{n_\theta}$. Depending on the type of the system at hand, we define the following* presets*:*

$$\begin{aligned}
\Omega(\mathcal{Z}) &:= \{x \in \mathcal{X} \mid f(x) \in \mathcal{Z}\}, \\
\Omega(\mathcal{Z}) &:= \{x \in \mathcal{X} \mid \forall w \in \mathcal{W} \quad f(x,w) \in \mathcal{Z}\}, \\
\Omega(\mathcal{Z}) &:= \{x \in \mathcal{X} \mid \forall \theta \in \Theta \quad f(x,\theta) \in \mathcal{Z}\}, \\
\Omega(\mathcal{Z}) &:= \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} : f(x,u) \in \mathcal{Z}\}, \\
\Omega(\mathcal{Z}) &:= \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} : \forall w \in \mathcal{W} \quad f(x,u,w) \in \mathcal{Z}\}, \\
\Omega(\mathcal{Z}) &:= \{x \in \mathcal{X} \mid \forall \theta \in \Theta \quad \exists u \in \mathcal{U} : f(x,u,\theta) \in \mathcal{Z}\}.
\end{aligned}$$

More precisely, the computation of the maximal invariant set $\mathcal{Z}_\infty$ can be performed by executing Algorithm 2.1, initialized with some set of $\mathcal{Z}_0$ (e.g. $\mathcal{Z}_0 = \mathcal{X}$). Analogue the maximum λ-contractive set can be computed by performing Algorithm 2.1 replacing $\Omega(\mathcal{Z}_i)$ by $\Omega(\lambda\mathcal{Z}_i)$. One should mention that the backpropagation algorithm is not guaranteed to terminate in finite time, hence in a practical implementation some numerical tolerances should

Algorithm 2.1 Backpropagation of sets

```
Input: $\mathcal{Z}_0$
Output: $\mathcal{Z}_1, \ldots, \mathcal{Z}_\infty$
1: $\mathcal{Z}_1 = \Omega(\mathcal{Z}_0)$
2: $i = 1$
3: while $\mathcal{Z}_i \neq \mathcal{Z}_{i-1}$ do
4:     $\mathcal{Z}_{i+1} = \Omega(\mathcal{Z}_i)$
5:     $i = i + 1$
6: end while
7: $\mathcal{Z}_\infty = \mathcal{Z}_i$
```

be introduced to terminate when the presets converged up to a numerical precision.

Example 2.1 *Figure 2.1 depicts the sets $\mathcal{Z}_0, \ldots, \mathcal{Z}_\infty$ occurring during set backpropagation. In the example the sets converge after five iterations, this set is the maximal invariant set, denoted by $\mathcal{Z}_\infty$.*

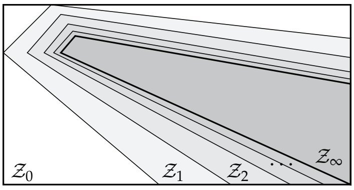


Figure 2.1: Example sets $\mathcal{Z}_0, \ldots, \mathcal{Z}_\infty$ occurred during set backpropagation.

2.2 Pólya's Theorem

One of the main ingredients for the techniques described in this thesis is based on a theorem by the famous Hungarian mathematician György Pólya, published in 1928, [Pól28]. It constitutes the theoretical basis for the treatment of positivity requirements on polynomials. This section provides some background information to Pólya's theorem, following [PR06, Ros09].

Definition 2.15 (Monomial) *Given the variables $\theta_1, \ldots, \theta_{n_\theta}$ and the multi-index exponents $\alpha = (\alpha_1, \ldots, \alpha_{n_\theta}) \in \mathbb{N}^{n_\theta}$, a* monomial *is defined as*

$$\theta^\alpha = \theta_1^{\alpha_1} \theta_2^{\alpha_2} \cdots \theta_{n_\theta}^{\alpha_{n_\theta}} .$$

The degree *of a monomial is given by the sum of its exponents,*

$$\deg(\theta^\alpha) = |\alpha| := \sum_{j=1}^{n_\theta} \alpha_i \,.$$

Definition 2.16 (Polynomial) *A* polynomial *is a linear combination of finitely many monomials,*

$$p(\theta) = \sum_{\alpha \in \mathbb{N}^{n_\theta}} c_\alpha \theta^\alpha = \sum_{\alpha \in \mathbb{N}^{n_\theta}} c_\alpha \theta_1^{\alpha_1} \theta_2^{\alpha_2} \cdots \theta_{n_\theta}^{\alpha_{n_\theta}} \,,$$

consisting of the terms $c_\alpha \theta^\alpha$ *with the real* coefficients $c_\alpha \in \mathbb{R}$. *The set of all nonzero coefficients is denoted by*

$$\operatorname{coef}(p) := \{c_\alpha \mid \alpha \in \mathbb{N}^{n_\theta} : c_\alpha \neq 0\} \,,$$

We denote the support *of p as all monomials with nonzero coefficients,*

$$\operatorname{supp}(p) := \{\theta^\alpha \mid \alpha \in \mathbb{N}^{n_\theta} : c_\alpha \neq 0\} \,,$$

the degree *or* order *of p as the highest degree of the supporting monomials,*

$$\deg(p) := \max_{\theta^\alpha \in \operatorname{supp}(p)} \deg(\theta^\alpha) \,.$$

Definition 2.17 (Homogeneous polynomial) *A* homogeneous polynomial, *or a* form, *is a polynomial whose supporting monomials all have the same degree,*

$$\deg(\theta^\alpha) = \deg(p) \quad \forall \theta^\alpha \in \operatorname{supp}(p) \,.$$

The set of all polynomials in n_θ variables with real coefficients is denoted by $\mathbb{R}[\theta] := \mathbb{R}[\theta_1, \ldots, \theta_{n_\theta}]$. $\mathbb{R}[\theta]$ is called *polynomial ring* over $\mathbb{R}$ and is commutative with the usual addition and multiplication.

Theorem 2.2 (Pólya's theorem) *Let* $p(\theta) \in \mathbb{R}[\theta]$ *be a homogeneous polynomial of degree d and let* Θ *be the standard simplex,*

$$\Theta := \left\{ \theta = [\theta_1, \ldots, \theta_{n_\theta}]^T \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_j = 1 \right\} .$$

If p is positive on Θ, *all coefficients of the* extended polynomial

$$p_{N_p}(\theta) = p(\theta) \cdot (\sum_{j=1}^{n_\theta} \theta_j)^{N_p}$$

are positive for a sufficiently large Pólya degree $N_p \in \mathbb{N}$.

Theorem 2.2 follows directly from the proof of the original version of the theorem, which was stated in Pólya's original work from 1928 in German, [Pól28]. The following theorem was taken from the book *Inequalities* in English, [HLP52].

Theorem 2.3 (Original version of Pólya's theorem) *If the form $p(\theta) \in \mathbb{R}[\theta]$ is positive in the positive orthant,*

$$\theta \geq 0, \quad \sum_{j=1}^{n_\theta} \theta_j > 0\,, \tag{2.6}$$

then p may be expressed as $p = \frac{p_{N_p}(\theta)}{h(\theta)}$ where $p_{N_p}(\theta) \in \mathbb{R}[\theta]$ and $h(\theta) \in \mathbb{R}[\theta]$ are forms with positive coefficients. In particular we may suppose that

$$h(\theta) = (\sum_{j=1}^{n_\theta} \theta_j)^{N_p} \tag{2.7}$$

for a suitable $N_p \in \mathbb{N}$.

Proof ([HLP52]) For notational simplicity we restrict ourselves to the case $n_\theta = 3$; no new point of principle arises for general n_θ.

The homogeneous polynomial $p(\theta) \in \mathbb{R}[\theta]$ is continuous and positive on the standard simplex Θ, and has a positive minimum

$$p^* := \min_{\theta \in \Theta} p(\theta) > 0$$

in Θ. We write

$$p(\theta) = \sum_{|\alpha|=d} c_\alpha \frac{\theta_1^{\alpha_1}\theta_2^{\alpha_2}\theta_3^{\alpha_3}}{\alpha_1!\alpha_2!\alpha_3!}\,, \tag{2.8}$$

with the summation being over all terms of degree d,

$$\alpha \geq 0\,, \quad \sum_{j=1}^{3} \alpha_j = d\,. \tag{2.9}$$

Furthermore, we define

$$\phi(\theta, t) := t^d \sum_{|\alpha|=d} c_\alpha \binom{\theta_1 t^{-1}}{\alpha_1}\binom{\theta_2 t^{-1}}{\alpha_2}\binom{\theta_3 t^{-1}}{\alpha_3}\,, \tag{2.10}$$

with $t > 0$ and where $\binom{\theta_j t^{-1}}{\alpha_j}$, $j = 1, 2, 3$, are the usual binomial coefficients, such that

$$\binom{\theta_j t^{-1}}{0} = 1 \quad \text{and} \quad t^{\alpha_j}\binom{\theta_j t^{-1}}{\alpha_j} = \frac{\theta_j(\theta_j - t)(\theta_j - 2t)\ldots(\theta_j - (\alpha_j - 1)t)}{1 \cdot 2 \cdot 3 \cdots \alpha_j}\,.$$

It is obvious that $\phi(\theta, t) \to p(\theta)$ for $t \to 0$. If we write $\phi(\theta, 0) = p(\theta)$, then $\phi(\theta, t)$ is continuous in

$$\theta \geq 0\,, \quad \sum_{j=1}^{3} \theta_j = 1\,, \quad 0 \leq t \leq 1\,. \tag{2.11}$$

Consequently, there exists an $\epsilon > 0$ such that

$$\phi(\theta, t) > \phi(\theta, 0) - 0.5p^* = p(\theta) - 0.5p^* \geq 0.5p^* > 0 \tag{2.12}$$

for all $t \in (0, \epsilon)$ and for all $\theta \in \Theta$.

Given a degree $\hat{d} > d$, we also have

$$(\sum_{j=1}^{3} \theta_j)^{\hat{d}-d} = (\hat{d} - d)! \sum_{|\beta| = \hat{d}-d} \frac{\theta_1^{\beta_1}\theta_2^{\beta_2}\theta_3^{\beta_3}}{\beta_1!\beta_2!\beta_3!}\,, \tag{2.13}$$

with the summation being over all terms of degree $\hat{d} - d$,

$$\beta \geq 0, \quad \sum_{j=1}^{3} \beta = \hat{d} - d\,. \tag{2.14}$$

By multiplying (2.8) and (2.13), we obtain

$$(\sum_{j=1}^{3} \theta_j)^{\hat{d}-d} p(\theta) = (\hat{d} - d)! \sum_{|\alpha|=d} \sum_{|\beta|=\hat{d}-d} c_\alpha \frac{\theta_1^{\alpha_1+\beta_1}\theta_2^{\alpha_2+\beta_2}\theta_3^{\alpha_3+\beta_3}}{\alpha_1!\beta_1!\alpha_2!\beta_2!\alpha_3!\beta_3!}\,. \tag{2.15}$$

Here we introduce

$$\gamma_j = \alpha_j + \beta_j\,,\ j = 1, 2, 3\,, \tag{2.16}$$

so that γ_j vary over

$$\gamma \geq 0\,, \quad \sum_{j=1}^{3} \gamma_j = \hat{d} \tag{2.17}$$

and α_j over

$$0 \leq \alpha \leq \gamma\,, \quad \sum_{j=1}^{3} \alpha_j = d\,. \tag{2.18}$$

We obtain

$$(\sum_{j=1}^{3}\theta_j)^{\hat{d}-d}p(\theta) = (\hat{d}-d)! \sum_{|\gamma|=\hat{d}} \frac{\theta_1^{\gamma_1}\theta_2^{\gamma_2}\theta_3^{\gamma_3}}{\gamma_1!\gamma_2!\gamma_3!} \sum_{|\alpha|=d}{}' c_\alpha \binom{\gamma_1}{\alpha_1}\binom{\gamma_2}{\alpha_2}\binom{\gamma_3}{\alpha_3}\ , \tag{2.19}$$

where $\sum'_{|\alpha|=d}$ indicates summation w.r.t. α over (2.18). But, since $\binom{\gamma_j}{\alpha_j} = 0$, if $\alpha_j > \gamma_j$, we may replace this summation by summation over (2.9), i.e. by $\sum_{|\alpha|=d}$. We thus yield

$$\begin{aligned}(\sum_{j=1}^{3}\theta_j)^{\hat{d}-d}p(\theta) &= (\hat{d}-d)! \sum_{|\gamma|=\hat{d}} \frac{\theta_1^{\gamma_1}\theta_2^{\gamma_2}\theta_3^{\gamma_3}}{\gamma_1!\gamma_2!\gamma_3!} \sum_{|\alpha|=d} c_\alpha \binom{\gamma_1}{\alpha_1}\binom{\gamma_2}{\alpha_2}\binom{\gamma_3}{\alpha_3} \\ &= (\hat{d}-d)!\hat{d}^d \sum_{|\gamma|=\hat{d}} \phi(\frac{\gamma}{\hat{d}};\frac{1}{\hat{d}})\frac{\theta_1^{\gamma_1}\theta_2^{\gamma_2}\theta_3^{\gamma_3}}{\gamma_1!\gamma_2!\gamma_3!}\,. \end{aligned} \tag{2.20}$$

The ϕ here is positive, by (2.12), if $\hat{d}$ is sufficiently large, and this proves the theorem. ■

The main idea of the proof is to construct a sequence $\phi(\theta, t)$ of real polynomials with two properties:

1. The sequence ϕ converges uniformly to p on Θ with increasing N_p.
2. The coefficients of the extended polynomial p_{N_p} are positive multiples of values of the sequence members, evaluated at certain points on the standard simplex.

If p is positive on Θ, then, for sufficiently large N_p, we have that also the members of the sequence are positive on the entire standard simplex. Since the coefficients of the extended polynomial are positive multiples of evaluations of the sequence members, this means in return that for sufficiently large N_p, the coefficients of all monomials of p_{N_p} are positive.

Pólya's Relaxation

During this thesis, we will repeatedly make use of the more obvious reverse of Pólya's theorem[1], i.e. positive coefficients of the extended polynomial mean positivity over the entire standard simplex. By applying *Pólya's relaxation* to a polynomial inequality, we mean the processing of the following steps:

[1]The presented usage of Pólya's theorem is implemented in YALMIP as one of the so called filters in the robust optimization framework, [Löf08].

1. Reformulate the inequality into a positivity constraint of a polynomial $p(\theta)$.

2. If required, homogenize the polynomial $p(\theta)$ by multiplying single monomials with $\sum_{j=1}^{n_\theta} \theta_j$ (which equals one on the standard simplex) until all monomials have the same degree.

3. Set the Pólya degree N_p, and compute the coefficients $\mathbf{C}_{N_p} = \operatorname{coef}(p_{N_p})$ of the extended polynomial $p_{N_p}(\theta)$.

4. Replace the polynomial inequality by $\mathbf{C}_{N_p} > 0$. In this step some conservatism may be introduced depending on the selection of N_p. By increasing the polynomial degree N_p, the relaxations become tighter until the exact problem is considered.

Example 2.2 *Consider the polynomial*

$$p(\theta) = a\theta_1^2 + b\theta_1\theta_2 + a\theta_2^2\,.$$

A sufficient condition for positivity over the standard simplex Θ is the positivity of the coefficients $\mathbf{C}_0 = \{a, b\}$. By multiplying $p(\theta)$ with $\sum_{j=1}^{n_\theta} \theta_j$, we obtain

$$p_1(\theta) = a\theta_1^3 + (a+b)\theta_1^2\theta_2 + (b+a)\theta_1\theta_2^2 + a\theta_2^3$$

and the less conservative condition of positive coefficients $\mathbf{C}_1 = \{a, a+b\}$. Note that the coefficients $\mathbf{C}_1$ of the extended polynomial are a conic combination of the coefficients $\mathbf{C}_0$, i.e. they lie in the cone which is spanned by the coefficients $\mathbf{C}_0$. Hence $\mathbf{C}_{i_1} > 0$ implies $\mathbf{C}_{i_2} > 0$ for all $i_2 \geq i_1$. Pólya's contribution was to show that by repeated multiplication with $\sum_{j=1}^{n_\theta} \theta_j$, the condition of positive coefficients indeed converges to the exact necessary and sufficient condition for positivity of the polynomial over the standard simplex. Figure 2.2 shows the resulting conditions on the coefficients of $p(\theta)$ for different Pólya degrees.

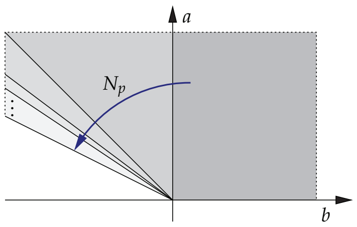


Figure 2.2: Abating conditions on the coefficients of $p(\theta)$ for increasing Pólya degrees $N_p = 0, 1, 3, 5, ..., \infty$.

Definition 2.18 (Polytopic Preset) *Let $\mathcal{Z} \subset \mathbb{R}^{n_x}$ be a polytope. A set $\Omega_{N_p}(\mathcal{Z}) \subset \mathbb{R}^{n_x}$ is called* polytopic preset *of $\mathcal{Z}$, if (i) it is a polytope and (ii) it contains all states which fulfil the polytopic constraints arising when applying Pólya's relaxation with degree N_p to the preset $\Omega(\mathcal{Z})$.*

Complexity Analysis

The authors of [HLP52] state:

> The theorem gives a systematic process for deciding whether a given form F is strictly positive for positive x. We multiply repeatedly by $\sum x_i$ and, if the form is positive, we shall sooner or later obtain a form with positive coefficients.

This quotation reveals the algorithmic nature of Pólya's theorem. The question remains how soon is sooner or later. In [PR01] the authors were able to derive an explicit bound for the Pólya degree N_p.

Theorem 2.4 (Qualitative Pólya's theorem) *Suppose that $p(\theta) \in \mathbb{R}[\theta]$ is a homogeneous polynomial of degree d and positive on the standard simplex Θ. Let the maximum of the scaled coefficients of $p(\theta)$ be denoted by c_{max} and the minimum of the polynomial over the simplex by p^*, i.e.*

$$c_{max} := \max_{|\alpha|=d} \frac{\alpha_1! \cdots \alpha_{n_\theta}!}{d!} |c_\alpha| , \qquad p^* := \min_{\theta \in \Theta} p(\theta) .$$

For all

$$N_p > \frac{d(d-1)}{2} \frac{c_{max}}{p^*} - d , \tag{2.21}$$

the extended polynomial $p_{N_p} = p(\theta) \cdot (\sum_{j=1}^{n_\theta} \theta_j)^{N_p}$ has positive coefficients.

Proof See [PR01]. ■

Example 2.3 *Consider again the polynomial*

$$p(\theta) = a\theta_1^2 + b\theta_1\theta_2 + a\theta_2^2 ,$$

with $a = 1, b = -0.5$. From Example 2.2 we know already that a Pólya degree of $N_p = 1$ suffices to verify positivity on the standard simplex. Here we want to know the bound on the Pólya degree stemming from Theorem 2.4. The minimum on the standard simplex is $p^* = 0.625$, while the maximum of the scaled coefficients is $c_{max} = 1$. By using Equation (2.21), the required Pólya degree can be determined to be $N_p > 0.666$, i.e. the bound is tight in this example.

Unfortunately, Theorem 2.4 is only of limited use for our purposes, since it requires a-priori knowledge of the minimum of the polynomial over the standard simplex. If the minimum of the polynomial is known, so is the positivity of the polynomial over the standard simplex. Note however that Theorem 2.4 requires the Pólya degree $N_p \to \infty$ for $p^* \to 0$.

Corollary 2.5 (Number of coefficients) *Let $d \in \mathbb{N}$ denote the degree of a homogeneous polynomial $p(\theta) \in \mathbb{R}[\theta]$. The degree of the extended polynomial is given by*

$$\hat{d} := \deg(p_{N_p}) = d + N_p \,,$$

and the extended polynomial possesses

$$n_{coef} \leq \sum_{i_1=1}^{n_\theta} \sum_{i_2=i_1}^{n_\theta} \cdots \sum_{i_{\hat{d}}=i_{\hat{d}-1}}^{n_\theta} 1 \in O(n_\theta^{d+N_p}) \qquad (2.22)$$

nonzero coefficients.

The correctness of Corollary 2.5 becomes comprehensible by noting that the monomials of p_{N_p} consist of $\hat{d}$ positions and that the indices $i_1, \ldots, i_{\hat{d}}$ denote the variables at all positions of each monomial, i.e. $\theta^\alpha = \theta_{i_1}\theta_{i_2}\cdots\theta_{i_{\hat{d}}}$. Furthermore, the variables in the monomial are arranged in a non-decreasing order to prevent redundancies.

The first inequality of (2.22) is tight if all monomials of degree d support the polynomial p. The O-notation on the other hand just gives a rough upper bound on the complexity; it ignores the required ordering of the monomials.

Example 2.4 *Consider a third time the polynomial*

$$p(\theta) = a\theta_1^2 + b\theta_1\theta_2 + a\theta_2^2 \,.$$

We have $d = 2, n_\theta = 2$. Hence the number of coefficients can be stated as $n_{coef} = N_p + 3$.

Finally one can state that for the number of coefficients n_{coef} to be small, either the polynomial has to be sparse, the number of variables n_θ has to be small, or the Pólya degree N_p has to be small.

Rational Functions

The application of Pólya's relaxation is not limited to polynomial functions, but extends to rational functions defined on the entire parameter simplex, as will be discussed next.

Definition 2.19 (Rational function) *A function $r\colon \mathcal{D} \to \mathbb{R}$ with the domain $\mathcal{D} \subseteq \mathbb{R}^{n_\theta}$ is called a* rational function, *if it can be expressed as*

$$r(\theta) = \frac{p_{nom}(\theta)}{p_{den}(\theta)},$$

with $p_{nom} \in \mathbb{R}[\theta]$ and $p_{den} \in \mathbb{R}[\theta]$ being polynomials in θ, and p_{den} is not the zero polynomial.

Corollary 2.6 (Extension of Pólya's theorem to rational functions) *Let $r\colon \mathcal{D} \to \mathbb{R}$ be a rational function with the domain $\mathcal{D} \subseteq \mathbb{R}^{n_\theta}$, and let Θ be the standard simplex. Assume $r(\theta)$ to be defined on the entire standard simplex, $\Theta \subseteq \mathcal{D}$. If $r(\theta) = p_{nom}(\theta)/p_{den}(\theta)$ is positive on Θ, all coefficients of the extended polynomials*

$$p_{nom}(\theta) \cdot (\sum_{j=1}^{n_\theta} \theta_j)^{N_p} \quad \textit{and} \quad p_{den}(\theta) \cdot (\sum_{j=1}^{n_\theta} \theta_j)^{N_p}$$

are either jointly positive or negative for a sufficiently large Pólya degree N_p.

For $r(\hat{\theta})$ to be positive at a given $\hat{\theta} \in \Theta$, both $p_{nom}(\hat{\theta})$ and $p_{den}(\hat{\theta})$ have to be either positive or negative. Since r is defined on the entire standard simplex Θ, it follows that $p_{den}(\theta) \neq 0 \quad \forall \theta \in \Theta$. Polynomials are continuous, sign changes of the denominator with varying scheduling θ are therefore excluded. Positivity of r hence requires the joint positivity or negativity of p_{nom} and p_{den} over the entire standard simplex. Application of Pólya's theorem yields the corollary.

3 Constrained Optimization

'The subject of optimization is a fascinating blend of heuristics and rigour, of theory and experiment.'

Roger Fletcher

OPTIMIZATION under constraints is part of our all lives, and we, consciously or unconsciously, perform optimizations on an everyday basis. What exactly is understood under the term optimization varies depending on the context. In the following we will discuss the meaning of optimization from a mathematical point of view.

Optimization is a major field in applied mathematics, devoted to the choice of the best element in a set of alternatives. What is considered the best element is determined by an objective function or cost function. Often the set of alternatives is not unlimited, but underlies some restrictions, leading to optimization under constraints, or constrained optimization. Constrained optimization is a common tool in control nowadays, and will repeatedly be applied in the succeeding chapters.

In this chapter basic notions and concepts of constrained optimization are presented. Starting with mathematical programming, parametric programming and finally the common optimization paradigms brand-and-bound and dynamic programming are discussed.

3.1 Mathematical Programming

The first section in this chapter is concerned with mathematical programming, or optimization. After considering general nonlinear optimization problems, common classes of optimization problems are introduced such as linear programs, quadratic programs, semidefinite programs and mixed-integer programs. For a good introduction into convex optimization see [BV04].

We consider the constrained *optimization problem*, or *mathematical program*, of the form

$$J^* = \inf_u J(u) \tag{3.1a}$$
$$\text{s.t. } g_i(u) \le 0\,, \quad i = 1, \dots, n_c\,. \tag{3.1b}$$

We call $u \in \mathbb{R}^{n_u}$ the *optimization variable* and $J\colon \mathbb{R}^{n_u} \to \mathbb{R}$ the *objective* or *cost function*. The *optimal value* of the optimization problem is denoted by J^*, while the functions $g_i\colon \mathbb{R}^{n_u} \to \mathbb{R}$, $i = 1, \dots, n_c$, denote the *constraints* of the optimization problem. A problem is called *unconstrained* if $n_c = 0$. The constraints, together with the domains of the objective and constraints, define the set of feasible points, or the *feasible set*,

$$\mathcal{U} = \left\{ u \in \operatorname{dom} J(u) \cap \bigcap_{i=1}^{n_c} \operatorname{dom} g_i(u) \;\middle|\; g_i(u) \le 0\,, \quad i = 1, \dots, n_c \right\}.$$

A constraint is called *redundant*, if the feasible set is unaffected by the inclusion/neglection of this constraint. The optimization problem (3.1) is called *feasible* if $\mathcal{U} \neq \emptyset$, and *infeasible* otherwise.

We call $u^* \in \mathcal{U}$ a *solution* to (3.1) or an *optimal point*, if $J(u^*) = J^*$ with $J^* > -\infty$. The optimization problem (3.1) is called *solvable*, and the optimal value J^* is attained, if the *optimal set*

$$\mathcal{U}^* = \{u \in \mathcal{U} \mid J(u) = J^*\}$$

is non-empty. For a given u, a constraint is called *active* if $g_i(u) = 0$, and *inactive* if $g_i(u) < 0$. If u is not specified, the *set of active constraints* (also called *optimal active set*) denotes the constraints which are active for all solutions of (3.1),

$$\boldsymbol{A} = \{i \in \{1, \dots, n_c\} \mid g_i(u^*) = 0 \quad \forall u^* \in \mathcal{U}^*\}\,. \tag{3.2}$$

With the *set of inactive constraints* we denote the complement of the set of active constraints, i.e.

$$\boldsymbol{I} = \{i \in \{1, \dots, n_c\} \mid \exists u^* \in \mathcal{U}^* : g_i(u^*) < 0\}\,. \tag{3.3}$$

We will use the set of (in-)active constraints also in subscript notation, i.e. $g_{\boldsymbol{A}} := \{g_i : i \in \boldsymbol{A}\}$ or $g_{\boldsymbol{I}} := \{g_i : i \in \boldsymbol{I}\}$.

Definition 3.1 (Locally optimal point) *A point $\bar{u}$ is called* locally optimal *if there exists an $\epsilon > 0$ such that $J(u) \geq J(\bar{u})$ holds for all elements of a ball $\mathcal{B}$ of radius ϵ, $\mathcal{B}(\bar{u}, \epsilon) = \{u \in \mathcal{U} : \|u - \bar{u}\| \leq \epsilon\}$.*

If the feasible set $\mathcal{U}$ is convex, and J is a convex function, (3.1) is called a *convex optimization problem*. The most striking consequence of convexity of the optimization problem (3.1) is that all local optimal points are solutions to (3.1). Moreover, there exist well-established solvers for convex optimization problems, which are based on oracles and/or efficient barriers. Given an oracle that verifies the set membership of a point u to the feasible set $\mathcal{U}$, convex optimization solvers either provide a close-to-optimal solution within polynomial time, or provide a certificate for the nonexistence of a solution.

In the remainder of this work we will occasionally make use of *epigraph reformulations*. The epigraph form of the optimization problem (3.1) is the problem

$$J^* = \inf_{\{u,t\}} \ t \tag{3.4a}$$

$$\text{s.t. } g_i(u) \leq 0\,, \quad i = 1, \ldots, n_c\,, \tag{3.4b}$$

$$J(u) - t \leq 0\,, \tag{3.4c}$$

where an *epigraph variable* $t \in \mathbb{R}$ is introduced. It can easily be shown that (3.4) is equivalent to (3.1) in the sense that (u^*, t^*) is a solution to (3.4) if and only if u^* is a solution to (3.1) and $J^* = t^*$.

Remark 3.1 *The main benefit of the epigraph formulation is the transformation into a problem with linear objective function. One example for a common epigraph reformulation is the minimization of a convex piecewise affine cost function under linear constraints, which can be transformed into a linear program by using an epigraph reformulation.*

Remark 3.2 *In this work we consider optimization problems with a continuous objective function J over a compact feasible set $\mathcal{U}$. Therefore the conditions of the* Weierstrass theorem *are fulfilled, [Ber95], and the optimal value is always attained. As a consequence, the infimum in* (3.1) *can be replaced by minimum.*

The most commonly used necessary conditions for optimal points are the first order Karush-Kuhn-Tucker conditions (or KKT conditions for short), which were discovered independently by Karush, [Kar39], and later by Kuhn and Tucker, [KT51]. For more details on convex optimization, see e.g. [BV04], [Ber03].

Linear Program

A *linear program* (LP) denotes an optimization problem with linear objective function and linear constraints. It can be stated in several different forms, e.g. as *inequality form LP*,

$$J^* = \min_u c^T u \tag{3.5a}$$
$$\text{s.t. } Au \leq b\,, \tag{3.5b}$$

with $u \in \mathbb{R}^{n_u}, c \in \mathbb{R}^{n_u}, A \in \mathbb{R}^{n_c \times n_u}$ and $b \in \mathbb{R}^{n_c}$. There exist efficient solvers for linear programs, based on the *simplex method*, [Dan98, Mur83], or on the *interior-point method*, [Kar84, NN94]. Common solvers for linear programs are NAG, [Num02], or CPLEX, [CPL94].

The feasible set of a linear program is given by the constraints (3.5b) which describe a polyhedron in the space of the optimization variables. When solving a linear program without redundant constraints, four different results can be obtained:

(*i*) the LP is infeasible,

(*ii*) the LP is unbounded ($J^* = -\infty$),

(*iii*) the LP is bounded and we obtain a unique solution u^*, and

(*iv*) the LP is bounded, but the solution is not unique.

In the first case, the feasible set is empty and we set $J^* = \infty$ by convention. In the second case the polyhedron is unbounded in a descent direction. In case (*iii*) the solution is attained at one of the vertices of the feasible set $\mathcal{U}$, and the number of active constraints equals the number of optimization variables. In case (*iv*) the optimal set $\mathcal{U}^*$ is a face of the feasible set, and the number of active constraints is greater than zero and smaller than n_u. In general we can also encounter redundant constraints, and instead of the regular behaviour discussed above, the linear program can be of degenerate nature.

Quadratic Program

A *quadratic program* (QP) refers to a convex optimization problem with quadratic objective function and linear constraints. It can be stated in the form

$$\min_u \frac{1}{2} u^T Q u + c^T u \tag{3.6a}$$
$$\text{s.t. } Au \leq b\,, \tag{3.6b}$$

with $u \in \mathbb{R}^{n_u}, Q \in \mathbb{R}^{n_u \times n_u}, c \in \mathbb{R}^{n_u}, A \in \mathbb{R}^{n_c \times n_u}$ and $b \in \mathbb{R}^{n_c}$. Note that Q has to be positive semidefinite for the objective function to be convex. If Q is positive definite, the solution to (3.6) is unique. It can be shown that a single negative eigenvalue of Q renders (3.6) NP-hard, [PV91]. From now on, and if not mentioned otherwise, we will implicitly assume a quadratic program to be convex. There exist efficient solvers for quadratic programs, based on the *active set method* or the *interior-point method*. Details on these solution methods can be found e.g. in [NW06, GT01]. Common solvers for quadratic programs are NAG, [Num02], or CPLEX, [CPL94].

Contrary to the LP case, the solution to a QP can also be attained at the interior of the feasible set $\mathcal{U}$. The number of active constraints at the QP solution can vary between 0 (solution lies in the interior) to n_c (all constraints are active).

Semidefinite Program

A large subfield within convex optimization is *conic optimization*, which is based on the concept of generalized inequality constraints. The element-wise inequalities are replaced by vector-valued inequalities stemming from a convex cone. If the generalized inequalities stem from the cone of positive semidefinite matrices $\mathbb{S}_{+}^{n}$, the cone problem is referred to as *semidefinite program* (SDP). Analogue to the inequality form LP, the *inequality form SDP* is defined as

$$\min_{u} c^T u \tag{3.7a}$$

$$\text{s.t. } A_1 u_1 + \cdots + A_{n_u} u_{n_u} \preceq B\,, \tag{3.7b}$$

with $u = \begin{bmatrix} u_1 & \cdots & u_{n_u} \end{bmatrix}^T \in \mathbb{R}^{n_u}, B, A_1, \dots, A_{n_u} \in \mathbb{S}^{n_c}$ and $c \in \mathbb{R}^{n_u}$. The constraint (3.7b) is called *Linear Matrix Inequality* (LMI). The term LMI became so popular in the control community that some authors (falsely) denote semidefinite programs as LMIs.

Semidefinite programs contain the quadratic program (3.6) and the linear program (3.5) as special cases. SDPs however are far more general and a large number of nonlinear convex optimization problems can be cast and solved as semidefinite programs. For an introduction to semidefinite programming and its various applications to control see [WSV00, BEGFB94, BV04]. Decent SDP solvers as SeDuMi, [Stu99], or SDPT3, [TTT99], are freely available, and can be addressed conveniently under Matlab via the neat interface Yalmip, [Löf04].

Mixed-Integer Program

Another class of optimization problems optimizes not only over continuous variables, but also over binaries or integer variables. This class of problems is called *mixed-integer programs*, or mixed-integer nonlinear problems (MINLP). A common special case is to consider binary variables instead of integers. Although the notation mixed-binary program would be more precise, the standard notation in the control literature denotes this class of problems also as mixed-integer programs, which can be stated in the form

$$\min_{u \in \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ub}}} \quad J(u) \tag{3.8a}$$

$$\text{s.t.} \quad g(u) \leq 0\,, \tag{3.8b}$$

with the optimization variable $u = [u_c^T \;\; u_b^T]^T, u_c \in \mathbb{R}^{n_{uc}}, u_b \in \{0,1\}^{n_{ub}}$.

Mixed-integer programs are usually solved with branch-and-bound techniques, as discussed in Section 3.3, with cutting-plane methods, or with branch-and-cut, a mixture of both. See [Sch86, Flo95] for further details.

Remark 3.3 *With the exception of trivial cases, mixed integer programs do not fall into the class of convex optimization problems, because the domain of the binary variables, $\{0,1\}$, is not convex.*

The main classes in mixed integer programming are the *mixed-integer linear program* (MILP) and the *mixed-integer quadratic program* (MIQP). They can both be stated analogously to their continuous counterpart, i.e.

$$\min_{u \in \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ub}}} \quad c^T u \tag{3.9a}$$

$$\text{s.t.} \quad Au \leq b\,, \tag{3.9b}$$

and

$$\min_{u \in \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ub}}} \quad \frac{1}{2} u^T Q u + c^T u \tag{3.10a}$$

$$\text{s.t.} \quad Au \leq b\,, \tag{3.10b}$$

the only difference being that some of the optimization variables are not continuous but binary variables, $u = [u_c^T \;\; u_b^T]^T, u_c \in \mathbb{R}^{n_{u_b}}, u_b \in \{0,1\}^{n_b}$. The modification of the domain of the optimization problem renders previously easily solvable LPs and QPs NP-hard. More details on mixed-integer quadratic programs can be found e.g. in [VRS87, Axe08].

Note that for fixed binary variables u_b, an MILP (/MIQP) reduces to an LP (/QP). Hence an MILP (/MIQP) can always be solved by enumerating all combinations of binary variables and comparing the solutions of all emerging LPs (/QPs). For the solution of mixed-integer problems under MATLAB, free software such as YALMIP, [Löf04], or commercial solvers such as CPLEX, [CPL94], can be used.

3.2 Parametric Programming

In this section we introduce the concept of parametric programming, which refers to the solution of optimization problems in dependence of a parameter. After considering the general nonlinear case, we recall the main results for parametric linear programs (pLP), parametric quadratic programs (pQP) and parametric mixed-integer programs.

Parametric programming is an extension of *sensitivity analysis*, where the dependence of the solution of an optimization problem to small perturbations of the problem data is investigated. Contrary to the sensitivity analysis, in parametric programming not only small perturbations are considered, but a whole set of possible parameter values.

Some authors in the literature use the term *multi-parametric programming* to emphasize the fact that the considered parameter is not a scalar, but a vector. Since the discussed concepts in this thesis are not affected by this distinction, we will not follow this convention, but use the term parametric programming for both cases. For a detailed description of the mathematical concept and properties of parametric programming, see [BGK+82].

We consider a *(multi-) parametric program* of the form

$$J^*(x) = \min_u J(u;x) \tag{3.11a}$$

$$\text{s.t. } g_i(u;x) \leq 0\,, \quad i = 1,\dots,n_c\,, \tag{3.11b}$$

with the optimization variable $u \in \mathbb{R}^{n_u}$, the *parameter* $x \in \mathbb{R}^{n_x}$, the cost function $J\colon \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \to \mathbb{R}$ and the constraints $g_i\colon \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \to \mathbb{R}$.

Contrary to the optimization problem (3.1), we are interested in solving (3.11) for a set of parameters, $x \in \mathcal{X}$. Instead of an optimal value we obtain an *optimal value function* over the parameter, $J^*\colon \mathbb{R}^{n_x} \to \mathbb{R}$. Concepts such as feasibility or optimality depend on both elements of the pair (u,x). Since our point of view is often that of a map for a given parameter x, we will define these concepts accordingly. The *feasible set map* denotes a point-to-set mapping on the parameter value,

$$\mathcal{U}(x) = \left\{ u \in \mathrm{dom}_u(J) \cap \bigcap_{i=1}^{n_c} \mathrm{dom}_u(g_i) \;\middle|\; g_i(u;x) \leq 0\,,\ i = 1,\ldots,n_c \right\}.$$

A parameter value x is called *feasible* if $\mathcal{U}(x) \neq \emptyset$, otherwise *infeasible*. The *set of feasible parameters* is denoted by

$$\mathcal{X}_f = \{x \in \mathcal{X} \mid \mathcal{U}(x) \neq \emptyset\}.$$

We call a parametric program *infeasible* if $\mathcal{U}(x) = \emptyset\ \forall x \in \mathcal{X}$, otherwise *feasible*. Analogue to the feasible set map, we denote the *optimal set map* by

$$\mathcal{U}^*(x) = \{u \in \mathcal{U}(x) \mid J(u;x) = J^*(x)\}.$$

A *solution* of a parametric program is not necessarily unique, and denotes any function $u^*\colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ which is a single representative of the optimal set map, $u^*(x) \in \mathcal{U}^*(x)$.

By *set of active constraints* we denote the constraints which are active for all solutions of (3.11),

$$\boldsymbol{A}(x) = \{i \in \{1,\ldots,n_c\} \mid g_i(u^*(x);x)) = 0 \quad \forall u^*(x) \in \mathcal{U}^*(x)\}\ .$$

By *set of inactive constraints* we denote the complement of the set of active constraints, i.e.

$$\boldsymbol{I}(x) = \{i \in \{1,\ldots,n_c\} \mid \exists u^*(x) \in \mathcal{U}^*(x) : g_i(u^*(x);x) < 0\}\ .$$

Note that the set of active constraints depends on the value of the parameter, such that there are parameter values with the same and others with a different set of active constraints.

Definition 3.2 (Critical region) *We call a* critical region *the set of all $x \in \mathcal{X}_f$ with the same set of active constraints,*

$$\mathcal{R} := \{x \in \mathcal{X}_f \mid \boldsymbol{A}(x) = \boldsymbol{A}\}.$$

Parametric programming is, in general, a difficult task. Already small examples with simple and smooth constraints can be provided, where the feasible set map $\mathcal{U}(x)$ and the optimal set map $\mathcal{U}^*(x)$ are not continuous in the parameter x. Moreover, the critical regions of the optimal set map are not necessarily easily storable objects. For instance, the critical regions could be semi-algebraic sets, i.e. defined by a finite number of polynomial equalities and inequalities. Therefore the computed critical regions are, in general, difficult to store and to evaluate.

Due to these limitations, the application of parametric programming is mainly restricted to tiny problems with few constraints and a small dimension of the parameter x. The main exception to this observation are parametric programs with polytopic constraints and a linear or quadratic objective functions. Here the critical regions are polyhedra, and can be stored and evaluated more easily. Subsequently, these subclasses are the most common representatives of parametric programming, and will be discussed next.

Parametric Linear Programming

By *(multi-)parametric linear program* (pLP) we denote a linear program, where the right-hand side of the constraints depends affinely on a parameter, i.e. a parametric optimization problem of the form

$$\min_u c^T u \tag{3.12a}$$

$$\text{s.t. } Au \leq b + Ex\,, \tag{3.12b}$$

with $u \in \mathbb{R}^{n_u}, x \in \mathbb{R}^{n_x}, c \in \mathbb{R}^{n_u}, A \in \mathbb{R}^{n_c \times n_u}, b \in \mathbb{R}^{n_c}$ and $E \in \mathbb{R}^{n_c \times n_x}$.

The solution of parametric linear programs was first considered in the Master's thesis of Orchard-Hays in 1952, and was published in [SG54] and [OH55]. A good overview of parametric linear programing is provided in [Gal95, GG97]. Different solvers were proposed for the solution of parametric linear programs, see e.g. [GN72]. A more recent method utilizes lexicographic perturbation to yield a unique solution to parametric linear programs, [JMK07].

Theorem 3.1 (Solution to parametric linear program) *Consider the parametric linear program* (3.12). *The following statements hold:*

(i) The set of feasible parameters $\mathcal{X}_f$ is a closed polyhedral set in $\mathbb{R}^{n_x}$, and $\mathcal{X}_f$ is partitioned into a finite number of polyhedral critical regions.

(ii) The optimal value function $J^*(x)$ is continuous, convex and piecewise affine over $\mathcal{X}_f$ and affine in each critical region $\mathcal{R}$.

(iii) The optimal set map $\mathcal{U}^*(x)$ contains a continuous piecewise affine solution $u^*(x)$.

Definition 3.3 *A parametric linear program* (3.12) *is called* primal degenerate *for a $x \in \mathcal{X}_f$ if there exists a $u^*(x) \in \mathcal{U}^*(x)$ such that the number of active constraints at the optimum is greater than the number n_u of optimization variables.*

Definition 3.4 *A parametric linear program* (3.12) *is called* dual degenerate *for a $x \in \mathcal{X}_f$ if its dual problem is primal degenerate.*

If a parametric linear program is dual degenerate, its solution $u^*(x)$ may not be unique for some $x \in \mathcal{X}_f$.

Parametric Quadratic Programming

By *(multi-)parametric quadratic program* (pQP) we denote a quadratic program, where the constraints and the cost function depend affinely on the parameter in the way which follows by

$$\min_u \frac{1}{2} u^T Q u + x^T F u \tag{3.13a}$$

$$\text{s.t. } Au \leq b + Ex\,, \tag{3.13b}$$

with $u \in \mathbb{R}^{n_u}, Q \in \mathbb{R}^{n_u \times n_u}, F \in \mathbb{R}^{n_x \times n_u}, A \in \mathbb{R}^{n_c \times n_u}, b \in \mathbb{R}^{n_c}$ and $E \in \mathbb{R}^{n_c \times n_x}$.

The mathematical properties of parametric quadratic programs were studied in [BGK+82]. There exist several pQP solvers following the ideas of [BMDP02, Bao02, TJB01].

Theorem 3.2 (Solution to parametric quadratic program) *Consider the parametric quadratic program* (3.13) *with $Q \succ 0$, and assume that LICQ holds. Then the following statements hold:*

(i) The set of feasible parameters $\mathcal{X}_f$ is a polyhedron in $\mathbb{R}^{n_x}$, and $\mathcal{X}_f$ is partitioned into a finite number of polyhedral critical regions.

(ii) The optimal value function $J^*(x)$ is continuous, convex and piecewise quadratic on polyhedra. Moreover, the LICQ ensures that $J^*(x)$ is continuously differentiable.

(iii) The optimizer $u^*(x)$ is continuous and piecewise affine on polyhedra, and it is affine in each critical region. Moreover, $u^*(x)$ is unique (since $Q \succ 0$).

Definition 3.5 *We define a constraint to be* weakly active, *if it is active and the corresponding Lagrange multiplier is zero.*

Definition 3.6 *We say that the* linear independence constraint qualification *(LICQ) holds, if the gradients of the set of active constraints, i.e. the corresponding rows of A (also denoted by $A_{\mathbf{A}}$), are linearly independent.*

If LICQ holds, primal degeneracy can not occur.

Parametric Mixed-Integer Programming

Parametric mixed-integer programs were also considered in the literature, the common classes being parametric mixed-integer linear programs (pMILP) and parametric mixed-integer quadratic programs (pMIQP). For details and solvers for pMILPs see [DP00, AP97].

In the following we consider the *(multi-)parametric mixed-integer quadratic program* (pMIQP) which is stated by

$$\min_{u \in \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ub}}} \quad u^T Q u + x^T F u \tag{3.14a}$$

$$\text{s.t.} \qquad Au \leq b + Ex\,, \tag{3.14b}$$

with $Q \in \mathbb{R}^{n_u \times n_u}, F \in \mathbb{R}^{n_x \times n_u}, A \in \mathbb{R}^{n_c \times n_u}, b \in \mathbb{R}^{n_c}$ and $E \in \mathbb{R}^{n_c \times n_x}$ and $b \in \mathbb{R}^{n_c}$. Contrary to a pQP, some of the optimization variables are binaries, $u = [u_c^T \;\; u_b^T]^T$, with $u_c \in \mathbb{R}^{n_{uc}}, u_b \in \{0,1\}^{n_{ub}}$ and $n_u = n_{uc} + n_{ub}$.

Theorem 3.3 (Solution to parametric mixed-integer quadratic program)
Consider the parametric mixed-integer quadratic program stated in (3.14). *The following statements hold:*

(i) The set of feasible parameters $\mathcal{X}_f$ is a union of polyhedra in $\mathbb{R}^{n_x}$, and $\mathcal{X}_f$ is partitioned into n_r critical regions $\mathcal{R}_r$, $r = 1, \ldots, n_r$, whose closure is of the form

$$\bar{\mathcal{R}}_r = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; x^T H_i x + q_i x \leq r_i\,, \quad i = 1, \ldots, n_c \right\}\,. \tag{3.15}$$

The critical regions are not necessarily convex, nor is the set of feasible parameters $\mathcal{X}_f$ necessarily convex or connected.

(ii) The optimal value function $J^*(x)$ is piecewise quadratic on $\mathcal{X}_f$, but, in general, neither continuous nor convex.

(iii) The optimizer $u^*(x)$ is piecewise affine on $\mathcal{X}_f$, and it is affine in each critical region. The optimizer $u^*(x)$ is, in general, not continuous.

Contrary to parametric mixed-integer linear programs, the critical regions of parametric mixed-integer quadratic program cannot be decomposed into convex polyhedra. This fact hinders the efficient storage and evaluation of the solution in a look-up table.

For fixed binary values, the pMIQP becomes a pQP. Hence one solution strategy for pMIQPs is to compute the solution to the pQPs for all combinations of binary variables. The minimum over the resulting cost functions determines the solution to the pMIQP, see [Bor03]. While each pQP results in polyhedral critical regions, the minimization between piecewise quadratic cost functions results in critical regions of the form (3.15).

A solver for the special case that binary variables enter only affinely in constraints can be found in [DBP02], while a solution approach for the general case (3.14) is indicated in [Bor03].

3.3 Optimization paradigms

This chapter is closed with a brief outlook on two optimization paradigms, branch-and-bound and dynamic programming. Both paradigms are general concepts which can be applied for the solution of certain types of optimization problems.

Branch-and-bound

This section introduces the concept of branch-and-bound, which was proposed by Land and Doig in 1960, [LD60], in the area of Operations Research. *Branch-and-bound* is a general concept for the solution of various optimization problems, e.g. combinatorial optimization, integer programming, nonlinear programming or nonconvex programming. Depending on the problem at hand, there exist specific algorithms which are all based on the same principle. See e.g. [Dak65] for a branch-and-bound algorithm for mixed integer programming problems. In the following we will state the main principle of branch-and-bound.

We consider the general optimization problem

$$\min_{u} J(u) \tag{3.16a}$$

$$\text{s.t. } u \in \mathcal{U} \tag{3.16b}$$

with the *optimization variable* $u \in \mathbb{R}^{n_u}$, the *objective function* $J\colon \mathbb{R}^{n_u} \to \mathbb{R}$ and the *set of candidate solutions* $\mathcal{U} \subseteq \mathbb{R}^{n_u}$. The set of candidate solutions is usually a discrete set consisting of many candidates, such that an enumeration of all candidates would be intractable.

As the name indicates, a branch-and-bound scheme comprises the two components branching and bounding. In the *branching* step the optimization problem is divided into a series of simpler subproblems. A recursive application of the branch step leads to a tree structure, whose nodes are related to subsets of $\mathcal{U}$. The root node covers the entire set of candidate solutions

$\mathcal{U}$, while the leaves correspond to single candidates. In the *bounding* step, suboptimal parts of this tree are cut, if certain criteria are met. These criteria are usually the satisfaction of bounds derived from relaxed versions of the optimization problem. In a successful application of branch-and-bound, the optimizer of (3.16) is found, while the number of investigated candidates is far less than the number of elements in $\mathcal{U}$.

Dynamic Programming

One of the most frequently used tools in this work is *dynamic programming* (DP). In the following we will describe the main idea of dynamic programming by presenting it in its basic form. More details on dynamic programming can be found in [Ber95].

Consider the discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k) \tag{3.17}$$

with the discrete time $k \in \mathbb{N}$, the (discrete or continuous) state $x_k \in \mathbb{R}^{n_x}$ and the control $u_k \in \mathbb{R}^{n_u}$. The state is constrained to a set $x_k \in \mathcal{X}_k$, and only controls within a set, possibly depending on the current state, are possible, $u_k \in \mathcal{U}_k(x_k)$. In dynamic programming we are interested in the computation of *control laws*, $u_k = \mu_k(x_k)$, which map states to controls. We call a control law *admissible*, if $\mu_k(x_k) \in \mathcal{U}_k(x_k)$ for all $x_k \in \mathcal{X}_k$. We call a sequence of control laws, $\boldsymbol{\pi} = \{\mu_0, \dots, \mu_{N-1}\}$, a *(control) policy*, and a policy belongs to the set of admissible policies $\boldsymbol{\pi} \in \Pi$, if it consists of admissible control laws.

Starting with an initial state x_0, we will sustain some *stage costs* $L_k(x_k, u_k)$ at each stage, which accumulate over time to the additive *cost function*

$$J(\boldsymbol{\pi}; x_0) = L_N(x_N) + \sum_{k=0}^{N-1} L_k(x_k, u_k)\,, \tag{3.18}$$

where N denotes the considered horizon, and $L_N(x_N)$ the terminal cost at the end of this horizon. The aim of dynamic programming is to compute a policy which minimizes the occurring cost function by solving the optimization problem

$$J^*(x_0) = \inf_{\boldsymbol{\pi}\in\Pi} J(\boldsymbol{\pi}; x_0)\,, \qquad \boldsymbol{\pi}^*(x_0) = \arg \inf_{\boldsymbol{\pi}\in\Pi} J(\boldsymbol{\pi}; x_0)\,. \tag{3.19}$$

We call $J^*(x_0)$ the *optimal value function*, and $\boldsymbol{\pi}^*(x_0)$ a (not necessarily unique) *optimal policy*.

Dynamic programming is based on the *Principle of Optimality*, which goes back to Bellman, [Bel57].

Principle of Optimality: Let $\boldsymbol{\pi}^* = \{\mu_0^*, \dots, \mu_{N-1}^*\}$ be an optimal policy for the system (3.17) by means of the cost function (3.18). Then for all $k \in \{1, \dots, N-1\}$, the policy tail $\boldsymbol{\pi}_k^* = \{\mu_k^*, \dots, \mu_{N-1}^*\}$ is an optimal policy regarding the corresponding *truncated cost function*

$$J_k(\pi_k; x_k) = L_N(x_N) + \sum_{i=k}^{N-1} L_i(x_i, u_i)\,. \tag{3.20}$$

Dynamic programming exploits the principle of optimality by resolving the optimization problem (3.19) into a sequence of simpler optimization problems. Instead of solving (3.19) directly, it is solved step-by-step going backwards in time.

Proposition 3.4 (Dynamic programming algorithm) *For any given initial state x_0, the optimal value $J^*(x_0)$ obtained by solving the optimization problem* (3.19) *equals the optimal value $J_0(x_0)$ obtained by executing the following algorithm. The algorithm is initialized with*

$$J_N(x_N) = L_N(x_N)\,. \tag{3.21}$$

Then the following optimization problems from $k = N-1$ to $k = 0$ are solved backwards in time

$$J_k(x_k) = \inf_{u_k \in \mathcal{U}_k(x_k)} L_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k))\,. \tag{3.22}$$

Furthermore, the sequence of the resulting control laws $\boldsymbol{\pi}^* = \{\mu_0^*, \dots, \mu_{N-1}^*\}$ is optimal.

The functions $J_k(x_k)$ are also referred to as *cost-to-go*. For the sake of completeness, we will restate a proof for Proposition 3.4, which holds under mild technical assumptions (i.e. that the functions J_k are well-defined and finite).

Proof ([Ber95]) Let $J_k^*(x_k)$ be the optimal cost for the optimization problem

$$J_k^*(x_k) = \inf_{\pi_k} L_N(x_N) + \sum_{i=k}^{N-1} L_i(x_i, \mu_i(x_i))\,. \tag{3.23}$$

For $k = N$, we define $J_N^*(x_N) = L_N(x_N)$. We will show by induction that $J_k^*(x_k) = J_k(x_k)$ as defined in (3.22) which for $k = 0$ incorporates Proposition 3.4. By definition we have $J_N^*(x_N) = J_N(x_N) = L_N(x_N)$. Under the assumption that $J^*_{k+1}(x_{k+1}) = J_{k+1}(x_{k+1})$ holds for some $k \in \{1, \dots, N-1\}$, it follows that

$$\begin{aligned}
J^*_k(x_k) &= \inf_{\{\mu_k, \pi_{k+1}\}} L_k(x_k, \mu_k(x_k)) + L_N(x_N) + \sum_{i=k+1}^{N-1} L_i(x_i, \mu_i(x_i)) \\
&= \inf_{\mu_k} L_k(x_k, \mu_k(x_k)) + \inf_{\pi_{k+1}} \left[L_N(x_N) + \sum_{i=k+1}^{N-1} L_i(x_i, \mu_i(x_i)) \right] \\
&= \inf_{\mu_k} L_k(x_k, \mu_k(x_k)) + J^*_{i+1}(f_k(x_k, \mu_k(x_k))) \\
&= \inf_{\mu_k} L_k(x_k, \mu_k(x_k)) + J_{k+1}(f_k(x_k, \mu_k(x_k))) \\
&= \inf_{u_k \in \mathcal{U}_k(x_k)} L_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k)) \\
&= J_k(x_k)
\end{aligned} \tag{3.24}$$

■

4 Optimal Control of Constrained Discrete-Time Systems

'The more unpredictable the world is the more we rely on predictions.'

Steve Rivkin

Model predictive control (MPC) is the subject of interest in this chapter. The basic concepts of model predictive control are explained for rather general nonlinear systems without going into the details of numerical computations.

In model predictive control, the applied control inputs are determined as the solution to some optimal control problems, which are based on model predictions over a finite-time horizon. Therefore the first sections of this chapter consider a series of finite-time optimal control problems and solution approaches to tackle them. Each of these optimal control problems is related to a different type of discrete-time dynamical system, or a different control objective. These sections on the different optimization problems are rather similar, and it is recommended that the interested reader picks the optimization problem related to his problem at hand without the need to read through all other sections.

Finally the employment of these optimization problems in model predictive control is presented. Extensions to classical MPC schemes are mentioned and references to more elaborated works are given.

4.1 Constrained Finite-Time Optimal Control

At first we consider the constrained finite-time optimal control problem. More precisely, we consider a discrete-time model of the system to be controlled,

$$x_{k+1} = f(x_k, u_k)\,, \tag{4.1}$$

with the discrete time $k \in \mathbb{N}$, the *state* $x_k \in \mathbb{R}^{n_x}$ and the *control (action)* $u_k \in \mathbb{R}^{n_u}$. The state is constrained to a set $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and only controls within a set, possibly depending on the current state, are possible, $u_k \in \mathcal{U}(x_k) \subseteq \mathbb{R}^{n_u}$. The constrained discrete-time system (4.1) shall be controlled by state feedback, $u_k = \mu(x_k)$, as illustrated in Figure 4.1.

Starting with our current state x_k, we assume that at each stage we will sustain some *stage costs* $L_i(x_{k+i}, u_{k+i})$, which accumulate over time to the additive *cost function*

$$J(\boldsymbol{U}_k; x_k) = L_N(x_{k+N}) + \sum_{i=0}^{N-1} L_i(x_{k+i}, u_{k+i})\,, \tag{4.2}$$

where N denotes the considered *prediction horizon*, and the term $L_N(x_N)$ denotes the *terminal cost* at the end of this horizon, and penalizes the assumed costs beyond the prediction horizon. For notational simplicity we denote the *sequence of control inputs* during the prediction horizon by

$$\boldsymbol{U}_k = \{u_k, u_{k+1}, \ldots, u_{k+N-1}\}\,. \tag{4.3}$$

At best, the finite-horizon cost function (4.2) with the terminal cost $L_N(x_N)$ is equivalent to the infinite horizon cost without the need to let $N \to \infty$, [NP97, MRRS00].

The objective of constrained finite-time optimal control is to apply a sequence of control actions (4.3), which is optimal by means of the finite-time cost function (4.2) for the constrained system (4.1). Or, to put it mathematically, the objective is to solve the *constrained finite-time optimal control* (CFTOC) problem

$$\begin{aligned}
J^*(x_k) = \min_{\boldsymbol{U}_k} \quad & J(\boldsymbol{U}_k; x_k) && (4.4a)\\
\text{s.t.} \quad & x_{k+i} \in \mathcal{X}, \quad i = 0, \ldots, N-1\,, && (4.4b)\\
& x_{k+N} \in \mathcal{X}_T\,, && (4.4c)\\
& u_{k+i} \in \mathcal{U}(x_{k+i}), \quad i = 0, \ldots, N-1\,, && (4.4d)\\
& x_{k+i+1} = f(x_{k+i}, u_{k+i}), \quad i = 0, \ldots, N-1\,, && (4.4e)
\end{aligned}$$

where $\mathcal{X}_T \subseteq \mathbb{R}^{n_x}$ is the *terminal set* describing the target set of the state at the

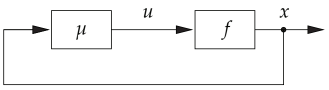


Figure 4.1: Control loop considered in constrained finite-time optimal control.

end of the prediction horizon. The choice of $\mathcal{X}_T$ and the terminal cost term $L_N(x_N)$ is typically determined by the satisfaction of stability guarantees in a receding horizon approach (more details on the receding horizon strategy will be given in Section 4.5). These stability guarantees can lead to conservatism if the resulting CFTOC problem resembles the infinite-horizon cost only poorly.

Remark 4.1 *Note that the system* (4.1) *as well as the stage costs in the cost function* (4.2) *are* time-invariant, *such that the CFTOC problem* (4.4) *is only depending on the initial state x_k and the applied input sequence, but not on time k. It would be straightforward to extend the problem to a time-varying optimal control problem, but this would necessitate a different solution at each sampling instance.*

Solution of CFTOC problems by mathematical programming

The CFTOC problem (4.4) is an optimization problem as discussed in Section 3.1 on mathematical programming. Hence the most straightforward approach to solve (4.4) is to insert the value of the current state x_k, and to solve (4.4) by using an appropriate optimization technique. In this approach the state feedback $u_k = \mu(x_k)$ in Figure 4.1 is an *implicit* relation specified by the CFTOC problem (4.4). We call the resulting J^* the *optimal value* and the solution

$$\boldsymbol{U}_k^* = \{u_k^*, u_{k+1}^*, \dots, u_{k+N-1}^*\}$$

the *optimal control sequence*. Depending on the discrete-time system at hand, and on the choice of the cost function, the CFTOC problem (4.4) can take the form of a specific optimization problem, e.g. a QP for a linear system with linear constraints and a quadratic cost function.

Solution of CFTOC problems by parametric programming

A different approach for solving the CFTOC problem (4.4) is to make use of the parametric optimization techniques presented in Section 3.2. In this approach the optimizer of the CFTOC problem is not a sequence of control actions, but a sequence of *control laws* which map the current state to future controls,

$$u_{k+i} = \mu_i(x_k), \quad i = 0, \dots, N-1\,. \tag{4.5}$$

We call $J^*(x_k)$ the *optimal value function*, and $\boldsymbol{\pi}^*(x_k) = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$ a (not necessarily unique) *optimal policy*. Note that the predicted future control laws depend on the current state x_k.

With parametric programming the state feedback $u_k = \mu(x_k)$ in Figure 4.1 can be obtained *explicitly*. The parametric programming approach has the advantage that the optimization does not have to be repeated for different states x_k, but that the parametric solution, once obtained, can be used for a whole range of states. On the other hand, the solution of parametric optimization problems is a difficult task, and only possible if the problem to solve is simple enough. Moreover, efficient solvers do only exist for parametric linear programs and parametric quadratic programs, such that in practice one basically is constrained to these problem classes.

Solution of CFTOC problems by dynamic programming

Another possibility to solve the CFTOC problem (4.4) utilizes dynamic programming, which was introduced in Section 3.3. Dynamic programming is based on the *Principle of Optimality*, and traces back to Bellman, [Bel57]. In dynamic programming we are interested in the computation of *control laws*, which map the future states to future controls,

$$u_{k+i} = \mu_i(x_{k+i}) \quad i = 0, \dots, N-1\,. \tag{4.6}$$

Note that in contrast to (4.5), the predicted future control laws are functions of the predicted future states. For the CFTOC problem considered so far the distinction between control actions / control laws of the current state on the one side and control laws of the predicted state on the other side is merely of conceptual interest and all three approaches yield the same sequence of control actions $\boldsymbol{U}_k^*$ for a specific x_k. We will see later on that this distinction indeed has practical significance if the prediction model differs from the system to be controlled. If uncertainties are taken into account by the prediction model, this distinction will even lead to a different sequence of control actions, see Section 4.2.

Dynamic programming exploits the principle of optimality by decomposing the optimization problem (4.4) into a sequence of simpler optimization problems. Instead of solving the CFTOC problem (4.4) directly, it is solved step-by-step going backwards in time. Contrary to the direct parametric solution of the whole CFTOC problem, the parametric solver optimizes only over a single control law in each iteration. Nevertheless, the dynamic programming approach is also based on parametric programming and as such limited to small problems, which can be transformed into pLPs or pQPs.

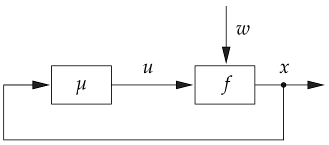


Figure 4.2: Control loop considered in constrained robust optimal control.

4.2 Constrained Robust Optimal Control

When controlling a dynamical system, an acquired model typically has a certain accuracy, and is only an approximation of the system at hand. Moreover, dynamical systems are affected by disturbances and process noise, and the perceived state of the system is often estimated and/or affected by measurement noise. Guarantees on stability and performance are therefore of limited use if they are not valid under the influence of some uncertainty.

A control system is called *robust* if stability and performance requirements are satisfied for a specified range of uncertainty. A whole branch of control is dedicated to the design of robust control systems, [ZDG96, SP05]. By taking uncertainties during the controller design into account, a robust controller is computed which meets the requirements as long as the uncertainty remains in the specified range. The robust control of constrained discrete-time systems probably started with the pioneering work of Witsenhausen, [Wit68]. In the following section we will review an extension of the CFTOC problem for uncertain systems, the constrained robust optimal control (CROC) problem.

We consider a discrete-time uncertain model of the system to be controlled,

$$x_{k+1} = f(x_k, u_k, w_k)\,, \tag{4.7}$$

with the discrete time $k \in \mathbb{N}$, the *state* $x_k \in \mathbb{R}^{n_x}$, the *control (action)* $u_k \in \mathbb{R}^{n_u}$ and the *uncertainty* $w_k \in \mathbb{R}^{n_w}$ (which models exogenous disturbances and parametric uncertainties). The state is constrained to a set $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and only controls within a set, possibly depending on the current state, are possible, $u_k \in \mathcal{U}(x_k) \subseteq \mathbb{R}^{n_u}$. Moreover we assume that the exact values of the uncertainty w_k are unknown, merely the uncertainty is known to be constrained to a bounded set, $w_k \in \mathcal{W} \subset \mathbb{R}^{n_w}$. The uncertain discrete-time system (4.7) shall be controlled by state feedback as illustrated in Figure 4.2.

Analogue to the CFTOC problem, we assume that starting with our current state x_k at each stage we will sustain some *stage costs* $L_i(x_{k+i}, u_{k+i})$, which accumulate over time to the additive *cost function*

$$J(\boldsymbol{U}_k, \boldsymbol{W}_k; x_k) = L_N(x_{k+N}) + \sum_{i=0}^{N-1} L_i(x_{k+i}, u_{k+i}), \tag{4.8}$$

where N denotes the considered *prediction horizon*, and the term $L_N(x_N)$ denotes the *terminal cost* at the end of this horizon, and penalizes the assumed costs beyond the prediction horizon. For notational simplicity we denote the *sequence of control inputs* and the *sequence of uncertainties* during the prediction horizon by

$$\boldsymbol{U}_k = \{u_k, u_{k+1}, \dots, u_{k+N-1}\}, \quad \text{and} \quad \boldsymbol{W}_k = \{w_k, w_{k+1}, \dots, w_{k+N-1}\},$$

respectively. Note that the cost function (4.8) depends implicitly on the sequence of uncertainties via (4.7).

If more information about the uncertainty such as the probability density function is available, this information can be exploited in the optimization problem, leading to stochastic optimization approaches [BIRS98]. In the following however it is assumed that no probability information about the uncertainty is available. In order to accommodate for the uncertainty, usually a min-max approach is chosen, minimizing the worst-case cost function while satisfying the constraints for all possible realizations of the uncertainty. Two approaches are distinguished, the former one being the *open-loop constrained robust optimal control* (OL-CROC) problem,

$$\begin{aligned} J^*(x_k) = \min_{\boldsymbol{U}_k} \max_{\boldsymbol{W}_k} \quad & J(\boldsymbol{U}_k, \boldsymbol{W}_k; x_k) && \text{(4.9a)} \\ \text{s.t.} \quad & x_{k+i} \in \mathcal{X}, \quad i = 0, \dots, N-1, && \text{(4.9b)} \\ & x_{k+N} \in \mathcal{X}_T, && \text{(4.9c)} \\ & u_{k+i} \in \mathcal{U}(x_{k+i}), \quad i = 0, \dots, N-1, && \text{(4.9d)} \\ & w_{k+i} \in \mathcal{W}, \quad i = 0, \dots, N-1, && \text{(4.9e)} \\ & x_{k+i+1} = f(x_{k+i}, u_{k+i}, w_{k+i}), \quad i = 0, \dots, N-1. && \text{(4.9f)} \end{aligned}$$

The latter one is called *closed-loop constrained robust optimal control* (CL-CROC) problem,

$$\begin{aligned} J^*(x_k) = \min_{u_k} \max_{w_k} \dots \min_{u_{k+N-1}} \max_{w_{k+N-1}} \quad & J(\boldsymbol{U}_k, \boldsymbol{W}_k; x_k) && \text{(4.10a)} \\ \text{s.t.} \quad & x_{k+i} \in \mathcal{X}, \quad i = 0, \dots, N-1, && \text{(4.10b)} \\ & x_{k+N} \in \mathcal{X}_T, && \text{(4.10c)} \\ & u_{k+i} \in \mathcal{U}(x_{k+i}), \quad i = 0, \dots, N-1, && \text{(4.10d)} \\ & w_{k+i} \in \mathcal{W}, \quad i = 0, \dots, N-1, && \text{(4.10e)} \\ & x_{k+i+1} = f(x_{k+i}, u_{k+i}, w_{k+i}), \quad i = 0, \dots, N-1. && \text{(4.10f)} \end{aligned}$$

The min-max formulation (4.9) is based on *open-loop* predictions, while the formulation (4.10) employs *closed-loop* predictions. The conceptual difference between open-loop and closed-loop predictions lies in the assumption of future control actions to be able to react to experienced disturbances (closed-loop predictions), or not (open-loop predictions). Depending on the system at hand, the OL-CROC problem typically results in simpler optimization problems than CL-CROC, but suffers from excessive conservatism, [Bem98].

The reason for this conservatism becomes understandable by considering the control input and the uncertainty as opponents in a dynamic game. The control input wants to minimize the costs, while the disturbance wants to maximize the cost function. In the open-loop setup, the control input has to reveal all his moves to the uncertainty. The uncertainty is able to take this information into account, when deciding about its own moves. Hence the uncertainty gains advantage just by the order of the moves, which can even lead to infeasibility of the optimization problem. In the closed-loop predictions on the other hand, the players' moves alternate, such that it is easier for the control input to mitigate the influence of the uncertainty.

Solution of CROC problems

Due to the considered uncertainty, constrained robust optimal control problems are typically more difficult to solve than CFTOC problems. A common approach for the solution of CROC problems is to minimize a tight upper bound of the cost function which coincides at the solution with the original problem, and thus to recast the CROC problem as a convex optimization problem, [BBM03, KBM96]. In the case of the CL-CROC problem this is typically accompanied by a more or less restrictive parametrization of the input as a function of the predicted state or uncertainty. These parametrizations reduce the computational effort required to solve the CROC problem. A common class of input parametrizations is the so-called affine disturbance feedback, [Löf03], which recently was shown to be optimal for a certain class of systems, [BIP09].

For some system and problem classes it is possible to solve the CROC problem parametrically, yielding control laws instead of control actions. The OL-CROC problem is typically solved by two parametric programs, and the obtained control laws are functions of the current state,

$$u_{k+i} = \mu_i(x_k)\,. \tag{4.11}$$

The CL-CROC problem is typically solved by dynamic programming. The control laws are functions of the predicted state,

$$u_{k+i} = \mu_i(x_{k+i}) = \mu_i(x_k, w_k, \dots, w_{k+i-1}), \tag{4.12}$$

and thus are responsive to the preceding realizations of the uncertainty. The previous control inputs are also functions of the current state x_k and the preceding uncertainties and thus are absorbed by (4.12). By comparing (4.11) with (4.12), the benefit of using closed-loop predictions becomes obvious: the control action can be adapted based on knowledge of previous uncertainty values, while the open-predictions assume the control to be insensitive to previous uncertainty values, yielding conservative predictions.

4.3 Constrained Parameter-Varying Optimal Control

Another extension of the CFTOC problem occurs, when the system dynamics depend on some *scheduling parameter* (not to be mistaken for the parameter in a parametric program). Contrary to uncertainty, a scheduling parameter is assumed to be accessible at the current time step, but its future values are unknown a-priori. Examples would be measurable disturbances, or time-varying system parameters. Knowledge of the scheduling parameter can and should be exploited to adapt the applied control actions and thus to improve control performance. In this section we will investigate the constrained finite-time optimal control problem for systems affected by such a scheduling parameter.

We consider a discrete-time parameter-varying model of the system to be controlled,

$$x_{k+1} = f(x_k, u_k, \theta_k), \tag{4.13}$$

with the discrete time $k \in \mathbb{N}$, the *state* $x_k \in \mathbb{R}^{n_x}$, the *control (action)* $u_k \in \mathbb{R}^{n_u}$ and the *scheduling parameter* $\theta_k \in \mathbb{R}^{n_\theta}$. The state is constrained to a set $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and only controls within a set, possibly depending on the current state, are possible, $u_k \in \mathcal{U}(x_k) \subseteq \mathbb{R}^{n_u}$. Moreover we assume that the scheduling parameter θ_k is measurable such that the current value of the scheduling parameter is known, but future values of the scheduling parameter are unknown a-priori, merely the scheduling parameter is known to be constrained to a bounded set, $\theta_k \in \Theta \subset \mathbb{R}^{n_\theta}$. The parameter-varying discrete-time system (4.13) shall be controlled by a state and scheduling parameter feedback as illustrated in Figure 4.3.

Analogue to the CFTOC problem, we assume that starting with our current state x_k at each stage we will sustain some *stage costs* $L_i(x_{k+i}, u_{k+i})$, which

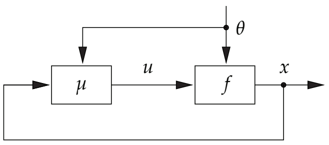


Figure 4.3: Control loop considered in constrained parameter-varying optimal control.

accumulate over time to the additive *cost function*

$$J(\boldsymbol{U}_k, \boldsymbol{T}_k; x_k, \theta_k) = L_N(x_{k+N}) + \sum_{i=0}^{N-1} L_i(x_{k+i}, u_{k+i}) \,, \tag{4.14}$$

where N denotes the considered *prediction horizon*, and the term $L_N(x_N)$ denotes the *terminal cost* at the end of this horizon, and penalizes the assumed costs beyond the prediction horizon. For notational simplicity we denote the *sequence of control inputs* during the prediction horizon by

$$\boldsymbol{U}_k = \{u_k, u_{k+1}, \dots, u_{k+N-1}\} \,,$$

and the unknown *sequence of future scheduling parameters* by

$$\boldsymbol{T}_k = \{\theta_{k+1}, \dots, \theta_{k+N-1}\} \,,$$

respectively. Note that the cost function (4.14) depends implicitly on the sequence of scheduling parameters via (4.13).

One typically wants to exploit the scheduling parameter information for control purposes, while guaranteeing constraint satisfaction for all realizations of the future scheduling parameter. Mathematically speaking this can be posed as a problem very similar to the CL-CROC problem, the *constrained parameter-varying optimal control* (CPVOC) problem,

$$\begin{aligned}
J^*(x_k, \theta_k) = &\min_{u_k} \max_{\theta_{k+1}} \min_{u_{k+1}} \dots \max_{\theta_{k+N-1}} \min_{u_{k+N-1}} \quad J(\boldsymbol{U}_k, \boldsymbol{T}_k; x_k, \theta_k) && \text{(4.15a)}\\
\text{s.t.} \quad & x_{k+i} \in \mathcal{X}, \quad i = 0, \dots, N-1 \,, && \text{(4.15b)}\\
& x_{k+N} \in \mathcal{X}_T \,, && \text{(4.15c)}\\
& u_{k+i} \in \mathcal{U}(x_{k+i}), \quad i = 0, \dots, N-1 \,, && \text{(4.15d)}\\
& \theta_{k+i} \in \Theta, \quad i = 0, \dots, N-1 \,, && \text{(4.15e)}\\
& x_{k+i+1} = f(x_{k+i}, u_{k+i}, \theta_{k+i}), \quad i = 0, \dots, N-1 \,. && \text{(4.15f)}
\end{aligned}$$

Note that from a game-theoretic point of view the order of the moves of the control input and the scheduling parameter changed, when compared to the

CL-CROC problem. The control input can make each move *after* the move of the scheduling parameter, reflecting the fact, that the scheduling parameter θ_{k+i} will be known to the controller at the time instance $k+i$.

Solution of CPVOC Problems

The constrained parameter-varying optimal control problem is again a constrained optimization problem, and thus could be solved with the mathematical programming tools described in Section 3.1.

We are interested in solving the CPVOC problem by means of dynamic programming, yielding control laws instead of control actions. Note that the final dynamic programming step differs from the previous ones, since a maximization over the current scheduling parameter is not necessary. Each control law of the optimal policy is a function of the predicted state and the predicted scheduling parameter,

$$u_{k+i} = \mu_i(x_{k+i}, \theta_{k+i}) = \mu_i(x_k, \theta_k, \dots, \theta_{k+i-1}) \,. \tag{4.16}$$

Contrary to the optimal policy of the CL-CROC problem, these control laws are in possession of knowledge of the current scheduling parameter values.

4.4 Constrained Time-Optimal Control

While the CROC problem and the CPVOC problem are variations of the CFTOC problem, which incorporate model uncertainties or scheduling parameters into the constrained finite-time optimal control problem, the constrained time-optimal control (CTOC) problem considers a different control objective. Instead of minimizing a cost function, the number of time steps to reach a target set shall be minimized. In the following, we will describe the constrained time optimal control problem for nonlinear discrete-time systems; extensions which take model uncertainties and/or scheduling parameters into account, are possible.

In constrained time-optimal control we consider a discrete-time model of the system to be controlled,

$$x_{k+1} = f(x_k, u_k) \,, \tag{4.17}$$

with the discrete time $k \in \mathbb{N}$, the *state* $x_k \in \mathbb{R}^{n_x}$ and the *control (action)* $u_k \in \mathbb{R}^{n_u}$. We assume the state to be constrained to a set $x_k \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, and only controls within a set, possibly depending on the current state, are applicable, $u_k \in \mathcal{U}(x_k) \subseteq \mathbb{R}^{n_u}$. For notational simplicity, we use a *sequence of control actions*

$$\boldsymbol{U}_k = \{u_k, u_{k+1}, \ldots, u_{k+N-1}\}\,.$$

Contrary to the CFTOC problem, we do not want to minimize a predicted cost function, but we want to apply a control sequence which minimizes the number of time steps needed to reach a target region $\mathcal{X}_T \subseteq \mathbb{R}^{n_x}$ from our current state x_k. The *constrained time-optimal control* (CTOC) problem (also called constrained minimum-time control problem) is defined as

$$\min_{\{\boldsymbol{U}_k, N\}} \quad N(\boldsymbol{U}_k; x_k) \tag{4.18a}$$

$$\text{s.t.} \quad x_{k+i} \in \mathcal{X}, \quad i = 0, \ldots, N-1\,, \tag{4.18b}$$

$$x_{k+N} \in \mathcal{X}_T\,, \tag{4.18c}$$

$$u_{k+i} \in \mathcal{U}(x_{k+i}), \quad i = 0, \ldots, N-1\,, \tag{4.18d}$$

$$x_{k+i+1} = f(x_{k+i}, u_{k+i}), \quad i = 0, \ldots, N-1\,, \tag{4.18e}$$

where N denotes the number of time steps needed to reach the target set.

Solution of CTOC problems

One strategy to solve the CTOC problem straightforward would be to start with $N = 0$ and then to increment N iteratively while checking if the resulting constraints of (4.18) are feasible for the current state x_k. As soon as the optimization problem becomes feasible for a certain N^*, apply *any* feasible sequence of control actions. Assuming it is possible to steer the state x_k to the target set, this procedure will eventually return an optimal solution $\boldsymbol{U}_k^*$ to the CTOC problem (4.18).

Nevertheless, there are some shortcomings of this strategy. The first is that the number of steps to reach the target set would have to be computed over and over again, while the sets of states, which require a certain number of steps, are constant in time. Consequently a common strategy is to precompute these sets of states, which can be mapped to the target set in a certain number of steps. This computation can be performed by applying the backwards propagation algorithm from page 16, initialized with the target set $\mathcal{X}_T$. The sets $\mathcal{Z}_0, \mathcal{Z}_1, \ldots, \mathcal{Z}_\infty$ are stored, where $\mathcal{Z}_0 = \mathcal{X}_T$, and $\mathcal{Z}_\infty$ denotes the maximum control invariant set. The determination of the minimum number of steps to reach the target set from x_k then reduces to some set membership tests, which can be stated as

$$\begin{aligned} N^* = \min_{i} \; & i && (4.19a)\\ \text{s.t. } & x_k \in \mathcal{Z}_i\,. && (4.19b) \end{aligned}$$

After determining N^*, a feasible control law has to be computed. The precomputed set collection $\{\mathcal{Z}_i\}_{i=0}^{\infty}$ can help to simplify this computation, because, in order to maintain time-optimality, it suffices if the succeeding state x_{k+1} is an element of $\mathcal{Z}_{N^*-1}$. One property of constrained time-optimal control is that all feasible sequences of control actions for N^* are optimal solutions of (4.18), i.e. all feasible solutions are considered equally good. However, this is an inappropriate assessment in a control setting since control practitioners want to be able to adjust the closed-loop behaviour to their preferences. Hence, often an auxiliary cost function $J(u_k;x_k)$ is introduced, leading to the following optimization problem:

$$\begin{aligned} J^*(x_k) = \min_{u_k} \quad & J(u_k;x_k) && (4.20a)\\ \text{s.t.} \quad & x_{k+1} \in \mathcal{Z}_{i-1}\,, && (4.20b)\\ & x_k \in \mathcal{X}\,, && (4.20c)\\ & u_k \in \mathcal{U}(x_k)\,, && (4.20d)\\ & x_{k+1} = f(x_k,u_k)\,, && (4.20e) \end{aligned}$$

where $\mathcal{Z}_{i-1}$ refers to $\mathcal{Z}_{N^*-1}$. Note the similarity of (4.20) to the CFTOC problem (4.4). When the same cost function is used, the difference is the number of variables and the number of constraints, which is typically smaller in the CTOC problem.

If the considered system and the cost function are such that the optimization problem (4.20) is a sufficiently small linear or quadratic program, it is possible to solve (4.20) parametrically. By applying a dynamic programming approach, the set collection $\{\mathcal{Z}_i\}_{i=0}^{\infty}$ and the corresponding control laws can be precomputed at once. Starting with $i = 1$ and the terminal set $\mathcal{Z}_0 = \mathcal{X}_T$, the solution of (4.20) provides a control law $\mu_1(x_k)$ and the set of feasible states $\mathcal{Z}_1$. Then i is incremented and the next iteration is proceeded until the maximum control invariant set $\mathcal{Z}_\infty$ is reached. Online, the set membership test (4.19) is executed and the control law $\mu_{N^*}(x_k)$ is evaluated.

4.5 Model Predictive Control

In this section the control method model predictive control (MPC) is presented. Different aspects such as reference tracking, move blocking or stability properties are discussed. Finally, explicit MPC is introduced.

The solution of a single finite-time optimal control problem for a given state x_k is not sufficient to control a constrained dynamical system in practice. Sooner or later the end of the prediction horizon is reached, such that new control inputs are needed. But there is a more pressing reason for an earlier re-computation of the control inputs: the model which we have identified for the system at hand is only of limited accuracy, and does not account for disturbances and noise in the process or measurements.

In order to add *feedback* to the control system, another approach is taken in which only the current control action u_k^* is applied, whereas the solution of the finite-time optimal control problem is repeated at each time step. Information contained in the current measurements of the system can thus be taken into account, allowing the system to react to unpredicted events and adding a certain robustness to the control. This approach is commonly referred to as the *receding horizon strategy* (because the prediction horizon recedes at every step). The receding horizon strategy ensures that at each time step the computed inputs depend on predictions of the same horizon length.

The solution of a constrained finite-time optimal control problem online at each time step in a receding horizon strategy is what is usually referred to as *Model Predictive Control* (MPC). In MPC the *control* inputs are computed by repeatedly solving optimization problems which incorporate finite-horizon *predictions* based on a discrete-time *model* of the system. More precisely, in MPC the following algorithm is executed repeatedly:

Algorithm 4.1 MPC Algorithm

```
1: Measure the current state $x_k$ (and the scheduling parameter $\theta_k$).
2: Determine the optimal control sequence $u_k^*, u_{k+1}^*, \ldots, u_{k+N-1}^*$.
3: Apply $u_k^*$.
4: Increment $k$.
```

In variations of MPC different optimization problems are solved repeatedly online, some of which were discussed earlier in this chapter. Examples for such variations of MPC are *robust MPC*, where instead of the CFTOC problem a CROC problem is solved in a receding horizon fashion, or *minimum-time MPC*, where the CTOC problem is solved instead. In the context of robust MPC, one distinguishes *open-loop MPC* and *closed-loop MPC* approaches, depending on the fact if in the prediction future control actions can react on experienced uncertainties, as in the CL-CROC problem, or not, as in the OL-CROC problem. In the following we will describe MPC in its standard form including the solution of the CFTOC problem at each time instance, but the reader should keep in mind that these variations exist, and that the consideration of other finite-horizon optimal control problems is possible.

Model predictive control traces back at least until 1963, when Propoi presented basic ideas of MPC, [Mac01, RM09]. Contrary to many other modern control techniques, model predictive control has been used in a vast amount of applications over the last decades, mainly in the process industry, [QB03]. The reasons for the success of MPC lie mainly in the flexibility towards the system and problem description, the ability to take constraints on the states and inputs directly into account, and, compared to other methods, the relatively small engineering effort to accomplish all this. Moreover, the choice of the objective function gives the opportunity to tune the behaviour of the closed-loop system to the needs of the application at hand. On the other hand, MPC requires the solution of optimization problems at each time step. Consequently, it is restricted to relatively slow systems and/or requires a considerable amount of computational power.

Choice of the Cost Function

Until now we did not specify the exact shape of the cost function. In general an additive cost function

$$J(\boldsymbol{U}_k; x_k) = L_N(x_{k+N}) + \sum_{i=0}^{N-1} L_i(x_{k+i}, u_{k+i}) \tag{4.21}$$

is used, as already introduced earlier in this chapter. The terminal cost $L_N(x_{k+N})$ and the stage costs $L_i(x_{k+i}, u_{k+i})$ are usually positive (semi-)definite functions. We will now discuss typical choices of the cost function.

A natural choice for the stage costs and terminal cost is a *quadratic cost function* over the predicted states and inputs,

$$L_i(x_{k+i}, u_{k+i}) = x_{k+i}^T Q x_{k+i} + u_{k+i}^T R u_{k+i}\,, \tag{4.22}$$

$$L_N(x_{k+N}) = x_{k+N}^T P x_{k+N}\,, \tag{4.23}$$

with the real, positive (semi-)definite weight matrices $Q \succeq 0, R \succ 0$ and $P \succ 0$. Usually diagonal matrices are chosen, but this is not necessary. The weight matrices may also vary over the prediction horizon, $Q \to Q_i$ and $R \to R_i$. Occasionally the quadratic terms of the cost function are written as $\|x\|_Q^2 := x^T Q x$. Some authors refer to the quadratic cost function also as the *quadratic norm*, though it is no norm in a strict sense. The quadratic cost function is motivated by its emulation of a generalized 'energy' of a signal. Furthermore, if the system is linear and the constraints polyhedral, the CFTOC problem is a quadratic problem, which can be solved easily with a QP solver (cf. Section 3.1). Given R is positive definite, the optimizer of the QP is unique. The quadratic terms penalize strong deviations from the origin much more than small deviations, leading to large control actions for a large error and smooth behaviour close to the origin. Finally, the tuning effort is modest, since the optimizer varies smoothly with the selected weight matrices.

Another common type of cost functions are induced by polyhedral norms as defined in Def. B.55 on page 260, usually the 1-norm or the ∞-norm. Some authors refer to these norms also as *linear norms*, though polyhedral norms result not in linear, but in *piecewise linear cost functions*. In this setup, the stage costs and terminal cost are

$$L_i(x_{k+i}, u_{k+i}) = \|Qx_{k+i}\|_p + \|Ru_{k+i}\|_p \,, \tag{4.24}$$
$$L_N(x_{k+N}) = \|Px_{k+N}\|_p \,, \tag{4.25}$$

with p denoting the polyhedral norm, and the real, full-column rank matrices Q, R and P denoting weight matrices. Although not linear, these cost functions are sometimes referred to as linear cost functions. The motivation for piecewise linear cost functions comes from economics, where the sum of absolute costs should be minimized (1-norm), or from worst-case minimizations (∞-norm). The main benefits of piecewise linear cost functions are of computational nature. If the system and constraints are linear, the CFTOC problem becomes a linear program, which can be solved easily with an LP solver. Moreover, the epigraph of such an optimization problem is a polyhedron which can be crucial for the application of dynamic programming. A price has to be paid for these benefits, the optimizer of an LP is not necessarily unique, and the tuning is more difficult, since the optimizer 'jumps' spontaneously from one vertex to the next. The deviations from the origin are penalized proportionally to their distance, which, compared to quadratic cost functions, leads to potentially less control action far from the origin, and more action close to the origin.

Reference Tracking

In most parts of this thesis we are dealing with the *regulator problem*, i.e. it is implicitly assumed that the control objective is to stabilize the system towards the origin. Often this is a realistic assumption, since we can shift any desired operating point $\bar{x}$ to the origin by working with state variables relative to this operating point $\tilde{x} = x - \bar{x}$. In the context of varying operating points and state constraints however, we might be interested in a *reference tracking* MPC scheme, which can be used for a set of reference points, [MBM09]. In this situation the reference point can be added as an additional parameter to the optimization problem, and the cost function is written as

$$L_i(x_{k+i}, u_{k+i}) = \|Q(x_{k+i} - x_{\text{ref},k+i})\|_p + \|R\Delta u_{k+i}\|_p \,, \tag{4.26}$$
$$L_N(x_{k+N}) = \|P(x_{k+N} - x_{\text{ref},k+N})\|_p \,, \tag{4.27}$$

where $x_{\text{ref},k+i}$ denotes the desired reference point and $\Delta u_{k+i} := u_{k+i} - u_{k+i-1}$ is the input rate. By penalizing the input rate instead of the inputs, the steady-state input required to track the reference points is not penalized, rather the variations in the input signal mitigating chattering. This is also known as *Δu-formulation*. Note that the number of parameters in the CFTOC problem increases by the number of reference variables and the number of inputs, increasing the complexity of the resulting explicit control law. An alternative to the Δu-formulation is to penalize the difference of the input to the steady-state input required to keep the system at the desired reference point.

Soft Constraints

The objective of the state constraints $x_{k+i} \in \mathcal{X}$ (also called hard constraints) in the CFTOC problem is to guarantee the future states to be members of a certain set. However, there are applications where uncertainties such as disturbances, measurement errors and/or model inaccuracies can render the constraint satisfaction impossible, leading to infeasibility of the CFTOC problem. Additionally to not being able to meet the state constraints, the control input can not be computed. One way to avoid this situation is the use of *soft constraints*, [Mac01]. The state constraints are replaced by

$$x_{k+i} + s_i \in \mathcal{X} \,, \tag{4.28}$$

with $s_i \in \mathbb{R}^{n_x}$ denoting a *slack variable*. The slack variables are penalized in the cost function (4.21), using a high weight. If designed properly, soft constraints operate exactly as hard constraints as long as the constraints can be met. If constraint satisfaction becomes impossible however, the CFTOC problem does not become infeasible, permitting the computation of control inputs. More details on soft constraints can be found in [KM00].

Move Blocking

Move blocking is an extension to Model Predictive Control, where the optimization variables are kept constant over several prediction steps. The benefit of move blocking is the reduction of optimization variables, leading to a simpler optimization problem at each time instance and thus to a faster computation of the control input. The drawback is that by fixing these optimization variables the number of degrees of freedom are reduced, which may result in a smaller feasible set, a higher cost function and even to unstable behaviour of the closed-loop system. More details about move blocking can be found e.g. in [CGKM07]. The most common move blocking strategy involves the definition of a *control horizon* N_c, which separates the sequence of control inputs

$$\boldsymbol{U}_k = \{u_k, u_{k+1}, \dots, u_{k+N-1}\}$$

into two parts. The control inputs within the control horizon $u_k, \dots, u_{k+N_c-1}$ can vary freely, while the remaining control actions are set to a common value, $u_{k+Nc} = \cdots = u_{k+N-1}$.

Stability and Feasibility

Unfortunately, MPC is not guaranteed to stabilize a dynamic system *per se*. Even worse, the dependence of stability on the tuning parameters N, Q, R, P and $\mathcal{X}_T$ can be not obvious, as was pointed out in [Löf03]. Additionally, MPC does not provide a guarantee for recursive feasibility.

Definition 4.1 (Recursive feasibility) *An MPC problem is called* feasible*, if for the current state x_k the underlying optimal control problem is feasible. An MPC controller is called* recursive feasible*, if a feasible MPC problem for the current state x_k implies feasible MPC problems for all future states x_{k+i}, $i \in \mathbb{N}$.*

There exists a range of possible remedies which can be applied to provide guarantees for stability and recursive feasibility. An overview about these methods is presented in [MRRS00]. Extensions to the case of non-continuous systems can be found e.g. in [Laz06]. Here we will just state the main ideas of the most common approaches.

One possibility is to use an *infinite prediction horizon*, $N \to \infty$. The optimal cost function

$$J^*(\boldsymbol{U}_k^*; x_k) = \sum_{i=0}^{\infty} L(x_{k+i}, u_{k+i}^*) \tag{4.29}$$

can be employed as a Lyapunov function, under the assumptions that the stage cost L is a continuous, positive definite function, and the dynamic system is a continuous, definite function. Furthermore, the sets $\mathcal{X}$ and $\mathcal{U}$ are assumed to be C-sets. Considering an infinite-horizon optimal control problem is desirable but often not possible, because of the complexity of the infinite-horizon problem and the limited computational power. If instead of $N = \infty$ the prediction horizon is selected large enough, stability and recursive feasibility are also guaranteed, [MRRS00].

A second possibility is the *terminal equality constraint*,

$$x_{k+N} = 0 \,. \tag{4.30}$$

With this additional constraint, the infinite-horizon cost (4.29) reduces to finite horizon, while it can still serve as a Lyapunov function. On the other hand the terminal equality constraint imposes a strong restriction on the CFTOC problem, leading to a considerably smaller feasible set and to a deterioration of control performance.

In order to mitigate the restrictions of the terminal equality constraint one can relax them to a *terminal inequality constraint*, $x_{k+N} \in \mathcal{X}_T$ with the C-set $\mathcal{X}_T$ denoting the terminal region, and apply a terminal region controller $\mu_T(x_k)$ if $x_k \in \mathcal{X}_T$. The resulting infinite-horizon cost (4.29) comprise the finite-horizon costs and a terminal cost term $L_N(x_k)$ for the costs under the terminal region controller $\mu_T(x_k)$. Under the assumptions

(*i*) $\mathcal{X}_T \subseteq \mathcal{X}$,

(*ii*) $\mu_T(x_k) \in \mathcal{U} \quad \forall x_k \in \mathcal{X}_T$,

(*iii*) $x_k \in \mathcal{X}_T \Rightarrow f(x_k, \mu_T(x_k)) \in \mathcal{X}_T$,

(*iv*) $L_N(f(x_k, \mu_T(x_k))) - L_N(x_k) \leq -L(x_k, \mu_T(x_k)) \quad \forall x_k \in \mathcal{X}_T$,

the resulting MPC controller is guaranteed to be stable and recursive feasible. Note that this approach, though to a minor extent than the terminal equality constraint, still comes with some conservatism. The conditions are merely sufficient.

Finally, one could analyze the stability and recursive feasibility *a-posteriori*. The benefit is that no adjustments of the CFTOC problem have to be performed which could result in a poor approximation of the infinite-horizon cost. On the other hand, a-posteriori analyses are difficult as long as the control law is not given in an analytical form. Hence this approach is mainly restricted to the case when the control law $u_k^* = \mu(x_k)$ is known explicitly.

Explicit Model Predictive Control

One important aspect of the receding horizon strategy is that only the current control action u_k^* is ever applied to the system, whereas the predicted control actions $u_{k+1}^*, \dots, u_{k+N-1}^*$ are only needed for the predictions itself.

The online solution of the CFTOC problem can therefore be seen as an evaluation of the *implicit* relation

$$u_k = \mu_k^*(x_k)\,. \tag{4.31}$$

By solving the optimization problem parametrically, we obtain this control law *explicitly*. The precomputed control laws are piecewise affine functions of the state and can be stored in a look-up table. This allows us to avoid the solution of an optimization problem at each time step, but to apply the receding horizon strategy by evaluating the look-up table instead. The application of parametric programming to solve constrained optimal control problems explicitly was initialized around the millennium, and is referred to as *Explicit MPC*, [BM99, PDB$^+$00, BBM02]. Explicit MPC is based on parametric programming, which allows to solve linear and quadratic programs over a range of parameters (see Section 3.1). The receding horizon strategy has the neat side-effect that for explicit MPC only the current control law needs to be stored and not the whole finite-horizon sequence of control laws. Efficient algorithms for the computation of explicit solutions to the CFTOC problem were developed for linear systems, [BBM00], for hybrid systems, [Bor03] and for uncertain linear systems, [BBM03].

Definition 4.2 (Region of explicit MPC controller) *A* region $\mathcal{R}$ *of an explicit MPC control law is a full-dimensional, closed and convex set of the state space, where the control law is a common, typically affine, function of the state.*

Usually the regions of an explicit MPC controller emerge from the closure of critical regions of the underlying parametric program, see Definition 3.2 on page 32. Note however that there is no one-to-one relation between controller regions and critical regions. Controller regions are full-dimensional, while critical regions can also be lower-dimensional. A region of the controller, as closure of a full-dimensional critical region, can also contain other, lower-dimensional critical regions on the boundary. Moreover, multiple critical regions may be merged to a single controller region, if they possess the same control law.

Definition 4.3 (Partition of explicit MPC controller) *A* partition $\mathcal{P} = \{\mathcal{R}_i\}_{i=1}^{n_r}$ *of an explicit MPC control law is a collection of n_r controller regions $\mathcal{R}_i \subset \mathbb{R}^{n_x}$, stemming from an underlying optimization problem.*

Definition 4.4 (Complexity of explicit MPC controller) *The term* complexity *usually refers to the number of regions of an explicit control law. Sometimes complexity is meant in a numerical way, i.e. the required number of operations to evaluate the explicit control law.*

Computing the explicit control law inherits several advantages. Replacing the solution of an optimization problem online by an evaluation of an explicit function can reduce the online numerical effort significantly. Hence it is possible to apply MPC to systems with a higher sampling rate, even in the range of microseconds, [BMC$^+$09, GPM09]. The computational equipment required to execute an MPC controller is reduced, the function evaluation can often be performed on a microprocessor, making an application of explicit MPC inexpensive compared to a traditional MPC scheme. The evaluation of the explicit control law can be performed quickly and reliable, and it is easier to derive realistic worst-case bounds for the online computation times. Moreover, the explicit form of the control law allows for an analysis of stability and performance of the closed-loop system, and to perform post-processing algorithms as reduction of the number of regions or an approximation of the optimal solution.

On the other hand, the computation of explicit control laws also exhibits drawbacks. The flexibility of MPC is partially lost, since explicit control laws can only be computed for systems of a certain type and size. The weight matrices can not be adapted online, but require a recomputation of the explicit control law. The gain in online computational time depends on the complexity of the control law. If the complexity of the resulting control law is too high, the function evaluation might be as time consuming as the solution of the optimization problem in the first place. Since the complexity of the control laws grows starkly (in the worst-case exponentially) with the number of parameters in the underlying parametric program, explicit MPC is mainly limited to problems of modest size.

Part II

Piecewise Affine Systems

5 Optimal Control of PWA Systems

'Among other advantages, it should be noted that the use of piecewise linear systems permits introducing thresholds and other discontinuities in a natural way that is not available in other algebraic approaches to nonlinear system theory.'

Eduardo Sontag

PIECEWISE AFFINE SYSTEMS are a system class commonly appearing in practical problems. In this chapter we briefly recapitulate constrained finite-time optimal control of piecewise affine systems based on a quadratic cost function. The explicit solution of the resulting mixed-integer quadratic program by means of parametric programming is considered. Properties of the resulting optimal state-feedback solution are stated, and an established solution approach is presented.

5.1 Introduction

During the last years a lot of effort has been spent on the development of control and analysis methodologies for hybrid systems. *Hybrid systems* are systems which contain continuous as well as discrete states, and are used to model nonlinear effects as saturation, switches and other logic elements, [Son81, BM99]. The dominating technique behind control of hybrid systems is Model Predictive Control (MPC), and a vast variety of controller synthesis methods are based on it, [ML99].

Around the millennium, the application of parametric programming to constrained optimal control problems, so-called *explicit MPC*, was initiated. Parametric programming allows one to solve the optimal control problems explicitly and moves the computational effort of MPC from online to offline. The pre-computed control laws are stored in a look-up table, which thereafter is evaluated online. The mathematical backgrounds of parametric programming were developed already in the 1970s and can be found in [BGK$^+$82]. Until today, several procedures were proposed which enable the employment of parametric programming for MPC. An introduction to explicit MPC for constrained linear and hybrid systems and further references are given e.g. in [Bor03, Bao05].

Model Predictive Control contains the minimization of a cost function to determine the optimal control inputs. A natural choice for this cost function is a quadratic function of the predicted states and inputs, reflecting a generalized energy concept in the control problem. Quadratic costs are chosen frequently by control engineers not only in MPC but also in Linear Quadratic Control, because they result in smooth control behaviour close to the reference values, and large control action far away from it. In the context of explicit MPC, the use of quadratic cost functions usually results in a smaller number of regions compared to polyhedral norm objectives, and thus to less complex control laws. In some cases, quadratic cost functions allow the use of a smaller prediction horizon and thus result in even less complex control laws. The computation of quadratic cost optimal control laws with dynamic programming can be found in [BBBM05].

5.2 Quadratic Cost Optimal Control of PWA Systems

A large class of hybrid systems can be described as *piecewise affine* (PWA) *systems*, [HdSB01]. Piecewise affine systems are defined by associating different dynamics with polyhedral regions in the state-input space,

$$\begin{aligned} x_{k+1} &= f_{PWA}(x_k, u_k) \\ &:= A_j x_k + B_j u_k + f_j \quad \text{if} \quad [x_k^T \;\; u_k^T]^T \in \mathcal{D}_j \,, \end{aligned} \tag{5.1}$$

with the discrete time $k \in \mathbb{N}$, the state $x_k \in \mathbb{R}^{n_x}$, the input $u_k \in \mathbb{R}^{n_u}$, and n_D polyhedral regions

$$\mathcal{D}_j := \left\{ \begin{bmatrix} x \\ u \end{bmatrix} \in \mathbb{R}^{n_x+n_u} \;\middle|\; P_{x,j}x + P_{u,j}u \le p_j \right\}$$

which assemble to a bounded polyhedral partition of the domain $\mathcal{D} = \bigcup_{j=1}^{n_D} \mathcal{D}_j$ in the state-input space. $A_j, B_j, P_{x,j}, P_{u,j}$ and f_j, p_j are real matrices and real vectors of appropriate dimensions, respectively. The polyhedral regions $\mathcal{D}_j$, $j = 1, \ldots, n_D$, define both regions in which a particular state update equation is valid as well as constraints on the state and the input. A PWA system is called *well-posed* if f_{PWA} is a single-valued function.

Under some technical assumptions, the PWA system representation is *equivalent* to several other models of hybrid systems and it was shown in [HdSB01] how one can convert one form into the other. Among others, the PWA description of a hybrid system is equivalent to the *mixed logical dynamical* (MLD) *system*, [MBB03, Bem04],

$$x_{k+1} = Ax_k + B_1u_k + B_2\delta_k + B_3z_k \tag{5.2a}$$
$$y_k = Cx_k + D_1u_k + D_2\delta_k + D_3z_k \tag{5.2b}$$
$$E_2\delta_k + E_3z_k \le E_1u_k + E_4x_k + e_5 \tag{5.2c}$$

with the state $x_k = [x_{k,c}^T \;\; x_{k,b}^T]^T \in \mathbb{R}^{n_{xc}} \times \{0,1\}^{n_{xb}}$, the input $u_k = [u_{k,c}^T \;\; u_{k,b}^T]^T \in \mathbb{R}^{n_{uc}} \times \{0,1\}^{n_{ub}}$ and the output $y_k = [y_{k,c}^T \;\; y_{k,b}^T]^T \in \mathbb{R}^{n_{yc}} \times \{0,1\}^{n_{yb}}$ comprising continuous and binary elements. $z_k \in \mathbb{R}^{n_z}$ and $\delta_k \in \{0,1\}^{n_\delta}$ denote continuous and binary auxiliary variables, and A, B_i, C, D_i, E_i and e_5 denote real matrices and a real vector of appropriate dimensions, respectively. An MLD system is called *well-posed* if δ_k and z_k are uniquely determined by (5.2c) for given x_k, u_k, implying that also x_{k+1} and y_k are uniquely determined. MLD systems comprise the continuous evolution of dynamical systems as well as discrete behaviour of automata. The equivalence of PWA and MLD system simplifies the implementation in a Matlab environment using the hybrid system description language HYSDEL, [TBB$^+$02].

In quadratic cost optimal control, a cost function is defined as a quadratic function of the states and inputs within the prediction horizon N,

$$J(\boldsymbol{U}_k; x_k) = x_{k+N}^T P x_{k+N} + \sum_{i=0}^{N-1} x_{k+i}^T Q x_{k+i} + u_{k+i}^T R u_{k+i} \,, \tag{5.3}$$

where $\boldsymbol{U}_k = \{u_k, \dots, u_{k+N-1}\}$ denotes the sequence of control actions, and the weight matrices $P, Q \in \mathbb{S}_+^{n_x}$ and $R \in \mathbb{S}_{++}^{n_u}$ are assumed to be positive (semi)definite (compare Section 4.5 on MPC with a quadratic cost function). Extensions to incorporate the tracking of a reference state x_{ref} are straightforward, but in the following we will restrict ourselves to the regulation problem for notational simplicity.

Consider the following *constrained finite-time optimal control* (CFTOC) problem for piecewise affine systems:

$$J^*(x_k) = \min_{\boldsymbol{U}_k} \quad J(\boldsymbol{U}_k; x_k) \tag{5.4a}$$

$$\text{s.t.} \quad x_{k+i+1} = f_{PWA}(x_{k+i}, u_{k+i}), \quad i = 0, \dots, N-1\,, \tag{5.4b}$$

$$x_{k+N} \in \mathcal{X}_T\,. \tag{5.4c}$$

Note that state and input constraints are included in the description of the PWA system (5.4b). The set $\mathcal{X}_T$ is a compact polyhedral *terminal set*, reflecting the set of admissible states at the final (finite) time instance N.

A classical implementation of MPC uses *online* optimization with a receding horizon policy, to compute the *optimal input sequence* $\boldsymbol{U}_k^*$, [Mac01]. The CFTOC problem (5.4) is solved at each time instance k for the current state x_k which amounts to solving a *mixed-integer quadratic program* (MIQP) as discussed in Section 3.1 on mixed-integer programming. From the resulting control sequence $\boldsymbol{U}_k^*$, only u_k^* is used as control input at time k, and the subsequent control inputs are discarded. In the next step, the prediction horizon is shifted and the procedure repeated. Due to the computational requirements, this approach becomes intractable for control systems requiring a high sampling rate.

Properties of the CFTOC Solution

A different approach is to solve the optimization problem (5.4) parametrically. The advantage of this approach is that the access to the optimal input sequence can be speed-up significantly. As shown in [BBBM05], the CFTOC problem (5.4) for PWA systems can be solved explicitly *off-line* using parametric programming (compare Section 3.2). The generated explicit solution is a look-up table whose structure is defined by the properties of the solution to the problem (5.4). Those are summarized by the following theorem, [Bor03]:

Theorem 5.1 (CFTOC solution for PWA systems) *The solution to the optimal control problem* (5.4) *with the quadratic cost function* (5.3) *is a PWA state feedback control law:*

$$u_k^*(x_k) = F_r x_k + g_r \quad \textit{if} \quad x_k \in \mathcal{R}_r \tag{5.5}$$

with $F_r \in \mathbb{R}^{n_u \times n_x}, g_r \in \mathbb{R}^{n_u},\ r = 1, \dots, n_r$*, and where the controller regions* $\mathcal{R}_r$*, whose closure is given by:*

$$\bar{\mathcal{R}}_r = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; x^T H_i x + q_i x \le r_i\,, \quad i = 1, \dots, n_c \right\}\,,$$

define a partition of the set of feasible states.

Once the explicit solution is computed, the on-line operation of the controller consists of the identification of the pre-computed affine feedback control law for a measured (estimated) state x_k by searching for a region $\mathcal{R}_i$ containing the state vector x_k. In this way, much of the computational effort is moved off-line, enabling optimal control of hybrid systems under much tighter time constraints.

Solution Approach to the CFTOC Problem

Solving the CFTOC problem beforehand for all possible initial states $x_k \in \mathcal{X}$, as discussed in Section 4.1, requires the solution of a *parametric mixed-integer quadratic program* (pMIQP), see Section 3.2. As stated in Theorem 5.1, the explicit solution is not defined on polytopes. One way to deal with the pMIQP related to the CFTOC problem (5.4) is to decompose it into parametric quadratic programs (pQPs) by fixing the modes m_i of the PWA system (5.1) in each prediction step i and to enforce a specific *switching sequence* $s = (m_0, \dots, m_i, \dots, m_{N-1})$. For each switching sequence a pQP of the form

$$\begin{aligned}
J^*(x_k) = \min_{\boldsymbol{U}_k} \quad & J(\boldsymbol{U}_k; x_k) && \text{(5.6a)}\\
\text{s.t.} \quad & x_{k+i+1} = f_{PWA}(x_{k+i}, u_{k+i})\,, \quad i = 0, \dots, N-1\,, && \text{(5.6b)}\\
& [x_{k+i}^T \ \ u_{k+i}^T]^T \in \mathcal{D}_{m_i}\,, \quad i = 0, \dots, N-1\,, && \text{(5.6c)}\\
& x_{k+N} \in \mathcal{X}_T\,. && \text{(5.6d)}
\end{aligned}$$

results. The number of possible switching sequences is $n_s = (n_D)^N$. For some switching sequences the corresponding pQPs might be infeasible due to the state-input constraints in every step. Following Theorem 3.2 on page 34, we can state: Each of the $n_p \le n_s$ remaining pQPs leads to a polyhedral partition $\mathcal{P}^p \subseteq \mathbb{R}^{n_x}$, $p = 1, \dots, n_p$, in the state space consisting of n_r^p polyhedral regions $\mathcal{R}_r^p \subseteq \mathbb{R}^{n_x}$, $r = 1, \dots, n_r^p$, and is associated with a PWA control law

$$u_k = \mu_r^p(x_k) = F_r^p x_k + g_r^p \quad \text{if} \quad x_k \in \mathcal{R}_r^p \qquad \text{(5.7)}$$

and with a piecewise quadratic (PWQ) cost function $J^p\colon \mathbb{R}^{n_x} \to \mathbb{R}$, which is a quadratic function in each region,

$$J_r^p(x_k) = x_k^T A_r^p x_k + (b_r^p)^T x_k + c_r^p \quad \text{if} \quad x_k \in \mathcal{R}_r^p\,. \tag{5.8}$$

Every cost function J^p is optimal for the related switching sequence; the remaining task is the determination of the switching sequence which yields the lowest cost, such that

$$J^*(x_k) = \min_p \; J^p(x_k) \tag{5.9}$$

is the solution to the CFTOC problem (5.4). Therefore it is suggested in [Bor03] to solve (5.6) parametrically offline for all possible switching sequences and to store the resulting cost functions $J^p,\ p = 1,\ldots,n_p$, as well as the resulting control laws $\mu^p,\ p = 1,\ldots,n_p$. Online at each time instance k, the control input u_k is determined by solving (5.9) for the current state x_k. This procedure is implemented in the *Multi-Parametric Toolbox* (MPT) 2.6.2 , [KGBM04].

6 Region Removal for Optimal Control of PWA Systems

'The ability to simplify means to eliminate the unnecessary so that the necessary may speak.'

Hans Hofmann

REDUNDANT REGIONS in the look-up table of an explicit MPC controller increase the required storage space and retard the efficient evaluation online. Consequently, the control law should contain little or no redundant regions. This chapter describes a post-processing algorithm for quadratic cost explicit model predictive controllers of piecewise affine systems. Redundant regions of explicit control laws are identified and removed. The algorithm reduces the complexity of the control laws without sacrificing optimality of the control performance. The resulting control laws need significantly less storage space and hence require also less online computational effort. The algorithm described in this chapter was implemented in MPT.

6.1 Introduction

The computation of optimal control laws for piecewise affine systems is described in Section 5.2. If a polyhedral norm is used in the cost function, it can be shown that the optimal control law is a piecewise affine function defined over polytopes. If instead a quadratic cost function is employed, the optimization problem to be solved is a pMIQP of the form (3.14), and the optimal control law is piecewise affine, but unfortunately not over polytopes. Therefore it is suggested in [Bor03] to determine the cost functions $J^p,\ p = 1,\dots,n_p$, and the corresponding control laws $\mu^p,\ p = 1,\dots,n_p$, for all possible switching sequences by solving the related pQP. Online the control action is determined by comparing all values $J^p(x_k)$ for the current state x_k. The obvious drawback is that instead of one, all n_p partitions with the total number of regions n_r together with the associated cost functions and the associated control laws have to be stored. However, many of the stored regions might be suboptimal and as such they will never be used while the online computational effort is increased due to a larger number of regions and due to the comparisons of cost functions. In order to lower the needed storage space and the number of online comparisons, we propose an algorithm for the detection and removal of unnecessary regions.

6.2 Removal of Redundant Regions

The basic task of the post-processing algorithm described in this chapter is to determine if a region $\mathcal{R}_r^p \subset \mathbb{R}^{n_x}$ is redundant and hence can be removed.

Definition 6.1 *We call a region $\mathcal{R}_r^p \subset \mathbb{R}^{n_x}$* dominated *by a finite set of controller regions $\{\mathcal{R}_i\}_{i=1}^n \subset \mathbb{R}^{n_x}$, iff*

(i) $\mathcal{R}_r^p \subseteq \{\mathcal{R}_i\}_{i=1}^n$

(ii) $\forall x \in \mathcal{R}_r^p, \exists\, i \in \{1,\dots,n\} : x \in \mathcal{R}_i \wedge J_i(x) \le J_r^p(x)$

A dominated region is also called redundant.

Definition 6.2 $\mathcal{R}_r^p \subset \mathbb{R}^{n_x}$ *is* partially dominated *by a finite set of controller regions $\{\mathcal{R}_i\}_{i=1}^n \subset \mathbb{R}^{n_x}$, iff there exists a nonempty common domain $\Delta\mathcal{R} := \mathcal{R}_r^p \cap \{\mathcal{R}_i\}_{i=1}^n \neq \emptyset$ such that*

(i) $\Delta\mathcal{R} \subset \mathcal{R}_r^p$

(ii) $\forall x \in \Delta\mathcal{R}, \exists i \in \{1,\dots,n\} : x \in \mathcal{R}_i \wedge J_i(x) \le J_r^p(x)$

In order to determine redundant regions, comparisons of the quadratic cost functions have to be executed which is equivalent to checking if the difference of two quadratic functions,

$$\begin{aligned}\Delta J =& J_r^p - J_{\hat{r}}^{\hat{p}} \\ =& x^T(A_r^p - A_{\hat{r}}^{\hat{p}})x + (b_r^p - b_{\hat{r}}^{\hat{p}})^T x + (c_r^p - c_{\hat{r}}^{\hat{p}}) \\ =& x^T \Delta A x + \Delta b^T x + \Delta c\,, \end{aligned} \tag{6.1}$$

and thus its minimum, is non-negative on the common domain. Note that the difference of two quadratic functions is not necessarily convex. The minimization of nonconvex quadratic functions over a polytope is known to be NP-hard, [PV91]. In order to determine the minimal value of the cost difference ΔJ, a minimization of a possibly nonconvex function over a polytope has to be performed. What makes this procedure computationally demanding is the fact that the number of these comparisons grows quadratically with the total number of regions of the handled control law.

In this chapter we follow an approach which involves several preprocessing steps, in order to prevent many unnecessary nonconvex minimizations and thus to reduce the required computation time. The remaining minimizations are performed using either a general branch-and-bound scheme, or an enumeration of the facets.

Algorithm 6.1 Region removal: Main algorithm

```
Input: $\mathcal{P}^p,\ p = 1,\dots,n_p$
Output: $\mathcal{P}^p,\ p = 1,\dots,n_p$
1: Initialization
2: for $i_{part} = 1 : n_p$ do
3:    for $j_{part} = i_{part} + 1 : n_p$ do
4:       Pre-select candidates
5:       Verify candidates
6:    end for
7: end for
8: Removal of redundant regions
```

The main algorithm for the removal of regions is presented as Algorithm 6.1. This algorithm consists of four phases. While the initial phase is performed once in the beginning and the actual removal of regions once in the very end of the algorithm, the pre-selection of candidates and the verification of these candidates are repeated for every possible combination of partitions, $\sum_{p=1}^{n_p}(p-1)$ times.

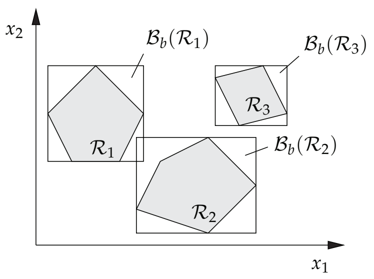

x_2
$\mathcal{B}_b(\mathcal{R}_1)$
$\mathcal{B}_b(\mathcal{R}_3)$
$\mathcal{R}_3$
$\mathcal{B}_b(\mathcal{R}_2)$
$\mathcal{R}_1$
$\mathcal{R}_2$
x_1

Figure 6.1: Three regions $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$ and their corresponding bounding boxes $\mathcal{B}_b(\mathcal{R}_1), \mathcal{B}_b(\mathcal{R}_2), \mathcal{B}_b(\mathcal{R}_3)$.

Initialization

In the initialization phase some technical tasks as pre-allocation of variables are executed. Then for each partition and each region, a *bounding box* is computed.[1] The computation of the bounding box of a polytope in $\mathbb{R}^{n_x}$ requires the computation of $2n_x$ Linear Programs (LPs), so the total number of LPs in the initialization is $2n_x(n_r + n_p)$. The bounding boxes are stored for later use.

Pre-select Candidates

In this phase of the algorithm, we have two partitions at hand, and want to determine candidate regions which potentially are dominated by regions of the other partition. Before the cost functions of two regions are compared by performing a possibly nonconvex minimization, a number of simple pre-processing steps are executed to decrease the number of required minimizations.

Bit by bit, each region of both partitions at hand is compared to each region of the opposite partition. First it is determined if the two regions at hand intersect. A necessary condition for the intersection of two regions is the intersection of their bounding boxes, which can easily be determined by value comparisons. This pre-test is visualized in Figure 6.1, where three regions and their corresponding bounding boxes are displayed. From the bounding boxes it is obvious that region $\mathcal{R}_3$ is intersecting neither region $\mathcal{R}_1$ nor region $\mathcal{R}_2$. However, if $\mathcal{R}_1$ and $\mathcal{R}_2$ intersect can not be decided from the bounding boxes. The bounding boxes facilitate an easy way to exclude many intersections between regions.

[1] In the default preferences of the Multi-Parametric Toolbox (MPT), the bounding box is already stored in the polytope object and this step can be skipped.

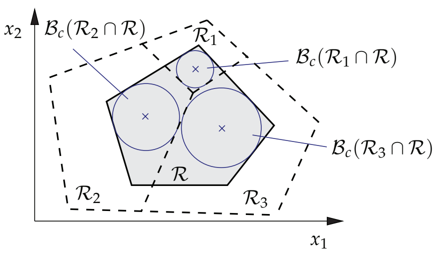

x_2
$\mathcal{B}_c(\mathcal{R}_2 \cap \mathcal{R})$
$\mathcal{R}_1$
$\mathcal{B}_c(\mathcal{R}_1 \cap \mathcal{R})$
$\mathcal{B}_c(\mathcal{R}_3 \cap \mathcal{R})$
$\mathcal{R}$
$\mathcal{R}_2$
$\mathcal{R}_3$
x_1

Figure 6.2: Intersection of a region $\mathcal{R}$ with three other regions $\mathcal{R}_i$ and their intersection Chebyshev balls $\mathcal{B}_c(\mathcal{R}_i \cap \mathcal{R})$.

If the bounding boxes of both regions intersect, the intersection of both regions is verified by computing the radius r_c of the Chebyshev ball, i.e. the largest ball contained in both regions. This computation requires only one LP per tested intersection. If the LP finishes with a positive radius r_c, the regions indeed intersect. An example for the intersection checking with Chebyshev balls is presented in Figure 6.2, where the intersection of a region $\mathcal{R}$ with three other regions (indicated by dashed lines) is visualized. Note also the Chebyshev balls with their Chebyshev centres.

While determining the intersections between all regions of the partitions at hand, possible candidates for removal are identified. In order to determine these candidates, the cost functions are evaluated in the Chebyshev centre x_c of all intersections. The Chebyshev centre is a by-product when computing the Chebyshev ball, and the evaluation of the cost functions in the Chebyshev centre is a cheap operation. This heuristic pre-test helps avoiding cost comparisons and thus saves computation time. Each region whose cost function in the Chebyshev centre is greater than the cost function of all intersecting regions, is a candidate for removal. If the cost function is lower in any of the intersection Chebyshev centres, the region is not dominated by the other partition, and does not have to be considered further.

Remark 6.1 *Regions, which fail in one of these Chebyshev tests, are not considered further, since this would require significantly more computational effort. The number of candidates would be increased significantly, and thus retard the execution of the proposed algorithm.*

In the end of this phase of the algorithm, we have for both partitions a list of candidate regions which potentially are redundant and have to be investigated further. Moreover, for each candidate we are in possession of a list with regions of the other partition, which intersect the candidate region.

Verify Candidates

In this phase of the algorithm, we are considering two partitions. For each partition we are in possession of a list of candidates, and for each candidate a list of intersecting regions of the opposite partition. All candidates are compared to their previously determined list of intersecting regions, and for each comparison, a possibly nonconvex minimization is carried out.

The algorithm offers two different methods, one for problems of lower dimensions and one suited for medium-size problems: When the low complexity option was chosen, all nonconvex comparisons are done by facet enumeration, otherwise by branch-and-bound.

Algorithm 6.2 Region removal: Facet enumeration

```
Input: $\Delta A, \Delta b, \Delta c, \mathcal{R}$
Output: partial dominance
 1: Check definiteness of $M$
 2: if positive definite then
 3:     return
 4: else
 5:     Check eigenvalues of $\Delta A$
 6:     if convex then
 7:         solve QP
 8:     else if concave then
 9:         Enumerate vertices
10:     else
11:         Enumerate facets
12:     end if
13: end if
```

Algorithm 6.2 represents the algorithm behind *facet enumeration*. Before starting with the actual cost comparison, a last pre-check is executed. Equation (6.1) can be rewritten as

$$\Delta J = \begin{pmatrix} x \\ 1 \end{pmatrix}^T \begin{bmatrix} \Delta A & \Delta b/2 \\ \Delta b^T/2 & \Delta c \end{bmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ 1 \end{pmatrix}^T M \begin{pmatrix} x \\ 1 \end{pmatrix} . \tag{6.2}$$

If the matrix M is positive definite, the cost difference function ΔJ is positive over the whole $\mathbb{R}^{n_x}$, and thus also over the intersection of the regions at hand. Hence the considered candidate is dominated in this intersection, and the next intersection is investigated.

Otherwise, the minimum of ΔJ over the intersection of both regions is determined. Therefore the convexity of the cost difference function ΔJ is computed by determining the eigenvalues of ΔA. If the difference function is convex, a quadratic program (QP) is solved to find the minimum. If the difference function ΔJ is concave, the minimal value is obtained at one of the vertices, and a vertex enumeration is performed.

In the case of an indefinite difference function, the minimum is attained on the boundary of the intersection polytope, but not necessarily at one of the vertices. Then the minimum on each face is computed separately. For all faces of the intersection polytope, the facet enumeration is performed by executing Algorithm 6.2. Compared to the intersection polytope, the dimension of the minimization problem on its faces is decreased by one, since one inequality is set to equality to describe the face.

The facet enumeration algorithm is applied recursively to all faces, then to all ridges and so on, until the difference functions become convex or concave on the facets. To avoid multiple checking of the same facets, a *directed facet enumeration* is performed, which means that all inequalities are numbered beforehand, and inequalities are only equalized in an ascending order. If on one facet the difference function obtains a negative value, the comparison is aborted for this region and the considered region is not dominated by the opposite partition. Otherwise the region is dominated by its counterpart, and the next intersection can be investigated.

For problems of higher dimensions, procedures as vertex enumeration and facet enumeration become more and more inefficient. Therefore another option is offered in the algorithm, which is based on branch-and-bound. Here also the eigenvalues are checked first to treat convex problems faster in a QP. If the cost difference function is not convex, a branch-and-bound algorithm, which is part of YALMIP, is used, [Löf04]. Nonconvex minimization based on branch-and-bound is explained e.g. in [Sahoo].

If a candidate region is identified as dominated by all intersecting regions of a partition, it is checked by vertex enumeration (or by polytope comparison for higher-dimensional problems) if the candidate region is a subset of the opposing partition. In this case the region is marked as redundant, otherwise the region is marked as *partially dominated*.

Remark 6.2 *Facet enumeration and directed facet enumeration can both be seen as specific branch-and-bound schemes with the whole polytope as root node and the vertices and local minima as leaves.*

Removal of Regions

The last part of the algorithm is performed only once after the completion of all comparisons. Regions which were marked as partially redundant, are checked again. All parts which are redundant, are removed and the convex hull of the remaining parts is saved. Finally the redundant regions are removed.

Remark 6.3 *This reduction of partially dominated regions is a computationally expensive operation, and thus only performed optionally.*

Remark 6.4 *The proposed algorithm is only removing regions which are dominated by a single partition. In fact there may also appear redundant regions which are not dominated by a single partition, but by a set of regions of different partitions. In order to remove also these regions, the computational effort would grow significantly. Many more nonconvex minimizations would have to be carried out (in fact for all intersections), in order to determine a set of regions, which dominate the region at hand.*

6.3 Numerical Example

In this section we will investigate the application of the proposed algorithms. In the first example, we compare the solution times of indefinite QPs by means of branch-and-bound and by means of directed facet enumeration. In the second example we apply the region removal algorithm to hybrid MPC controllers of different prediction horizons.

Example 6.1 (Facet Enumeration vs. Branch-And-Bound)

In the first example the computation times of directed facet enumeration (Algorithm 6.2) are compared to the computation times of Yalmip's internal branch-and-bound solver. Both methods were tested in the task to determine the positivity of the minima of random indefinite quadratic functions over random simplices. The dimension of the simplices was increased from $n_x = 2$ up to $n_x = 5$, while for each dimension 1000 random samples were investigated.

The total computation times for 1000 samples of each tested dimension and for both methods are depicted in Figure 6.3. While the use of directed facet enumeration is computationally beneficial for problems of dimension $n_x = 2$ and $n_x = 3$, for larger problems the branch-and-bound scheme is faster. In all tested instances both methods delivered the same results concerning the positivity of the quadratic function on the simplex.

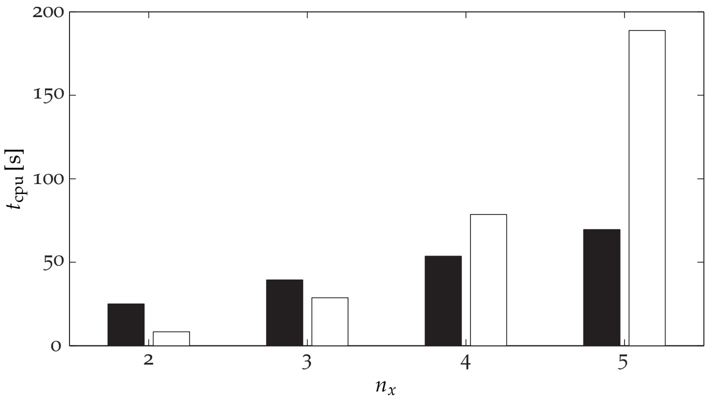


Figure 6.3: Total computation times for solving 1000 random indefinite QPs vs. the state dimension n_x with branch-and-bound (black bars) and directed facet enumeration (white bars).

Example 6.2 (Region Removal for the Sine-Cosine System)

In order to indicate potentially achievable complexity reductions, we examine the application of the proposed region removal algorithm to optimal control laws for the constrained PWA sine-cosine system presented in [BM99],

$$\begin{aligned} x_{k+1} &= \tfrac{4}{5}\begin{bmatrix}\cos\alpha_k & -\sin\alpha_k\\ \sin\alpha_k & \cos\alpha_k\end{bmatrix} x_k + \begin{bmatrix}0\\1\end{bmatrix} u_k,\\ \alpha_k &= \begin{cases} \frac{\pi}{3} & \text{if } \begin{bmatrix}1 & 0\end{bmatrix} x_k \geq 0,\\ -\frac{\pi}{3} & \text{if } \begin{bmatrix}1 & 0\end{bmatrix} x_k < 0,\end{cases}\end{aligned} \tag{6.3a}$$

under the constraints

$$\|x_k\|_\infty \leq 10, \quad |u_k| \leq 1 \quad \forall k \in \mathbb{N}. \tag{6.3b}$$

The CFTOC problem (5.4) was solved for different prediction horizons N, using the weights $Q = I, R = 1$ and a quadratic cost function[2]. How many regions the resulting optimal control laws consist of, and how many regions remain after the application of the region removal algorithm, is shows graphically in Figure 6.4. An exponential relation between the prediction horizon and the number of redundant regions can be observed, culminating in a reduction from 7512 to 522 regions at a prediction horizon of $N = 8$. Although this example is by no means sufficient to substantiate this relation in general, there is a certain probability based on the fact that the number of

[2]All computations have been performed on a 3 GHz Pentium 4 with MPT 2.6.2, [KGBM04], and the commercial NAG solver, [Num02].

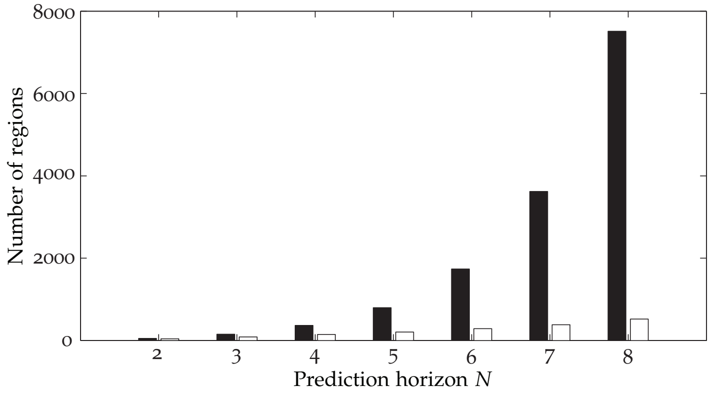


Figure 6.4: Number of regions vs. the prediction horizon N before (black bars) and after (white bars) applying Algorithm 6.1.

switching sequences s grows exponentially with the prediction horizon N, such that also the likelihood of overlapping regions increases. The number of redundant controller regions is in fact problem dependent.

The computation times of the region removal algorithm and, as point of reference, of the related pMIQP for computing the optimal control law are listed in Table 6.1. The removal of regions was performed using directed facet enumeration (Alg. 6.2). The time needed to perform the complexity reduction increased roughly quadratically with the prediction horizon. A worst-case bound for the computation time can be stated as $O(n_r^2)$, which is only tight, if all controller regions overlap.

6.4 Conclusions and Outlook

An algorithm for the identification and removal of redundant regions in explicit model predictive control laws, which arise during the solution of the quadratic cost CFTOC problem for piecewise affine systems, was developed. It proceeds partition-by-partition and involves a number of preprocessing steps to reduce the inherent computational effort of nonconvex minimizations. Two possible alternatives for the comparison of quadratic functions, directed facet enumeration and a general branch-and-bound scheme, have been included in the algorithm. The application of the algorithm was demonstrated in a numerical example as indicator for the potentially possible reduction of complexity.

prediction horizon	time[1] of pMIQP in s	time[1] of Alg. 6.1 in s
2	2	1
3	8	3
4	11	10
5	24	30
6	64	100
7	200	345
8	521	1273

Table 6.1: Computation times of the region removal for different prediction horizons N.

A similar region removal algorithm was presented in [AB06]. Here the piecewise quadratic (PWQ) cost functions of the partitions are approximated from above by PWA functions based on an initial Delauney triangulation of the common domain, [YF05]. Redundant regions are identified by comparing PWQ cost functions with the PWA approximations of other regions. If necessary, the triangulation grid is refined and the procedure recursively repeated for each subset. The advantage of this procedure is the avoidance of direct comparisons of PWQ functions and thus of nonconvex minimization problems. On the other hand, a possibly very fine triangulation grid of each partition has to be determined. A possible branch for future research would be the comparison of the algorithms presented in this chapter with the method in [AB06], and the investigation if one could benefit from merging them.

7 Mechanical System with Backlash

Backlash:
1. A sudden backward motion.
2. A reaction, objection or outcry, especially of a violent or abrupt nature.
3. The distance through which one part of connected machinery, as a wheel, piston, or screw, can be moved without moving the connected parts, resulting from looseness in fitting or from wear.

Wiktionary, the free dictionary

Most publications on hybrid systems focus on theoretical or computational aspects, and as such contain only small simulation examples, but no experimental verifications of the proposed methods. The objective of this project was (*i*) to verify the usefulness of considering the hybrid nature of a dynamical system by *demonstrating the application of hybrid control* methods to a real laboratory system, and (*ii*) to *establish a hybrid benchmark system* which then can be used by other researchers or interested users to compare different identification, estimation and control schemes for hybrid systems.

A mechanical system with backlash was selected, because it is a system well studied by many control practitioners. This interest has been motivated by the fact that backlash in mechanical systems can cause severe performance degradation and lead to instability of the control system. Furthermore, high impact forces in backlash systems can lead to a lower durability of the components and to strokes and peaks in the output. This work was supported by the European Commission research project FP6-IST-511368 *Hybrid Control (HYCON)*.

7.1 Introduction

Backlash is a common problem in mechanical systems occurring whenever there is a gap in the transmission link, e.g. in the differential gearbox of a power train. This transmission gap causes problems, when the system input changes from acceleration to braking or vice versa. During a short time interval the driving torque will not be transmitted to the load. When the backlash gap is traversed, sudden contact will cause a large change in the torque exercised on the load, causing undesirable bumps and possible damages of the mechanical elements in contact. Furthermore, improperly designed control for mechanical systems with backlash may cause undesired vibrations, thus limiting performance and causing additional wear of the mechanical parts.

The problem of controlling mechanical systems with backlash has been considered for a long time by the control community. Surveys of this topic define the scopes of [Lag01] and [NG02]. A detailed treatment of different approaches towards control and estimation of mechanical systems with backlash is given in [Lag04]. New developments have emerged in the theory of *hybrid systems*, [MBB03], and (relatively) recent achievements in the area of *model predictive control* (MPC), [BBBM05].

A natural way to model a mechanical system with backlash is to distinguish between two operating modes, namely the "backlash mode", when the two mechanical parts are not in contact, and the "contact mode", when the contact between the two mechanical parts is established and the transmission of the momentum takes place. The inherent switching between these two modes makes this system a prime example of a hybrid system and motivates the "hybrid approach" to modelling and control of mechanical systems with backlash.

A model predictive control strategy is particularly convenient, when the system to be controlled is subject to *constraints*. The *explicit solution* to an MPC problem, which is obtained off-line, makes the application of this control scheme possible also for systems requiring a fast control sampling rate. The approach has proven to be successful for constrained linear systems, as well as for a class of hybrid systems, namely *piecewise affine* (PWA) systems. The application of MPC to mechanical systems containing backlash is particularly attractive, since it enables a control design incorporating constraints which could increase the safety and reduce the wear of mechanical parts, while preserving satisfactory control performance. An application of MPC to automotive powertrains with backlash has been reported in [LE05], where the authors deploy a linear acceleration controller for the system in contact mode and MPC to traverse the backlash gap.

In this chapter, we demonstrate a comparative study of three different control strategies on a mechanical system with elastic shaft and backlash in the transmission. We compare the performance of a classical LQR design with a switched observer to MPC designed for a linear prediction model (and applied to the actual system with backlash) and MPC designed using a *hybrid prediction model* of the system. Unlike [LE05], we present a more holistic approach by modelling the mechanical system as a hybrid system and computing the explicit MPC for the entire system, i.e. we design a single explicit MPC controller for both modes. The emphasis of the whole project is on potential benefits that could be obtained by applying MPC to such systems, in particular, related to the satisfaction of different constraints. Furthermore, our aim is also to demonstrate a systematic hybrid control design procedure based on freely available MPC controller design tools, [KGBM04] and [Löf04]. In the comparison, both performance and complexity of the controllers are taken into account.

This chapter is structured as follows. After a brief description (Section 7.2) of the physical system at hand, Section 7.3 describes the modelling of the experimental system in terms of a PWA system as well as the parameter estimation. In Section 7.4 a switched observer is introduced to recover unmeasurable states. After describing the design of different controllers and their complexity in Section 7.5, a comparison of experimental results (Section 7.6) is given.

7.2 Mechanical System with Backlash

In order to be able to evaluate control and estimation results not only in theory, but in a real application, a laboratory physical model of a mechanical system with backlash has been constructed. This laboratory setup, which models an automotive powertrain system, can be seen in Figure 7.1. The main elements of the experimental system are two rotating masses, the backlash element, a spring, two DC motors and two encoders which provide position measurements. The system is driven by one motor while the other motor of the same type is used on the load side. The rotating masses represent the inertia of the motor and the load. The spring connecting both sides (see Figure 7.1) has been included to model the flexibility of the shaft. In contrast to automotive powertrains, where shafts are usually rather stiff, we have chosen an elastic shaft, to make the laboratory-scale experiment more demanding. The backlash gap size of the backlash element can be changed to four different values, either 2°, 4°, 6° or 10°. The measured signals are the angles of the motor shaft θ_m and the load shaft θ_l, which are obtained from two incremental encoders with a resolution of 2000 counts per revolution, i.e. approximately 0.0031 rad. This resolution is sufficient to measure

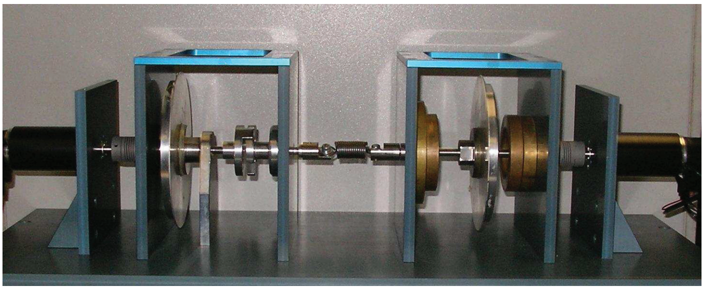


Figure 7.1: The experimental setup of a mechanical system with backlash.

the position of the system in backlash mode for the smallest backlash gap used ($2^\circ \approx 0.035$ rad). However, if only sensors with a lower resolution are available, Kalman filters as proposed by [LE03a] can be used.

7.3 Modelling and Parameter Identification

The mechanical system with backlash, which is described in Section 7.2, has been modelled using a first-principle model. The modelling and parameter identification procedure is illustrated in this section.

Modelling of Hybrid Systems

Recognizing the hybrid nature of the system makes the process of modelling, control and state estimation more accurate and systematic. The models presented will be given in the form of *piecewise-affine* (PWA) *systems*

$$x_{k+1} = f_{\text{PWA}}(x_k, u_k) = A_j x_k + B_j u_k + f_j, \quad \text{if} \quad \begin{bmatrix} x_k \\ u_k \end{bmatrix} \in \mathcal{D}_j, \tag{7.1a}$$

with

$$\mathcal{D}_j := \left\{ \begin{bmatrix} x \\ u \end{bmatrix} \in \mathbb{R}^{n_x+n_u} \;\middle|\; P_{x,j}x + P_{u,j}u \leq p_j \right\} \tag{7.1b}$$

where $k \in \mathbb{N}$ denotes the discrete time, $x_k \in \mathbb{R}^n$ the state vector, $u_k \in \mathbb{R}^m$ the control vector and $\{\mathcal{D}_j\}_{j=1}^{n_D}$ a bounded polyhedral partition of the state-input space. The polyhedral inequalities $P_{x,j}x + P_{u,j}u \leq p_j$ define both regions in which a particular state-update equation is valid as well as constraints on the state and input variables. $A_j, B_j, P_{x,j}, P_{u,j}$ and f_j, p_j are real matrices or real vectors of appropriate dimensions. Under some technical assumptions, the

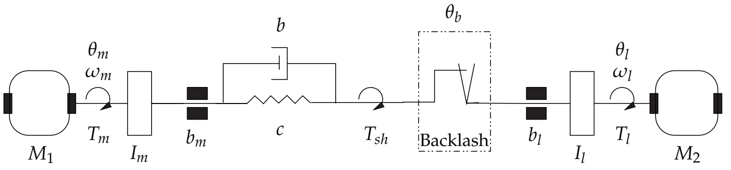


Figure 7.2: Schematic representation of a rotating mechanical system with backlash.

PWA system representation is *equivalent* to several other models of hybrid systems and one can convert one into the other, [HdSB01].

Backlash Model

A schematic representation of the rotating mechanical system with backlash is shown in Figure 7.2. The motor M_1 is the driving motor. The second motor, M_2, can be used to model disturbance torques caused e.g. by different road friction. The angle, rotational speed and torque are denoted by θ_m, ω_m and T_m for the motor shaft, likewise θ_l, ω_l and T_l for the load shaft. The inertia I_m represents the motor flywheel, whereas the inertia I_l represents the load. The dampers b_m and b_l represent viscous friction. The spring-damper combination with spring coefficient c and damping factor b models a flexible shaft with damping.An important parameter in a rotating mechanical system with backlash is the size of the backlash gap, which shall be denoted as 2α. The position in the backlash gap is denoted by θ_b. The torque of the shaft connecting the spring-damper and the backlash element is denoted by T_{sh}.

The configuration of Figure 7.2 can be described approximately by the following differential equations, using balance of moments:

$$I_m\dot{\omega}_m = -b_m\omega_m + T_m - T_{sh}\,, \tag{7.2a}$$

$$I_l\dot{\omega}_l = -b_l\omega_l + T_{sh} + T_l\,. \tag{7.2b}$$

The load torque T_l acts as a disturbance and is considered to be zero under nominal conditions. When shaft damping is taken into account, the shaft torque, T_{sh}, is given by

$$T_{sh} = \begin{cases} c(\Delta\theta - \theta_b) + b(\omega_m - \omega_l) & \text{(Contact)} \\ 0 & \text{(Backlash)} \end{cases} \tag{7.3}$$

Here, the angle $\Delta\theta := \theta_m - \theta_l$ is the total shaft displacement. The backlash position angle θ_b can be described by the following nonlinear differential

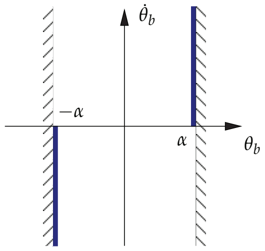


Figure 7.3: Distribution of backlash and contact mode. Thick line: contact mode, white space: backlash mode

equation, [NGG97],

$$\dot{\theta}_b = \begin{cases} \max[0, \Delta\dot{\theta} + \frac{c}{b}(\Delta\theta - \theta_b)] & \text{if } \theta_b = -\alpha \\ \Delta\dot{\theta} + \frac{c}{b}(\Delta\theta - \theta_b) & \text{if } |\theta_b| < \alpha \\ \min[0, \Delta\dot{\theta} + \frac{c}{b}(\Delta\theta - \theta_b)] & \text{if } \theta_b = \alpha \end{cases}, \tag{7.4}$$

where α is half the backlash gap size. From this differential equation for θ_b conditions can be derived which define when the system is in backlash mode. The system is in backlash mode if one of the following three conditions holds

$$|\theta_b| < \alpha\,, \tag{7.5a}$$

$$\theta_b = \alpha \wedge \Delta\dot{\theta} + \frac{c}{b}(\Delta\theta - \theta_b) < 0\,, \tag{7.5b}$$

$$\theta_b = -\alpha \wedge \Delta\dot{\theta} + \frac{c}{b}(\Delta\theta - \theta_b) > 0\,. \tag{7.5c}$$

Conditions (7.5b) or (7.5c) become true when the system is in positive or negative contact mode, respectively, and the backlash elements starts to move away from the driving shaft. The distribution of the backlash and the contact mode vs. the backlash angle θ_b and its derivative $\dot{\theta}_b$ is visualized in Figure 7.3. While the contact mode is marked by the two thick lines, the backlash mode consists of the remaining reachable space.

This leads to the state update equation:

$$\dot{x}(t) = \begin{cases} A_{co}x(t) + Bu(t) & \text{(Positive contact)} \\ A_{bl}x(t) + Bu(t) & \text{(Backlash)} \\ A_{co}x(t) + Bu(t) & \text{(Negative contact)} \end{cases} \tag{7.6}$$

where the state is $x = \begin{bmatrix} \omega_m & \omega_l & \theta_m & \theta_l & \theta_b \end{bmatrix}^T$, and the system matrices are given by

$$A_{co} = \begin{bmatrix} -\frac{b_m+b}{I_m} & \frac{b}{I_m} & -\frac{c}{I_m} & \frac{c}{I_m} & \frac{c}{I_m} \\ \frac{b}{I_l} & -\frac{b_l+b}{I_l} & \frac{c}{I_l} & -\frac{c}{I_l} & -\frac{c}{I_l} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},\ A_{bl} = \begin{bmatrix} -\frac{b_m}{I_m} & 0 & 0 & 0 & 0 \\ 0 & -\frac{b_l}{I_l} & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & \frac{c}{b} & -\frac{c}{b} & -\frac{c}{b} \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{K}{J_m} & 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

Note that since $\Delta\dot{\theta} = \omega_m - \omega_l$ can be expressed in terms of the state variables, Equation (7.4) together with Equation (7.6) describes a continuous-time PWA system defined over a polyhedral partition. For using discrete-time model predictive control, the continuous model from Equation (7.6) still has to be discretized. This can be done for each linear subsystem separately either straightforward via the Euler discretization or with a more sophisticated discretization algorithm. Finally, a discrete-time hybrid model

$$x_{k+1} = \begin{cases} \tilde{A}_{co}x_k + \tilde{B}u_k & \text{(Positive contact)} \\ \tilde{A}_{bl}x_k + \tilde{B}u_k & \text{(Backlash)} \\ \tilde{A}_{co}x_k + \tilde{B}u_k & \text{(Negative contact)} \end{cases} \tag{7.7}$$

can be obtained where the ˜ denotes matrices of the corresponding linear, discrete-time model.

One should be aware that the discretization of continuous-time PWA models introduces some modelling errors; mode changes for instance can be reflected by a continuous-time PWA model at any time, whereas they can only be reflected at each sampling instance by a discrete-time PWA model. The discrete-time PWA model (7.7) is thus only an approximation of the continuous-time model (7.6).

The PWA model (7.7) was described in the modelling language HYSDEL, alongside MATLAB, [TBB+02]. For the simulation of PWA systems and the design and simulation of controllers for PWA systems the Multi-Parametric Toolbox (MPT) was used, see [KGBM04].

A linear prediction model of the mechanical system with backlash can be obtained by *ignoring* the backlash gap, and thus by considering solely the contact mode of the discrete-time PWA model (7.7), resulting in the linear state-update equation

$$x_{k+1} = \tilde{A}_{co}x_k + \tilde{B}u_k\,. \tag{7.8}$$

Parameter Identification

In order to identify the parameters for the model, the backlash element was removed from the experimental system and the shafts were connected, [Alt07]. This leads to a system with almost linear behaviour governed by the following differential equations:

$$I_m \dot{\omega}_m = -b_m \omega_m - c(\theta_m - \theta_l) - b(\omega_m - \omega_l) + T_m \,, \tag{7.9a}$$
$$I_l \dot{\omega}_l = -b_l \omega_l + c(\theta_m - \theta_l) + b(\omega_m - \omega_l) \,. \tag{7.9b}$$

The numerical values for the parameter can now be obtained using linear identification methods, e.g. with the System Identification Toolbox, [Lju06]. Since we used a first principle modelling approach, the parameters of the linear system are the same as the corresponding contact mode parameters in the hybrid model. Therefore, the identified parameters of the linear system can be used in the hybrid model that includes backlash. Afterwards the backlash mode parameters can be extracted by means of physical relations. For the identification of the system in contact mode, a pseudo-random binary signal has been used with an amplitude of $\pm 5\,\mathrm{V}$.

7.4 State Estimation

In the experimental setup considered here, only the motor and load positions are directly available through measurements. For the purpose of state-feedback control, the remaining states, i.e. the position in backlash θ_b and the rotational speed of motor and load, ω_m and ω_l, need to be estimated. This motivates the design of a state estimator as described in this section.

In applications where the switching signal is assumed to be available or can be derived easily by observing sign changes in the motor input signal, switching between linear observers can be used to recover the states of the system, [AC01]. Due to the high elasticity of the spring in the experimental system at hand, a considerable time delay between sign changes in the input signal and the actual switching may occur. In order to cope with this, a switching observer is designed, which does not rely on an external switching sequence, [LE03b].

The size of the backlash gap 2α is considered to be known. However, in general this gap size may be unknown. In these cases this parameter can either be identified in the model identification process or estimated during the system operation. In [LE03b] it is argued that in most cases on-line estimation is the only feasible option.

Since the piecewise affine model (7.6) comprises only two distinct linear dynamics, the design of an *extended Kalman filter* (EKF) is similar to the design of two Kalman filters, one for each linear dynamic. A linearization of the nonlinear dynamics at each time step in a classical EKF is nothing more than using the linear dynamics valid in each region. The advantage of this decoupled design, proposed in [LE03b], is that linear steady state gains can be calculated as stationary solutions to the Riccati equation. In general this is not possible in EKF design.

For the switched observer we choose the following structure

$$\dot{\hat{x}}(t) = \begin{cases} A_{co}\hat{x}(t) + Bu(t) + L_1\Delta y(t) & \text{(Contact)} \\ A_{bl}\hat{x}(t) + Bu(t) + L_2\Delta y(t) & \text{(Backlash)} \end{cases}, \tag{7.10a}$$

$$\hat{y}(t) = \begin{bmatrix}0 & 0 & 1 & 1 & 0\end{bmatrix} \hat{x}(t), \tag{7.10b}$$

and the switching conditions:

$$\text{(Contact)}: \begin{cases} \quad\ \hat{\theta}_b = -\alpha \ \wedge\ \hat{\omega}_m - \hat{\omega}_l + \frac{c}{b}(\Delta\hat{\theta} - \hat{\theta}_b) < 0 \\ \vee\ \hat{\theta}_b = \alpha \quad \wedge\ \hat{\omega}_m - \hat{\omega}_l + \frac{c}{b}(\Delta\hat{\theta} - \hat{\theta}_b) > 0 \end{cases}, \tag{7.11a}$$

$$\text{(Backlash)}: \begin{cases} \quad\ \hat{\theta}_b = -\alpha \ \wedge\ \hat{\omega}_m - \hat{\omega}_l + \frac{c}{b}(\Delta\hat{\theta} - \hat{\theta}_b) \geq 0 \\ \vee\ |\hat{\theta}_b| < \alpha \\ \vee\ \hat{\theta}_b = \alpha \quad \wedge\ \hat{\omega}_m - \hat{\omega}_l + \frac{c}{b}(\Delta\hat{\theta} - \hat{\theta}_b) \leq 0 \end{cases}, \tag{7.11b}$$

where $\hat{x}(t) = [\hat{\omega}_m,\ \hat{\omega}_l,\ \hat{\theta}_m,\ \hat{\theta}_l,\ \hat{\theta}_b]^T \in \mathbb{R}^5$ is the estimated state vector at time t, $\Delta y(t) = y(t) - \hat{y}(t)$ is the output estimation error at time t and the real matrices $L_1, L_2 \in \mathbb{R}^{5\times 2}$ are Kalman filter gains, computed for each linear subsystem separately. Following the notation above, the estimated difference between motor and load angle is denoted by $\Delta\hat{\theta} = \hat{\theta}_m - \hat{\theta}_l$.

While this rather straightforward state estimation scheme was shown to work satisfactorily in an experimental validation, [Los07], there is still space for future improvements. In a more sophisticated estimation setup, one could e.g. follow the Luenberger-type observer approach in [JHW07], or employ a moving horizon estimation scheme, [FMM02].

7.5 Controller Design

In this section the design of an explicit MPC controller based on the PWA model presented in Section 7.3 is described. For the comparison we also design an MPC controller based on the linear model (7.8) describing the system in *contact mode* (i.e. ignoring the backlash mode). The goal we wish to achieve with MPC is to track a speed reference for the load, while satisfying certain constraints. A general description of MPC can be found in Section 4.5, while more specific properties of the quadratic cost optimal control problem for piecewise affine systems are discussed in Section 5.2. The model of the mechanical system with backlash includes the position in backlash, θ_b, and a single MPC controller based on this model is designed. This is in contrast to the approach described in [LE05], where MPC is combined with an additional LQ controller. This incorporation allows us to use all available information from the state observer presented in the previous section and may lead to better control performance.

During operation, the following constraints need to be fulfilled:

$$|u| \leq 5\,\text{V}\,, \tag{7.12a}$$

$$|\theta_b| \leq \alpha = 5^\circ\,, \tag{7.12b}$$

$$|\Delta\omega| \leq \frac{\pi}{2}\,\text{rad/s}\,, \tag{7.12c}$$

where the last one, Equation (7.12c), is added as a *soft constraint* in the MPC design. We introduce constraints on the difference between ω_m and ω_l in order to reduce the impact forces between the mechanical parts when traversing from backlash to contact mode. While it is physically impossible to exceed the input and the backlash angle constraints (7.12a) and (7.12b), the safety constraint on $\Delta\omega$ may be violated. In order to avoid feasibility problems that might occur from this constraint, we define this constraint on the speed difference as a soft constraint. In the sequel we will sometimes refer to this constraint as *safety constraint*.

Since we are interested only in the reference tracking of a particular state (the speed of the load ω_l), we modify the general formulation by augmenting the state vector in the following way:

$$x_{\text{aug},k} := \begin{bmatrix}\omega_{m,k} & \omega_{l,k} & \Delta\theta_k & \theta_{b,k} & u_{k-1} & \omega_{l,\text{ref}}\end{bmatrix}^T \in \mathbb{R}^6\,,$$

where $k \in \mathbb{N}$ denotes discrete time, $\Delta\theta_k := \theta_{m,k} - \theta_{l,k}$ is the relative angle between motor and load shaft, u_{k-1} is the control input in the previous sampling interval and $\omega_{l,\text{ref}}$ is a reference value for the rotational speed of the load. Instead of computing the steady-state values for u_k, we use $\Delta\boldsymbol{U}_k := \{\Delta u_k, \ldots, \Delta u_{k+N_c-1}\}$ with $\Delta u_{k+i} := u_{k+i} - u_{k+i-1}$, $i = 0, \ldots, N_c - 1$, as the optimization variable. The formulation of the underlying CFTOC problem is the following:

$$\min_{\Delta\boldsymbol{U}_k, s} \|s\|_{q_s}^2 + \sum_{i=0}^{N-1} \|\omega_{l,k+i} - \omega_{l,\text{ref}}\|_Q^2 + \sum_{i=0}^{N_c-1} \|\Delta u_{k+i}\|_R^2\,, \tag{7.13a}$$

$$\text{s.t. } x_{\text{aug},k+i+1} = f_{\text{DYN}}(x_{\text{aug},k+i}, u_{k+i})\,, \quad i = 0, \ldots, N-1\,, \tag{7.13b}$$

$$|\omega_{l,k+i} - \omega_{m,k+i}| \leq \Delta\omega_{\max} + s\,, \quad i = 0, \ldots, N-1\,, \tag{7.13c}$$
$$s \geq 0\,, \tag{7.13d}$$
$$\text{(7.12a)} \quad \text{and} \quad \text{(7.12b)}\,, \tag{7.13e}$$

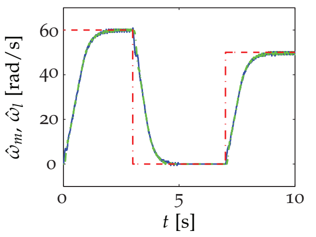
where s is a slack variable introduced to enforce the soft constraints and $\Delta\omega_{\max}$ is the maximal difference allowed between the rotational speeds at the load and the motor. The abbreviations $\|x\|_Q^2 := x^T Q x$ were used.

The state-update function f_{DYN} may be easily derived from the PWA model (7.7) or the linear model (7.8), respectively. For all control schemes a large weight is put on the term penalizing the tracking error in the cost function. For MPC we used a prediction horizon $N = 4$ and discrete-time models with a sampling period of $T_s = 0.04\,\text{s}$.

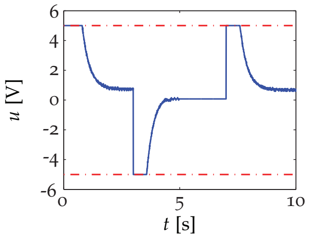
Several measures were necessary to reduce the complexity of the explicit solution sufficiently to be able to evaluate the control law within the sampling time. The augmentation of the state vector, needed for the tracking of arbitrary references, increases the dimension of the explicit solution. In order to reduce the complexity of the parametric problem, a prediction model of lower order is employed, where the absolute shaft angles are substituted by the relative angle, $\Delta\theta = \theta_m - \theta_l$ (compare $x_{\text{aug},k}$).

Furthermore, a move-blocking strategy with the control horizon $N_c = 2$ is used, i.e. the optimization problem (7.13) is simplified by considering only u_k and u_{k+1} as optimization variables and set the value $u_{k+i} = u_{k+1}$ for $i = 2, \ldots, N-1$. The move-blocking strategy introduces more constraints to the CFTOC problem, and thus results in some suboptimality of the solution and a potential reduction of the feasible set. On the other hand, the move-blocking strategy was able to reduce the number of regions of the explicit hybrid MPC controller from 21593 to 3289, i.e. by a factor of more than 6.

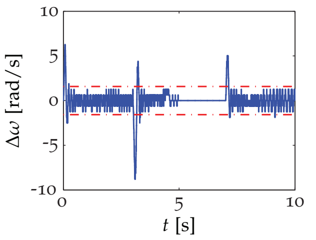
After the controller computation the region removal algorithm from Chapter 6 was applied, which resulted in the removal of merely 78 regions. One should note that the move-blocking strategy renders many switching sequences infeasible, since the future control control actions are set to an equal value. Consequently the potential for a region removal is already quite low, which explains the low number of identified redundant regions. With the described setup, we obtained an explicit MPC based on the hybrid model comprising 3211 polyhedral regions in 6 dimensions. With the linear prediction model, the obtained explicit linear MPC controller comprises 101 polyhedral regions.



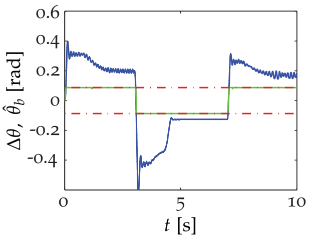
(a) Estimation of rotational speed of motor shaft $\hat{\omega}_m$ (—), load shaft $\hat{\omega}_l$ (– –) and reference speed ω_{ref} (– ·)



(b) Input voltage u and input constraints (– ·)



(c) Estimated speed difference $\Delta\hat{\omega}$ and safety constraints (– ·)



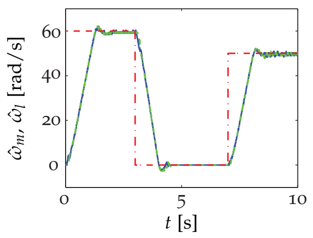
(d) Estimated backlash angle θ_b (—), shaft torsion $\Delta\theta$ (– –) and backlash gap α (– ·)

Figure 7.4: Experimental results: State evolution of the backlash system under LQ control.

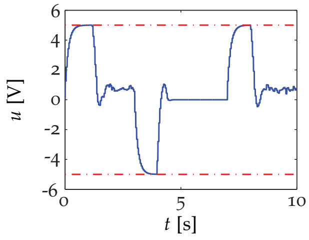
7.6 Comparison of Different Control Strategies

This section describes the comparison of different control strategies applied to the mechanical system with backlash. As mentioned in the introduction, the performance of a standard LQ controller shall be compared to MPC based on the constrained linear model and to MPC using the hybrid PWA model of the backlash system, as described in Section 7.5. Therefore a control experiment is considered where the control aim is to have the load shaft follow a certain speed reference trajectory.

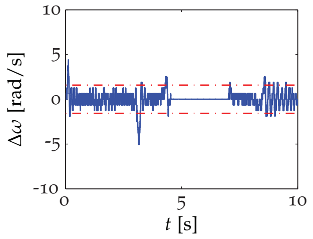
The experiments were conducted on the laboratory-scale system using MATLAB Real-Time Workshop with xPC Target. A computer, connected to the experimental system was used to run observer and controller. The sampling time of this xPC Target platform was chosen to be $T_{\text{sim}} = 5\,\text{ms}$ in order to obtain appropriate state estimations, whereas the MPC controllers were updated every $T_s = 40\,\text{ms}$. All three controllers have been tuned in such a way, that they show a similar performance in tracking the reference signal.



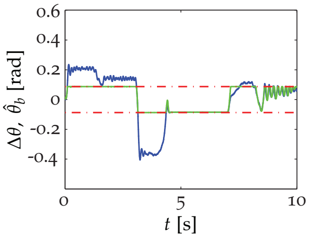
(a) Estimation of rotational speed of motor shaft $\hat{\omega}_m$ (—), load shaft $\hat{\omega}_l$ (– –) and reference speed ω_{ref} (– ·)



(b) Input voltage u and input constraints (– ·)



(c) Estimated speed difference $\Delta\hat{\omega}$ and safety constraints (– ·)



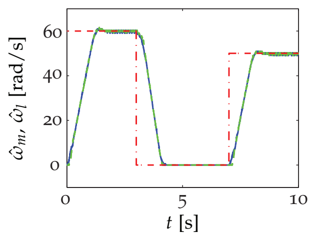
(d) Estimated backlash angle θ_b (—), shaft torsion $\Delta\theta$ (– –) and backlash gap α (– ·)

Figure 7.5: Experimental results: State evolution of the backlash system under linear MPC.

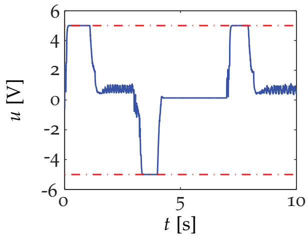
Therefore, the evaluation of these three control strategies can be done by considering the complexity issue and the satisfaction of constraints (7.12).

LQ control with a switching observer admits a very simple control law and provides a satisfying tracking performance. However, the constraint (7.12c) on the speed difference between the load and the motor cannot be enforced without sacrificing performance. While the LQ controller was tuned less aggressive than the MPC controllers, the closed-loop behaviour shows a significant violation of the safety-constraint, see Figure 7.4.

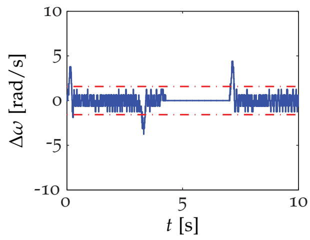
MPC based on the linear model (Figure 7.5) is relatively simple in structure and shows a satisfying control performance. While being tuned more aggressively than the LQ controller and thus achieving a faster reference tracking, the linear MPC controller nevertheless comes quite close to fulfilling the safety constraint on $\Delta\omega$ (Figure 7.5 (c)). However, violations of the safety constraint still occur. Moreover there are small overshoots whenever the experimental system enters the backlash mode. This comes as no surprise since the backlash mode is not included in the linear prediction model.



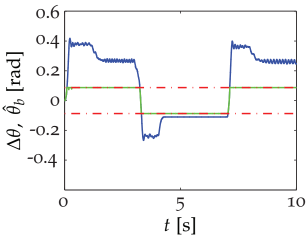
(a) Estimation of rotational speed of motor shaft $\hat{\omega}_m$ (—), load shaft $\hat{\omega}_l$ (– –) and reference speed ω_{ref} (– ·)



(b) Input voltage u and input constraints (– ·)



(c) Estimated speed difference $\Delta\hat{\omega}$ and safety constraints (– ·)



(d) Estimated backlash angle θ_b (—), shaft torsion $\Delta\theta$ (– –) and backlash gap α (– ·)

Figure 7.6: Experimental results: State evolution of the backlash system under hybrid MPC.

In order to reduce the occurrence of the spikes in $\Delta\omega$ even further and to improve control performance, a full hybrid model was used for the predictions, leading to a hybrid MPC framework as described in Section 7.5. A faster, smoother response and a slight reduction of safety constraint violations can be observed (Figure 7.6). Knowledge of the backlash mode seems to be able to remove the small overshoots during the mode transitions. Nevertheless, the hybrid MPC controller is not capable to consistently satisfy the safety constraint.

7.7 Addendum: Robustness Issues

When we published the result of this project initially in [RBB$^+$07], we reported robustness issues with the hybrid MPC approach, which in some runs led to performance degradations and occasionally even to a loss of stability. After the publication of [RBB$^+$07], the robustness issues of the hybrid

MPC approach were analyzed in detail, [Chu08]. It turned out that the behaviour was caused by a simple implementation error in a rate transition block within Simulink, independent of the methods discussed previously. Since this implementation error was removed the robustness issues disappeared entirely.

7.8 Conclusion and Outlook

A mechanical system with backlash has been specified and built, providing a benchmark system for hybrid control and identification strategies with this experimental setup. Modern control strategies for hybrid systems can be verified and their realizability on a real system can be investigated. A hybrid system benchmark problem was created, [Alt07], which can be accessed by other researchers via the webpage

```
https://control.ee.ethz.ch/~backlash/
```

The full design cycle containing modelling, estimation and control design for this rotational system with backlash has been performed. An observer, based on [LE03b], was implemented and tested in experiments with the laboratory setup. The aim was to design a controller that tracks the rotational speed of the load shaft while minimizing bumps or damages that can occur when the system is operated and backlash is not taken into account. This was realized by constraining the difference in speed between the drive and the load $\omega_m - \omega_l$.

Three different controllers were designed and investigated in a typical start-stop scenario. A standard LQ controller, where constraints cannot be considered directly, was compared to model predictive control for a linear and a hybrid model. The different effectiveness of the investigated controllers in handling constraints was shown. While the constraints are violated significantly with LQ control, MPC with a linear model shows only small violations of the constraints. With a hybrid model, accounting for backlash in the system, violations of the constraints could be reduced even further.

What are the lessons learnt? What can be deduced from this experiment to the general application of hybrid control? The experiments on the mechanical system with backlash have shown that considering the hybrid nature of the system in the MPC scheme is a way to improve control performance and constraint satisfaction of the system. However, the improvements over the linear MPC scheme are not significant for the system at hand. In practice, the use of the more complex hybrid prediction model should at least be put into question. This brings us to the more general question for factors which should influence the consideration of a certain mode in the prediction model.

Compared to other systems affected by backlash, the selected size of the backlash angle is quite high. Nevertheless, the backlash is traversed within a few time steps such that the prediction errors of the linear MPC approach by not considering the backlash mode are rather low. The larger the size of the backlash gap, the larger the possible benefit of including the backlash mode on the control performance. Therefore the size of the backlash gap, or in general the time spent in a certain mode, should influence the decision if a mode is considered in the prediction model or not.

Another significant factor is the quality of estimation. Though the used incremental encoders are accurate, it is not possible to measure the mode of the system, due to the spring. Rather, the correct mode as well as some of the system states have to be estimated. The hybrid control practitioner should keep in mind that a sophisticated model of a hybrid system can be of no use, if the mode and/or state of the hybrid system can not be measured or estimated reliably.

Another factor are the nonlinearities of the system at hand. While the benefits of considering the hybrid nature of the backlash system in the MPC scheme were demonstrated, there is still space for improvements, especially the constraint on the speed difference is not guaranteed with the current scheme. This can be explained by the fact that the hybrid model is only an approximation of the true dynamic behaviour of the system. Nonlinearities, as they are caused by the DC motors and by occurring mechanical friction, are not taken into account. The general question should be how large the influence of the hybrid nature of the system is in comparison to other neglected nonlinearities. It might turn out that the neglected nonlinearities undermine all efforts spent to model the hybrid modes of a system accurately. A robust approach might be appropriate.

8 Autonomous Vehicle Steering (I)

'The dream of cars driving themselves is becoming a reality. Before, the question was whether it was possible. Now we know it is.'

Sebastian Thrun

Autonomous steering of automobiles is the subject of interest in this chapter. A case study on different MPC schemes for autonomous vehicle steering is presented. The objective of this project was the reduction of the online computational effort of the control scheme, while maintaining the control performance of nonlinear MPC, [KFB+06]. For this task the nonlinear lateral vehicle model was approximated by a linear model, a hybrid model and a hybrid parameter-varying (HPV) model. A comparison between controllers using prediction models varying from the full nonlinear model, as an indication for the maximal achievable performance, to a linear model was performed. In the investigated scenarios, the displacement of a car on an icy road due to a side wind gust shall be mitigated, and a double lane-change maneuver shall be performed autonomously. This project was supported by the European Commission research project FP6-IST-511368 *Hybrid Control* (HYCON).

8.1 Introduction

In recent years, the number of road accident fatalities in developed countries were on the decline. This is mainly due to the improved safety of automobiles and to the development of driver assistance systems like the Antilock Braking System or the Electronic Stability Program. Nevertheless, road accidents are still one of the major causes of death among young people. The development and improvement of driver assistance systems is consequently an area of active research.

Autonomous Vehicle Steering is a driver assistance system which could not only support the driver in emergency situations and thus improve road traffic safety, but it could also be part of an autonomous driving system replacing the driver in a long-term perspective. Contrary to the aforementioned systems, autonomous steering does not utilize the brakes but the steering angle of the wheels to control the lateral dynamics and thus the vehicle trajectory.

Trigonometric relations between angles and forces lead to a nonlinear control problem, further complicated by the presence of varying road friction and longitudinal dynamics, which influence the lateral dynamics of the vehicle. Robust control techniques are one possible way to deal with these complications. The application of robust control techniques for autonomous steering has been studied intensively, for instance in [AGS+95, Ack99, STS04].

Keviczky et al. presented a way to apply nonlinear model predictive control (MPC) to autonomous vehicle steering, [KFB+06]. This predictive control approach is able to achieve high performance while respecting constraints on the state and the input signal. For this purpose, a nonlinear model together with the commercial solver NPSOL is used for the *nonlinear optimization*. Although remarkable control performance could be achieved, the high computational burden is a drawback for practical implementations.

In this chapter, three simplified models are derived, all of them allow the use of *quadratic programming* (QP) solvers instead of nonlinear programming (NP) solvers, and thus a reduction in computational complexity. The emphasis lies on a hybrid parameter-varying model with parameter-varying system matrices to improve the accuracy of the model. As a direct competitor a purely hybrid model with constant system matrices in each region is considered. The improvements due to the hybrid nature of the aforementioned models are illustrated in a comparison to a linear model. The MPC schemes are compared in two simulated driving scenarios.

This chapter is structured as follows. Section 8.2 shows how to derive the nonlinear model of the lateral vehicle dynamics and its approximation by a piecewise affine parameter-varying model, a piecewise affine model and

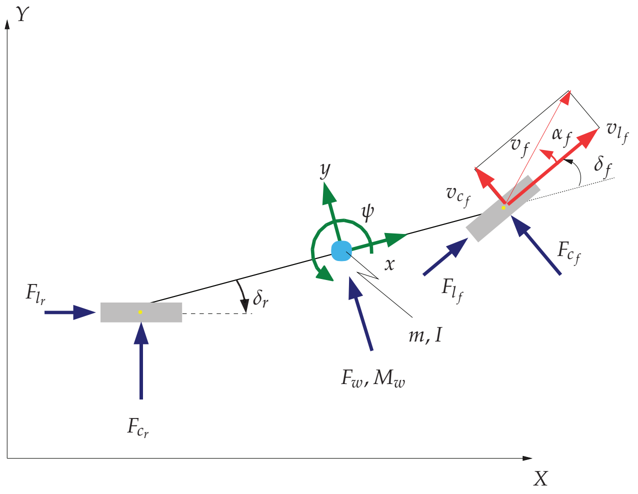


Figure 8.1: Scheme for single-track model.

a linear model, [JR03]. In Section 8.3, the design of the compared MPC schemes is exposed. The application of the MPC schemes in a side wind rejection scenario is examined in Section 8.4. Section 8.5 presents the application of the model predictive control schemes to the double lane-change maneuver and the required adaptations to the MPC schemes. Finally conclusions are drawn.

8.2 Modelling

This section describes the modelling of the lateral vehicle dynamics. Firstly, a nonlinear model is derived, which is then approximated by two hybrid models, one of them parameter dependent, and one linear model.

Nonlinear Model

A nonlinear model of the lateral dynamics has been presented in [KFB+06]. In order to clarify our approximations of this nonlinear model, its derivation is stated in this section. Figure 8.1 shows a standard single-track model of a car, as derived in [MA91]. Here, the width of the car is neglected and only the length is considered. The distance between the centre of mass and the front wheels is a and the distance to the rear wheels is b. The model assumes the mass to be centred. All forces and moments are transformed to this centre of mass in order to obtain ordinary differential equations describing the lateral motion. Two coordinate systems are defined, one relative to the car position, (x, y, ψ), and one fixed to the road, (X, Y, ψ).

A balance of the moments and forces leads to the following differential equations:

$$m\ddot{y} = -mv_x\dot{\psi} + 2F_{y,f} + 2F_{y,r} + F_w\,, \tag{8.1a}$$
$$I\ddot{\psi} = 2aF_{y,f} - 2bF_{y,r} + M_w\,. \tag{8.1b}$$

The external moment M_w and force F_w model the effect of sidewind, acting on the car's body. The dependence on the wind speed v_w is assumed to be

$$F_w = \frac{2.5\pi}{2}v_w^2\,, \tag{8.2}$$
$$M_w = \left(\frac{2.5\pi}{2} - 3.3\left(\frac{\pi}{2}\right)^3\right)v_w^2 + \frac{a-b}{2}F_w\,. \tag{8.3}$$

The forces $F_{y,i}$ in (8.1), where $i = \{f, r\}$ denotes the front or rear wheels, are the transformed tire forces, resulting from the contact between tire and ground,

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \end{bmatrix} = \begin{bmatrix} \cos(\delta_i) & -\sin(\delta_i) \\ \sin(\delta_i) & \cos(\delta_i) \end{bmatrix} \begin{bmatrix} F_{l,i} \\ F_{c,i} \end{bmatrix}. \tag{8.4}$$

The most important part of the lateral vehicle model is the contact behaviour of the tires with the ground. Several studies have investigated the behaviour of the contact forces for different road conditions and tire geometries. It can be distinguished between two basic types of tire models, static and dynamic tire models. In this project the tire dynamics are neglected and the static Pacejka tire model is used, [BNP87, BO97].

In the Pacejka tire model, the longitudinal and corner forces $F_{l,i}, F_{c,i}$ for all tires are calculated as functions of the slip ratio s_i, the adhesion coefficient μ, the slip angle α_i and the normal tire forces $F_{z,i}$. The slip ratio s_i is defined as the difference between the actual longitudinal velocity at the axle of the wheel and the equivalent rotational velocity of the tire, see e.g. [Raj06]. Assuming constant normal tire forces leads to the force dependencies

$$F_{l,i} = F_{l,i}(\alpha_i, s_i, \mu), \quad F_{c,i} = F_{c,i}(\alpha_i, s_i, \mu), \quad i = \{f, r\}\,.$$

The cornering and longitudinal tire forces $F_{c,i}$ and $F_{l,i}$ are the main nonlinearities in the lateral dynamical system (8.1). The exact shape of the assumed nonlinear relations are provided by the Pacejka tire force model. As an example for the shape of the tire forces in the Pacejka tire force model, Figure 8.2 shows the corner tire force $F_{c,i}$ for $\mu = 0.9$.

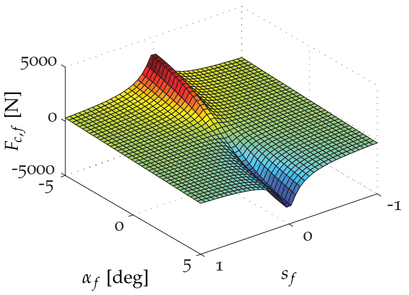


Figure 8.2: Cornering tire force $F_{c,f}(\alpha_f, s_f)$ as function of slip angle α_f and slip s_f for $\mu = 0.9$.

As illustrated in Figure 8.1, the slip angles α_i depend on the longitudinal and lateral speed of the tires:

$$\alpha_i = \tan^{-1}\left(\frac{v_{c,i}}{v_{l,i}}\right) . \tag{8.5}$$

These velocities again relate to the velocity of the vehicle in the following way:

$$v_{l,f} = (\dot{y} + a\dot{\psi})\sin(\delta_f) + v_x \cos(\delta_f) , \tag{8.6a}$$
$$v_{c,f} = (\dot{y} + a\dot{\psi})\cos(\delta_f) - v_x \sin(\delta_f) , \tag{8.6b}$$
$$v_{l,r} = (\dot{y} - b\dot{\psi})\sin(\delta_r) + v_x \cos(\delta_r) , \tag{8.6c}$$
$$v_{c,r} = (\dot{y} - b\dot{\psi})\cos(\delta_r) - v_x \sin(\delta_r) . \tag{8.6d}$$

Using the state vector

$$x = \begin{bmatrix} Y & \dot{Y} & \psi & \dot{\psi} \end{bmatrix}^T ,$$

where Y belongs to the fixed coordinates and is related to the car coordinates via

$$\dot{Y} = v_x \sin(\psi) + \dot{y}\cos(\psi) , \tag{8.7}$$

we conclude from (8.5) – (8.7) that $\alpha_i = \alpha_i(x, \delta_i)$ and consequently $F_{y,i} = F_{y,i}(x, \delta_i, s_i, \mu)$. Moreover, the lateral vehicle dynamics can be modelled by nonlinear differential equations including (8.1) – (8.7). The resulting system is visualized in a block diagram in Figure 8.3, and can be stated as

$$\frac{d}{dt}\begin{bmatrix} Y \\ \dot{Y} \\ \psi \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{Y} \\ \frac{1}{m}(2F_{y,f} + 2F_{y,r} + F_w)\cos(\psi) + (v_x \sin(\psi) - \dot{Y})\dot{\psi}\tan(\psi) \\ \dot{\psi} \\ \frac{1}{J}(2aF_{y,f} - 2bF_{y,r} + M_w) \end{bmatrix} . \tag{8.8}$$

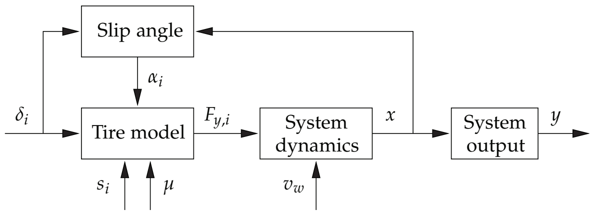


Figure 8.3: Block diagram of the open-loop system.

Hybrid Parameter-Varying Model

This section describes the linearizations made to yield a hybrid parameter-varying (HPV) model of the lateral car dynamics. Linear Parameter-Varying (LPV) models and the design of LPV gain-scheduling controllers will be described in detail in Chapter 9. For the moment it is sufficient to regard HPV models as an extension of PWA models, inspired by LPV models. HPV systems can be stated via

$$\begin{aligned} \dot{x} &= A_j(\theta)x + B_j(\theta)u + f_j(\theta)\,, \\ y &= C_j(\theta)x + D_j(\theta)u\,, \end{aligned} \qquad \text{if} \quad \begin{bmatrix} x \\ u \end{bmatrix} \in \mathcal{D}_j \tag{8.9}$$

with the state $x(t) \in \mathbb{R}^{n_x}$, the input $u(t) \in \mathbb{R}^{n_u}$ and the output $y(t) \in \mathbb{R}^{n_y}$, and where the index $j = 1, \ldots, n_D$ denotes one of the parameter-dependent, polyhedral regions

$$\mathcal{D}_j := \left\{ \left[\begin{array}{c} x \\ u \end{array} \right] \in \mathbb{R}^{n_x+n_u} \;\middle|\; P_{x,j}(\theta)x + P_{u,j}(\theta)u \leq p_j(\theta) \right\}$$

within the state-input space. The system matrices $A_j(\theta), B_j(\theta), C_j(\theta), D_j(\theta)$ and the affine term $f_j(\theta)$ of each region $\mathcal{D}_j$ are not constant, but depend on a time-varying parameter vector

$$\theta(t) = \begin{bmatrix} \theta_1(t) & \theta_2(t) & \ldots & \theta_{n_\theta}(t) \end{bmatrix} \in \Theta \subseteq \mathbb{R}^{n_\theta}\,. \tag{8.10}$$

The parameter vector $\theta(t)$ itself depends on a vector of scheduling variables $\rho \in \mathbb{R}^{n_\rho}$,

$$\theta(t) = f_\rho(\rho(t)). \tag{8.11}$$

While the scheduling variables ρ are measured or estimated signals, Equation (8.11) allows the definition of an favourable parameter space $\Theta \subseteq \mathbb{R}^{n_\theta}$.

In order to obtain such a hybrid parameter-varying (HPV) model, the non-linear model (8.8) is linearized. The following assumptions were made for these linearizations:

- constant longitudinal speed v_x,
- constant normal tire forces $F_{z,f}$, $F_{z,r}$,
- small angles: ψ, $\delta_f \ll 1$,
- front wheel steering: $s_r = \delta_r = 0$.

With these assumptions, system (8.8) can be linearized to

$$\frac{d}{dt}\begin{bmatrix} Y \\ \dot{Y} \\ \psi \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{Y} \\ \frac{1}{m}(2F_{y,f} + 2F_{y,r} + F_w) \\ \dot{\psi} \\ \frac{1}{J}(2aF_{y,f} - 2bF_{y,r} + M_w) \end{bmatrix}, \tag{8.12}$$

where the lateral forces $F_{y,i}$ can be computed from the longitudinal and corner forces of the tires via the linearized force transformation (cf. (8.4))

$$F_{y,i} = F_{l,i}\delta_i + F_{c,i}\,, \quad i \in \{f, r\}\,. \tag{8.13}$$

These tire forces shall now be approximated. In the Pacejka tire force model, the longitudinal and corner forces are nonlinear functions of the slip ratios s_i, the slip angles α_i and the friction coefficient μ. The slip angles α_i are depending in a nonlinear way on the state and input of the system, see (8.5) and (8.6). These relations can be linearized to

$$\alpha_f = -\delta_f - \psi + (\dot{Y} + a\dot{\psi})/v_x\,, \tag{8.14a}$$

$$\alpha_r = -\delta_r - \psi + (\dot{Y} - b\dot{\psi})/v_x\,, \tag{8.14b}$$

where the relation between car coordinate $\dot{y}$ and global coordinate $\dot{Y}$ from (8.7) has been linearized to

$$\dot{Y} = \dot{y} + v_x\psi\,. \tag{8.15}$$

The hybrid parameter-varying model (8.9) is piecewise-affine in its state x and its control input u, whereas its system matrices can depend on the scheduling signals ρ in any functional manner. In our case the control input u is the steering angle δ_f such that the longitudinal forces $F_{l,i}$ contribute to the input matrix $B(\theta)$, see (8.13). From (8.12) to (8.14b) follows that in order to obtain a model which depends on the state x and the control input δ_f in a piecewise affine manner, we have to approximate the longitudinal tire forces $F_{l,i}$ to be independent of the slip angles α_i.

The selected approximations of the tire forces are illustrated in the following figures. The longitudinal force $F_{l,f}$ can be approximated by evaluating the Pacejka tire force for $\alpha_f = 0$. Figure 8.4 shows the longitudinal force $F_{l,f}$ and

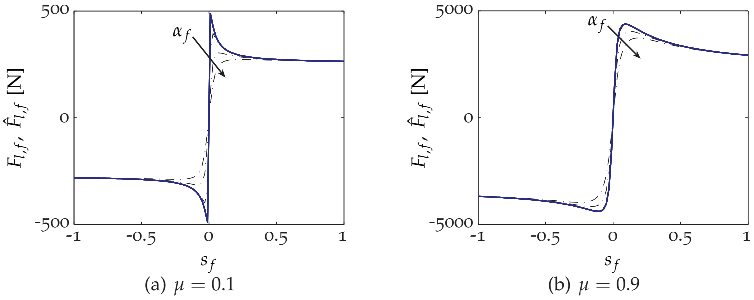


Figure 8.4: Longitudinal tire force of the front wheels $F_{l,f}$ ($- \cdot -$) and its approximation $\hat{F}_{l,f}$ (—) vs. slip ratio s_f. The arrows point into the direction of increasing slip angle α_f.

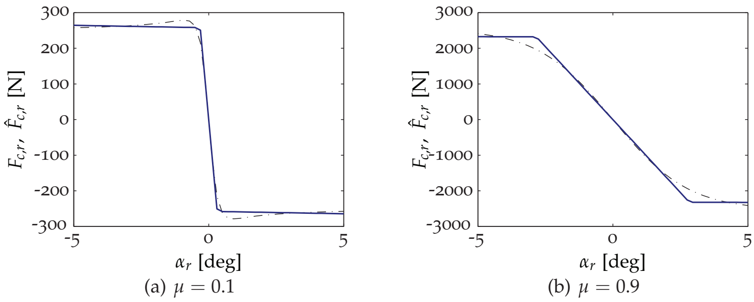
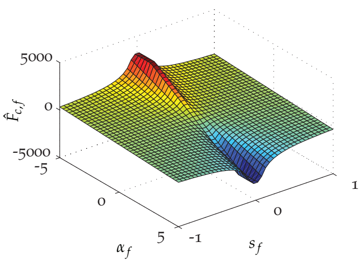


Figure 8.5: Corner tire force for rear wheels $F_{c,r}$ ($- \cdot -$) and its approximation $\hat{F}_{c,r}$ (—) vs. slip angle α_r.

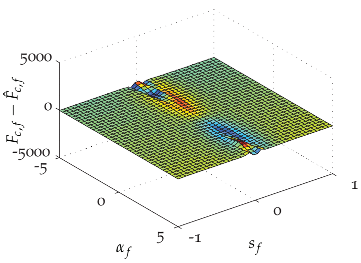
the selected approximation $\hat{F}_{l,f}$ vs. s_f for two different values of the friction coefficient μ. The approximation of the longitudinal force $F_{l,r}$ can be set to zero as δ_r is zero, cf. (8.13).

The approximations of the corner forces $F_{c,f}$ and $F_{c,r}$, as shown in Figure 8.5 and Figure 8.6, can depend on α_i in a piecewise affine way, since they are not multiplied by the control input δ_f in (8.12). The hybrid aspect of the HPV model makes it particular suitable to model the saturating effects of the cornering forces. Note that the approximation of $F_{c,f}$ is far more difficult as $F_{c,f}$ depends on three variables (α_f, s_f and μ), whereas $F_{c,r}$ only depends on two ($s_r = 0$).

Finally, the force approximations can be stated as piecewise affine functions of the slip angles,



(a) Approximation $\hat{F}_{c,f}$ for $\mu = 0.9$.



(b) Approximation error for $\mu = 0.9$.

Figure 8.6: Approximation of cornering tire force for front wheels $F_{c,f}$ and approximation error vs. slip angle α_f and slip ratio s_f for $\mu = 0.9$.

$$\hat{F}_{c,f} = \theta_1 \alpha_f + \theta_2\,, \qquad \hat{F}_{l,f} = \theta_5\,, \tag{8.16a}$$

$$\hat{F}_{c,r} = \theta_3 \alpha_r + \theta_4\,, \qquad \hat{F}_{l,r} = 0\,, \tag{8.16b}$$

where the scheduling parameters $\theta_1, \ldots, \theta_5$ depend on the scheduling signals $\rho = [s_f \;\; \mu \;\; v_w]^T$, and on possible saturations of the force approximations

$$\tilde{F}_{c,f} = c_1 (\exp(\frac{c_2 \mu}{|s_f + \varepsilon| + c_3}) - 1)\alpha_f\,, \tag{8.17}$$

$$\tilde{F}_{c,r} = k_1 (\exp(k_2 \mu) - 1)\alpha_r\,, \tag{8.18}$$

using the coefficients c_1, c_2, c_3, k_1, k_2.

Inserting the force approximations (8.16) into the linearized model (8.12) yields the matrices of the *Hybrid Parameter-Varying Model*:

$$A_j(\theta) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2(\theta_1+\theta_3)}{m v_x} & \frac{-2(\theta_1+\theta_3)}{m} & \frac{2(\theta_1 a - \theta_3 b)}{m v_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2(\theta_1 a - \theta_3 b)}{J v_x} & \frac{-2(\theta_1 a - \theta_3 b)}{J} & \frac{2(\theta_1 a^2 + \theta_3 b^2)}{J v_x} \end{bmatrix}, \quad B_j(\theta) = \begin{bmatrix} 0 \\ \frac{2(\theta_5 - \theta_1)}{m} \\ 0 \\ \frac{2a(\theta_5-\theta_1)}{J} \end{bmatrix}$$

$$f_j(\theta) = \begin{bmatrix} 0 \\ \frac{2\theta_2 + 2\theta_4 + \theta_6}{m} \\ 0 \\ \frac{2a\theta_2 - 2b\theta_4 + \theta_7}{J} \end{bmatrix}, \quad C_j(\theta) = I_4\,, \quad D_j(\theta) = O_4\,, \tag{8.19}$$

with the scheduling parameters

$$\theta_1 = \begin{cases} c_6\,, & \tilde{F}_{c,f} > c_5\mu \\ c_1(\exp(\frac{c_2\mu}{|s_f+\varepsilon|+c_3}) - 1)\,, & |\tilde{F}_{c,f}| < c_5\mu \\ c_6\,, & \tilde{F}_{c,f} < -c_5\mu \end{cases} \tag{8.20a}$$

$$\theta_2 = \begin{cases} c_4\mu\,, & \tilde{F}_{c,f} > c_5\mu \\ 0\,, & |\tilde{F}_{c,f}| < c_5\mu \\ -c_4\mu\,, & \tilde{F}_{c,f} < -c_5\mu \end{cases} \tag{8.20b}$$

$$\theta_3 = \begin{cases} k_4\,, & \tilde{F}_{c,r} > k_5\mu \\ k_1(\exp(k_2\mu) - 1)\,, & |\tilde{F}_{c,r}| < k_5\mu \\ k_4\,, & \tilde{F}_{c,r} < -k_5\mu \end{cases} \tag{8.20c}$$

$$\theta_4 = \begin{cases} k_3\mu\,, & \tilde{F}_{c,r} > k_5\mu \\ 0\,, & |\tilde{F}_{c,r}| < k_5\mu \\ -k_3\mu\,, & \tilde{F}_{c,r} < -k_5\mu \end{cases} \tag{8.20d}$$

$$\theta_5 = F_{l,f}(\alpha_f = 0)\,, \tag{8.20e}$$

$$\theta_6 = F_w\,, \quad \theta_7 = M_w\,, \tag{8.20f}$$

and with the constants $c_1, \ldots, c_5$ and $k_1, \ldots, k_5$.

The values of the scheduling parameters depend solely on the scheduling signals, $\theta = f_\rho(\rho)$, whereas the distinction of the hybrid modes $\mathcal{D}_j$ is not only depending on the scheduling signals ρ, but for each value of the scheduling signal a polytopic function in the state space. Taking into account that the distinction of cases for the parameters in (8.20) yields 9 different modes, it becomes clear that the model (8.19) - (8.20) is indeed a hybrid parameter-varying model of the form (8.9). Note also that $C_j(\theta) = I_4$ and $D_j(\theta) = O_4$ are constant matrices while $A_j(\theta), B_j(\theta)$ and $f_j(\theta)$ are parameter-varying with $\theta = [\theta_1, \ \ldots, \ \theta_7]^T$ as parameters.

Hybrid Model

The following section describes the hybrid model derived from the HPV model (8.19) - (8.20). The HPV model makes use of parameter-varying system matrices to adjust to different values of the scheduling variables s_i, μ and v_w. The parameter dependence facilitates a more accurate model, but at the price of non-constant system matrices. In order to achieve a hybrid model with multiple, but parameter-independent system matrices, the scheduling variables are replaced by constant values. The following values have been chosen:

$$s_f = 0\,, \quad \mu = 0.1\,, \quad v_w = 0\,\mathrm{m/s}\,.$$

The front slip ratio and the sidewind speed have been set to $s_f = v_w = 0$, because this represents their average values. The road friction on the other hand has been fixed to $\mu = 0.1$, which means the controller is assuming to drive permanently on ice. The worst case is assumed here to ensure accuracy and thus safety on icy roads, albeit at the price of a worse performance when driving on roads with higher adhesion.

By fixing the scheduling variables in (8.20), the parameters are restricted to discrete values, such that we obtain the *Hybrid Model*:

$$A_j = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2(\theta_1+\theta_3)}{mv_x} & \frac{-2(\theta_1+\theta_3)}{m} & \frac{2(\theta_1 a-\theta_3 b)}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2(\theta_1 a-\theta_3 b)}{Jv_x} & \frac{-2(\theta_1 a-\theta_3 b)}{J} & \frac{2(\theta_1 a^2+\theta_3 b^2)}{Jv_x} \end{bmatrix}, \quad B_j = \begin{bmatrix} 0 \\ \frac{2(\theta_5-\theta_1)}{m} \\ 0 \\ \frac{2a(\theta_5-\theta_1)}{J} \end{bmatrix}$$

$$f_j = \begin{bmatrix} 0 \\ \frac{2\theta_2+2\theta_4+\theta_6}{m} \\ 0 \\ \frac{2a\theta_2-2b\theta_4+\theta_7}{J} \end{bmatrix}, \quad C_j = I_4, \quad D_j = O_4, \tag{8.21}$$

with the discrete parameter values

$$\theta_1 = \begin{cases} c_6, & \tilde{F}_{c,f} > 0.1c_5 \\ c_1(\exp(\frac{0.1c_2}{|\varepsilon|+c_3}) - 1), & |\tilde{F}_{c,f}| < 0.1c_5 \\ c_6, & \tilde{F}_{c,f} < -0.1c_5 \end{cases} \tag{8.22a}$$

$$\theta_2 = \begin{cases} 0.1c_4, & \tilde{F}_{c,f} > 0.1c_5 \\ 0, & |\tilde{F}_{c,f}| < 0.1c_5 \\ -0.1c_4, & \tilde{F}_{c,f} < -0.1c_5 \end{cases} \tag{8.22b}$$

$$\theta_3 = \begin{cases} k_4, & \tilde{F}_{c,r} > 0.1k_5 \\ k_1(\exp(0.1k_2) - 1), & |\tilde{F}_{c,r}| < 0.1k_5 \\ k_4, & \tilde{F}_{c,r} < -0.1k_5 \end{cases} \tag{8.22c}$$

$$\theta_4 = \begin{cases} 0.1k_3, & \tilde{F}_{c,r} > 0.1k_5 \\ 0, & |\tilde{F}_{c,r}| < 0.1k_5 \\ -0.1k_3, & \tilde{F}_{c,r} < -0.1k_5 \end{cases} \tag{8.22d}$$

$$\theta_5 = F_{l,f}(\alpha_f = 0), \tag{8.22e}$$

$$\theta_6 = 0, \quad \theta_7 = 0, \tag{8.22f}$$

and with the constants $c_1, \dots, c_5$ and $k_1, \dots, k_5$. In contrast to the nonlinear and the HPV model, online information about the current scheduling signal $\rho = [s_f \;\; \mu \;\; v_w]^T$ is not required by the hybrid model.

Linear Model

Finally, a linear model of the lateral dynamics is considered. This model can be obtained from the hybrid model (8.21) - (8.22) by neglecting the saturating effects of the tire forces. Subsequently no distinction of discrete modes is considered in (8.22) and the hybrid model simplifies to the *Linear Model*:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2(\theta_1+\theta_3)}{mv_x} & \frac{-2(\theta_1+\theta_3)}{m} & \frac{2(\theta_1 a-\theta_3 b)}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2(\theta_1 a-\theta_3 b)}{Jv_x} & \frac{-2(\theta_1 a-\theta_3 b)}{J} & \frac{2(\theta_1 a^2+\theta_3 b^2)}{Jv_x} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{2(\theta_5-\theta_1)}{m} \\ 0 \\ \frac{2a(\theta_5-\theta_1)}{J} \end{bmatrix},$$

$$f = \begin{bmatrix} 0 \\ \frac{2\theta_2+2\theta_4+\theta_6}{m} \\ 0 \\ \frac{2a\theta_2-2b\theta_4+\theta_7}{J} \end{bmatrix}, \quad C = I_4, \quad D = O_4, \tag{8.23}$$

with the fixed parameter values

$$\theta_1 = c_1(\exp(\frac{0.1c_2}{|\varepsilon| + c_3}) - 1), \tag{8.24a}$$
$$\theta_2 = 0, \tag{8.24b}$$
$$\theta_3 = k_1(\exp(0.1k_2) - 1), \tag{8.24c}$$
$$\theta_4 = 0, \tag{8.24d}$$
$$\theta_5 = F_{l,f}(\alpha_f = 0), \tag{8.24e}$$
$$\theta_6 = 0, \quad \theta_7 = 0 \tag{8.24f}$$

and with the constants $c_1, \dots, c_5$ and $k_1, \dots, k_5$.

8.3 Controller Design

In this section the design of four MPC controllers, using the discrete-time versions of the prediction models presented in Section 8.2, is described. Model predictive control was chosen due to its straightforward design methodology and its ability to deal with constraints, [GPM89]. For the simulation of the car in the control scenario, the nonlinear model (8.8) is used. The objective is to reduce the yaw angle ψ and the displacement Y of the car in a coordinate system fixed with respect to the road, i.e. to keep the car on the road and pointing in the desired direction.

A standard MPC approach is applied using the quadratic cost function,

$$J(\boldsymbol{U}_k; x_k, \rho_k) = \sum_{i=1}^{N} \|x_{k+i}\|_Q^2 + \sum_{i=0}^{N_c-1} \|\Delta u_{k+i}\|_R^2 ,$$

which penalizes the predicted state vector $x_k = \begin{bmatrix} Y_k & \dot{Y}_k & \psi_k & \dot{\psi}_k \end{bmatrix}^T \in \mathbb{R}^4$ and the predicted input rate $\Delta u_k = \Delta\delta_{f,k} \in \mathbb{R}$ at time $k \in \mathbb{N}$. For the ease of notation we introduce the sequence of optimization variables $\boldsymbol{U}_k := \{\Delta u_k, \Delta u_{k+1}, \ldots, \Delta u_{k+N_c-1}\}$. The scheduling variables ρ_k are assumed to stay constant over the prediction horizon.

The following controller parameters yield reasonable simulation results. A prediction horizon of $N = 5$ and a control horizon of $N_c = 2$ is chosen with a sampling time of $T_s = 0.05\,\mathrm{s}$. The weights are chosen as

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad R = 1 \,.$$

Using the above stated parameters, we can formulate the constrained finite-time optimal control (CFTOC) problem, which is solved at each time step k,

$$\min_{\boldsymbol{U}_k} \quad J(\boldsymbol{U}_k; x_k, \rho_k) \tag{8.26a}$$

$$\text{s.t.} \quad u_{\min} \leq u_{k+i} \leq u_{\max}, \quad i = 0, \ldots, N-1 \,, \tag{8.26b}$$

$$\Delta u_{\min} \leq \Delta u_{k+i} \leq \Delta u_{\max}, \quad i = 0, \ldots, N_c - 1 \,, \tag{8.26c}$$

$$u_{k+i} = u_{k+i-1} + \Delta u_{k+i}, \quad i = 0, \ldots, N_c - 1, \tag{8.26d}$$

$$u_{k+i} = u_{k+i-1}, \quad i = N_c, \ldots, N-1, \tag{8.26e}$$

$$x_{k+i+1} = f_{\mathrm{DYN}}(x_{k+i}, u_{k+i}, \rho_k), \quad i = 0, \ldots, N-1, \tag{8.26f}$$

denoting the particular prediction model by f_{DYN}. Note that a discretization procedure has to be used to derive this discrete-time model f_{DYN} from the corresponding continuous-time model in Section 8.2. We employ a *receding horizon* control strategy, i.e. only the first input $u_k = u_{k-1} + \Delta u_k$ is applied at time k, and another optimization is performed at the next time step.

Integral action is not regarded in the MPC schemes, because it mainly acts in a time scale where control should remain a task of the driver, [Ack01]. See Section 4.5 for more details on model predictive control.

It is a known issue in MPC that closed-loop stability is not guaranteed under the control law computed as the solution to the CFTOC problem (8.26). This is mainly due to the fact that we are optimizing over a limited time horizon, while the long-term effects of the control actions are not taken into account.

As discussed in Section 4.5, it is possible to extend the CFTOC formulation by including terminal set constraints, $x_{k+N} \in \mathcal{X}_T$ and adapt the terminal cost term. However, since augmentations of the CFTOC problem for ensuring stability usually lead to performance degradations, we did not include it in our formulation. A detailed treatment of stability in MPC lies out of the focus of this project and can be found e.g. in [MRRS00]. A recent application of an MPC scheme to Active Steering with guaranteed stability is in [FBT$^+$08].

Nonlinear Model Predictive Control

In the first MPC approach, the nonlinear model (8.8) is employed to predict the behaviour of the system. Since model predictive control utilizes discrete-time models, the nonlinear model (8.8) is discretized by an explicit Euler algorithm (whereby discretization errors are introduced into the predictions). The resulting nonlinear discrete-time model is inserted as $f_{\text{DYN}}(x_{k+i}, u_{k+i}, \rho_k)$ in the CFTOC problem (8.26). The nonlinear model requires the scheduling signal ρ to be measured or estimated by an underlying traction control system. Since the future scheduling signal values are unknown, the scheduling signal is assumed to stay constant, i.e. $\rho_k = \rho_{k+1} = \cdots = \rho_{k+N-1}$. At every time instance k of the nonlinear MPC algorithm, the following actions are executed:

1. Measure or estimate the current state x_k and the current values of the scheduling variables $\rho_k = \begin{bmatrix} s_{f,k} & \mu_k & v_{w,k} \end{bmatrix}^T$.
2. Solve the CFTOC problem (8.26) using the discrete-time counterpart of the nonlinear model (8.8) as prediction model $f_{\text{DYN}}(x_{k+i}, u_{k+i}, \rho_k)$. The optimization is performed by the commercial solver NPSOL, [GMSW].
3. Apply the optimal control action u_k^*.
4. Increment k.

Locally Linear Model Predictive Control

The second model employed for prediction is the hybrid parameter-varying model (8.19) – (8.20). Hereby the HPV model is used to determine a local linearization of the nonlinear system at each time step. Note that an online linearization of the nonlinear model at each time instance is not performed, but the HPV model is evaluated using the measured scheduling signal. In order to obtain a linear approximation, the scheduling signal ρ_k as well as the system matrices are assumed constant over the prediction horizon. More precisely, in every time step k of the locally linear MPC algorithm, the following actions are executed:

1. Measure or estimate the current state x_k and the current values of the scheduling variables $\rho_k = \begin{bmatrix} s_{f,k} & \mu_k & v_{w,k} \end{bmatrix}^T$.
2. Evaluate the system matrices $A_j(\theta_k), B_j(\theta_k)$ and $f_j(\theta_k)$ in (8.19) to obtain the current continuous-time linearization.
3. Discretize the current linearization with respect to time yielding the prediction model $f_{\text{DYN}}(x_{k+i}, u_{k+i}, \rho_k)$.
4. Solve the resulting CFTOC problem (8.26). By assuming the system matrices to stay constant over the prediction horizon, the CFTOC problem at hand is a quadratic program (QP). In order to solve this QP, the solver CPLEX was chosen, using the MATLAB interface YALMIP, [Löf04].
5. Apply the optimal control action u_k^*.
6. Increment k.

Explicit Hybrid Model Predictive Control

As a third controller candidate, an explicit hybrid model predictive controller was computed. For this purpose the hybrid model (8.21) – (8.22) was discretized using an explicit Euler algorithm. With a discrete-time hybrid model, the CFTOC problem (8.26) becomes a mixed-integer quadratic program (MIQP). As discussed in Section 5.2, it is possible to solve the MIQP parametrically in order to obtain an explicit controller which contains the optimal control law as a piecewise affine (PWA) function of the state, see also [BBBM05]. The explicit controller can be stored in a look-up table and allows a quick online access, thus leading to a reduction of computational effort. Instead of solving an optimization problem at each time step k, the look-up table is evaluated. No online information about the scheduling variables is needed. More information about explicit MPC can be found in Section 4.5.

The Multi-Parametric Toolbox (MPT) was used to solve the CFTOC problem (8.26) parametrically with the discrete-time counterpart of the hybrid model (8.21) – (8.22) as prediction model $f_{\text{DYN}}(x_{k+i}, u_{k+i})$, [KGBM04]. The obtained explicit hybrid MPC controller comprises 9669 regions.

Online, in every time step k of the explicit hybrid MPC algorithm, the following actions are executed:

1. Measure or estimate the current state x_k.
2. Evaluate the precomputed look-up table to obtain the optimal control action u_k^*.
3. Apply the optimal control action u_k^*.

4. Increment k.

Explicit Linear Model Predictive Control

Finally the linear model (8.23) – (8.24) from Section 8.2 was used for model predictive control. Again the model was discretized to obtain an discrete-time model for the predictions. With a linear discrete-time model, the CFTOC problem (8.26) becomes a QP, which can be solved parametrically since the system matrices are constant. Analogue to the explicit hybrid MPC approach, an explicit linear MPC controller comprising 145 regions was computed using the discrete-time counterpart of the linear discrete-time model (8.23) – (8.24). Online, the procedure in every time step k of the explicit linear MPC algorithm does not differ from the explicit hybrid MPC approach:

1. Measure or estimate the current state x_k.
2. Evaluate the precomputed look-up table to obtain the optimal control action u_k^*.
3. Apply the optimal control action u_k^*.
4. Increment k.

8.4 Side Wind Rejection Scenario

This section shows the simulation results for the four previously described model predictive controllers and compares the different methods in terms of performance and complexity. The problem considered here is the rejection of side wind disturbances acting on a vehicle.

In the control scenario proposed in [KFB⁺06], a car is driving with constant longitudinal speed $v_x = 15\,\mathrm{m/s}$ on a dry road. It will be assumed that a tracking control system ensures a constant longitudinal velocity even in different driving situations and under disturbances. After $t = 0.5\,\mathrm{s}$ a sidewind gust appears with $v_w = 10\,\mathrm{m/s}$ which acts as a disturbance on the vehicle. In this benchmark example the road is assumed to have an initial friction coefficient (adhesion) of $\mu = 0.9$. The road conditions change at $t = 2\,\mathrm{s}$ passing from a normal road to an icy road with $\mu = 0.1$, while the sidewind is still persistent.

The slip ratios are functions of the longitudinal dynamics, and they depend on the controllers in closed loop. However, in order to be able to focus on the lateral dynamics, the slip ratios are considered as external signals, and an underlying traction control system is assumed to produce the front slip

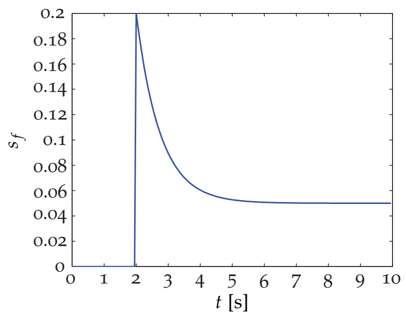


Figure 8.7: Slip ratio s_f of the front wheels during scenario.

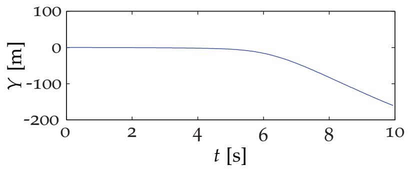


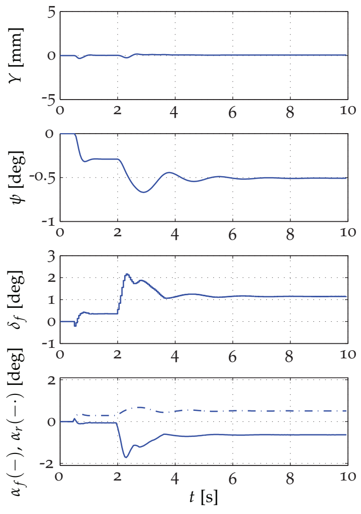
Figure 8.8: Lateral displacement Y in the open-loop simulation.

ratios shown in Figure 8.7, [KFB+06]. The slip of the rear wheels is assumed to be zero as front wheel steering is used and no braking occurs. Hence the vector of scheduling variables is given by $\rho(t) = [s_f, \ \ \mu, \ \ v_w]^T$.

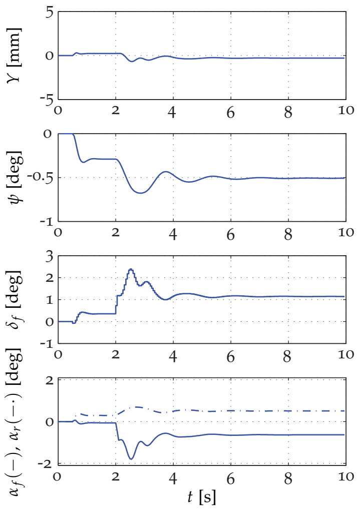
The control task is to mitigate lateral displacements of the vehicle due to the sidewind gust, and to stabilize the vehicle trajectory. The control input is the steering angle of the front wheels δ_f. As in [KFB+06], the input and the input rate are constrained to

$$u_{\text{max/min}} = \pm 3\,\text{deg}\,, \qquad\qquad \Delta u_{\text{max/min}} = \pm 25\,\text{deg/s}\,.$$

Figure 8.8 shows the lateral displacement of the vehicle for the simulated case of no reaction of the driver. Within seconds, the vehicle leaves the road and entirely changes its direction. Previous investigations have shown that a human driver has severe difficulties in stabilizing the car in this situation, when only front-wheel steering is available, [HAF00]. Even in the case of 4-wheel steering this might lead to dangerous situations. This shows the immanent need for additional steering control. In order to investigate the best performance achievable with MPC, the nonlinear model is employed for prediction in nonlinear model predictive control. In later stages the hybrid models and finally the linear model is used for prediction and its performance will be investigated and compared to the controller with the full nonlinear prediction model.



(a) NMPC.



(b) Locally linear MPC.

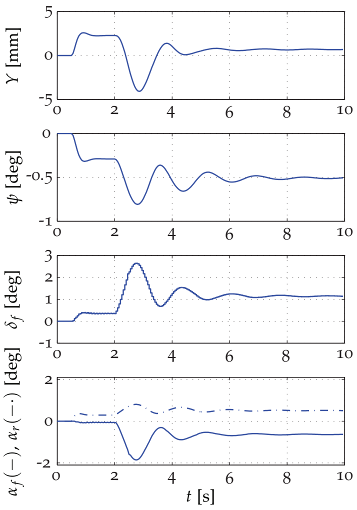
Figure 8.9: Simulation results for the side wind rejection scenario using nonlinear MPC and locally linear MPC.

Nonlinear Model Predictive Control

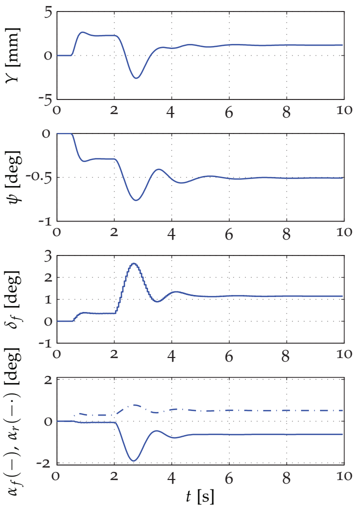
Figure 8.9(a) shows the simulation results with the nonlinear prediction model. The evolution of the displacement Y, the yaw angle ψ, the steering angle δ_f and the slip angles α_f and α_r during the side wind scenario are displayed. A peak deviation of merely 0.3 mm in the displacement Y can be noted. The nonlinear MPC scheme requires the solution of a nonlinear optimization problem at each time instance. The solution of these nonlinear optimization problems took 0.15 s on average and 0.38 s in the worst case with the commercial solver NPSOL, [GMSW], on a 3 GHz Pentium 4 processor. Similar results were obtained in [KFB$^+$06]. However, both algorithms take far too long for a real application, when a sampling time of 0.05 s is needed.

Locally Linear Model Predictive Control

In Figure 8.9(b), the simulation results with the locally linear MPC controller are visualized. Due to the parameter dependence of the HPV model, the absolute peak deviation can be kept below 0.7 mm. A steady-state offset of 0.3 mm must be noted.



(a) Explicit hybrid MPC.



(b) Explicit linear MPC.

Figure 8.10: Simulation results for the side wind rejection scenario using explicit hybrid MPC and explicit linear MPC.

The solution of the quadratic problem took on average 0.06 s and in the worst case 0.08 s on a 3 GHz Pentium 4 processor. This is still larger than the sampling time, but means a speedup by a factor 2 to 3 compared to the nonlinear MPC. It must be mentioned that the procedure was implemented using Matlab and that substantial improvements can be achieved with a more sophisticated implementation, e.g. in C.

Explicit Hybrid Model Predictive Control

The simulation results for the explicit hybrid MPC controller can be seen in Figure 8.10(a), where a maximal displacement of 4 mm, but also a steady-state error can be noted. While the nonlinear and the HPV model adapt to the values of the scheduling signal, the hybrid and the linear model are parameter independent. The advantage of parameter-dependent system matrices can be seen by comparing the displacement of the car during the first two seconds of the investigated scenario, while being controlled by the nonlinear/locally linearized MPC controller (Figure 8.9) vs. the hybrid/ linear MPC controller (Figure 8.10). During this part of the scenario the road friction coefficient is $\mu = 0.9$.

The nonlinear model and the hybrid parameter-varying model adapt to this value of the road friction coefficient, while the purely hybrid model does not. This adaptation leads to 6 times smaller displacements. By using the explicit hybrid MPC controller, the computational effort can be reduced even further. The evaluation of the lookup table facilitates a worst-case computation time of 0.02 s at each time instance. An implementation in a real car would be possible, and further reductions in the online computational times are conceivable with a more sophisticated implementation.

Explicit Linear Model Predictive Control

The simulation results for the explicit linear MPC controller can be seen in Figure 8.10(b). A peak deviation of 2.4 mm and a steady-state deviation of 1.2 mm can be noted. Using the presented linear and hybrid approximation of the nonlinear dynamics and the same specified horizons, the linear prediction model yielded a smaller peak deviation than the hybrid model, but a higher steady-state error. Due to the small number of regions, the computational time at each time instance was much less then $0.01\,s$ on 3 GHz Pentium 4 processor.

8.5 Double Lane-Change Manoeuver

After testing the four MPC schemes in the side wind rejection scenario, this section presents their application to autonomous vehicle steering in a double lane-change maneuver on snow, [KFB+06]. In this control scenario, a car is driving with constant longitudinal speed $v_x = 15\,\mathrm{m/s}$ on a snowy road ($\mu = 0.3$). The double lane change shall be proceeded by following a reference trajectory in Y and ψ, assuming that this reference is supplied by a trajectory planning system. The control objective is to track the reference trajectory as closely as possible. The vehicle is assumed to coast during the maneuver, no braking or accelerating is proceeded. These assumptions leave us with the scheduling variable $\rho(t) = \mu(t)$. The control input is again the steering angle of the front wheels δ_f, which is now constrained to

$$u_{\max/\min} = \pm 30\,\mathrm{deg}\,, \qquad \Delta u_{\max/\min} = \pm 20\,\mathrm{deg/s}\,.$$

During the double lane-change scenario, the small angle assumption ($\psi, \delta_f \ll 1$), which forms the basis for all approximation models, is not valid anymore. This led to a significant reduction of control performance and unnatural high slip angles α_f in first simulations when employing the approximation models. In order to prevent the system from leaving the area of acceptable model accuracy, the front slip angle α_f is constrained to $\alpha_{\max/\min} = \pm 2\,\mathrm{deg}$. This constraint is implemented as a *soft constraint*, meaning that α_f violating its constraints is not strictly prevented, but mitigated by resulting in an increase in the cost function. The advantage of using soft constraints is that the optimization problem still remains feasible in case the desired constraint can not be fulfilled. The soft constraint for α_f is also included in the nonlinear MPC to increase the comparability between the MPC schemes.

The basic structure of the CFTOC problem (8.26) is unchanged, while the cost function and the controller parameters do change. The cost function for the optimal control problem is chosen analogue to (8.25), with the difference that not the state itself, but its difference to the reference value is penalized. The resulting cost function of the CFTOC problem is then

$$J(\boldsymbol{U}_k; x_k, \rho_k, \boldsymbol{X}_{\mathrm{ref},k}) = \sum_{i=1}^{N} \|x_{k+i} - x_{\mathrm{ref},k+i}\|_Q^2 + \sum_{i=1}^{N-1} \|s_{k+i}\|_{q_s}^2 + \sum_{i=0}^{N_c-1} \|\Delta u_{k+i}\|_R^2 \tag{8.27}$$

with the sequence of reference values, $\boldsymbol{X}_{\mathrm{ref},k} = \{x_{\mathrm{ref},k+1}, \ldots, x_{\mathrm{ref},k+N}\}$, and the slack variables, $s_{k+i} \in \mathbb{R}$, $i = 1, \ldots, N-1$, for the inclusion of the soft constraints

$$\alpha_{\min} \leq \alpha_{f,k+i} + s_{k+i} \leq \alpha_{\max}, \quad i = 0, \ldots, N-1 \tag{8.28}$$

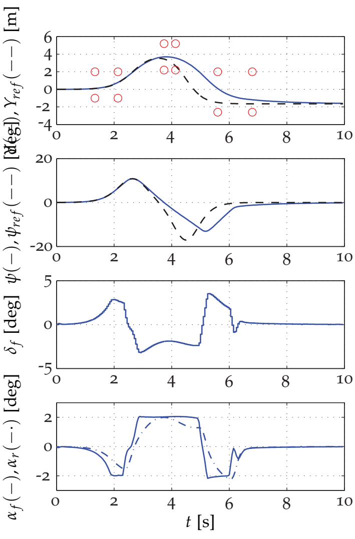
to the CFTOC problem (8.26).

The following control settings yield reasonable simulation results: Again a prediction horizon of $N = 4$ and a control horizon of $N_c = 2$ is chosen with a sampling time of $T_s = 0.05\,\mathrm{s}$. The weights are chosen as

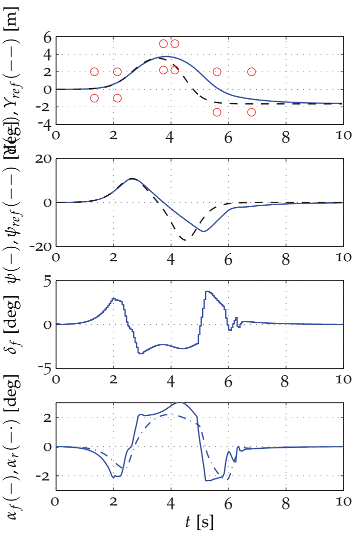
$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad q_s = 1000 \quad \text{and} \quad R = 1\,.$$

Nonlinear MPC

Besides the changes described above, the nonlinear MPC approach is the same as for the side-wind scenario. The online computation times of the nonlinear MPC in the double lane-change maneuver were 0.08 s on average, but with a worst case of 0.30 s. Figure 8.11(a) shows the simulation results: The maximal deviation from the reference trajectory is $\Delta Y_{\max} = 2.61\,\mathrm{m}$ and $\Delta\psi_{\max} = 10.2\,\mathrm{deg}$, respectively. The cumulated cost function, evaluated for the actual states and inputs during the simulation, amounts to $J_{\mathrm{cum}} = 183$.



(a) NMPC.



(b) Locally linear MPC.

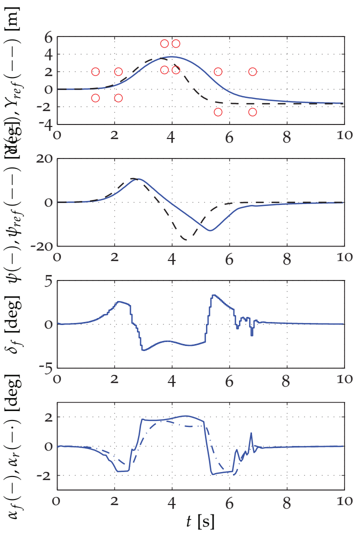
Figure 8.11: Simulation results during the double lane-change scenario using nonlinear MPC and locally linear MPC.

Locally Linear Model Predictive Control

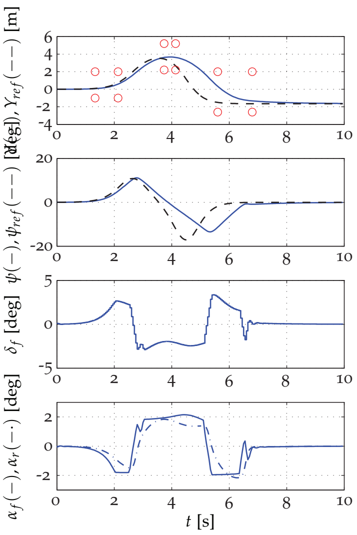
When employing the hybrid parameter-varying prediction model, the online computation time was 0.03 s on average and 0.06 s in the worst case. The maneuver under the locally linear MPC controller is presented in Figure 8.11(b). A violation of the soft constraints for the front slip angle appeared. As maximal deviations, $\Delta Y_{\max} = 2.64\,\mathrm{m}$ and $\Delta\psi_{\max} = 9.8\,\mathrm{deg}$ were observed, and the cumulated costs are $J_{\mathrm{cum}} = 195$.

Explicit Hybrid Model Predictive Control

While the online MPC schemes utilizing the nonlinear and the HPV model basically can be re-used for the double lane-change scenario (after adapting the cost function and the controller parameters), the explicit solutions to the optimization problems for the hybrid and the linear prediction models had to be recomputed. The explicit hybrid MPC for the double-lane change scenario comprises 24699 regions. The increase in the number of regions compared to the side wind rejection scenario is due to the fact that tracking is now included in the design, increasing the dimension of the optimization problem by the reference signals. The online computations are on average



(a) Explicit hybrid MPC.



(b) Explicit linear MPC.

Figure 8.12: Simulation results during the double lane-change scenario using explicit hybrid MPC and explicit linear MPC.

0.03 s and in the worst case less than 0.04 s, and the simulation results are shown in Figure 8.12(a). The maximal lateral deviation is $\Delta Y_{\max} = 3.08$ m and the maximal yaw deviation $\Delta\psi_{\max} = 12.1$ deg. The overall costs cumulate to $J_{\text{cum}} = 274$.

The hybrid MPC schemes obtained a 'delayed' version of the lateral position in the nonlinear MPC scheme. The reason for this delay is not only the model simplifications, but also different knowledge about the future reference trajectory: While the nonlinear and the HPV model account for the full information of the reference trajectory during the prediction horizon, the explicit MPC schemes utilize the values of the current reference and assume it to stay constant during the prediction. Therefore changes in future references are not predicted in the optimization and the reaction to reference changes is delayed. By including future references in the optimization problem, this drawback could be prevented, but to the price of an even higher dimension of the optimization problem and thus a much more complex control law.

Explicit Linear Model Predictive Control

The explicit controller for the linear prediction model comprises 1443 regions and results in an online computation time of less than 0.01 s on average as

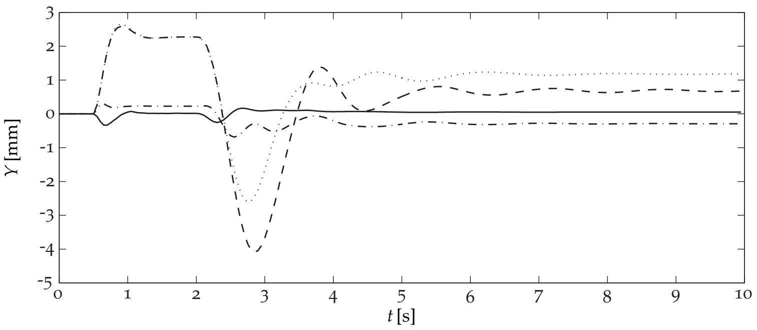


Figure 8.13: Vehicle position Y during side wind rejection scenario under model predictive control with nonlinear model (—), hybrid parameter-varying model ($- \cdot -$), hybrid model ($- -$) and linear model ($\cdots$).

well as in worst case. The results are shown in Figure 8.12(b). The maximal errors are $\Delta Y_{\max} = 2.94\,\mathrm{m}$ and $\Delta\psi_{\max} = 11.5\,\mathrm{deg}$. The cumulated costs are $J_{\mathrm{cum}} = 235$.

The hybrid MPC shows a somewhat surprisingly weak performance compared to the linear MPC. The difference between the two models is the saturation incorporated in the hybrid model. Both models are based on a road friction of $\mu = 0.1$, where saturation appears already at smaller forces than for the road friction of $\mu = 0.3$ used in the double lane-change maneuver. Hence taking the saturation effects into account led to a less accurate model for a road friction of $\mu = 0.3$ and in the occurred state values.

8.6 Conclusions and Outlook

Model Predictive Control with four different models has been applied for side wind rejection and in a double lane-change maneuver. In order to simplify the nonlinear model, a linear model, a purely hybrid model and a locally linear model based on a hybrid parameter-varying model have been proposed and verified in simulations.

Figure 8.13 shows the comparison of the displacements for all four prediction models during the side wind rejection scenario. The best results are obtained with a nonlinear model predictive control scheme, using the full nonlinear model. Due to the costs and time needed for solving the nonlinear optimization problem, this is hardly implementable on a real vehicle. The locally linear MPC scheme based on the hybrid parameter-varying model showed a tradeoff between satisfying control performance and complexity. Although still computationally demanding, its computation time was slightly larger

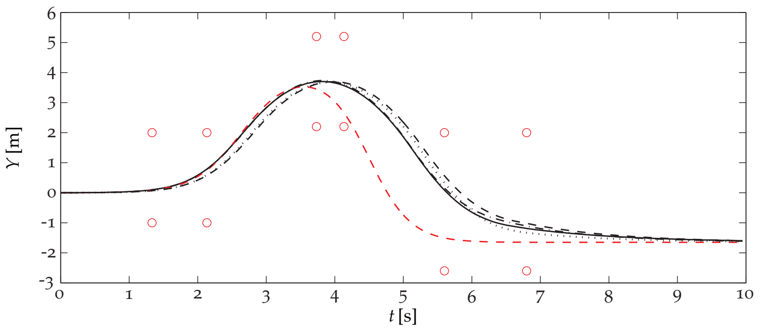


Figure 8.14: Vehicle position Y during double lane-change maneuver under model predictive control with nonlinear model (—), hybrid parameter-varying model (– · –), hybrid model (– –) and linear model (· · ·).

than the sampling time and the approach seems to be useable with a more sophisticated implementation. The locally linear MPC scheme yields a satisfying control performance, but still needs modern computing equipment. The purely hybrid and the linear models could achieve only worse performance within this comparison, but still stabilize the systems and lead to displacements of less than 5 mm. For both prediction models, an explicit solution was computed, which afterwards can be stored in a look-up table. The on-line computations are then reduced to evaluate this look-up table, which means a significant reduction of on-line computational effort.

For the double lane-change scenario, the controllers were enhanced by tracking of reference signals. Figure 8.14 shows the evolution of the lateral position during the maneuver for all four MPC schemes. Here basically the same observations as in the side wind rejection scenario could be made, in addition to the following remarks: Taking the saturation effects into account for a different road friction led to a performance loss of the hybrid MPC. Additionally, by including the reference values, the complexity of the hybrid MPC was increased such that an evaluation of the look-up table is nearly as time consuming as solving the optimization problem online.

In future research, the proposed control strategies should be tested in a real car application. Towards this goal the hybrid parameter-varying MPC scheme has to be implemented in a more efficient manner (e.g. in C) to be usable within the sampling time of 0.05 s. Thereby a non-constant vehicle speed should also be considered.

During the work on this project, one aspect became more and more obvious: The inclusion of a varying parameter in the system matrices of the HPV model improved the quality of the predictions substantially, and resulted in a better control performance than the hybrid and linear MPC approaches.

On the other hand, the use of a hybrid or linear model allows for the computation of explicit control laws by using standard parametric solvers (see Section 3.2 on parametric programming). A similar approach for parameter-varying systems did not exist, which motivated the main part of this dissertation: The computation of explicit model predictive control laws for linear parameter-varying systems.

Part III

Linear Parameter-Varying Systems

9 LPV Systems in Control

Gain scheduling
A method of eliminating influences of variations in the process dynamics of a control system by changing the parameters of the regulator as functions of auxiliary variables which correlate well with those dynamics.

The free dictionary

Linear parameter-varying (LPV) systems lie at the heart of this thesis. In this chapter we will review classical gain-scheduling approaches, which initiated the academic interest in the class of LPV systems. Then we will summarize important system properties, and examine existing approaches for control systems design using LPV models. Finally we will formulate the motivation for Part III of this thesis: High-speed (close-to-)optimal control of constrained LPV systems.

9.1 Introduction

In practice, the vast majority of systems to be controlled exhibits nonlinear behaviour. Control inputs are seldom unbounded, but saturate. States are required to be within certain limits. System dynamics may change rapidly, due to switches, logics, hysteresis, backlash or saturation; or smoothly with the state or the point of operation. Often system gains and parameters do depend on the amplitude of signals.

A classical approach to deal with nonlinearities is to linearize around an operating point, and compute a *linear controller* for the linearized model of the plant. While the benefits of applying linear control techniques are apparent, there is also a drawback in the described linearization approach. The linearized model of the nonlinear system is only accurate in a neighbourhood around the operating point, while the system might be operated in a larger part of the state space, with entirely different dynamics.

Gain scheduling is a direct extension of the linearization approach. As such, gain scheduling shares the central aspect of the linearization approach, i.e. the application of *linear control techniques* to control a nonlinear plant. In gain scheduling instead of one, a family of linear controllers is used to control the system. Each controller is designed for a linearized model at a specific operating point. *Scheduling variables* are used to determine what region the system is in and which controller, or which blend of controllers, is applied. The interpolation of the linear controllers obeys a function of the scheduling variables, the *scheduling function*. The resulting controllers have a linear structure, but its parameters depend in a nonlinear way on the scheduling variables, hence the name *linear parameter-varying*. Gain scheduling belongs to the class of adaptive control techniques, though no online system identification is involved (the scheduling can be considered as an adaptation of the linear model), [ÅW95].

Similar to the linearization approach, gain scheduling enjoys a broad acceptance in practice. The past has seen numerous successful applications, especially in the aerospace and automotive industries, [SL91, PSSU80]. This success is based upon the fact that gain scheduling – apart from being a straightforward control approach – typically yields a reasonably good control performance. The concentration on locally linear models during the controller design hinders the introduction of conservatism due to model uncertainties, as it is inherent e.g. in robust control techniques.

On the other hand, gain scheduling is mainly limited to slowly varying scheduling variables, [SA90, SA91]. The linear controllers are designed for locally accurate models, and fast variations of the scheduling variable can be problematic. Stability of the local linear models under the corresponding linear controllers does not imply stability of the nonlinear system under the interpolated gain-scheduling controller; and one can construct simple examples, where the mere variation between interpolations of two stable linear systems can result in instability, [ABGH96]. The limitation to 'sufficiently slowly' varying scheduling variables, and the lack of strong, precise guarantees motivated the development of more holistic approaches, considering not only local linear models, but making use of *linear parameter-varying models* of the nonlinear plant.

9.2 LPV Systems

The study of Linear Parameter-Varying (LPV) systems is motivated by their use in gain-scheduling control techniques for nonlinear systems, [SA90], [SA91], [AGB95]. Classical gain-scheduling approaches work with an interpolation of controller gains among a set of linear time-invariant (LTI) controllers, which are designed for linearized models of the system. While those gain-scheduling techniques work well in practice, it is hard to give precise stability/performance statements taking changes in the system dynamics into account. This drawback led to the development of the LPV gain-scheduling framework. LPV systems account for changes in the system dynamics by parameter-varying system matrices. The parameters lie in a bounded set, such that an LPV system describes a *family* of linear systems. In the LPV gain-scheduling framework, the interpolation is brought from the controller level to the modelling level, and controllers are designed for an entire LPV system. The LPV framework constitutes a useful theoretical foundation and allows statements on stability and performance which take variations of the scheduling parameter directly into account.

In the remainder of this work, we will consider discrete-time LPV systems in a *polytopic representation*. Polytopic descriptions are among the most common in the LPV framework, [AGB95]. Note however that there exists a series of other descriptions, e.g. input-output LPV representations, linear fractional representations, where the model depends rationally on the scheduling parameter, the behavioural description or kernel representations, [Tót08]. Moreover, those representations are not necessarily equivalent, [THVdH07].

Definition 9.1 (LPV systems) *Let $k \in \mathbb{Z}$ denote discrete time. We define the following* LPV systems*:*

$$x_{k+1} = A(\theta_k)x_k + B(\theta_k)u_k\,, \tag{9.1a}$$

with

$$A(\theta_k) = \sum_{j=1}^{n_\theta} A_j\theta_{k,j}\,, \qquad B(\theta_k) = \sum_{j=1}^{n_\theta} B_j\theta_{k,j}\,, \tag{9.1b}$$

and

$$\theta_k \in \Theta := \left\{ \theta_k \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_{k,j} = 1 \right\} . \tag{9.1c}$$

The variables $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $\theta_k \in \mathbb{R}^{n_\theta}$ denote the state, the control input, and the time-varying scheduling parameter, respectively. The system matrices $A(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_x}$ and $B(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_u}$ are known to lie in polytopes with the description (9.1b)*, where $A_j \in \mathbb{R}^{n_x \times n_x}$, $B_j \in \mathbb{R}^{n_x \times n_u}$ denote the jth vertices of the corresponding polytope. The scheduling parameter $\theta_k = [\,\theta_{k,1}\ \dots\ \theta_{k,n_\theta}\,]^T \in \mathbb{R}^{n_\theta}$ is constrained to the standard simplex in* (9.1c)*.*

We assume that the state x_k is either measurable or observable. The essential assumption behind LPV control is that the scheduling parameter is measured online and *known to the controller*, while future values are not known. This assumption is referred to as the LPV paradigm by some authors.

In the LPV framework there are three scenarios regarding the a-priori knowledge of future scheduling parameters:

($S1$) *The scheduling parameter varies arbitrarily fast.*
In this scenario no further information about future scheduling parameter values are given. This scenario was one of the main motivations for the development of LPV gain-scheduling approaches: Being able to give guarantees on the stability and achievable performance, *independent* of the movement of the scheduling parameter (inside some bounds). This scenario is considered in the control methods developed in Chapter 10, Chapter 11 and Chapter 13.

($S2$) *The scheduling parameter varies with a bounded rate of variation.*
In this scenario it is assumed that the scheduling parameter varies only with a limited speed. This knowledge can be utilized for the controller design to increase control performance. Control methods which assume this scenario are presented in Chapter 12.

($S3$) *The scheduling parameter varies slowly enough to be considered constant.*
In this scenario the parameter variations are much slower than the system dynamics. This scenario is a common assumption in classical gain-scheduling approaches. This scenario is a special case of ($S2$) and as such also considered in Chapter 12.

Stability Properties of LPV Systems

In the following we present some results concerning the stability and stabilizability of LPV systems. If not stated otherwise, these results are taken from [BM08b]. Initially we consider an arbitrarily fast varying scheduling parameter (Scenario ($S1$)). We start with the notion of quadratic stability of autonomous LPV systems. Note that in the case of autonomous LPV systems, there is no difference between the scheduling parameter θ_k and an uncertainty w_k, because there is no control input which could take advantage of knowledge of the scheduling parameter.

Theorem 9.1 (Quadratic stability, autonomous LPV system) *The autonomous LPV system*

$$x_{k+1} = A(\theta_k)x_k\,,$$

where the parameter-dependent matrix $A\colon \mathcal{D}_\theta \to \mathbb{R}^{n_x \times n_x}$ is a function with the domain $\mathcal{D}_\theta \in \mathbb{R}^{n_\theta}$, is quadratically stable, *if there exists a positive definite* Lyapunov matrix $P \succ 0$, *such that*

$$A(\theta)^T PA(\theta) - P \prec 0 \tag{9.2}$$

holds $\forall \theta \in \mathcal{D}_\theta$.

The conditions of Theorem 9.1 imply the existence of a quadratic Lyapunov function $V(x) = x^T Px$. The LMI (9.2) in Theorem 9.1 has to be satisfied for all possible values of the scheduling parameter, which might be difficult to verify in practice. Fortunately, the conditions simplify in the case of *autonomous polytopic LPV systems*:

$$x_{k+1} = A(\theta_k)x_k = \sum_{j=1}^{n_\theta} A_j \theta_{k,j} x_k\,, \tag{9.3a}$$

where

$$\theta_k \in \Theta := \left\{ \theta_k \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_{k,j} = 1 \right\}. \tag{9.3b}$$

For the class of autonomous polytopic LPV systems, quadratic stability is easier to verify due to the following theorem:

Theorem 9.2 (Quadratic stability, autonomous polytopic LPV system) *The autonomous polytopic LPV system* (9.3) *is* quadratically stable, *if there exists a positive definite Lyapunov matrix $P \succ 0$, such that*

$$A_j^T PA_j - P \prec 0\,, \quad j = 1, \ldots, n_\theta$$

holds.

Quadratic Lyapunov functions are a common vehicle to verify the stability of LPV systems. This is due to their simple structure (also in higher dimensions), and due to the possibility to verify quadratic stability relatively easy.

However, quadratic stability is only sufficient but not necessary for the stability of autonomous polytopic LPV systems. A necessary and sufficient for the asymptotic stability of autonomous polytopic LPV systems is the existence of a polyhedral Lyapunov function.

Theorem 9.3 (Polyhedral stability, autonomous polytopic LPV system) *Consider the autonomous polytopic LPV system* (9.3)*. The following statements are equivalent:*

1. *The autonomous polytopic LPV system is asymptotically stable.*
2. *The autonomous polytopic LPV system is exponentially stable.*
3. *The autonomous polytopic LPV system admits a polyhedral norm $\|Px\|_\infty$ as a Lyapunov function.*
4. *All vertex systems $x_{k+1} = A_j x_k$ of* (9.3) *share a common polyhedral Lyapunov function $\|Px\|_\infty$.*

After reviewing stability properties of autonomous polytopic LPV systems, we return to the initial class of interest, to LPV systems in a polytopic representation, (9.1). The next theorem follows from Theorem 9.1, and is a straightforward extension of a similar theorem in [BM08b].

Theorem 9.4 (Quadratic stabilizability, polytopic LPV system) *Consider the polytopic LPV system* (9.1)*. The LPV system is* quadratically stabilizable, *if there exists a matrix $Y \succ 0$ and a parameter-dependent matrix $F\colon \Theta \to \mathbb{R}^{n_u \times n_x}$, such that*

$$\begin{bmatrix} -Y & A(\theta)Y + B(\theta)F(\theta) \\ YA(\theta)^T + F(\theta)^T B(\theta)^T & -Y \end{bmatrix} \prec 0\,, \quad \forall \theta \in \Theta \tag{9.4}$$

holds.

Proof Assume the state feedback to be of the form $u_k = K(\theta_k)x_k$ with the parameter-dependent state feedback matrix $K\colon \Theta \to \mathbb{R}^{n_u \times n_x}$. Inserting the state feedback and the system equation (9.1) in (9.2) yields

$$(A(\theta) + B(\theta)K(\theta))^T P \left(A(\theta) + B(\theta)K(\theta)\right) - P \prec 0\,, \quad \forall \theta \in \Theta\,.$$

By pre- and post-multiplying this equation with $Y := P^{-1}$, and by substituting $F(\theta_k) := K(\theta_k)Y$, we obtain

$$(YA(\theta)^T + F(\theta)^T B(\theta)^T)Y^{-1}(A(\theta)Y + B(\theta)F(\theta) - Y \prec 0\,, \quad \forall \theta \in \Theta\,.$$

Applying Schur's complement formula completes the proof. ■

Similar to Theorem 9.1, it might be difficult to verify the LMI (9.4) for all scheduling parameter values in practice. There are two special cases, when this verification can be performed easily: the case of a constant input matrix B and an affine controller matrix $F(\theta_k) = \sum_{j=1}^{n_\theta} \theta_{k,j} F_j$, and the case of a robust (parameter-independent) controller F.

In both situations, the LMI (9.4) depends affinely on the scheduling parameter, and can thus be verified by considering the vertices only, i.e.

$$\begin{bmatrix} -Y & A_j Y + B F_j \\ Y A_j^T + F_j^T B^T & -Y \end{bmatrix} \prec 0\,, \quad j = 1, \ldots, n_\theta\,, \tag{9.5}$$

or

$$\begin{bmatrix} -Y & A_j Y + B_j F \\ Y A_j^T + F^T B_j^T & -Y \end{bmatrix} \prec 0\,, \quad j = 1, \ldots, n_\theta\,, \tag{9.6}$$

respectively.

If both the input matrix and the controller depend on the scheduling parameter, the vertex condition is only necessary, but not sufficient to guarantee constraint satisfaction over the entire parameter simplex Θ. In similar situations, gridding approaches were recommended to cope with LMIs polynomially depending on the scheduling parameter, [AA98]. One should be aware that such gridding approaches *per se* guarantee the desired properties only at the grid points, such that a guarantee for constraint satisfaction between the grid points requires additional measures.

A different approach is to use a sufficient condition for the satisfaction of LMIs such as (9.4), which depend polynomially on the scheduling parameter over the parameter simplex. This approach is facilitated by a matrix-valued version of Pólya's relaxation, [SH04], which is discussed in Section 2.2. By applying Pólya's relaxation to the LMI (9.4), we obtain sufficient conditions for quadratic stabilizability of (9.1) in the form of a parameter-independent LMI.

Quadratic stabilizability is only sufficient for stabilizability of an LPV system, and several approaches were proposed to mitigate the inherent conservatism. We will mention one extension to Theorem 9.4, which may be of use in the case of a limited rate of parameter variation (Scenarios $(S2)$ and $(S3)$). Instead of ensuring stabilizability by means of a quadratic Lyapunov function, one might use a parameter-dependent Lyapunov function of the form $V(x) = x^T P(\theta) x$, [AA98]. The gained freedom might indeed result in a reduction of conservatism, but is computationally more demanding, such that the authors of [AA98] resort to gridding approaches.

Relation to Common System Classes

How do other system classes relate to LPV systems? Many system classes can easily be embedded into the LPV framework:

- If the scheduling parameter is constant, $\theta_k = \bar{\theta},\ \forall k \in \mathbb{N}$, an LPV system is a *linear system*.
- It is a common misconception to equate LPV systems with *linear time-varying* (LTV) systems. The conceptual difference between both system classes lies in the possible behaviour. For each nonconstant scheduling parameter trajectory, an LPV system represents a specific LTV system, such that a LPV system contains a continuum of LTV systems, depending on the scheduling parameter θ. From an control engineer point of view, the values of the scheduling parameter trajectory typically are assumed to be known a-priori in the LTV framework, while they are assumed unknown a-priori in the LPV framework.
- *Bilinear systems* can be regarded as a special case of an LPV system, where the scheduling parameter equals either the input or the state of the system.
- With a small extension of Definition 9.1, namely the inclusion of a parameter-dependent affine term in the state-update equation (9.1a), LPV systems even comprise the class of *piecewise affine systems*. In return, piecewise affine systems can be regarded as special case of LPV systems, where the scheduling parameter only attains discrete values depending on the state and input of the system. The extension requires either the ability of the LPV control method to cope directly with an affine term, or an external treatment of the steady-state offset. Both approaches are possible with the procedures proposed in the following chapters.

9.3 LPV Modelling and Quasi-LPV Systems

This section is concerned with the embedding of nonlinear systems into the LPV framework. In this context, the concept of quasi-LPV systems is introduced.

In practice, the scheduling of the system matrices is often not affine as in Equation (9.1b), but there is a nonlinear dependence on some measurable scheduling variables $\rho_k \in \mathcal{P}_\rho \subseteq \mathbb{R}^{n_\rho}$. In order to obtain an LPV system of the form (9.1), the scheduling parameter θ_k is introduced, such that the system matrices depend affinely on θ_k, and such that a *scheduling function*

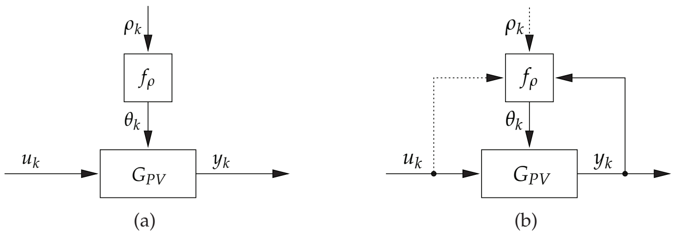


Figure 9.1: LPV system (a) and quasi-LPV system (b).

$f_\rho \colon \mathbb{R}^{n_\rho} \to \mathbb{R}^{n_\theta}$ describes the relation between the scheduling parameter and the scheduling variables,

$$\theta_k = f_\rho(\rho_k)\,. \tag{9.7}$$

In LPV controller design methods this scheduling function is usually ignored. Instead, a controller is designed which works for all $\theta_k \in \Theta$ independent of how θ_k was determined. The scheduling function, however, plays a significant role in LPV modelling. In this context, the definition of LPV systems (Definition 9.1) is sometimes extended by the scheduling function (9.7) and the set of possible scheduling variables $\mathcal{P}_\rho$,[Kwi07], and the LPV system is defined by the tuple

$$\big(A(\theta), B(\theta), f_\rho, \mathcal{P}_\rho\big)\,, \quad \text{with} \quad \theta = f_\rho(\rho),\ \rho \in \mathcal{P}_\rho\,. \tag{9.8}$$

In this definition of LPV systems, the scheduling variable ρ is considered to be an external variable. In the majority of instances however, the scheduling variable includes information of the output or even the input of the system. To distinguish those systems, the concept of *quasi-LPV systems* was established. Figure 9.1 depicts LPV systems with exclusively external scheduling variables and quasi-LPV systems with internal and external scheduling variables. The dotted lines indicate optional dependencies.

The quasi-LPV framework can be used to embed nonlinear systems. In this embedding process, nonlinearities in the plant dynamics are disguised as time-varying parameters, that are subsequently used as scheduling parameters. How to embed nonlinear systems into the quasi-LPV framework is a research topic on its own, because of two characteristics: non-uniqueness and overbounding.

Definition 9.2 (Equivalence of LPV systems) *Two LPV systems of the form* (9.8) *are called* equivalent *if the scheduling variables are limited to the same set $\mathcal{P}_\rho$, and if their system matrices are the same for all scheduling variables $\rho \in \mathcal{P}_\rho$, i.e.*

$$\big[A(f_\rho(\rho)) \quad B(f_\rho(\rho))\big] = \big[\tilde{A}(\tilde{f}_\rho(\rho)) \quad \tilde{B}(\tilde{f}_\rho(\rho))\big] \quad \forall \rho \in \mathcal{P}_\rho\,.$$

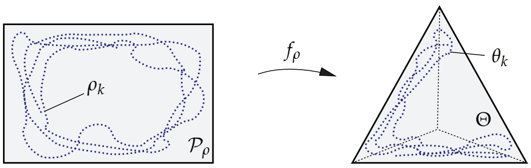


Figure 9.2: Example for overbounding.

Two LPV models can be equivalent, but at the same time they can be differently advantageous for control purposes. One reason is the so-called *overbounding* which means that the LPV model contains behaviour which is never actually put in execution by the nonlinear plant. In Figure 9.2 such a case of overbounding is displayed. On the left-hand side we see the scheduling variable ρ_k observed while the nonlinear plant is in operation. The scheduling parameter values corresponding to this scheduling trajectory are shown on the right-hand side: The parameter simplex Θ is not entirely occupied, some parts are never visited. Most LPV control approaches on the other hand are designed for the whole parameter simplex, such that overbounding introduces conservatism to the controller. Hence the objective in LPV modelling is to derive a model which contains as little overbounding as possible. In the ideal case the LPV model contains only behaviour which is possible with the nonlinear plant, in that case we call the LPV model *exact*.

Stability and Performance Properties of Quasi-LPV Systems

In the following we want to discuss stability and performance properties of quasi-LPV systems. Especially, we are interested in the question, if and how guarantees of stability and performance of an LPV system carry over to a nonlinear system embedded in the quasi-LPV framework.

Theorems 9.1 and 9.4 ensure stability or stabilizability, respectively, for *all* possible scheduling parameters $\theta \in \Theta$ of an LPV system (Scenario ($S1$)). Hereby it is not important, if the scheduling variables ρ, which determine the actual values of the scheduling parameter, are purely external or incorporate parts of the system state and the input. Similar statements can be made related to performance guarantees, or in the case of limitations to the rate of parameter variation. Consecutively, stability and performance guarantees of an LPV system do not differ from stability and performance guarantees of the corresponding quasi-LPV system. Moreover, they imply stability and performance of a nonlinear plant embedded in the quasi-LPV framework – under the assumption that the scheduling matrices of the quasi-LPV system represent the true dynamics of the nonlinear plant.

This assumption is not only technical, but has some important practical implications due to the boundedness of the parameter simplex (9.1c). In the following we will illustrate possible issues with the boundedness of the parameter simplex by means of two small example systems.

Example 9.1 *Consider the following nonlinear autonomous systems:*

$$x_{k+1} = 0.5\sin(x_k)x_k\,, \quad \textit{and} \quad x_{k+1} = 0.5x_k^2\,. \tag{9.9}$$

It is possible to embed both systems into the quasi-LPV framework by using the state x_k as scheduling variable ρ_k. This way we obtain the quasi-LPV system

$$x_{k+1} = A(\theta_k)x_k = (A_1\theta_{k,1} + A_2\theta_{k,2})\,x_k = (0.5\theta_{k,1} - 0.5\theta_{k,2})\,x_k\,, \tag{9.10}$$

with either of the scheduling functions,

$$\theta_k = \begin{bmatrix} 0.5 + 0.5\sin(x_k) \\ 0.5 - 0.5\sin(x_k) \end{bmatrix}, \quad \textit{and} \quad \theta_k = \begin{bmatrix} 0.5 + x_k \\ 0.5 - x_k \end{bmatrix}, \tag{9.11}$$

respectively. Application of Theorem 9.2 reveals that the quasi-LPV system (9.10) *admits the quadratic, radially unbounded Lyapunov function $V(x) = x^T x$, and is thus globally asymptotically stable. Note that the former scheduling function in* (9.11) *will deliver for all states $x_k \in \mathbb{R}^{n_x}$ an element of the parameter simplex, such that the quasi-LPV system together with the first scheduling function represents the true dynamics of the first nonlinear plant independent of the current state x_k. It follows that all stability and performance guarantees for the quasi-LPV system* (9.10) *carry over to the first nonlinear system* (9.9)*, such that we can infer that* (9.9) *is globally asymptotically stable. With the latter scheduling function on the other hand, the true dynamics of the second nonlinear plant are only represented by the quasi-LPV system for $x_k \in [-0.5, 0.5]$ (for which $\theta_k \in \Theta$), and we can* not *infer global asymptotic stability of the second nonlinear system, but only asymptotic stability within a certain domain of attraction.*

The previous example illustrates the impact of the boundedness of the scheduling function on the transfer of stability and performance properties from the quasi-LPV system to a nonlinear system. In the case of a bounded scheduling function, it is often possible to scale the scheduling function in order to obtain a subset of the parameter simplex as the image. Subsequently, we can infer global stability properties of the nonlinear system from the LPV system. On the other hand, if the scheduling function is unbounded, we can not have $\theta_k \in \Theta$ for all scheduling variables ρ. For consistency of the quasi-LPV system with the nonlinear system we have to respect additional constraints on the scheduling variable ρ, hence on the state or the input of the nonlinear system. *A systematic controller design should take the resulting state and input constraints directly into account*. In order to provide statements on the stability of the nonlinear system based on the quasi-LPV system, one has to verify the perpetual satisfaction of these constraints. Eventually the inducible stability and performance properties are limited to a certain domain of attraction.

One should also note that there is a gap between stability of a nonlinear system, and stability of a quasi-LPV system, since knowledge of the scheduling function f_ρ is ignored to determine stability of the latter. Therefore stability of the quasi-LPV system is only sufficient for stability of the nonlinear system. Furthermore quadratic stability of an LPV system is only sufficient for stability of the LPV system.

The distinction between LPV systems and quasi-LPV systems plays an important role for modelling and system identification, [Kwi07]. For the controller synthesis however it is common practice to *neglect* the dependencies on internal variables and to treat quasi-LPV systems not differently from LPV systems. Consequently for the remainder of this work we will use the term LPV systems for both LPV and quasi-LPV systems.

9.4 Control Systems Design Using LPV Models

After the introduction of LPV systems we investigate the steps which are needed to control a dynamical system using an LPV model. The complete design cycle to control a dynamical system contains the steps:

1. Modelling
2. Observer design
3. Controller synthesis

Finally the performance of the closed-loop system is assessed, potentially requiring another iteration of the design steps. In the following, not all of these steps will be covered in the depth they deserve. Nevertheless, the main aspects shall be mentioned and references to the literature shall be given.

LPV Modelling

For the application of a model-based control technique, a model of the dynamical system is required. Modelling and identification of nonlinear plants using LPV models has gained quite some interest in the past, resulting in a high number of publications. An overview about different identification methods and more references can be found in [Tót08]. Furthermore, a novel identification approach based on orthonormal basis functions is presented.

A different approach is to identify a nonlinear model or a set of local linear models, and to derive an LPV model from those. A survey on different ways to model an LPV system based e.g. on local linear models or on a full nonlinear model can be found in [Kwi07]. Moreover, this reference contains some tools to support the control engineer in the task of LPV modelling: An automated generation of affine LPV models from a nonlinear state-space model, the assessment of LPV models using a measure for overbounding, and the reduction of overbounding and of the number of scheduling parameters by a method called parameter set mapping which is based on a principal component analysis.

Some recent promising developments which are not covered in the references above are an identification approach using neural networks, [LAW08], an identification approach using instrumental variables, [AW09], and an optimization-based LPV approximation from a set of local linear models, [PL09].

Observer design for LPV Systems

In the polytopic LPV representation described in Section 9.2, we intentionally omitted the system output. The control methods we consider in this thesis are *state-feedback* methods and require knowledge of the state of the system. This might be a serious limitation and the extension of the proposed methods to support output feedback should be considered as a direction of future research. However, state feedback does typically not require that all states are measured; if the system is observable, state observers can be applied, [SM67]. Moreover, the past values of the scheduling parameter values are known, such that the state observation for LPV systems does not differ from the state observation for LTV systems, and classical techniques can be used. It is straightforward to modify a Luenberger observer or a Kalman filter to varying system matrices, [Bes07]. More specific material on the state observation and state estimation for LPV systems can be found in [BDRK00, DBK00, BM03].

A possible alternative is moving horizon estimation (MHE) which allows the incorporation of constraints in the estimation procedure. Details on moving horizon estimation for nonlinear systems can be found in the thesis from Rao, [Rao00], which is straightforward to adapt to LPV systems. However, this technique requires some computational power, similar to what is needed for model predictive control.

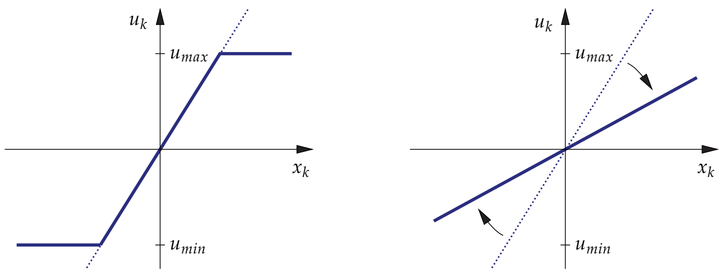
Controller Synthesis

The development of controller synthesis methods for LPV gain-scheduling followed the propagation of semidefinite programming (see Section 3.1) in the mid 1990s. Most of these gain-scheduling methods stem from frequency methods such as mixed sensitivity design and $\mathcal{H}_\infty$-optimal control for LTI systems and extend ideas from robust control of uncertain systems to the case of LPV systems. In difference to classical interpolation techniques, the interpolation is brought to the modelling level, allowing for a-priori statements related to stability and control performance. Stability is guaranteed by the determination of a quadratic, possibly parameter-varying, Lyapunov function for the closed-loop system. As a measure of performance the induced $\mathcal{L}_2$ gain between the exogenous inputs and some fictitious outputs of the closed-loop system is taken. The synthesis of stabilizing LPV gain-scheduling controllers can be formulated as semidefinite programs as was shown e.g. in [ABGH96, AG95].

Many publications followed investigating alternative formulations which extended and/or improved the LPV gain-scheduling approaches, among others [IS01, ZD98, Sch96, Sch01]. The resulting controller typically possesses the same model order as the generalized plant. In applications, however, the situation can occur when one wants to use a gain-scheduling controller with a fixed structure (e.g. a PID structure). The design of LPV gain-scheduling controllers of fixed structure forms the background of [CW06, FW04, FW06].

9.5 Control of Constrained LPV Systems

The previous section provided some aspects of the complete control system design cycle for LPV gain scheduling, i.e. modelling, observer design and controller synthesis. In the following we will investigate possible strategies in the case of significant constraints on the state and input. Since the modelling and the observer design can be adapted rather easily, we will focus solely on the controller synthesis. More precisely, the objective of the subsequent part of this thesis is *high-speed (close-to-) optimal control of constrained LPV systems*. We will approach this objective in three steps, considering successively the building blocks input constraints, output constraints and high-speed approaches.



(a) Saturate-and-Cut. (b) Detuning.

Figure 9.3: Strategies to deal with input constraints. Unconstrained control law ($\cdots$), modified control law (—).

Control of Input-Constrained LPV Systems

Constraints are one of the most often occurring nonlinearities in control, and as such they were already mentioned in the introduction of this chapter. In practice nearly every plant is affected by limitations on the inputs. Consequently many different strategies to deal with input constraints were developed, including:

- *Saturate-and-Clip*
  In this approach a controller is designed as for the unconstrained system. Whenever the control action of the unconstrained control exceeds the input limitations, it is replaced by the closest point in the set of admissible inputs. An example for such a saturated state-feedback control law is visualized in Figure 9.3(a) for a single-input system. The dotted line indicates the control law for the unconstrained system, while the solid line corresponds to the control law for the system with input constraints. In the case of a static state-feedback controller, this strategy often results in a reasonable, but not optimal control performance. There are cases however, in which the saturation leads to windup effects, possibly deteriorating the control performance or even resulting in an unstable closed-loop system.

- *Detuning*
  In this approach the aggressiveness of the unconstrained controller is reduced to avoid input saturation. This strategy is visualized in Figure 9.3(b), where the original control law is indicated by the dotted line, and the detuned control law by the solid line. There are two major drawbacks of detuning. (*i*) The detuned controller will violate the input constraints for large state values, resulting in a limited region in the state space, for which the detuned control input is admissible. (*ii*) The

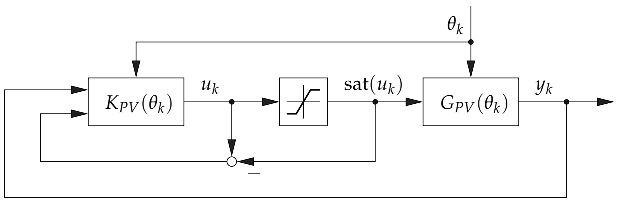


Figure 9.4: LPV anti-windup controller structure.

controller gains are also reduced near the origin, not only in regions where the input eventually saturates, thus reducing the overall control performance.

- *Anti-windup techniques*
  There are a number of other, more or less sophisticated, anti-windup techniques which deal with input constraints. The control structure of a common LPV anti-windup scheme is depicted in Figure 9.4. Note that in most anti-windup schemes, the anti-windup element only becomes active when the input saturates, $u_k \neq \mathrm{sat}(u_k)$. For an overview of practical anti-windup strategies see [GS03] or [KG02]. An anti-windup technique more specifically for LPV gain scheduling can be found in [WGP00]. In this approach, the input saturation is embedded in the quasi-LPV framework, such that a gain scheduling controller is computed, which takes the input saturation directly into account.

Control of Input- and Output-Constrained LPV Systems

An LPV system might not only be affected by input constraints, but also by state constraints. Constraints on the states of a system can either be imposed by the application at hand, or they are introduced somewhat artificially to enforce the state to stay in a region where the model reflects the dynamics of the system sufficiently well. As discussed in the section on quasi-LPV systems, the embedding of nonlinear systems into the LPV framework can lead to state and input constraints, which ensure the validity of the quasi-LPV model. Among the strengths of model predictive control (see Section 4.5) is the ability to take state and input constraints explicitly into account. The model predictive control community has considered LPV systems in the past, enabling finite-horizon optimal control under state and input constraints, [LA00b], [CFZ03].

Since the future scheduling parameter values are unknown, the underlying optimization is similar to robust MPC for uncertain systems, such that it is no surprise that most MPC approaches for LPV systems trace back to the robust MPC approach of Kothare et al. for uncertain systems, [KBM96]. In this approach a quadratic upper bound on the infinite-horizon cost was minimized by solving a semi-definite program at each sampling instance. The benefit of employing MPC was the incorporation of constraints, and the adaptation of the quadratic cost function to the current system state, enabling less conservative control compared to approaches utilizing a single constant quadratic cost function.

Lu and Arkun adapted this approach to the case of LPV systems, the so-called *Quasi-Min-Max MPC*, [LA00b]. At each sampling instance, after measuring the state x_k and the scheduling parameter θ_k, the following semidefinite program is solved:

$$\min_{\gamma, u_k, Y_k, F_k, X} \quad \gamma \tag{9.12a}$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & \star & \star & \star \\ A(\theta_k)x_k + B(\theta_k)u_k & Y_k & \star & \star \\ Q^{1/2}x_k & O & \gamma I & \star \\ R^{1/2}u_k & O & O & \gamma I \end{bmatrix} \succeq O \tag{9.12b}$$

$$|u_{k,j}| \leq u_{j,max}, \quad j = 1, \dots, n_u \tag{9.12c}$$

$$\|C(A(\theta_k)x_k + B(\theta_k)u_k)\|_2 \leq y_{max} \tag{9.12d}$$

$$\begin{bmatrix} Y_k & \star & \star & \star \\ A_jY_k + B_jF_k & Y_k & \star & \star \\ Q^{1/2}Y_k & O & \gamma I & \star \\ R^{1/2}F_k & O & O & \gamma I \end{bmatrix} \succeq O, \quad j = 1, \dots, n_\theta \tag{9.12e}$$

$$\begin{bmatrix} X & \star \\ F_k^T & Y_k \end{bmatrix} \succeq O, \quad \text{with} \quad X_{jj} \leq u_{j,max}^2, \quad j = 1, \dots, n_u \tag{9.12f}$$

$$\begin{bmatrix} Y_k & \star \\ C(A_jY_k + B_jF_k) & y_{max}^2 \end{bmatrix} \succeq O, \quad j = 1, \dots, n_\theta \tag{9.12g}$$

The main idea behind Quasi-Min-Max MPC is to decompose the quadratic infinite-horizon cost function into current and future costs. The future costs are approximated from above by a quadratic function,

$$V_k(x_{k+1}) = x_{k+1}^T P_k x_{k+1}\,, \tag{9.13}$$

where $P_k = \gamma Y_k^{-1}$; and the future control inputs are assumed to be *parameter-independent* state-feedback control laws,

$$u_{k+i} = K_k x_{k+i}\,, \quad i = 1, \dots, N-1\,, \tag{9.14}$$

where $F_k = K_k Y_k$. The constraints (9.12e) - (9.12g) ensure

- the decrease of the quadratic upper bound (9.13) by some stage costs,

$$V_k(x_{k+i+1}) - V_k(x_{k+i}) \leq -x_{k+i}^T Q x_{k+i} - u_{k+i}^T R u_{k+i} \,, \tag{9.15}$$

- the satisfaction of the input constraints $|u_{k+i,j}| \leq u_{j,max}$,
- the satisfaction of output constraints $\|y_{k+i+1}\|_2 \leq y_{max}$,

at each time step $i \geq 1$ independent of the future scheduling parameter values. Hereby the invariance of the ellipsoid $\mathcal{E} = \{z \mid z^T P z \leq \gamma\}$ is exploited in order to infer satisfaction of input and output constraints for all times from the satisfaction for all $z \in \mathcal{E}$. The constraints (9.12b) - (9.12d) ensure the minimization of the quadratic upper bound on the infinite-horizon cost,

$$J(u_k; x_k, \theta_k) = x_k^T Q x_k + u_k^T R u_k + V_k(x_{k+1}) \,, \tag{9.16}$$

while respecting the constraints on the current input u_k and the output constraints at step $k+1$.

Remark 9.1 *Note that the predicted future state-feedback control laws* (9.14) *are parameter-independent, i.e. robust and not scheduling. In the case of an LPV-A system (an LPV system, where the input matrix B does not depend on the scheduling parameter), the constraints of* (9.12) *can be adapted to facilitate* parameter-dependent *state-feedback control laws:* $u_{k+i} = \sum_{j=1}^{n_\theta} \theta_{k+i,j} K_{k,j} x_{k+i}, \quad i = 1, \ldots, N-1$.

Several extensions followed, incorporating a bounded rate of parameter variation, finite-horizon predictions, closed-loop state-feedback control laws and /or a parameter-varying terminal weighting matrix, [LA00a], [CFF02], [PJ04], [LW06], [SS06]. The common trait amongst these references is the minimization of a quadratic upper bound on the infinite-horizon cost and the solution of semi-definite programs in each iteration. This is often accompanied by simplifications such as the restriction to input constraints, a parameter-independent input matrix and/or the use of parameter-independent state feedback laws in the predictions.

High-Speed Control of Constrained LPV Systems

The main drawback of MPC is, in general, the computational effort needed to solve the optimization problem at each sampling instance. This effort can prevent the application of MPC to systems with a high sampling rate, or at least make such an application expensive, since the necessary computational equipment has to be provided. Quasi-Min-Max MPC requires the solution of a semidefinite program in each sampling interval, such that its application is limited to systems with sufficiently slow dynamics. In the subsequent chapters we will propose novel methods based on explicit MPC, which enable (close-to-)optimal control of constrained LPV systems with a high sampling rate.

Before we start with the explicit MPC schemes for LPV systems, we present alternative solution strategies to the stated problem. These strategies share the property of being straightforward workarounds, but also inherit different disadvantages. Nevertheless, due to their simplicity, they might be of use for the control practitioner and shall be covered here for the sake of completeness.

Approach 1: Grid and Interpolate

Whenever there is a complex nonlinear function to be approximated, one of the first tools of the engineer is a gridding and interpolation approach. Also the MPC control law for LPV systems can be regarded as such a nonlinear function of the state and the scheduling parameter. A straightforward approach is thus, similar to classical gain scheduling, to grid the parameter simplex, and compute an explicit control law for each grid point, assuming the parameter to stay constant over the prediction horizon. Online we measure the scheduling parameter, evaluate the neighbouring look-up tables and interpolate the neighbouring grid-point solutions.

The benefits of this approach are that at each grid point we are dealing with a linear system instead of an LPV system, simplifying the computation of the explicit solutions. Linear systems allow for a greater flexibility in the cost function, we are not restricted to polyhedral norms, but can also employ quadratic cost functions. The selection of grid points can be adapted to the needs of the problem at hand. It is obvious that this approach is especially beneficial in Scenario ($S3$) when the scheduling parameter varies only marginally over the prediction horizon. But it is also possible to adapt the gridding method proposed above for Scenario ($S1$), by employing a dynamic programming approach.

Nevertheless, there are no guarantees for stability, constraint satisfaction or control performance in between grid points. Those properties have to be verified by extensive simulations or an a-posteriori analysis. Therefore this approach can be regarded as an analogon to classical gain scheduling for the case of constrained parameter-varying systems.

Approach 2: Treat parameter as uncertainty

Another solution approach is to *ignore* the scheduling parameter information and to treat the scheduling parameter as a parametric uncertainty. This allows us to solve the closed-loop constrained robust control (CL-CROC) problem from Section 4.2 explicitly, and use the resulting explicit robust control law to control the LPV system, [BBM03].

The benefits are that there is no loss of guarantees, i.e. if robust stability and robust performance could be verified for the uncertain system, they also hold for the LPV system. Furthermore, no knowledge of the scheduling parameter is required, possibly saving sensing equipment.

One objective of the numerical examples in the following chapters was to justify the utilization of the scheduling parameter in the control system. Therefore in some of the examples the robust approach was positioned as a competitor to the proposed control schemes. As will be seen, the effects of ignoring scheduling parameter information can include a performance degradation and/or a smaller feasible set. More importantly, there are systems which are stabilizable by taking the scheduling parameter into account, but which are not stabilizable with a robust approach.

10 Optimal Control of Constrained LPV-A Systems

'If you can't solve a problem, then there is an easier problem you can solve: find it.'

George Pólya

Before considering the general class of LPV systems, we restrict ourselves to so-called LPV-A systems, linear discrete-time systems with a parameter-varying state transition matrix. LPV-A systems are a subclass of LPV systems where the input matrix is constant. The reason for considering LPV-A systems separately is that they allow for a simpler computation of the state-feedback control laws, and, as will be shown later, the considered closed-loop MPC problem can be solved optimally. We propose a closed-loop max-min MPC algorithm based on dynamic programming, to compute the explicit solution of the constrained parameter-varying optimal control (CPVOC) problem. The considered approach enables the controller to exploit parameter information to improve performance compared to a standard robust approach where no uncertainty knowledge is used, while keeping the benefits of fast online computations. The off-line computational burden is similar to what is required for computing explicit control laws for uncertain or nominal LTI systems.

10.1 Problem Statement

The class of systems we consider is the class of LPV-A systems, linear discrete-time systems with parameter-varying system matrices, which are defined by the state-update equation

$$x_{k+1} = A(\theta_k)x_k + Bu_k\,, \tag{10.1a}$$

with

$$A(\theta_k) = \sum_{j=1}^{n_\theta} A_j \theta_{k,j}\,, \tag{10.1b}$$

and

$$\theta_k \in \Theta := \left\{ \theta_k \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_{k,j} = 1 \right\}. \tag{10.1c}$$

The discrete time is denoted by $k \in \mathbb{Z}$, whereas the variables $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $\theta_k \in \mathbb{R}^{n_\theta}$ denote the state, the control input, and the time-varying scheduling parameter, respectively. We assume that the state x_k is either measurable or observable. The system matrix $A(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_x}$ is known to lie in a polytope with the description (10.1b), where $A_j \in \mathbb{R}^{n_x \times n_x}, B_j \in \mathbb{R}^{n_x \times n_u}$ denote the jth vertices of the corresponding polytope. The scheduling parameter $\theta_k = [\theta_{k,1} \;\; \dots \;\; \theta_{k,n_\theta}]^T \in \mathbb{R}^{n_\theta}$ is constrained to the standard simplex (10.1c). This polytopic description is a common assumption in the LPV framework, see e.g. [AGB95].

Furthermore, the LPV-A system (10.1) is constrained, $u_k \in \mathcal{U}$ and $x_k \in \mathcal{X}$. The constraint sets $\mathcal{U}$ and $\mathcal{X}$ are assumed to be bounded polyhedra,

$$u_k \;\in\; \mathcal{U} = \{u_k \in \mathbb{R}^{n_u} \mid H_u u_k \le \mathbf{1}\}\,, \tag{10.2a}$$
$$x_k \;\in\; \mathcal{X} = \{x_k \in \mathbb{R}^{n_x} \mid H_x x_k \le \mathbf{1}\}\,, \tag{10.2b}$$

which contain the origin in their interiors, since we are interested in the regulator problem.

Remark 10.1 *For ease of notation, we restrict ourselves to separate constraints on the state and inputs in* (10.2)*. It is straightforward to modify the presented algorithm in this chapter to the case of mixed polytopic constraints, i.e.* $E_x x + E_u u \le f_{xu}$.

Remark 10.2 *The proposed procedure in this chapter can easily be extended to systems, where the scheduling parameter lives in arbitrary polytopes instead of the standard simplex* (10.1c)*. Nonlinear dependencies of the system matrices on the scheduling signal can be embedded in a conservative way into the shown framework by defining* Θ *appropriately.*

Remark 10.3 *LPV systems with varying input matrix* $B(\theta)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_u}$ *are the subject of Chapter 11. Note that LPV systems with varying input matrix*

*$B(\theta)\colon \mathbb{R}^{n_\theta} \rightarrow \mathbb{R}^{n_x \times n_u}$ can also be reformulated to LPV-A systems by the Δu-*formulation*, [Bar83, BMS07],*

$$\begin{pmatrix} x_{k+1} \\ u_{k+1} \end{pmatrix} = \begin{bmatrix} A(\theta_k) & B(\theta_k) \\ 0 & I \end{bmatrix} \begin{pmatrix} x_k \\ u_k \end{pmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \Delta u_k \,. \tag{10.3}$$

The price to pay is an increase in the system dimension by the number of inputs, and the introduction of an input delay which counteracts the idea of the input to depend on the current parameter. But the Δu-formulation also exhibits advantages as the possibility to constrain and/or penalize input variations, and zero steady-state errors when tracking reference steps.

The essential assumption behind LPV control is that the scheduling parameter is measured online and *known to the controller*. Future values are however only known to be constrained to the standard simplex. In this setting, we want to compute an explicit state feedback control law

$$u_k = \mu(x_k, \theta_k)\,, \tag{10.4}$$

for the described class of LPV-A systems, which makes use of the information of the current θ_k. This assumption is referred to as the LPV paradigm by some authors. For the control problem to make sense, it is assumed that system (10.1) is controllable and observable for all admissible θ_k, [SM67, BBS03].

To compute the control law (10.4) within a model predictive control scheme, a finite-horizon cost function is to be minimized. According to standard MPC, our cost function is defined as

$$J(\boldsymbol{\pi}_N; x_k, \theta_k, \boldsymbol{T}_k) = \|Px_{k+N}\|_p + \sum_{i=0}^{N-1} \|Qx_{k+i}\|_p + \|Ru_{k+i}\|_p\,, \tag{10.5}$$

where p denotes a polyhedral norm, e.g. the 1-norm or the ∞-norm. Polyhedral norms enable a parametric solution to the stated problem using dynamic programming. For the minimization of the cost function (10.5) we have to consider the current as well as the unknown future parameter values, as the state trajectories are parameter-dependent. We will optimize over finite-horizon *control policies*

$$\boldsymbol{\pi}_N := \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$$

under the influence of the unknown *sequence of future scheduling parameters*

$$\boldsymbol{T}_k := \{\theta_{k+1}, \ldots, \theta_{k+N-1}\}\ .$$

In a closed-loop MPC approach (cf. Section 4.3), one assumes that the future control action u_{k+1} is calculated optimally over the horizon $N-1$ not until x_{k+1} and θ_{k+1} are available. But as the future values of the scheduling parameters are unknown, all possible cases must be considered in order to accommodate for the worst-case scenario. This way it is assured that the actual cost function will be less or equal to the computed one, no matter how the scheduling parameters evolve. The optimization problem to solve in a closed-loop MPC approach is thus

$$\mu^*(x_k,\theta_k) = \arg\min_{\mu_0}\max_{\theta_{k+1}}\min_{\mu_1}\cdots\max_{\theta_{k+N-1}}\min_{\mu_{N-1}} J(\boldsymbol{\pi}_N,\boldsymbol{T}_k;x_k,\theta_k) \tag{10.6a}$$

$$\begin{aligned} \text{s.t.}\quad & \forall i\in\{0,\dots,N-1\} \\ & x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B\mu_i(x_{k+i},\theta_{k+i})\,, && (10.6b)\\ & \mu_i(x_{k+i},\theta_{k+i}) \in \mathcal{U}\,, && (10.6c)\\ & x_{k+i}\in\mathcal{X}\,, && (10.6d)\\ & \theta_{k+i}\in\Theta\,. && (10.6e) \end{aligned}$$

Remark 10.4 *It is straightforward to add terminal state constraints $x_{k+N}\in\mathcal{X}_T$ to the optimization problem* (10.6) *in order to obtain a-priori stability guarantees. However, for complexity reasons we recommend not to use terminal state constraints, but to verify stability a-posteriori (for more details on stability guarantees see the discussion in Section 10.3).*

10.2 Computation of Explicit MPC Controllers

Here we propose a *dynamic programming* (DP) procedure to solve (10.6) by iterating backwards in time. For more details on dynamic programming, see Section 3.3. We start at the prediction horizon N with the initial cost function

$$J_N^*(x_{k+N}) = \|Px_{k+N}\|_p\,. \tag{10.7}$$

We iterate backwards in time, with the iteration index i decreasing from $N-1$ to 1. At each iteration of the dynamic programming procedure we solve the parametric optimization problem

$$J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \max_{\theta_{k+i}}\min_{\mu_i}\|R\mu_i(x_{k+i},\theta_{k+i})\|_p + J_{i+1}^*(x_{k+i+1}) \tag{10.8a}$$

$$\begin{aligned} \text{s.t.}\quad & x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B\mu_i(x_{k+i},\theta_{k+i})\,, && (10.8b)\\ & \mu_i(x_{k+i},\theta_{k+i}) \in \mathcal{U}\quad \forall\,\theta_{k+i}\in\Theta\,, && (10.8c)\\ & x_{k+i+1}\in\mathcal{X}_{i+1}\quad \forall\,\theta_{k+i}\in\Theta\,, && (10.8d) \end{aligned}$$

$$x_{k+i} \in \mathcal{X}\,, \tag{10.8e}$$
$$\theta_{k+i} \in \Theta\,, \tag{10.8f}$$

where the polytopic set $\mathcal{X}_{i+1}$ denotes all states for which the optimal value function J^*_{i+1} of the previous DP iteration is finite.

The constraint (10.8e) on the parameter of the parametric optimization problem ensures the satisfaction of the state constraints (10.2b). For the parametric optimization problem at hand it is not a constraint on the optimization variable, but determines the set of states where a solution is wanted.

Theorem 10.1 (Solution properties of the DP iterations) *Consider the parametric optimization problem* (10.8). *The following statements hold:*

(i) The set of feasible states $\mathcal{X}_i$ is a closed polyhedral set in $\mathbb{R}^{n_x}$, and $\mathcal{X}_i$ is partitioned into polyhedral critical regions.

(ii) The optimal value function $J^*_i(x_{k+i})$ is continuous, convex and piecewise affine over $\mathcal{X}_i$, and affine in each critical region $\mathcal{R}$.

(iii) The optimal solution $\mu^*_i(x_{k+i}, \theta_{k+i})$ is a continuous piecewise affine function of the state x_{k+i} and an affine function of the scheduling parameter θ_{k+i}, i.e. of the form

$$\mu^*_i(x_{k+i}, \theta_{k+i}) = \sum_{j=1}^{n_\theta} \theta_{k+i,j}\mu^*_{i,j}(x_{k+i}) \tag{10.9}$$

with $\mu^*_{i,j}(x_{k+i})$ being continuous and polyhedral piecewise affine over $\mathcal{X}_i$.

Remark 10.5 *If one denotes the parameter dependence of the optimal solution linear or affine is a matter of perspective. While* (10.9) *by itself is clearly a linear function in θ_{k+i}, the scheduling parameters are restricted to the parameter simplex Θ, which does not contain the origin. Considering that the elements of θ_{k+i} are not independent, we can substitute one element of θ_{k+i} via $\sum_{j=1}^{n_\theta} \theta_{k+i,j} = 1$ in Equation* (10.9) *and obtain a function affine in the remaining elements of θ_{k+i}.*

In the following we will proof Theorem 10.1 by first showing that (10.9) is the optimal input parametrization, and then reformulating the parametric optimization problem (10.8) into a parametric linear program, whose solution properties were stated in Theorem 3.1.

Proof For the time being assume that the set $\mathcal{X}_{i+1}$ is a polyhedron and the optimal cost function of the previous DP iteration J^*_{i+1} is a continuous, convex and polyhedral piecewise affine function. Introducing the epigraph variable t_i, we apply an epigraph reformulation,

$$\begin{aligned} J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \max_{\theta_{k+i}} \min_{\{\mu_i,t_i\}} & \ t_i(x_{k+i},\theta_{k+i}) && (10.10a)\\ \text{s.t.}\quad & x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B\mu_i(x_{k+i},\theta_{k+i})\,, && (10.10b)\\ & \|R\mu_i(x_{k+i},\theta_{k+i})\|_p + J_{i+1}^*(x_{k+i+1}) \le t_i(x_{k+i},\theta_{k+i}) \quad \forall\, \theta_{k+i} \in \Theta\,, && (10.10c)\\ & \mu_i(x_{k+i},\theta_{k+i}) \in \mathcal{U} \quad \forall\, \theta_{k+i} \in \Theta\,, && (10.10d)\\ & x_{k+i+1} \in \mathcal{X}_{i+1} \quad \forall\, \theta_{k+i} \in \Theta\,, && (10.10e)\\ & x_{k+i} \in \mathcal{X}\,, && (10.10f)\\ & \theta_{k+i} \in \Theta\,. && (10.10g) \end{aligned}$$

Note that the constraints (10.10c) - (10.10e) describe a polyhedron in the space spanned by μ_i, t_i and x_{k+i+1}. Hence those constraints can be rewritten by a number of polytopic inequalities. By inserting the state-update equation to replace x_{k+i+1} in J_{i+1}^*, we can rewrite the optimization problem as

$$\begin{aligned} J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \max_{\theta_{k+i}} \min_{\{\mu_i,t_i\}} & \ t_i(x_{k+i},\theta_{k+i}) && (10.11a)\\ \text{s.t.}\quad & C_{\mu t}\begin{bmatrix}\mu_i(x_{k+i},\theta_{k+i})\\ t_i(x_{k+i},\theta_{k+i})\end{bmatrix} \le c - C_x\left(\sum_{j=1}^{n_\theta} A_j\theta_{k+i,j}x_{k+i}\right) \quad \forall\, \theta_{k+i} \in \Theta\,, && (10.11b)\\ & x_{k+i} \in \mathcal{X}\,, && (10.11c)\\ & \theta_{k+i} \in \Theta\,, && (10.11d) \end{aligned}$$

with $C_{\mu t}, C_x$ and c being matrices and a vector of appropriate dimensions describing the hyperplanes of this polyhedron.

When solving the optimization problem (10.11) parametrically, we are only interested in regions which are full-dimensional with respect to the state x_{k+i}, i.e. where the constraints (10.11c) are inactive. Thus the set of active constraints (in the following denoted by the subscript $\boldsymbol{A}$) in each controller region will be a subset of the constraints (10.11b), and we can conclude on the structure of the optimization variables

$$\begin{aligned} \begin{bmatrix}\mu_i(x_{k+i},\theta_{k+i})\\ t_i(x_{k+i},\theta_{k+i})\end{bmatrix} &= \left(C_{\mu t,\boldsymbol{A}}\right)^{-1}\left(c_{\boldsymbol{A}} - C_{x,\boldsymbol{A}}\left(\sum_{j=1}^{n_\theta} A_j\theta_{k+i,j}x_{k+i}\right)\right)\\ &= \sum_{j=1}^{n_\theta}\theta_{k+i,j}\left(\left(C_{\mu t,\boldsymbol{A}}\right)^{-1}\left(c_{\boldsymbol{A}} - C_{x,\boldsymbol{A}}A_jx_{k+i}\right)\right)\\ &=: \sum_{j=1}^{n_\theta}\theta_{k+i,j}\begin{bmatrix}\mu_{i,j}(x_{k+i})\\ t_{i,j}(x_{k+i})\end{bmatrix}\,. && (10.12) \end{aligned}$$

This equation shows that $\mu_i(x_{k+i}, \theta_{k+i})$ and $t_i(x_{k+i}, \theta_{k+i})$ depend affinely on the scheduling parameter in each controller region. Hence we can conclude that the affine input parametrization (10.12) is the optimal input parametrization for the optimization problem (10.11). Next we insert this input parametrization (10.12) together with the state update equation (10.1) into (10.10). Convexity of the polyhedral norm $\|\cdot\|_p$ as well as of J^*_{i+1} allows us to factor out the scheduling parameter, yielding the optimization problem

$$\begin{aligned}
J_i^*(x_{k+i}) &= \|Qx_{k+i}\|_p + \max_{\theta_{k+i}} \min_{\{\mu_{i,j}, t_{i,j}\}} \sum_{j=1}^{n_\theta} \theta_{k+i,j} t_{i,j}(x_{k+i}) && (10.13a)\\
\text{s.t.} \quad & \sum_{j=1}^{n_\theta} \theta_{k+i,j} \big(\|R\mu_{i,j}(x_{k+i})\|_p + && (10.13b)\\
& \quad + J^*_{i+1}(A_j x_{k+i} + B\mu_{i,j}(x_{k+i})) - t_{i,j}(x_{k+i}) \big) \le 0 \qquad \forall\, \theta_{k+i} \in \Theta, &&\\
& \sum_{j=1}^{n_\theta} \theta_{k+i,j} \mu_{i,j}(x_{k+i}) \in \mathcal{U} \quad \forall\, \theta_{k+i} \in \Theta, && (10.13c)\\
& \sum_{j=1}^{n_\theta} \theta_{k+i,j} (A_j x_{k+i} + B\mu_{i,j}(x_{k+i})) \in \mathcal{X}_{i+1} \quad \forall\, \theta_{k+i} \in \Theta, && (10.13d)\\
& x_{k+i} \in \mathcal{X}, && (10.13e)\\
& \theta_{k+i} \in \Theta. && (10.13f)
\end{aligned}$$

The constraints (10.13b) - (10.13d) are satisfied $\forall \theta_{k+i} \in \Theta$ if and only if they are satisfied at the vertices of the parameter simplex. Moreover, the maximum is attained at a vertex of the parameter simplex, such that the (10.13) can be restated as

$$\begin{aligned}
J_i^*(x_{k+i}) &= \|Qx_{k+i}\|_p + \max_{j} \min_{\{\mu_{i,j}, t_{i,j}\}} t_{i,j}(x_{k+i}) && (10.14a)\\
\text{s.t.} \quad & \|R\mu_{i,j}(x_{k+i})\|_p + J^*_{i+1}(A_j x_{k+i} + B\mu_{i,j}(x_{k+i})) \le t_{i,j}(x_{k+i}) &&\\
& \qquad \forall\, j \in [1, n_\theta], && (10.14b)\\
& \mu_{i,j}(x_{k+i}) \in \mathcal{U} \quad \forall\, j \in [1, n_\theta], && (10.14c)\\
& A_j x_{k+i} + B\mu_{i,j}(x_{k+i}) \in \mathcal{X}_{i+1} \quad \forall\, j \in [1, n_\theta], && (10.14d)\\
& x_{k+i} \in \mathcal{X}. && (10.14e)
\end{aligned}$$

Since we are not interested in the optimization variables $\mu_{i,j}(x_{k+i})$ *per se*, but only in the cost function $J_i^*(x_{k+i})$, the optimization problem can be simplified by replacing $t_{i,j}(x_{k+i})$ by $\bar{t}_i(x_{k+i}) := \max_j t_{i,j}(x_{k+i})$, followed by solving the optimization problem

$$J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \min_{\{\mu_{i,j}, \bar{t}_i\}} \bar{t}_i(x_{k+i}) \qquad (10.15a)$$

$$\begin{aligned} \text{s.t.} \quad & \|R\mu_{i,j}(x_{k+i})\|_p + J^*_{i+1}(A_j x_{k+i} + B\mu_{i,j}(x_{k+i})) \leq \bar{t}_i(x_{k+i}) \\ & \qquad \forall j \in [1, n_\theta] , && (10.15b) \\ & \mu_{i,j}(x_{k+i}) \in \mathcal{U} \quad \forall j \in [1, n_\theta] , && (10.15c) \\ & A_j x_{k+i} + B\mu_{i,j}(x_{k+i}) \in \mathcal{X}_{i+1} \quad \forall j \in [1, n_\theta] , && (10.15d) \\ & x_{k+i} \in \mathcal{X} , && (10.15e) \end{aligned}$$

instead. Note that the parametric optimization problem (10.15) is a parametric linear program, which leads to the remaining properties of Theorem 10.1. The initial assumption of a polytopic set $\mathcal{X}_{i+1}$ and a convex, polyhedral piecewise affine J^*_{i+1} can be shown by induction. ■

Basically, the unknown future parameters are dealt with by performing an epigraph reformulation, and parameter-dependent constraints are replaced by a constraint for each vertex of the parameter simplex. Due to the epigraph reformulation (cf. Equation (3.4)), the objective function is converted to a constraint. In order to stay in the class of parametric linear programs, which require polyhedral constraints, we are limited to objective functions which are representable by convex piecewise affine functions. Unfortunately, quadratic cost functions are not possible, since we have no efficient methods for parametric programming for problems with quadratic constraints.

By using polyhedral norms, the optimal cost functions J^*_i are convex, polyhedral piecewise affine functions of the state x_{k+i}, such that in every iteration the optimization problem (10.15) can be solved parametrically with respect to x_{k+i}. Contrary to the closed-loop MPC approach for uncertain systems, the predicted future inputs are functions of the future scheduling parameters, $u_{k+i} = \mu_i(x_{k+i}, \theta_{k+i})$.

Using the terminology of the robust optimization community, in every step of the proposed dynamic programming procedure, we are solving the *Affinely Adjustable Robust Counterpart* (AARC) of an uncertain linear program in a parametric fashion. By restricting the input matrix B to be constant, this AARC is of *fixed recourse*, ensuring computational tractability. For more details on this robust optimization framework, see e.g. [BTGGN04].

The *final step of the dynamic programming procedure* differs from the previous steps. As the parameter θ_k is measured and known, this information can, and should, be taken into account instead of considering the worst case. The problem to solve in the final DP step is thus the nonlinear parametric program

$$\begin{aligned}
J^*(x_k,\theta_k) = \min_{\mu_0} & \|R\mu_0(x_k,\theta_k)\|_p + J_1^*(x_{k+1}) && \text{(10.16a)}\\
\text{s.t.}\quad & x_{k+1} = A(\theta_k)x_k + B\mu_0(x_k,\theta_k)\,, && \text{(10.16b)}\\
& \mu_0(x_k,\theta_k) \in \mathcal{U}\,, && \text{(10.16c)}\\
& x_{k+1} \in \mathcal{X}_1\,, && \text{(10.16d)}\\
& x_k \in \mathcal{X}\,, && \text{(10.16e)}\\
& \theta_k \in \Theta\,, && \text{(10.16f)}
\end{aligned}$$

Unfortunately, some of the constraints of this optimization problem depend bilinearly on the parametric variables (x_k,θ_k), which prevents a standard parametric solution strategy. One way around this is to solve the optimization problem not parametrically in (x_k,θ_k), but in the *uncontrolled successor state*,

$$z_k := (\sum_{j=1}^{n_\theta} A_j\theta_{k,j})x_k\,, \qquad \text{(10.17)}$$

which was first introduced in [BLM08] and in generalized form constitutes a cornerstone of [BRBM08]. By solving the optimization problem parametrically in the uncontrolled successor state z_k, the optimization problem (10.16) can be rewritten as

$$\begin{aligned}
J^*(z_k) = \min_{\{\mu,t_k\}} & \; t_k(z_k) && \text{(10.18a)}\\
\text{s.t. } & C_{\mu t}[\mu(z_k),t_k(z_k)]^T \le c - C_x z_k\,, && \text{(10.18b)}
\end{aligned}$$

which can be solved parametrically with respect to the uncontrolled successor state as a standard parametric linear program. The solution is an explicit control law

$$u_k = \mu(z_k) = \begin{cases} F_1 z_k + g_1\,, & z_k \in \mathcal{R}_1\,,\\ \quad\vdots & \quad\vdots \\ F_{n_r} z_k + g_{n_r}\,, & z_k \in \mathcal{R}_{n_r} \end{cases} \qquad \text{(10.19)}$$

with $F_r \in \mathbb{R}^{n_u\times n_x}, g_r \in \mathbb{R}^{n_u}$ forming the feedback in the rth controller region $\mathcal{R}_r$. Online, all we have to do is to compute the uncontrolled successor state z_k, which is completely determined by the measured state x_k and scheduling parameter θ_k, and evaluate the look-up table to obtain the optimal control input u_k.

The DP procedure finishes with a piecewise affine control law, not defined over a set of current states x_k, but of feasible uncontrolled successor states z_k,

$$\mathcal{Z}_f = \{z \in \mathbb{R}^{n_x} \mid E_z z \le f_z\}\,. \qquad \text{(10.20)}$$

However, it might be interesting to compute a region which tells us which actual initial states are feasible, moreover, which initial states are *admissible*, such that for all parameter values the uncontrolled successor state is feasible. This way we guarantee that a solution exists for all initial states of the admissible state set, independent from the parameter value. For a fixed state x_k, the set of all possible uncontrolled successor states (10.20) is a polytope, where A_j determines the jth vertex. For convexity reasons it is sufficient to check if all vertices of this polytope lie in the polytope (10.20), such that the set of admissible initial states can be stated as

$$\mathcal{X}_f = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; \begin{bmatrix} E_z A_1 \\ \vdots \\ E_z A_{n_\theta} \end{bmatrix} x \leq \begin{bmatrix} f_z \\ \vdots \\ f_z \end{bmatrix} \right\}. \tag{10.21}$$

10.3 Stability

This section is concerned with stability of the resulting closed-loop system, when explicit control laws are applied to LPV-A systems. Note that the proposed procedure does not guarantee stability a-priori, which is a classical issue in finite-horizon MPC. As discussed in Section 4.5, there are known variations of the model predictive control scheme which can be employed, for example *dual mode MPC* or the (overly conservative) *terminal equality constraint*. One can guarantee (i) asymptotic stability, (ii) constraint satisfaction, and (iii) recursive feasibility *a-priori* for all feasible states, by considering a *dual mode* approach and by choosing the terminal state constraints $\mathcal{X}_T$ and the polyhedral terminal cost $L_N(x_{k+N})$ appropriately, [MRRS00]. From [MRRS00] we have the following conditions for asymptotic stability:

A1: $\mathcal{X}_T \subseteq \mathcal{X}, \mathcal{X}_T$ closed and contains the origin.

A2: $\mu_T(x_k, \theta_k) \in \mathcal{U}\ \forall x_k \in \mathcal{X}_T\ \forall \theta_k \in \Theta$.

A3: $x_{k+1} = A(\theta_k)x_k + B(\theta_k)\mu_T(x_k, \theta_k) \in \mathcal{X}_T\ \forall x_k \in \mathcal{X}_T\ \forall \theta_k \in \Theta$

A4: $L_N(x_k) - L_N(x_{k+1}) \leq \|Qx_k\|_p + \|R\mu_T(x_k, \theta_k)\|_p\ \forall x_k \in \mathcal{X}_T\ \forall \theta_k \in \Theta$.

Furthermore, it is a well-known fact, that stability is preserved in the case of a suboptimal solution, as long as the suboptimality of the cost function does not exceed one stage cost, [SMR99]. Consider the following procedure, which is based on [BLM09] and [BBM02]:

1. Compute an asymptotically stabilizing terminal region parameter-varying state-feedback controller

$$u_k = \mu_T(x_k, \theta_k) = K(\theta_k)x_k = \sum_{j=1}^{n_\theta} K_j \theta_{k,j} x_k \tag{10.22}$$

   for the unconstrained system (10.1), e.g. by the procedure in Section 13.

2. Determine a polytopic λ-contractive terminal region $\mathcal{X}_T$ by pre-image computations, such that

$$\forall x_k \in \mathcal{X}_T, \forall \theta_k \in \Theta \ \exists \mu_T(x_k, \theta_k) \in \mathcal{U} : A(\theta_k)x_k + B(\theta_k)\mu_T(x_k, \theta_k) \in \lambda \mathcal{X}_T \tag{10.23}$$

   holds for some $\lambda \in [0, 1)$.

3. Scale the Minkowski function

$$\psi_{\mathcal{X}_T}(x_k) := \min_{\alpha}\{\alpha \in \mathbb{R}_+ \mid x_k \in \alpha \mathcal{X}_T\}, \tag{10.24}$$

   induced by the terminal region $\mathcal{X}_T$, by a factor $\beta^* \in \mathbb{R}_+$, which can be determined by the linear program

$$\beta^* = \min_{\beta} \beta \tag{10.25a}$$

$$\text{s.t. } \beta(1-\lambda) \geq \|Qv_i\|_p + \|RK_jv_i\|_p$$

$$\forall v_i \in \text{vert}(\mathcal{X}_T) \ \forall j \in \{1, \ldots, n_\theta\}. \tag{10.25b}$$

4. Define $L_N(x_{k+N}) := \beta^* \psi_{\mathcal{X}_T}(x_{k+N})$.

Theorem 10.2 *Assume that there exists a terminal region control $\mu_T(x_k, \theta_k)$ of the form* (10.22)*, which renders the polytope $\mathcal{X}_T$ λ-contractive as in* (10.23) *with $\lambda \in [0, 1)$. Then, for the terminal region $\mathcal{X}_T$ and the terminal cost $L_N(x_k) = \beta^* \psi_{\mathcal{X}_T}(x_k)$, as defined in* (10.24)-(10.25)*, the conditions A1 to A4 are satisfied, such that we have asymptotic stability, constraint satisfaction and recursive feasibility for all feasible points.*

Proof Conditions A1 to A3 follow immediately from the properties of the λ-contractive terminal set $\mathcal{X}_T$.

Condition A4 follows from

$$
\begin{aligned}
&\beta\psi_{\mathcal{X}_T}(x_k) - \beta\psi_{\mathcal{X}_T}(x_{k+1}) \geq \|Qx_k\|_p + \|RK(\theta_k)x_k\|_p \quad \forall x_k \in \mathcal{X}_T \; \forall\theta_k \in \Theta \\
\Leftarrow\ &\beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \|Qx_k\|_p + \|RK(\theta_k)x_k\|_p \quad \forall x_k \in \mathcal{X}_T \; \forall\theta_k \in \Theta \\
\Leftarrow\ &\beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \|Qx_k\|_p + \sum_{j=1}^{n_\theta} \theta_{k,j}\|RK_j x_k\|_p \quad \forall x_k \in \mathcal{X}_T \; \forall\theta_k \in \Theta \\
\Leftrightarrow\ &\sum_{j=1}^{n_\theta} \theta_{k,j}\beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \sum_{j=1}^{n_\theta} \theta_{k,j}\|Qx\|_p + \sum_{j=1}^{n_\theta} \theta_{k,j}\|RK_j x_k\|_p \\
&\quad \forall x_k \in \mathcal{X}_T \; \forall\theta_k \in \Theta \\
\Leftrightarrow\ &\beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \|Qx_k\|_p + \|RK_j x_k\|_p \quad \forall x_k \in \mathcal{X}_T \; \forall j \in \{1,\dots,n_\theta\} \\
\Leftrightarrow\ &\beta(1-\lambda) \geq \|Qv_i\|_p + \|RK_j v_i\|_p \quad \forall v_i \in \mathrm{vert}(\mathcal{X}_T) \; \forall j \in \{1,\dots,n_\theta\} \qquad \blacksquare
\end{aligned}
$$

Note that a symmetric terminal set $\mathcal{X}_T$ implies a Minkowski function which can be expressed as $L_N(x_{k+N}) = \|Px_{k+N}\|_\infty$ with some matrix P. However, the described procedures work also for convex polyhedral piecewise affine terminal cost $L_N(x_{k+N})$. For more results on stability of LPV-A systems, see e.g. [BM08b].

The drawback of adding terminal state constraints is that in general they lead to a loss of performance, a smaller feasible space and an increase in complexity of the resulting control law. While the former two effects can be mitigated by extending the prediction horizon, this typically leads to a further increase in the complexity of the control law. In order to avoid these downsides in practical implementations, a possibility is to omit the terminal state constraints and to verify stability a-posteriori, following the theory described in [BM08b].

This stability analysis is performed in the space of the uncontrolled successor state z_k, where the LPV-A system (10.1) under the PWA control law $u_k = \mu(z_k)$ from Equation (10.19) corresponds to the piecewise affine closed-loop system

$$
\begin{aligned}
z_{k+1} &= A(\theta_{k+1})\left(z_k + B\mu(z_k)\right) \\
&= \begin{cases} A(\theta_{k+1})(I+BF_1)z_k + A(\theta_{k+1})Bg_1\,, & z_k \in \mathcal{R}_1 \\ \vdots & \vdots \\ A(\theta_{k+1})(I+BF_{n_r})z_k + A(\theta_{k+1})Bg_{n_r}\,, & z_k \in \mathcal{R}_{n_r} \end{cases}
\end{aligned}
\tag{10.26}
$$

with polytopic uncertainty.

The stability analysis consists of three steps, which can be easily automated. In the following we will focus on the regulator problem, i.e. the stabilization to the origin. Note that all other reference values in the state space correspond to a set of reference values in z–space, depending on the scheduling parameter θ_k. Asymptotic stability to reference values $\bar{x} \neq 0$ requires constant scheduling parameter values (Scenario ($S3$)). For varying scheduling parameter values, instead of asymptotic stability to some reference value $\bar{x} \neq 0$, only ultimate boundedness to a target set can be certified.

Definition 10.1 *Let $\boldsymbol{I}$ be the index set of all controller regions containing the origin,*

$$\boldsymbol{I} := \{ r \in \{1, \dots, n_r\} \mid 0 \in \mathcal{R}_r \} \ .$$

The index set $\boldsymbol{I}$ is single-valued if the origin is contained in the interior of a controller region, and multi-valued if the origin lies on the facet of several controller regions.

1. *Invariance of the origin*
   At first we require the origin to be invariant, i.e. to be an equilibrium point of the closed-loop system. This is the case, if the condition

   $$g_r = 0 \quad \forall r \in \boldsymbol{I} \tag{10.27}$$

   holds. Since $\mathcal{X}$ and $\mathcal{U}$ include the origin, there always exists a controller which fulfils this condition. If Condition (10.27) is violated, the origin is *not* an equilibrium point, and at most ultimate boundedness to a target set can be present.

2. *Contractiveness of a target set*
   After verifying invariance of the origin, the size of the region of attraction shall be determined. First we establish contractiveness of a target set. If the index set $\boldsymbol{I}$ of regions containing the origin is single-valued, a robust invariant target set can be determined by a set backpropagation as in Algorithm 2.1, starting with $\mathcal{Z}_0 = \mathcal{R}_r$ and repeatedly using the contractive preset

   $$\Omega(\mathcal{Z}_i, \mathcal{R}_r) = \left\{ z_k \in \mathbb{R}^{n_x} \middle| \begin{array}{l} A_j(I + BF_r)z_k + A_jBg_r \in \lambda\mathcal{Z}_i,\ j = 1, \dots, n_\theta \\ z_k \in \mathcal{R}_r \end{array} \right\}, \tag{10.28}$$

   with $\lambda \in [0, 1)$ denoting the *contraction ratio*. If $\mathcal{Z}_i \subseteq \Omega(\mathcal{Z}_i, \mathcal{R}_r)$, the algorithm is terminated with $\mathcal{T} := \mathcal{Z}_i$, otherwise we set $\mathcal{Z}_{i+1} = \Omega(\mathcal{Z}_i, \mathcal{R}_r)$ and perform the next iteration. If the index set $\boldsymbol{I}$ is multi-valued, the backpropagation runs in parallel for each regarded region $r \in \boldsymbol{I}$, starting with $\mathcal{Z}_0 = \{\mathcal{R}_r\}_{r \in \boldsymbol{I}}$, and merging the presets after each iteration $\mathcal{Z}_{i+1} = \bigcup_{r \in \boldsymbol{I}} \Omega(\mathcal{Z}_i, \mathcal{R}_r)$. The resulting target region $\mathcal{T}$ is λ-contractive, and asymptotically stable due to the following proposition.

   Proposition 10.3 *Let $\mathcal{T} \subseteq \bigcup_{r \in \boldsymbol{I}} \mathcal{R}_r \subset \mathbb{R}^{n_x}$ be a polytope containing the origin in its interior and let Condition (10.27) hold. If $\forall r \in \boldsymbol{I}$, all vertices*

v_r^i of $\mathcal{T} \bigcap \mathcal{R}_r$ are mapped into $\lambda\mathcal{T}$, $0 \leq \lambda < 1$, then $\mu\mathcal{T}$ is λ-contractive $\forall \mu \in [0,1]$.

Proof Consider any $\hat{z}_k \in \mu(\mathcal{T} \bigcap \mathcal{R}_r) \Rightarrow \tilde{z}_k = \hat{z}_k/\mu \in (\mathcal{T} \bigcap \mathcal{R}_r) \Leftrightarrow \exists \alpha_r^i \in \mathbb{R}_+, \sum_i \alpha_r^i = 1 : \tilde{z}_k = \sum_i \alpha_i v_r^i \Rightarrow \hat{z}_k = \mu \sum_i \alpha_i v_r^i \Rightarrow \hat{z}_{k+1} \in \mu \sum_i \alpha_i \lambda \mathcal{T} = \mu\lambda\mathcal{T}$. ■

Proposition 10.3, together with the properties of the gauge function $\psi_{\mathcal{T}}(z)$ induced by $\mathcal{T}$ suffices to establish asymptotic stability inside $\mathcal{T}$ by using $\psi_{\mathcal{T}}(z)$ as a Lyapunov function.

3. *Reachability analysis*
Finally a reachability analysis is performed to determine all states which are mapped to the target region $\mathcal{T}$. In this reachability analysis, the backpropagation algorithm (Algorithm 2.1) is performed, using again the preset (10.28). We start the backpropagation with $\mathcal{Z}_0 = \mathcal{T}$. In each iteration i we compute $\Omega(\mathcal{Z}_i, \mathcal{R}_r)$, $r = 1, \ldots, n_r$ and merge the resulting presets to $\mathcal{Z}_{i+1} = \bigcup_{r=1}^{n_r} \Omega(\mathcal{Z}_i, \mathcal{R}_r)$.

   The iterations terminate when $\mathcal{Z}_{i+1} \subseteq \mathcal{Z}_i$ or when the entire feasible space is covered. The resulting region of attraction is denoted by $\mathcal{Z}_\infty$. All uncontrolled successor states $z_k \in \mathcal{Z}_\infty$ are controlled to the target set $\mathcal{T}$ and eventually to the origin by construction. The required algorithms to perform the backpropagations boil down to polytopic manipulations and can be adapted from the algorithms given in the references mentioned above. The set of stable states can be determined similar to (10.21) by ensuring that the uncontrolled successor state is in $\mathcal{Z}_\infty$ independent of the current parameter.

In the following, we will show that stability properties of the uncontrolled successor state z_k imply stability properties of the actual state x_k.

Lemma 10.4 *The uncontrolled successor state obeys the following upper and lower bounds:*

(a) For every LPV-A system (10.1) *there exists an $\epsilon_1 > 0$, such that the uncontrolled successor state* (10.17) *can be bounded by*

$$\|z_k\| \leq \epsilon_1 \|x_k\| \,. \tag{10.29a}$$

(b) Under the assumption of (10.27) *and for a small-enough $\delta_2 > 0$, there exists an $\epsilon_2 > 0$ such that*

$$\|z_k\| < \delta_2 \quad \Rightarrow \quad \|x_{k+1}\| \leq \epsilon_2 \|z_k\| \tag{10.29b}$$

holds.

Proof (a) Using the definition of the uncontrolled successor state, it follows that

$$\begin{aligned}\|z_k\| &= \|A(\theta_k)x_k\| \\ &\leq \|A(\theta_k)\|\|x_k\| \\ &\leq \max_{\theta_k\in\Theta} \|A(\theta_k)\|\|x_k\| =: \epsilon_1\|x_k\| \,.\end{aligned}$$

(b) If δ_2 is small enough, the uncontrolled successor state is an element of $\bigcup_{r\in\boldsymbol{I}}\mathcal{R}_r$. Using the state-update equation, the control law (10.19) and Condition (10.27), we infer

$$\begin{aligned}\|x_{k+1}\| &= \|z_k + B\mu(z_k)\| \\ &\leq \| \max_{r\in\boldsymbol{I}}(I+BF_r)z_k\| \\ &\leq \max_{r\in\boldsymbol{I}} \|(I+BF_r)\|\|z_k\| =: \epsilon_2\|z_k\| \,.\end{aligned}$$ ■

Proposition 10.5 *Lyapunov stability and asymptotic stability of* $x_k = 0$ *follow directly from Lyapunov stability and asymptotic stability of* $z_k = 0$*, respectively.*

Proof Proposition 10.5 is trivial, if the state-update matrix $A(\theta)$ is invertible. If $A(\theta)$ is singular, it possesses a (parameter-dependent) null space, containing states x_k which are not necessarily zero if $z_k = 0$. Fortunately, we just need to let pass one time step in order to show Proposition 10.5 in this case.

With Lemma (10.4) and under the assumption of $\epsilon \leq \delta_2$, we obtain from the Lyapunov stability of $z_k = 0$:

$$\begin{aligned}\|x_k\| \leq \delta' := \frac{\delta}{\epsilon_1} \quad &\Rightarrow \quad \|z_k\| \leq \delta \\ &\Rightarrow \quad \|z_{k+i}\| \leq \epsilon \quad \forall i \in \mathbb{N} \\ &\Rightarrow \quad \|x_{k+i+1}\| \leq \epsilon_2\epsilon =: \epsilon' \quad \forall i \in \mathbb{N} \,.\end{aligned}$$

Similarly, we infer asymptotic stability of $x_k = 0$ from asymptotic stability of $z_k = 0$, using again Lemma (10.4),

$$\begin{aligned}\|x_k\| \leq \delta' := \frac{\delta}{\epsilon_1} \quad &\Rightarrow \quad \|z_k\| \leq \delta \\ &\Rightarrow \quad \lim_{k\to\infty} \|z_k\| = 0 \\ &\Rightarrow \quad \lim_{k\to\infty} \epsilon_2\|x_{k+1}\| = 0 \\ &\Leftrightarrow \quad \lim_{k\to\infty} \|x_k\| = 0 \,.\end{aligned}$$ ■

Remark 10.6 *Note that Condition* (10.27) *is used in Lemma 10.4(b), and is thus also required for Proposition 10.5 to hold. However, Condition* (10.27) *is contained in the assumptions of Proposition 10.5, since it is a necessary condition for stability of* $z_k = 0$.

Remark 10.7 *The assumption* $\epsilon \leq \delta_2$ *is not a necessary condition for Lyapunov stability of* $x_k = 0$. *There will always be a small-enough* ϵ', *such that* $\epsilon \leq \delta_2$ *is fulfilled.*

10.4 Numerical Examples

This section consists of three numerical examples, demonstrating the application of the proposed algorithm and comparing it to Quasi-Min-Max MPC, explicit robust MPC and to explicit MPC for piecewise affine systems.

Example 10.1 (Comparison with Quasi-Min-Max MPC)

In the first example the potential of Explicit LPV-A MPC in reducing the online computational effort is demonstrated. The example system is taken from [LA99]. It represents an unstable LPV-A system of the form (10.1) with the system matrices

$$\begin{aligned} A_1 &= \begin{bmatrix} 1.3333 & -0.6667 & 1.3333 & -0.6667 \\ 0.1 & 0 & 0 & 0 \\ 1.3333 & -0.6667 & 1.3333 & -0.6667 \\ 0.1 & 0 & 0 & 0 \end{bmatrix}, \\ A_2 &= \begin{bmatrix} 1.3333 & -0.6667 & 1.3333 & -0.6667 \\ 1 & 0 & 1 & 0 \\ 1.3333 & -0.6667 & 1.3333 & -0.6667 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \\ B &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

The states are constrained to be $x_{k,i} \leq 1.14$, $i \in [1,4]$, and the input is constrained to be $u_k \leq 4.15$. In this example, the states of the LPV-A system shall be regulated from the initial state

$$x_0 = \begin{bmatrix} -0.3964 & 0.4377 & -1.0905 & 1.1137 \end{bmatrix}$$

to the origin by means of two control methods: The explicit MPC scheme presented in Section 10.2, and the Quasi-Min-Max MPC scheme from Section 9.5, which was proposed together with the original example in [LA99]. The evolution of the scheduling parameter is depicted in Figure 10.1.

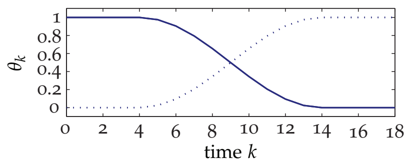


Figure 10.1: Evolution of the scheduling parameter, $\theta_{k,1}$ (—), $\theta_{k,2}$ ($\cdots$), in Example 10.1.

The Multi-Parametric Toolbox (MPT) and Yalmip were used to compute the explicit control law, [KGBM04, Löf08]. The weight matrices

$$Q = \operatorname{diag}(\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}), \quad R = 0.1, \quad P = Q,$$

and a prediction horizon of $N = 4$ were chosen. The ∞-norm was used in the cost function (10.5). The control law was computed for all states in the hyperbox $-10 \leq x_{k,i} \leq 1.14$, $i \in [1,4]$. Afterwards, the resulting controller regions were merged using a greedy merge, resulting in a total number of 740 regions. For this control law a binary search tree was generated, which can be evaluated under C with small computational effort, [TJB03]. MPT supports the export of binary search trees into C code, which can then be compiled as MEX functions callable from within Matlab. A pure C implementation would even further decrease the required computation times.

The solution of the semi-definite programs within the Quasi-Min-Max MPC scheme was performed by SeDuMi, [Stu99], interfaced via Yalmip [Löf04]. Since the input matrix B is constant, the future state feedback control laws were chosen parameter-dependent, in order to reduce the conservatism of the approach. Apart from the selected cost function, which is quadratic in the Quasi-Min-Max MPC scheme, the setup is identic to the explicit LPV-A MPC scheme.

The simulation results for both control methods are depicted in Figure 10.2. It can be seen that the closed-loop trajectories of the states and the inputs are nearly the same, although two different objective functions were minimized (quadratic upper bound on the infinite horizon cost vs. piecewise linear finite-horizon cost). Under both control methods, the required input and state constraints are satisfied.

The actual difference lies in the required online computational effort. While explicit MPC requires a computation time of less than 0.02 ms in each step, Quasi-Min-Max MPC requires between 0.2 s and 0.3 s, which is more than 4 orders of magnitude difference[1] longer.

[1] All computations were performed on a 3 GHz Pentium 4 processor using Matlab 7.

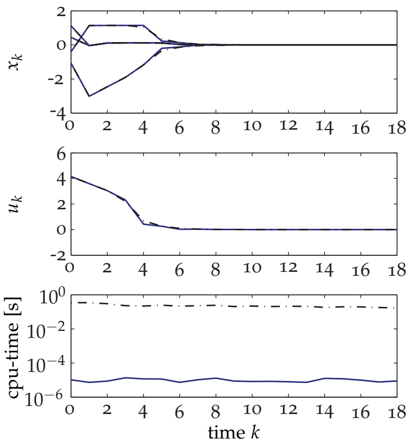


Figure 10.2: Closed-loop trajectories of the states and the input, and the required online CPU time in Example 10.1. Comparison of Explicit LPV-A MPC (—) and Quasi-Min-Max MPC (− · −).

Example 10.2 (Comparison with robust MPC)

In the second example explicit control laws for a system of the form (10.1) were computed, once utilizing knowledge the scheduling parameter θ_k and once treating the scheduling parameter as uncertainty. In the latter case, the closed-loop constrained robust optimal control (CL-CROC) problem is solved explicitly, resulting in the control input being independent of the parameter, $u_k = \mu(x_k)$. It makes sense to assume that utilizing the current scheduling parameter information improves performance, and this intuition has already been verified in several examples, for instance in [LA99]. Therefore we focus here on the question how the complexity of the two resulting explicit control laws relates to each other.

The example system is taken from [LA00b], and consists of the matrices

$$A_1 = \begin{bmatrix} 0.2730 & 0.0660 & 0.3021 & -0.5012 \\ 0.2717 & 0.4416 & 0.5602 & -0.7123 \\ 0.3051 & -0.7865 & 0.7651 & 0.3121 \\ 0.7962 & -0.1452 & 0.5231 & -0.9345 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 0.2093 & -0.1981 & -0.2394 & 0.5671 \\ 0.2717 & 0.4598 & 0.5602 & 1.3782 \\ -0.4700 & 0.6700 & -0.8600 & -1.2400 \\ 0.3456 & -0.6312 & -1.4594 & 1.8936 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.2300 & 0.2601 & 0.1213 & 1.3452 \end{bmatrix}^T.$$

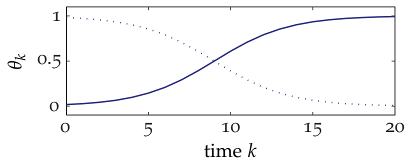
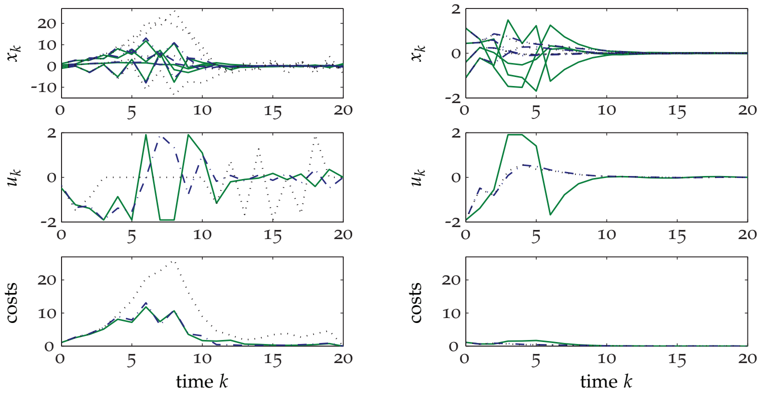


Figure 10.3: Evolution of the scheduling parameter, $\theta_{k,1}$ (—) and $\theta_{k,2}$ ($-\cdot$), in Example 10.2.



(a) Explicit robust MPC. (b) Explicit LPV-A MPC.

Figure 10.4: Closed-loop trajectories of the states and the input, and the actual costs in Example 10.2. Comparison of explicit LPV-A MPC and explicit robust MPC with prediction horizon $N = 1$ (—), $N = 2$ ($- \cdot -$) and $N = 3$ ($\cdots$).

The control aim is to regulate the system from the initial point

$$x_0 = \begin{bmatrix} -0.3964 & 0.4377 & -1.0905 & 1.1137 \end{bmatrix}^T$$

to the origin, while the state is bounded to $\|x_k\|_\infty \leq 20$, the input is bounded to $|u_k| \leq 1.91$, and the parameter/uncertainty follows the trajectory shown in Figure 10.3. As weights for the MPC schemes, $Q = I$ and $R = 0.01$ were chosen. The terminal cost matrix P was selected to be equal to Q, and the norm, with $p = \infty$, was minimized in the cost function.

The simulation results are depicted in Figure 10.4. It can be observed that the explicit robust controllers have far more difficulties to handle the system than the explicit LPV-MPC controllers. Moreover, the control performance of explicit LPV-MPC increases with longer prediction horizons, whereas the control performance of explicit robust MPC decreases.

Horizon	Explicit LPV-A MPC		Explicit Robust MPC	
	regions	costs	regions	costs
1	32	10.14	244	69.26
2	838	5.64	4392	71.15
3	6550	5.50	43168	161.33

Table 10.1: Number of regions and cumulated costs for different prediction horizons in Example 10.2.

Table 10.1 summarizes the complexity of the computed control laws and the cumulated costs in the control scenario for different prediction horizons. As expected, using the parameter information improves the control performance, but also the complexity of the resulting control law decreased by a factor of 4 – 8. This is mainly due to the uncontrolled successor state z_k in the final step of the proposed DP algorithm, which indeed reduced the number of regions of the control law. While the min-max steps of the CL-CROC problem and the max-min steps of the CPVOC problem differ mainly in the number of optimization variables, the parametric linear program in the final step of the CPVOC problem is not a max-min optimization problem, such that the number of constraints is lower compared to the final step of the CL-CROC problem.

Example 10.3 (Comparison with hybrid MPC)

In the third example we examine the application of the proposed procedure by computing explicit control laws for the nonlinear Hénon map. An LPV model and a PWA model is derived to investigate when a parameter-varying model could be of use.

The Hénon map is a nonlinear second-order system and a popular example for chaotic systems, [Hén76]. It is defined as

$$x_{k+1,1} = -a(x_{k,1})^2 + x_{k,2} + 1\,, \tag{10.30a}$$

$$x_{k+1,2} = bx_{k,1}\,, \tag{10.30b}$$

where the second subscript indicates the element of the state vector. When the coefficients are $a = 1.4,\ b = 0.3$, the system has an unstable fixed point at

$$\bar{x} = (-1/4 + \frac{\sqrt{609}}{28})\begin{bmatrix}1 & 0.3\end{bmatrix}^T \approx \begin{bmatrix}0.63 & 0.19\end{bmatrix}^T.$$

Already small deviations from this fixed point lead to chaotic behaviour, and the system moves along a so-called chaotic attractor. A chaotic attractor has the property that during an infinite amount of time, the system is getting arbitrary close to every point on the attractor.

In order to control the Hénon map, we introduce an input to obtain the controlled Hénon map,

$$x_{k+1,1} = -a(x_{k,1})^2 + x_{k,2} + 1 + u\,, \tag{10.31a}$$
$$x_{k+1,2} = bx_{k,1} + cu\,, \tag{10.31b}$$

The input coefficient in the controlled Hénon map (10.31) is set to $c = 0.1$. A linear controller can stabilize this system to the fixed point from a surrounding domain of attraction, [Vin97].

In the following, two methods are used to compute explicit control laws for the controlled Hénon map: One is to model the Hénon map by an LPV model and apply the method described above. As an alternative approach the Hénon map is approximated by a PWA model and the associated optimal control law is computed.

LPV model of the Hénon map

For the computation of an explicit MPC controller with the proposed method, we have to embed the Hénon map (10.31) into an LPV-A system of the form (10.1). Due to the affine term in (10.31), this is not directly possible. However, the proposed algorithm easily extends to LPV-A systems with affine terms in the state prediction. In a first step, the Hénon map is rewritten as

$$x_{k+1} = \begin{bmatrix} -ax_{k,1} & 1 \\ b & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ c \end{bmatrix} u_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\,. \tag{10.32}$$

The next step is the selection of a suitable scheduling function $\theta_k = f_\rho(\rho_k)$, with the scheduling variable $\rho_k = x_{k,1}$, and where we want the scheduling parameter θ_k to vary in $[0, 1]$. This involves the decision for a domain of $x_{k,1}$, for which the LPV model represents the dynamics of the nonlinear system accurately. On the one hand we want the state transition matrix $A(\theta_k)$ to vary as little as possible to mitigate the introduced conservatism of the LPV approach, on the other hand we want to make the domain as large as possible. As the chaotic attractor lies in $[-1.5, 1.5]$, this interval was chosen and led to the description

$$x_{k+1} = \begin{bmatrix} 3a\theta_k - 1.5a & 1 \\ b & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ c \end{bmatrix} u_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\,, \tag{10.33}$$
$$\theta_k = (1.5 - x_{k,1})/3\,. \tag{10.34}$$

Note that the first entry of the state transition matrix can take values in the interval $[-2.1, 2.1]$. The parameter causes such a severe change of dynamics,

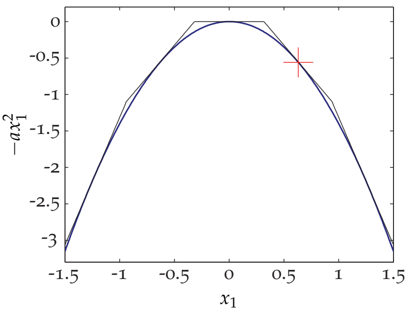


Figure 10.5: PWA approximation of the quadratic term of the Hénon map.

that a robust min-max MPC scheme, assuming uncertain parameter θ_k, failed to stabilize (10.33) to $\bar{x}$.

PWA model of the Hénon map

It is also possible to obtain a piecewise affine approximation of (10.31) and compute the optimal control for this system. For this approach, the quadratic term in (10.31) has to be approximated by a piecewise affine function. The quadratic term as well as a possible piecewise affine approximation with 5 affine terms is shown in Figure 10.5. The used approximation is

$$-a(x_{k,1})^2 \approx \begin{cases} 2.5ax_{k,1} + \frac{25a}{16}, & x_{k,1} < -0.94, \\ 1.26ax_{k,1} + 0.4a, & -0.94 \leq x_{k,1} < -0.32, \\ 0, & -0.32 \leq x_{k,1} < 0.32, \\ -1.26ax_{k,1} + 0.4a, & 0.32 \leq x_{k,1} < 0.94, \\ -2.5ax_{k,1} + \frac{25a}{16}, & 0.94 \leq x_{k,1} . \end{cases} \tag{10.35}$$

The affine terms were selected to be tangential to the quadratic term, which is especially important at the equilibrium point $\bar{x}$, where approximation errors would lead to steady-state errors, when stabilizing to $\bar{x}$. With this approximation a piecewise affine model of the Hénon map can be derived, consisting of five regions with different dynamics.

The approximation would become more accurate by using more affine terms. However, with a more complex PWA model, the complexity of the explicit controller grows rapidly, and the computations become intractable already for very short prediction horizons. For more details on piecewise affine systems see Chapter 5.

Model type	Controller regions	Computation time[5]
PWA	344	181 secs
LPV	93	24 secs

Table 10.2: Comparison of optimal control laws in Example 10.3.

Comparison of optimal control laws

For both models of the nonlinear system the optimal control laws were computed that minimize the cost function (10.5). In both cases, the weight matrices $Q = I, R = 0.1$ and a prediction horizon of $N = 4$ was chosen. The terminal cost matrix P was selected to be equal to Q, and the 1-norm was minimized in the cost function. Instead of penalizing the weighted 1-norm of the state, the difference to the fixed point $\bar{x}$ was penalized, which trivially can be incorporated in both algorithms.

The Multi-Parametric Toolbox (MPT) and Yalmip were used to compute the explicit control laws, [KGBM04, Löf04].[2] The complexity of the resulting control laws can be seen in Table 10.2. The explicit control law for the LPV model was computed faster[3] and resulted in less regions than the explicit control law for the PWA model. This is due to transitions between the regions of the PWA model, which have to be handled in a combinatorial fashion. The most time in the PWA case was spent to remove redundant regions from overlapping partitions. This phenomenon of overlapping partitions does not appear in the LPV case. Moreover, the cost function and feasible state set of

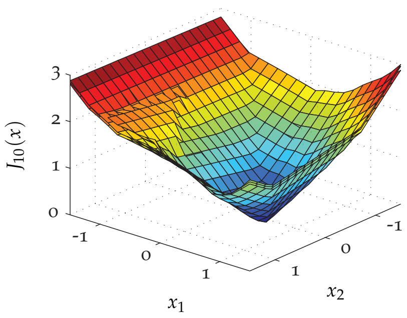


Figure 10.6: Actual 10-steps-costs of LPV, PWA and optimal nonlinear control in Example 10.3.

[2] A complete implementation of the example can be found at http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php?n=Examples.Examples
[3] All computations were performed on a 3 GHz Pentium 4.

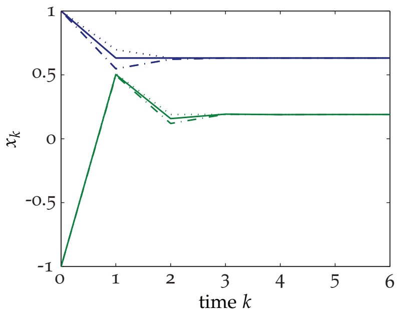


Figure 10.7: State evolution from $x_0 = [1, -1]^T$ under LPV (—), PWA (– ·) and optimal nonlinear (· · ·) control in Example 10.3.

the controller for the LPV model are convex, which allows a faster evaluation of the look-up table online, [Bor03]. The ultimate boundedness of the controlled Hénon map under explicit LPV-MPC to a target region in z-space has been verified for the box $[-1.5, 1.5]^2$ following the reachability analysis in Section 10.3.

Both control laws were tested in simulations by controlling the system from 400 initial points x_0, uniformly distributed over the box $[-1.5, 1.5]^2$, to the fix-point $\bar{x}$. Since the approximation with a PWA model leads to a nonzero steady-state input $\bar{u}_{\text{PWA}}$, this was subtracted from the control signal of the PWA controller during simulations. The actual simulated costs accumulated over 10 steps can be seen in Figure 10.6. The truly optimal solution, based on solving the optimal control problem for the nonlinear system model, is also shown. The solution was computed using the global branch-and-bound based solver available in Yalmip. The differences are hardly visible, relative to the optimal control, the LPV control exhibits an average cost increase of 2.3 % and the PWA control of 3.9 %.

For a closer look, the state evolution of the system under LPV-, PWA- and optimal control, starting from $x_0 = [1, -1]^T$, is shown in Figure 10.7. Indeed, the PWA control has a small steady-state error due to modelling errors. These errors would vanish, if the region including the fixed point would be a tangent in the fixed point. However, this is only possible for the regulation to certain points, but not for tracking of arbitrary reference values.

The question remains in which cases modelling a nonlinear system with an LPV model instead of a PWA model should be preferred. A PWA model comprises a *finite* number of different linearizations of the nonlinear system appointed on a *spatial* basis. The more considered linearizations, the more complex the resulting control law, which limits us to nonlinear systems which can be approximated using a small number of different linearizations.

The closed-loop control performance is determined by the accuracy of the PWA approximation of the nonlinear system. If a nonlinear system can be approximated accurately by a PWA model using only a few different linearizations, this approach should be considered. An LPV model on the other hand comprises a *continuum* of different linearizations of the nonlinear system appointed on a *temporal* basis. Hence LPV models cover a broader range of dynamic behaviour, but as long as the future appointed linearizations are unknown, some uncertainty is introduced to the predictions. The more varying the state transition matrix of the LPV model, the more severe is the introduced conservatism. Consequently an LPV model should be considered if the uncertainty in the future predictions is not severe and/or if a simple and accurate approximation with an PWA model is not possible.

10.5 Conclusions

In this chapter a method was proposed to compute explicit control laws for LPV-A systems, linear time-discrete systems with a parameter-varying state transition matrix. For these LPV-A systems an affine parametrization of the input was shown to be optimal and consequently used in a dynamic programming approach similar to explicit min-max MPC for uncertain systems. The benefits of using an explicit solution by means of the online computational effort were shown in a comparison with Quasi-Min-Max MPC. Our approach fits between two different explicit MPC approaches to approximately solve nonlinear optimal control problems, which are the robust and the PWA approach. A drawback of explicit control laws is that the number of controller regions grows exponentially with the prediction horizon and the number of states. As the suggested approach is based on parametric programming, it participates in this drawback, and is thus only tractable for systems of a limited size. However, the final step tends to reduce the number of regions and the resulting number of regions is typically smaller than the number of regions one would obtain when solving the CL-CROC problem (assuming no knowledge of the uncertainty).

Our procedure enables an alternative to constrained optimal control of piecewise affine systems, when PWA models are approximating nonlinear systems. As an example, optimal control laws were computed for an LPV model and a PWA model of the nonlinear Hénon map. The resulting explicit control laws were compared by means of control performance and complexity. Though conservatism is introduced in the LPV approach by considering the worst case for future parameter values, no approximation errors are introduced as in the approximation by a PWA model and the cumbersome incorporation of region transitions in the prediction is omitted.

11 (Sub-)Optimal Control of Constrained LPV Systems

‘Every now and then go away, have a little *relaxation*, for when you come back to your work your judgement will be surer.´

Leonardo Da Vinci

This chapter is concerned with the computation of explicit state feedback control laws for the more general class of discrete-time LPV systems. Contrary to the previous chapter, the input matrix is now assumed to depend affinely on the scheduling parameter. We demonstrate how one can reformulate the MPC problem for LPV systems into a series of parametric linear programs in a closed-loop min-max MPC scheme based on dynamic programming. A *relaxation* technique is employed to reformulate constraints which are polynomial in the scheduling parameters to parameter-independent constraints. The algorithm allows the computation of explicit control laws for linear parameter-varying systems and enables the controller to exploit information about the scheduling parameter. This improves the control performance compared to a standard robust approach where no uncertainty knowledge is used, while keeping the benefits of fast online computations. In order to investigate different variations of the proposed procedure, several examples are included.

11.1 Problem Statement

The class of systems we consider is the class of LPV systems, linear discrete-time systems with parameter-varying system matrices, which are defined by the state-update equation

$$x_{k+1} = A(\theta_k)x_k + B(\theta_k)u_k \,, \tag{11.1a}$$

with

$$A(\theta_k) = \sum_{j=1}^{n_\theta} A_j\theta_{k,j} \,, \qquad B(\theta_k) = \sum_{j=1}^{n_\theta} B_j\theta_{k,j} \,, \tag{11.1b}$$

and

$$\theta_k \in \Theta := \left\{ \theta_k \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_{k,j} = 1 \right\} . \tag{11.1c}$$

The discrete time is denoted by $k \in \mathbb{Z}$, whereas the variables $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $\theta_k \in \mathbb{R}^{n_\theta}$ denote the state, the control input, and the time-varying scheduling parameter, respectively. We assume that the state x_k is either measurable or observable. The system matrices $A(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_x}$ and $B(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_u}$ are known to lie in polytopes with the description (11.1b), where $A_j \in \mathbb{R}^{n_x \times n_x}, B_j \in \mathbb{R}^{n_x \times n_u}$ denote the jth vertices of the corresponding polytope. The scheduling parameter $\theta_k = [\theta_{k,1} \;\; \dots \;\; \theta_{k,n_\theta}]^T \in \mathbb{R}^{n_\theta}$ is constrained to the standard simplex (11.1c). This polytopic description is a common assumption in the LPV framework, see e.g. [AGB95].

Furthermore, the LPV system (11.1) is constrained, $u_k \in \mathcal{U}$ and $x_k \in \mathcal{X}$. The constraint sets $\mathcal{U}$ and $\mathcal{X}$ are assumed to be bounded polyhedra,

$$u_k \;\in\; \mathcal{U} = \{u_k \in \mathbb{R}^{n_u} \mid H_u u_k \le \mathbf{1}\} \,, \tag{11.2a}$$

$$x_k \;\in\; \mathcal{X} = \{x_k \in \mathbb{R}^{n_x} \mid H_x x_k \le \mathbf{1}\} \,, \tag{11.2b}$$

which contain the origin in their interiors, since we are interested in the regulator problem.

Remark 11.1 *For ease of notation, we restrict ourselves to separate constraints on the state and inputs in* (11.2)*. It is straightforward to modify the presented algorithm in this chapter to the case of mixed polytopic constraints, i.e.* $E_x x + E_u u \le f_{xu}$.

Remark 11.2 *With a change of coordinates, arbitrary polytopes can be transformed into (higher dimensional) standard simplices* Θ*. In Chapter 12 a possibility to generalize the standard simplex to arbitrary polytopes without an increase in the dimension will be shown. Nonlinear dependencies of the system matrices on the scheduling signal can be embedded in a conservative way into the shown framework by defining* Θ *appropriately.*

The essential assumption behind LPV control is that the scheduling parameter is measured online and *known to the controller*. Future values are however only known to be constrained to the standard simplex. This assumption is referred to as the LPV paradigm by some authors.

For the described class of LPV systems we want to compute an explicit state feedback control law

$$u_k = \mu(x_k, \theta_k), \tag{11.3}$$

which makes use of the information of the current θ_k. For the control problem to make sense, it is assumed that system (11.1) is controllable and observable for all admissible θ_k, [SM67, BBS03].

To compute the control law (11.3) within a model predictive control scheme, a finite-horizon cost function is to be minimized. We will optimize over finite-horizon *control policies*

$$\boldsymbol{\pi}_N := \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$$

under the influence of the unknown *sequence of future scheduling parameters*

$$\boldsymbol{T}_k := \{\theta_{k+1}, \ldots, \theta_{k+N-1}\} \ .$$

According to standard MPC, our cost function is defined as

$$J(\boldsymbol{\pi}_N; x_k, \theta_k, \boldsymbol{T}_k) = \|Px_{k+N}\|_p + \sum_{i=0}^{N-1} \|Qx_{k+i}\|_p + \|Ru_{k+i}\|_p\,, \tag{11.4}$$

where p denotes a polyhedral norm, e.g. the 1-norm or the ∞-norm. Polyhedral norms[1] enable a parametric solution to the stated problem using dynamic programming. For the minimization of the cost function (11.4) we have to consider the current as well as the unknown future parameter values, as the state trajectories are parameter-dependent.

In a closed-loop MPC approach (cf. Section 4.3), one assumes that the future control action u_{k+1} is calculated optimally over the horizon $N-1$ not until x_{k+1} and θ_{k+1} are available. But as the future values of the scheduling parameters are unknown, all possible cases must be regarded in order to accommodate for the worst-case scenario. This way it is assured that the actual cost function will be less or equal to the computed one, no matter how the scheduling parameters evolve. The optimization problem to solve in a

[1]Quadratic cost functions are not possible since our procedure relies on epigraph reformulations, which would render the original problem a parametric quadratically constrained quadratic program, for which no efficient solution techniques are known.

closed-loop MPC approach is thus

$$\begin{aligned}
\mu^*(x_k,\theta_k) = \ & \arg\min_{\mu_0}\max_{\theta_{k+1}}\min_{\mu_1}\cdots\max_{\theta_{k+N-1}}\min_{\mu_{N-1}} J(\boldsymbol{\pi}_N,\boldsymbol{T}_k;x_k,\theta_k) && (11.5\text{a})\\
\text{s.t.}\ \ & \forall i\in\{0,\dots,N-1\}\\
& x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i},\theta_{k+i})\,, && (11.5\text{b})\\
& \mu_i(x_{k+i},\theta_{k+i}) \in \mathcal{U}\,, && (11.5\text{c})\\
& x_{k+i}\in\mathcal{X}\,, && (11.5\text{d})\\
& \theta_{k+i}\in\Theta\,. && (11.5\text{e})
\end{aligned}$$

Remark 11.3 *It is straightforward to add terminal state constraints $x_{k+N}\in\mathcal{X}_T$ to the optimization problem* (11.5)*. However, for complexity reasons we recommend not to use terminal state constraints, but to verify stability a-posteriori (see the discussion in Section 11.3).*

11.2 Computation of Explicit MPC Controllers

Following the dynamic programming procedure from Section 10.2, we would obtain an optimization problem similar to (10.11), but with the scheduling parameter appearing in the constraint matrix, $C_{\mu t} = C_{\mu t}(\theta_{k+i})$. Parametric optimization problems with a parameter-dependent constraint matrix are known to be difficult problems, and are far less understood than parametric linear programs with a parameter-dependent objective function and right-hand side, [BGK+82]. Some interesting aspects were presented in [KP03] for the case of a single parameter. The solution for a single critical region was computed by solving an extended linear program. This solution however turns out to be a rational function of the scheduling parameter with a degree equal to the number of active constraints. Each coefficient of this rational function is an optimization variable in the linear program, which would render the optimal control laws very complex, already for tiny LPV systems. Moreover, the critical regions are generally not polyhedra, and the optimal cost function is not a convex piecewise affine function. These obstacles cumber the computation, the storage and the efficient evaluation of the optimal solution and render the optimal solution to (11.5) impractical.

Consequently we are proposing a *suboptimal* solution by restricting the parametrization of the input. In the following we will assume the control law to be an affine function of the scheduling parameter,

$$u_{k+i} = \mu_i(x_{k+i},\theta_{k+i}) = \sum_{j=1}^{n_\theta}\theta_{k+i,j}\mu_{i,j}(x_{k+i})\,. \qquad (11.6)$$

Remark 11.4 *Note that in principle any polynomial in the parameter θ_k and also rational functions are possible with our proposed method. The choice of the input parametrization is a tradeoff between solution complexity and control performance, and depends on the problem at hand. For an investigation of different input parametrizations see also Example 11.1 on page 187.*

Remark 11.5 *It is also possible to choose different parametrizations at different steps of the dynamic programming procedure. One promising approach is e.g. to choose the predicted future control laws to be parameter-independent, and to use an affine parametrization for the actually applied control law in the final DP step.*

In the case of the affine parametrization (11.6), the function $\mu_{i,j}(x_{k+i})$ corresponds to the control law in the jth vertex of the parameter simplex (11.1c). In order to simplify notation, we introduce the basis

$$\boldsymbol{U}_i := \{\mu_{i,1}, \mu_{i,2}, \dots, \mu_{i,n_\theta}\}\,.$$

Instead of minimizing over the control policy $\boldsymbol{\pi}_N = \{\mu_0, \mu_1, \dots, \mu_{N-1}\}$, we are minimizing over the policy of basis functions $\{\boldsymbol{U}_0, \boldsymbol{U}_1, \dots, \boldsymbol{U}_{N-1}\}$. Since these basis functions are parameter-independent, the considered problem becomes a robust min-max optimization problem. Eventually, the optimization problem to solve in closed-loop MPC can be stated as

$$\begin{aligned}
\mu^*(x_k, \theta_k) = {} & \arg\min_{\mu_0} \min_{\boldsymbol{U}_1} \max_{\theta_{k+1}} \cdots \min_{\boldsymbol{U}_{N-1}} \max_{\theta_{k+N-1}} J(\boldsymbol{\pi}_N, \boldsymbol{T}_k; x_k, \theta_k) && \text{(11.7a)}\\
\text{s.t.}\quad & \forall i \in \{0, \dots, N-1\} \\
& x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i}, \theta_{k+i})\,, && \text{(11.7b)}\\
& \mu_i(x_{k+i}, \theta_{k+i}) \in \mathcal{U}\,, && \text{(11.7c)}\\
& x_{k+i+1} \in \mathcal{X}\,, && \text{(11.7d)}\\
& \theta_{k+i} \in \Theta\,. && \text{(11.7e)}
\end{aligned}$$

Analogue to Section 10.2, we solve (11.7) by a *dynamic programming* (DP) procedure iterating backwards in time. We start at the prediction horizon N with the initial cost-to-go function

$$J_N^*(x_{k+N}) = \|Px_{k+N}\|_p\,. \tag{11.8}$$

Then at each iteration we use

$$x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i}, \theta_{k+i}) \tag{11.9}$$

to substitute x_{k+i+1} in $J_{i+1}^*(x_{k+i+1})$. As θ_{k+i} is unknown at time instance k, we consider the worst case, which leads to

$$
\begin{aligned}
J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \min_{\boldsymbol{U}_i} \max_{\theta_{k+i}} \|R\mu_i(x_{k+i},\theta_{k+i})\|_p + J_{i+1}^*(x_{k+i+1}) \quad && \text{(11.10a)}\\
\text{s.t.} \quad x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i},\theta_{k+i})\,, && \text{(11.10b)}\\
\mu_i(x_{k+i},\theta_{k+i}) \in \mathcal{U} \quad \forall\,\theta_{k+i} \in \Theta\,, && \text{(11.10c)}\\
x_{k+i+1} \in \mathcal{X}_{i+1} \quad \forall\,\theta_{k+i} \in \Theta\,, && \text{(11.10d)}\\
x_{k+i} \in \mathcal{X}\,, && \text{(11.10e)}\\
\theta_{k+i} \in \Theta\,, && \text{(11.10f)}
\end{aligned}
$$

In order to determine the worst-case parameters of (11.10), we first apply an epigraph reformulation to the optimization problem in order to transfer the parameter dependence to the constraints. Introducing the epigraph variable $t_i(x_{k+i})$ this leads to the following semi-infinite optimization problem

$$
\begin{aligned}
J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \min_{\{\boldsymbol{U}_i,t_i\}} t_i(x_{k+i}) && \text{(11.11a)}\\
\text{s.t.} \quad x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i},\theta_{k+i})\,, && \text{(11.11b)}\\
\|R\mu_i(x_{k+i},\theta_{k+i})\|_p + J_{i+1}^*(x_{k+i+1}) \le t_i(x_{k+i}) \quad \forall\,\theta_{k+i} \in \Theta\,, && \text{(11.11c)}\\
\mu_i(x_{k+i},\theta_{k+i}) \in \mathcal{U} \quad \forall\,\theta_{k+i} \in \Theta\,, && \text{(11.11d)}\\
x_{k+i+1} \in \mathcal{X}_{i+1} \quad \forall\,\theta_{k+i} \in \Theta\,, && \text{(11.11e)}\\
x_{k+i} \in \mathcal{X}\,, && \text{(11.11f)}\\
\theta_{k+i} \in \Theta\,. && \text{(11.11g)}
\end{aligned}
$$

Remark 11.6 *There exist two special cases which facilitate a different solution approach to* (11.11)*. If the control law is chosen to be parameter-independent, the constraints in* (11.11) *depend affinely on the scheduling parameter, such that those constraints are fulfilled if and only if they are fulfilled at the vertices of the parameter simplex. Thus they can be substituted by the vertex constraints to yield a parameter-independent pLP. For a constant input matrix B with parameter-varying control law, the resulting constraints depend affinely on the scheduling parameter, which again allows for a substitution by the vertex constraints. This case was tackled in detail in Chapter 10.*

In general, we will be faced with a non-constant input matrix $B(\theta_k)$ and an affinely or even polynomially parameterized input, $u_{k+i} = \mu_i(x_{k+i},\theta_{k+i})$. Hence the constraints of (11.11) are polynomial in the scheduling parameters and the vertex constraints are *not* sufficient to ensure constraint satisfaction over the whole simplex. However, the constraint satisfaction of the semi-infinite optimization problem (11.11) can be ensured, conservatively, over the whole parameter simplex following Pólya's theorem.

We will make use of the more obvious reverse of Pólya's theorem[2], i.e. positive coefficients of the extended polynomial mean positivity over the whole simplex. This reverse is also known as Pólya's relaxation and is elaborated in Section 2.2. If all coefficients $\mathbf{C}_{N_p}$ of the extended polynomial p_{N_p} are non-negative, the constraints (11.11c) – (11.11e) are satisfied. Hence the semi-infinite optimization problem (11.11) can be transformed into the following parametric linear program:

$$\begin{aligned} J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + &\min_{\{\boldsymbol{U}_i, t_i\}} t_i(x_{k+i}) && \text{(11.12a)} \\ \text{s.t.} \quad &\mathbf{C}_{N_p}(\boldsymbol{U}_i, t_i; x_{k+i}) \geq 0\,, && \text{(11.12b)} \\ &x_{k+i} \in \mathcal{X}\,. && \text{(11.12c)} \end{aligned}$$

Note that the coefficients $\mathbf{C}_{N_p}$ of the extended polynomial lie in the cone which is spanned by the coefficients of the polynomial constraints in (11.11), and the piecewise affine dependence of the coefficients on the state is preserved.

Remark 11.7 *The choice of the Pólya degree N_p for the Pólya relaxation is a trade-off between solution complexity and introduced conservatism. The larger the Pólya degree, the less conservative the relaxation, but also the greater the number of constraints in* (11.12). *See also Section 2.2, or Example 11.1 for a comparison of different Pólya relaxations.*

By using polyhedral norms, the optimal cost-to-go functions J_i^* are convex piecewise affine functions of the state x_{k+i}, such that in every iteration the optimization problem (11.11) can be formulated as the parametric linear program (11.12) and solved parametrically with respect to x_{k+i}. Contrary to the closed-loop MPC approach for uncertain systems, the future inputs are functions of the future scheduling parameters.

Remark 11.8 *Instead of using an affine or even polynomial input parametrization, it is also possible to select a* parameter-independent control law *in all but the final dynamic programming iterations, following a robust approach. This approach typically results in less complex control laws, since instead of Pólya's relaxation a vertex enumeration as in the case of LPV-A systems is sufficient, and since* (11.6) *simplifies to a single control law. On the other hand, this approach will result in a more conservative control law, since the predicted future inputs are assumed to be parameter-independent (see also Example 11.2).*

The *final step of the dynamic programming procedure* differs from the preceding steps, since knowledge of the current scheduling parameter values can be

[2]The presented usage of Pólya's theorem is implemented in YALMIP as one of the so-called filters in the robust optimization framework, [Löf08].

exploited to improve control performance. The problem we want to solve is

$$\begin{aligned}
\mu^*(x_k,\theta_k) = \arg\min_{\mu} \quad & \|R\mu(x_k,\theta_k)\|_p + J_1^*(x_{k+1}) && (11.13a)\\
\text{s.t.} \quad & x_{k+1} = A(\theta_k)x_k + B(\theta_k)\mu(x_k,\theta_k)\,, && (11.13b)\\
& \mu(x_k,\theta_k) \in \mathcal{U} \quad \forall\,\theta_k \in \Theta\,, && (11.13c)\\
& x_{k+1} \in \mathcal{X}_1 \quad \forall\,\theta_k \in \Theta\,, && (11.13d)\\
& x_k \in \mathcal{X}\,, && (11.13e)\\
& \theta_k \in \Theta\,, && (11.13f)
\end{aligned}$$

which is again a parametric optimization problem with a parametric constraint matrix. Yielding an explicit solution to (11.13) is more demanding than in the previous DP steps, since we are not only interested in minimizing the worst-case cost, but in minimizing the costs for all possible scheduling parameter values. The input parametrization should not only be able to cope with the worst-case scheduling parameter, but be as non-restrictive as possible for all scheduling parameter values. Ideally, one would prefer the control law to consider only the measured scheduling parameter θ_k, and not be conservative due to other possible values of the scheduling parameter.

One possible approach which has this property, is of course to store the cost-to-go $J_1^*(x_{k+1})$ and to solve

$$\begin{aligned}
u_k^* = \arg\min_{\{u_k,t_k\}} \quad & t_k && (11.14a)\\
\text{s.t.} \quad & x_{k+1} = A(\theta_k)x_k + B(\theta_k)u_k\,, && (11.14b)\\
& \|Ru_k\|_p + J_1^*(x_{k+1}) \le t_k\,, && (11.14c)\\
& u_k \in \mathcal{U}\,, && (11.14d)\\
& x_{k+1} \in \mathcal{X}_1\,, && (11.14e)
\end{aligned}$$

online at each sampling instance when the scheduling parameter is known. Since the linear program (11.14) is minimizing only over the current control action, the number of optimization variables is low and (11.14) can be solved quite quickly. Moreover, there exist dual LP solvers which are tailor-made for problems as (11.14), with a small number of optimization variables and a potentially large number of constraints.

Computing an entirely explicit solution without the need to solve linear programs online is more complicated. Again we are dealing with a parametric optimization problem with parameters in the constraint matrix. Due to the scheduling parameter in the input matrix $B(\theta)$, it is unfortunately not possible to remove the scheduling parameter entirely from the optimization problem by solving the problem parametrically in the uncontrolled successor state as it was possible in the case of LPV-A systems. By the introduction of an input parametrization as in (11.6), some conservatism is introduced. To counteract, one can choose a more flexible input parametrization in the final DP step, to the price of a more complex control law. Finally, there is still the need to transform the optimization problem (11.13) to a parametric linear program, which can be solved efficiently.

There exist several options on how to select the input parametrization and how to transform the optimization problem. This selection is a tradeoff between control performance and complexity, and unfortunately, there is no distinct superior approach. We list three options here.

1. *Grid-and-interpolate*
   One option is to grid the parameter simplex and to compute one explicit control law for each grid point. Online these control laws are then interpolated. In contrast to the other two options, each control law is determined based on its local grid point only. Additional measures are required to ensure constraint satisfaction for intermediate values of the scheduling parameter.

   This approach offers flexibility in terms of the dependence on the scheduling parameter; the interpolated control law is a piecewise affine function of the scheduling parameter. On the other hand it requires a relatively large storage space, since for each grid point a separate control law is needed.

2. *Average cost minimization*
   Another option is to consider again a grid of scheduling parameter values, but to fit a single parametrized control law to the averaged costs at those grid points. Unfortunately, in this approach the limited structure in the scheduling parameter is compensated by creating many regions in the state space, resulting again in complex control laws. In order to counteract one can decrease the number of grid points. Possible grid points are e.g. the vertices of the parameter simplex, resulting in the optimization problem

$$J^*(z_k) = \min_{\{U_0, t\}} \; t(z_k) \tag{11.15a}$$

$$\text{s.t.} \quad \frac{1}{n_\theta} \sum_{j=1}^{n_\theta} \|R\mu(z_k, e_j)\|_p + \|J_1^*(z_k + B_j \mu(z_k, e_j))\|_p \leq t(z_k)\,, \tag{11.15b}$$

$$\mu(z_k, \theta_k) \in \mathcal{U} \quad \forall\, \theta_k \in \Theta\,, \tag{11.15c}$$

$$z_k + B(\theta_k)\mu(z_k, \theta_k) \in \mathcal{X}_1 \quad \forall\, \theta_k \in \Theta\,. \tag{11.15d}$$

   This approach makes use of the *uncontrolled successor state* z_k from Equation (10.17).

3. *Worst-case cost minimization*
   Our preferred approach makes also use of the uncontrolled successor state, combined with a worst-case minimization. By parameterizing the parametric problem not in the measured state x_k, but in the uncontrolled successor state z_k, the parameter dependence of $A(\theta_k)$ can directly be taken into account. In lack of an equivalent scheme for the input, we employ the input parametrization

$$u_k = \mu(z_k, \theta_k) = \sum_{j=1}^{n_\theta} \theta_{k,j}\mu_{0,j}(z_k) \tag{11.16}$$

   and minimize the worst-case gain-scheduled cost, i.e. solve the semi-infinite optimization problem

$$\begin{aligned} J^*(z_k) = \min_{\{\boldsymbol{U}_0,t,s\}} \quad & t(z_k) + s(z_k) && (11.17a)\\ \text{s.}t. \quad & \|R\mu(z_k,\theta_k)\|_p \ + J_1^*(z_k + B(\theta_k)\mu(z_k,\theta_k)) \leq t(z_k) \quad \forall\, \theta_k \in \Theta\,, && (11.17b)\\ & \mu(z_k,\theta_k) \in \mathcal{U} \quad \forall\, \theta_k \in \Theta\,, && (11.17c)\\ & z_k + B(\theta_k)\mu(z_k,\theta_k) \in \mathcal{X}_1 \quad \forall\, \theta_k \in \Theta\,, && (11.17d)\\ & \epsilon \sum_{j=1}^{n_\theta} \|Q(z_k + B_j\mu(z_k,e_j))\|_p \leq s(z_k)\,, && (11.17e) \end{aligned}$$

   where s denotes an additional epigraph variable and e_j the jth standard basis vector. For many problems, the worst-case cost depend solely on a single control law $\mu_{0,j}(z_k)$ at one vertex of the parameter simplex. The remaining optimization variables are not unique, when solving the parametric problem (11.17), which can lead to undesired control laws at the other vertices of the parameter simplex. Therefore a small regularization weight, $0 < \epsilon \ll 1$, which penalizes the sum of the successor states at all vertices of the parameter simplex, is added as an ad-hoc measure. While this virtually does not change the shape of the cost function, it turns out that the actual achieved performance is improved, because the non-uniqueness of the vertex solutions is mitigated. The polynomial dependence on the scheduling parameter can again be treated by employing Pólya's theorem to transform (11.17) into a parametric linear program.

   In our test simulations, this approach turned out to be a good tradeoff between complexity and control performance. Note however that it is merely a heuristic, and for specific systems other approaches might be more successful. Moreover, the use of the uncontrolled successor state might decrease the feasible space, since artificial combinations of state-update and input matrices are considered.

When applying the computed control law online, in each step the state x_k and the scheduling parameter θ_k are measured and used to compute the uncontrolled successor state z_k, which is then inserted in the parametric solution to obtain the control input u_k. Analogue to Section 10.2 is the set of feasible uncontrolled successor states,

$$\mathcal{Z}_f = \{z \in \mathbb{R}^{n_x} \mid E_z z \leq f_z\}, \tag{11.18}$$

and the set of *admissible* initial states, i.e. which result in a feasible uncontrolled successor state for all parameter values,

$$\mathcal{X}_f = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; \begin{bmatrix} E_z A_1 \\ \vdots \\ E_z A_{n_\theta} \end{bmatrix} x \leq \begin{bmatrix} f_z \\ \vdots \\ f_z \end{bmatrix} \right\}. \tag{11.19}$$

11.3 Stability

This section is concerned with stability of the resulting closed-loop system, when explicit control laws are applied to LPV systems. Note that the proposed procedure does not guarantee stability a-priori, which is a classical issue in finite-horizon MPC. As discussed in Section 4.5, there are known variations of the model predictive control scheme which can be employed, for example *dual mode MPC* or the (overly conservative) *terminal equality constraint*. One can guarantee (i) asymptotic stability, (ii) constraint satisfaction, and (iii) recursive feasibility *a-priori* for all feasible states, by considering a *dual mode* approach and by choosing the terminal state constraints $\mathcal{X}_T$ and the polyhedral terminal cost $L_N(x_{k+N})$ appropriately, [MRRS00]. From [MRRS00] we have the following conditions for asymptotic stability:

A1: $\mathcal{X}_T \subseteq \mathcal{X}, \mathcal{X}_T$ closed and contains the origin.

A2: $\mu_T(x_k, \theta_k) \in \mathcal{U}\ \forall x_k \in \mathcal{X}_T\ \forall \theta_k \in \Theta$.

A3: $x_{k+1} = A(\theta_k)x_k + B(\theta_k)\mu_T(x_k, \theta_k) \in \mathcal{X}_T\ \forall x_k \in \mathcal{X}_T\ \forall \theta_k \in \Theta$

A4: $L_N(x_k) - L_N(x_{k+1}) \leq \|Qx_k\|_p + \|R\mu_T(x_k, \theta_k)\|_p\ \forall x_k \in \mathcal{X}_T\ \forall \theta_k \in \Theta$.

Furthermore, it is a well-known fact, that stability is preserved in the case of a suboptimal solution, as long as the suboptimality of the cost function does not exceed one stage cost, [SMR99]. Consider the following procedure, which is based on [BLM09] and [BBM02]:

1. Compute an asymptotically stabilizing terminal region parameter-varying state-feedback controller

$$u_k = \mu_T(x_k, \theta_k) = K(\theta_k)x_k = \sum_{j=1}^{n_\theta} K_j \theta_{k,j} x_k \tag{11.20}$$

   for the unconstrained system (11.1), e.g. by the procedure in Section 13.

2. Determine a polytopic λ-contractive terminal region $\mathcal{X}_T$ by pre-image computations, such that

$$\forall x_k \in \mathcal{X}_T, \forall \theta_k \in \Theta \ \exists \mu_T(x_k, \theta_k) \in \mathcal{U} : A(\theta_k)x_k + B(\theta_k)\mu_T(x_k, \theta_k) \in \lambda \mathcal{X}_T \tag{11.21}$$

   holds for some $\lambda \in [0, 1)$.

3. Scale the Minkowski function

$$\psi_{\mathcal{X}_T}(x_k) := \min_\alpha \{\alpha \in \mathbb{R}_+ \mid x_k \in \alpha \mathcal{X}_T\}, \tag{11.22}$$

   induced by the terminal region $\mathcal{X}_T$, by a factor $\beta^* \in \mathbb{R}_+$, which can be determined by the linear program

$$\beta^* = \min_\beta \beta \tag{11.23a}$$

$$\begin{aligned} \text{s.t. } & \beta(1 - \lambda) \geq \|Qv_i\|_p + \|RK_j v_i\|_p \\ & \qquad \forall v_i \in \text{vert}(\mathcal{X}_T) \ \forall j \in \{1, \dots, n_\theta\}. \end{aligned} \tag{11.23b}$$

4. Define $L_N(x_{k+N}) := \beta^* \psi_{\mathcal{X}_T}(x_{k+N})$.

Theorem 11.1 *Assume that there exists a terminal region control $\mu_T(x_k, \theta_k)$ of the form* (11.20)*, which renders the polytope $\mathcal{X}_T$ λ-contractive as in* (11.21) *with $\lambda \in [0, 1)$. Then, for the terminal region $\mathcal{X}_T$ and the terminal cost $L_N(x_k) = \beta^* \psi_{\mathcal{X}_T}(x_k)$, as defined in* (11.22)-(11.23)*, the conditions A1 to A4 are satisfied, such that we have asymptotic stability, constraint satisfaction and recursive feasibility for all feasible points.*

Proof Conditions A1 to A3 follow immediately from the properties of the λ-contractive terminal set $\mathcal{X}_T$. Condition A4 follows from

$$\begin{aligned} & \beta \psi_{\mathcal{X}_T}(x_k) - \beta \psi_{\mathcal{X}_T}(x_{k+1}) \geq \|Qx_k\|_p + \|RK(\theta_k)x_k\|_p \quad \forall x_k \in \mathcal{X}_T \ \forall \theta_k \in \Theta \\ \Leftarrow \ & \beta(1 - \lambda)\psi_{\mathcal{X}_T}(x_k) \geq \|Qx_k\|_p + \|RK(\theta_k)x_k\|_p \quad \forall x_k \in \mathcal{X}_T \ \forall \theta_k \in \Theta \end{aligned}$$

$$\begin{aligned}
&\Leftarrow \beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \|Qx_k\|_p + \sum_{j=1}^{n_\theta} \theta_{k,j}\|RK_jx_k\|_p \quad \forall x_k \in \mathcal{X}_T \; \forall \theta_k \in \Theta \\
&\Leftrightarrow \sum_{j=1}^{n_\theta} \theta_{k,j}\beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \sum_{j=1}^{n_\theta} \theta_{k,j}\|Qx\|_p + \sum_{j=1}^{n_\theta} \theta_{k,j}\|RK_jx_k\|_p \\
&\qquad \forall x_k \in \mathcal{X}_T \; \forall \theta_k \in \Theta \\
&\Leftrightarrow \beta(1-\lambda)\psi_{\mathcal{X}_T}(x_k) \geq \|Qx_k\|_p + \|RK_jx_k\|_p \quad \forall x_k \in \mathcal{X}_T \; \forall j \in \{1,\dots,n_\theta\} \\
&\Leftrightarrow \beta(1-\lambda) \geq \|Qv_i\|_p + \|RK_jv_i\|_p \quad \forall v_i \in \mathrm{vert}(\mathcal{X}_T) \; \forall j \in \{1,\dots,n_\theta\} \qquad \blacksquare
\end{aligned}$$

Note that a symmetric terminal set $\mathcal{X}_T$ implies a Minkowski function which can be expressed as $L_N(x_{k+N}) = \|Px_{k+N}\|_\infty$ with some matrix P. However, the described procedures work also for convex polyhedral piecewise affine terminal cost $L_N(x_{k+N})$.

The drawback of adding terminal state constraints is that in general they lead to a loss of performance, a smaller feasible space and an increase in complexity of the resulting control law. While the former two effects can be mitigated by extending the prediction horizon, this typically leads to a further increase in the complexity of the control law. See Example 11.5 for possible consequences of ensuring stability a-priori. In order to avoid these downsides in practical implementations, a possibility is to omit the terminal state constraints and to verify stability a-posteriori, following the theory described in [BM08b].

Fortunately, stability of the control law (11.16) can also be verified a-posteriori by performing a reachability analysis of the closed-loop system in the space of the uncontrolled successor state towards a λ-contractive target region $\mathcal{T}$ around the origin, [BM08b, Bla94, Bla95]. The closed-loop system is given by

$$\begin{aligned}
z_{k+1} &= A(\theta_{k+1})x_{k+1} = A(\theta_{k+1})\{z_k + B(\theta_k)\mu(z_k,\theta_k)\} \\
&= \begin{cases} A(\theta_{k+1})(I + B(\theta_k)F_1(\theta_k))z_k + A(\theta_{k+1})B(\theta_k)g_1(\theta_k)\,, & z_k \in \mathcal{R}_1 \\ \vdots & \vdots \\ A(\theta_{k+1})(I + B(\theta_k)F_{n_r}(\theta_k))z_k + A(\theta_{k+1})B(\theta_k)g_{n_r}(\theta_k)\,, & z_k \in \mathcal{R}_{n_r} \end{cases}
\end{aligned} \tag{11.24}$$

where the polyhedral controller regions in the space of the uncontrolled successor state z_k are denoted by $\mathcal{R}_1,\dots,\mathcal{R}_{n_r}$. If the uncontrolled successor state is not utilized in the control law, a similar stability analysis can be performed in the regular state space. In the following we will focus on the regulator problem, i.e. the stabilization to the origin. Note that all other reference values in the state space correspond to a set of reference values in z–space, depending on the scheduling parameter θ_k. Asymptotic stability to reference values $\bar{x} \neq 0$ requires constant scheduling parameter values (Scenario ($S3$)).

For varying scheduling parameter values, instead of asymptotic stability to some reference value $\bar{x} \neq 0$, only ultimate boundedness to a target set can be certified.

Definition 11.1 *Let $\boldsymbol{I}$ be the index set of all controller regions containing the origin,*

$$\boldsymbol{I} := \{r \in \{1, \ldots, n_r\} \mid 0 \in \mathcal{R}_r\} \ .$$

The index set $\boldsymbol{I}$ is single-valued if the origin is contained in the interior of a controller region, and multi-valued if the origin lies on the facet of several controller regions.

The stability analysis can be performed in three steps; the first verifies the origin to be an equilibrium point, the second considers the stability of the target region $\mathcal{T}$, and in the third a reachability analysis is performed.

1. *Invariance of the origin*
   At first we require the origin to be invariant, i.e. to be an equilibrium of the closed-loop system. This is the case, if the condition

   $$g_r(\theta_k) = 0 \quad \forall \theta_k \in \Theta\,,\ r \in \boldsymbol{I} \tag{11.25}$$

   holds. Since $\mathcal{X}$ and $\mathcal{U}$ include the origin, there always exists a controller which fulfils this condition. If Condition (11.25) is violated, the origin is *not* an equilibrium point, and at most ultimate boundedness to a target set can be present.

2. *Contractiveness of the target region*
   We infer asymptotic stability of the origin from the existence of a λ-contractive target region $\mathcal{T}$. The contractive preset of a set $\mathcal{Z}_i \subset \mathbb{R}^{n_x}$ w.r.t. the closed-loop system (11.24) is given by

   $$\Omega(\mathcal{Z}_i, \mathcal{R}_r) = \left\{ z_k \in \mathcal{R}_r \,\middle|\, \begin{array}{c} A_j(I + B(\theta_k)F_r(\theta_k))z_k + A_jB(\theta_k)g_r(\theta_k) \in \lambda\mathcal{Z}_i \\ j = 1, \ldots, n_\theta\,, \quad \forall \theta_k \in \Theta \end{array} \right\}, \tag{11.26}$$

   where $\lambda \in [0, 1)$ denotes the *contraction ratio*. For an efficient treatment of the sets $\mathcal{Z}_i$ we require them to be polytopes. Since the contractive presets $\Omega(\mathcal{Z}_i, \mathcal{R}_r)$ are convex but not polytopes, we determine the polytopic preset $\Omega_{N_p}(\mathcal{Z}_i, \mathcal{R}_r)$ of $\mathcal{Z}_i$ by applying Pólya's relaxation to (11.26). If the index set $\boldsymbol{I}$ of regions containing the origin is single-valued, a robust invariant target set can be determined by a set backpropagation as in Algorithm 2.1, starting with $\mathcal{Z}_0 = \mathcal{R}_r$ and repeatedly computing the polytopic presets. If the index set $\boldsymbol{I}$ is multi-valued, the backpropagation runs in parallel for each regarded region $r \in \boldsymbol{I}$, starting with a polytope $\mathcal{Z}_0 \subseteq \bigcup_{r\in \boldsymbol{I}} \mathcal{R}_r$ containing the origin in the interior and determining the largest polytope contained in the union of the polytopic presets after each iteration $\mathcal{Z}_{i+1} \subseteq \bigcup_{r\in \boldsymbol{I}} \Omega(\mathcal{Z}_i, \mathcal{R}_r)$.

If $\mathcal{Z}_i \subseteq \mathcal{Z}_{i+1}$, the algorithm is terminated with $\mathcal{T} := \mathcal{Z}_i$, and the resulting target region $\mathcal{T}$ is λ-contractive with regard to the closed-loop system (11.24). It follows from the succeeding proposition that the existence of a λ-contractive polytope $\mathcal{T}$ induces asymptotic stability of the origin.

Proposition 11.2 *Let $\mathcal{T} \subseteq \bigcup_{r\in \boldsymbol{I}} \mathcal{R}_r \subset \mathbb{R}^{n_x}$ be a polytope containing the origin in its interior and let Condition* (11.25) *hold. If $\forall r \in \boldsymbol{I}$, all vertices v_r^i of $\mathcal{T} \bigcap \mathcal{R}_r$ are mapped into $\lambda\mathcal{T}$, $0 \leq \lambda < 1$, then $\mu\mathcal{T}$ is λ-contractive $\forall \mu \in [0,1]$.*

Proof Consider any $\hat{z}_k \in \mu(\mathcal{T} \bigcap \mathcal{R}_r) \Rightarrow \tilde{z}_k = \hat{z}_k/\mu \in (\mathcal{T} \bigcap \mathcal{R}_r) \Leftrightarrow \exists \alpha_r^i \in \mathbb{R}_+, \sum_i \alpha_r^i = 1 : \tilde{z}_k = \sum_i \alpha_i v_r^i \Rightarrow \hat{z}_k = \mu \sum_i \alpha_i v_r^i \Rightarrow \hat{z}_{k+1} \in \mu \sum_i \alpha_i \lambda \mathcal{T} = \mu\lambda\mathcal{T}$. ∎

Proposition 11.2, together with the properties of the gauge function $\psi_{\mathcal{T}}(z)$ induced by $\mathcal{T}$ suffices to establish asymptotic stability inside $\mathcal{T}$ by using $\psi_{\mathcal{T}}(z)$ as a Lyapunov function.

3. *Reachability analysis*
A reachability analysis can be performed to check which states are mapped into the target set $\mathcal{T}$ under the computed control law. In this reachability analysis, the backpropagation algorithm 2.1 is performed, using again the polytopic preset arising from applying Pólya's relaxation to (11.26). We start the backpropagation with $\mathcal{Z}_0 = \mathcal{T}$. In each iteration i we compute $\Omega_{N_p}(\mathcal{Z}_i, \mathcal{R}_r)$, $r = 1, \ldots, n_r$ and merge the resulting presets to $\mathcal{Z}_{i+1} = \bigcup_{r=1}^{n_r} \Omega_{N_p}(\mathcal{Z}_i, \mathcal{R}_r)$.

   The iterations terminate when $\mathcal{Z}_{i+1} \subseteq \mathcal{Z}_i$ or when the entire feasible space is covered. The resulting region of attraction is denoted by $\mathcal{Z}_\infty$. All uncontrolled successor states $z_k \in \mathcal{Z}_\infty$ are controlled to the target set $\mathcal{T}$ and eventually to the origin by construction. The required procedures to perform the backpropagations boil down to polytopic manipulations and can be adapted from the algorithms given in the references mentioned above.

In the following, we will show that stability properties of the uncontrolled successor state z_k imply stability properties of the actual state x_k.

Lemma 11.3 *The uncontrolled successor state obeys the following upper and lower bounds:*

(a) *For every LPV system* (11.1) *there exists an* $\epsilon_1 > 0$*, such that the uncontrolled successor state* (10.17) *can be bounded by*

$$\|z_k\| \le \epsilon_1 \|x_k\| \,. \tag{11.27a}$$

(b) *Under the assumption of* (11.25) *and for a small-enough* $\delta_2 > 0$*, there exists an* $\epsilon_2 > 0$ *such that*

$$\|z_k\| < \delta_2 \quad \Rightarrow \quad \|x_{k+1}\| \le \epsilon_2 \|z_k\| \tag{11.27b}$$

holds.

Proof (a) Using the definition of the uncontrolled successor state, it follows that

$$\|z_k\| = \|A(\theta_k)x_k\| \le \|A(\theta_k)\|\|x_k\| \le \max_{\theta_k \in \Theta} \|A(\theta_k)\|\|x_k\| =: \epsilon_1 \|x_k\| \,.$$

(b) If δ_2 is small enough, the uncontrolled successor state is an element of $\bigcup_{r\in \boldsymbol{I}} \mathcal{R}_r$. Using the state-update equation, the control law (11.3) and Condition (11.25), we infer

$$\begin{aligned}\|x_{k+1}\| &= \|z_k + B(\theta_k)\mu(z_k,\theta_k)\| \\ &\le \| \max_{r\in \boldsymbol{I},\theta_k\in\Theta} (I + B(\theta_k)F_r(\theta_k))z_k\| \\ &\le \max_{r\in \boldsymbol{I},\theta_k\in\Theta} \|(I + B(\theta_k)F_r(\theta_k))\|\|z_k\| =: \epsilon_2\|z_k\| \,.\end{aligned}$$ ■

Proposition 11.4 *Lyapunov stability and asymptotic stability of* $x_k = 0$ *follow directly from Lyapunov stability and asymptotic stability of* $z_k = 0$*, respectively.*

Proof Proposition 11.4 is plain, if the state-update matrix $A(\theta)$ is invertible. If $A(\theta)$ is singular, it possesses a (parameter-dependent) null space, containing states x_k which are not necessarily zero if $z_k = 0$. Fortunately, we just need to let pass one time step in order to show Proposition 11.4 in this case.

With Lemma (11.3) and under the assumption of $\epsilon \le \delta_2$, we obtain from the Lyapunov stability of $z_k = 0$:

$$\begin{aligned}\|x_k\| \le \delta' := \frac{\delta}{\epsilon_1} \quad &\Rightarrow \quad \|z_k\| \le \delta \\ &\Rightarrow \quad \|z_{k+i}\| \le \epsilon \quad \forall i \in \mathbb{N} \\ &\Rightarrow \quad \|x_{k+i+1}\| \le \epsilon_2\epsilon =: \epsilon' \quad \forall i \in \mathbb{N} \,.\end{aligned}$$

Similarly, we infer asymptotic stability of $x_k = 0$ from asymptotic stability of $z_k = 0$, using again Lemma (11.3),

$$\begin{aligned} \|x_k\| \leq \delta' := \frac{\delta}{\epsilon_1} \quad &\Rightarrow \quad \|z_k\| \leq \delta \\ &\Rightarrow \quad \lim_{k\to\infty} \|z_k\| = 0 \\ &\Rightarrow \quad \lim_{k\to\infty} \epsilon_2 \|x_{k+1}\| = 0 \\ &\Leftrightarrow \quad \lim_{k\to\infty} \|x_k\| = 0\,. \end{aligned}$$

Note that Condition (11.25) has to be fulfilled for the origin to be an equilibrium point of the closed-loop system (11.24), and thus already for the stability of $z_k = 0$. ■

Remark 11.9 *The assumption $\epsilon \leq \delta_2$ is not a necessary condition for Lyapunov stability of $x_k = 0$. There will always be a small-enough ϵ', such that $\epsilon \leq \delta_2$ is fulfilled.*

11.4 Numerical Examples

This section consists of four examples which show different aspects and variations of the proposed explicit MPC scheme. In the first example different input parametrizations and Pólya relaxations are compared by means of controlling a nonlinear system. The second example focusses on the influence of different input parametrizations on the optimal cost-to-go functions, while the third example examines different options for the final DP step. Finally, the fourth example shows the benefits in terms of online computational times compared to Quasi-Min-Max MPC.

Example 11.1 (Comparison of Different Relaxations)

In the first example the application of the explicit LPV MPC is demonstrated. It shall be investigated, how the proposed method compares to explicit robust MPC and nonlinear MPC in terms of control performance and controller complexity. We also investigate different input parametrizations and Pólya relaxations of different degrees. We consider the following nonlinear system:

$$x_{k+1,1} = 0.85 x_{k,1} + u_k\,, \tag{11.28a}$$
$$x_{k+1,2} = (0.25 - 0.55(0.1 x_{k,2})^2) x_{k,1} + 0.65 x_{k,2} + (-1 + 2(0.1 x_{k,2})^2) u_k\,, \tag{11.28b}$$

under the constraints

$$-0.5 \leq u_k \leq 1, \qquad \begin{pmatrix} -10 \\ -10 \end{pmatrix} \leq x_k \leq \begin{pmatrix} 8 \\ 8 \end{pmatrix} \tag{11.29}$$

This system can be modelled as an quasi-LPV system of the form (11.1) by defining the following scheduling parameter

$$\theta_k = \left[1 - (0.1x_{k,2})^2 \quad (0.1x_{k,2})^2\right]^T , \tag{11.30}$$

resulting in the parameter-varying system matrices (11.1b) with the vertices

$$A_1 = \begin{bmatrix} 0.85 & 0 \\ 0.25 & 0.65 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 0.85 & 0 \\ -0.3 & 0.65 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The Multi-Parametric Toolbox (MPT) and Yalmip were used to compute the control laws, [KGBM04, Löf08]. The weight matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 0.01 , \quad P = Q ,$$

and a prediction horizon of $N = 3$ were chosen. The ∞-norm was used in the cost function.

Five different controllers for this system were compared in terms of complexity and control performance. By considering the scheduling parameter (11.30) as an unknown, bounded parametric uncertainty, an explicit robust controller was computed, in the following indicated by `rob`. This robust MPC scheme was presented in detail in [BBM03], where it is derived as the solution to the closed-loop constrained robust optimal (CL-CROC) problem (see also Section 4.2).

Following the proposed procedure in this chapter, three explicit LPV-MPC controllers were computed, with

(*i*) `aff2` – an affine input parametrization and Pólya degree $N_p = 2$,

(*ii*) `aff10` – an affine parametrization and Pólya degree $N_p = 10$, and

(*iii*) `quad2` – a quadratic parametrization and Pólya degree $N_p = 2$.

In the final DP step the optimization problem (11.17) was solved using the uncontrolled successor state z_k and minimizing the worst-case cost. Finally, the truly optimal solution, based on solving the optimal control problem for the nonlinear model online with the global branch-and-bound based solver

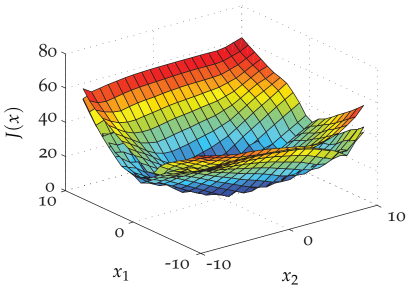


Figure 11.1: Actual costs in Example 11.1 simulated over 40 steps.

Controller:	`rob`	`aff2`	`aff10`	`quad2`
No. of regions:	31	53	55	117
Avg. cost increase:	23.3 %	0.4 %	0.4 %	0.2 %

Table 11.1: Complexity of the explicit control laws and average cost increase compared to nonlinear MPC, Example 11.1.

in Yalmip, was also used (`nl`). Stability of the closed-loop systems under explicit LPV control was verified for the whole feasible space (11.29) following the reachability analysis presented in Section 11.3.

All control laws were tested in simulations by controlling the system from 400 initial points, uniformly distributed over the feasible space. Figure 11.1 shows the actual simulated costs accumulated over 40 steps. It can be observed that the robust control yields a higher cost, while the other control laws are in about the same range.

This observation is quantified in Table 11.1, which reveals that there is virtually no difference in performance between the LPV controllers `aff2` and `aff10`. The quadratic controller `quad2` shows a slightly better behaviour. One has to mention here that we lack a guarantee of obtaining a better accumulated cost when using a less conservative approximation, since we optimize worst-case performance over a finite horizon.

Table 11.1 also shows the complexity of the explicit control laws. There is an increase in complexity from the robust to the affine parametrization and then to the quadratic parametrization. This is due to the number of piecewise affine control laws $\mu_{i,j}(z_k)$ needed to compose the explicit control laws. In this example the choice of the Pólya degree only had a small influence on the complexity of the resulting controllers.

Example 11.2 (Comparison of future input parametrization)

In the second example we want to investigate the influence of different input parametrizations on the normal dynamic programming iterations. In order to focus on the normal dynamic programming iterations, we do not perform the final DP step, but examine the cost-to-go $J_1^*(x_{k+1})$, which is the result of the normal DP steps.

The second example system is a marginally stable LPV system of the form (11.1), and was taken from [PRSDM05]. The system matrices of this LPV system are stated as

$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{11.32a}$$

$$A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}, \tag{11.32b}$$

and the system is subject to state and input constraints,

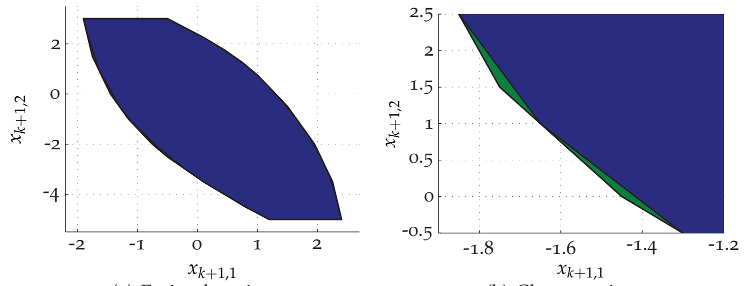
$$\begin{aligned} x_k \in \mathcal{X} &= \left\{ x_k \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} -10 \\ -10 \end{pmatrix} \leq x_k \leq \begin{pmatrix} 8 \\ 8 \end{pmatrix} \right\}, \\ u_k \in \mathcal{U} &= \{ u_k \in \mathbb{R} \mid -0.5 \leq u_k \leq 1 \} . \end{aligned}$$

Contrary to [PRSDM05], we assume that the LPV paradigm holds, i.e. the current scheduling parameter value is known to the controller. Moreover, a terminal state constraint was added, enforcing $\|x_{k+N}\|_\infty \leq 1$. We do not recommend the use of terminal state constraints for the controller design, but we want to add it in this example to demonstrate that the choice of the input parametrization can affect the size of the polytopic presets. The ∞-norm is minimized in the cost function (11.4). The weight matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 0.01, \qquad P = Q$$

and a prediction horizon of $N = 5$ were chosen, i.e. four subsequent normal DP iterations were computed.

Three different input parametrizations were tested. The first one is a parameter-independent input, denoted by `rob`. In this approach Pólya's relaxation is not required, but an epigraph formulation is sufficient to perform the DP steps. In the second approach an affine input parametrization with a Pólya degree of $N_p = 0$ is used, denoted by `aff0`, and the third input parametrization is quadratic with a Pólya degree of $N_p = 0$, denoted by `quad0`. All three approaches were compared against a lower bound on the cost-to-go, denoted by `grid5`. This lower bound is based on a uniform grid of the parameter simplex with 5 grid points. The optimization problem (11.10) at



(a) Entire domain. (b) Close-up view.

Figure 11.2: Domain of the cost-to-go functions $J^*_{1,\texttt{rob}}$ (blue), $J^*_{1,\texttt{aff2}}$ (green) in Example 11.2.

each DP step is solved separately for each grid point, and the maximum of the single cost-to-go functions was taken as cost-to-go in the succeeding DP step.

A general fact concerning the solution of parametric optimization problems is that the less restricted a parametric program is, the larger the feasible set and the lower the optimal cost function. In our case we are dealing with a series of parametric optimization problems, the difference being the flexibility of the control inputs. The robust parametrization is a special case of the affine parametrization and the affine parametrization is a special case of the quadratic parametrization. Hence we know already that the domains of the three cost-to-go functions J^*_1 are nested with

$$\operatorname{dom}(J^*_{1,\texttt{rob}}) \subseteq \operatorname{dom}(J^*_{1,\texttt{aff0}}) \subseteq \operatorname{dom}(J^*_{1,\texttt{quad0}}) \tag{11.33}$$

and

$$J^*_{1,\texttt{rob}}(x_{k+1}) \geq J^*_{1,\texttt{aff0}}(x_{k+1}) \geq J^*_{1,\texttt{quad0}}(x_{k+1}) \qquad \forall x_{k+1} \in \mathcal{X}\,. \tag{11.34}$$

Figure 11.2 depicts the domain of the cost-to-go $J^*_{1,\texttt{rob}}$ and $J^*_{1,\texttt{aff0}}$. With an affine input parametrization, the domain of the cost-to-go is slightly larger than with a robust approach. The domains of the cost-to-go of the quadratic input parametrization, $J^*_{1,\texttt{quad0}}$, and of the gridding approach, $J^*_{1,\texttt{grid5}}$, coincide with the one of the affine input parametrization.

More details on the cost-to-go functions are stated in Table 11.2. The number of regions of the cost-to-go functions is roughly doubled between the robust, the affine and the quadratic input parametrization. The average and the maximum cost increase relative to $J^*_{1,\texttt{grid5}}$, which is a lower bound on the optimal cost-to-go function J^*_1, is stated. The optimality gap is smaller than 4.68%

Approach:	rob	aff0	quad0
No. of regions:	304	544	1085
Avg. cost increase:	0.16 %	0.002 %	0.00003 %
Max. cost increase:	4.68 %	0.20 %	0.009 %

Table 11.2: Complexity of the cost-to-go functions $J^*_{1,\texttt{rob}}$, $J^*_{1,\texttt{aff2}}$ and $J^*_{1,\texttt{quad2}}$, and cost increase relative to the gridding approach grid5 in Example 11.2.

with the parameter-independent approach rob, smaller than 0.2% with the affine input parametrization aff0, and smaller than 0.009% with the quadratic input parametrization quad0. Again we are dealing with a tradeoff between control performance and complexity, which has to be decided based on the problem at hand.

Example 11.3 (Comparison of final DP approaches)

After examining in the previous example the choice of different input parametrizations in the normal DP steps, the third example is devoted to the final DP step. As will be seen, the success of the different approaches depends on the system at hand, hence we are considering not only a single example, but 20 random LPV systems. The example systems consist of two states and one input. The scheduling parameters live in standard 2-simplices, and the elements of the system matrices are uniformly distributed random numbers in the interval $[-1, 1]$, i.e. also unstable systems are considered. The state and input constraints were chosen to be

$$\mathcal{X} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 10\}\,, \quad \mathcal{U} = \{u \in \mathbb{R} \mid |u| \leq 1\}\ .$$

The setup of the four approaches is as follows. The cost function (11.4) using the maximum norm $p = \infty$ is minimized. The stage cost matrices are

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\,, \quad P = Q\,, \quad R = 1\,,$$

and the prediction horizon was set to $N = 3$. In all DP steps of all approaches an affine input parametrization and a Pólya degree of $N_p = 2$ was chosen. While the normal DP steps are exactly the same in all approaches, the final DP steps differ in the utilization of the uncontrolled successor state z_k and/or in the minimization of the average vertex costs instead of worst-case minimizations.

In particular, the first approach is the one considered in Section (11.2) with the uncontrolled successor state z_k, the input parametrization (11.16) and

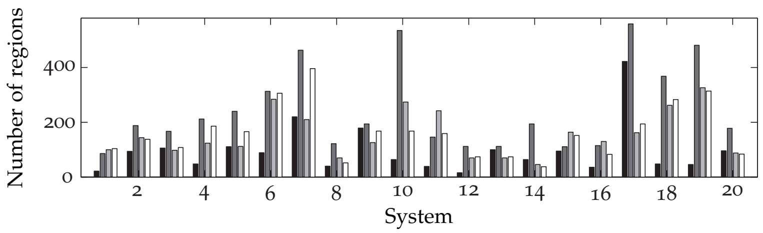


Figure 11.3: Complexity of the explicit control laws eMPCzWC (black), eMPCzAC (dark grey), eMPCxWC (light grey), and eMPCxAC (white) for 20 random example systems.

minimizing the worst-case cost as in (11.17). In the following, this control approach is denoted by eMPCzWC. The second approach uses also the input parametrization (11.16) with the uncontrolled successor state z_k, but instead of minimizing the worst-case cost, the average of the vertex costs is minimized, i.e. the optimization problem (11.15) is solved. The obtained controllers will be denoted by eMPCzAC. We were also interested in the role of the uncontrolled successor state z_k, which is why the last two approaches are not using the uncontrolled successor state, but employ an affine input parametrization as in (11.6). The third type of controllers, based on the minimization of the worst-case cost as in (11.17), will be denoted by eMPCxWC, while the last control approach, which is based on the minimization of the average vertex costs as in (11.15), will be denoted by eMPCxAC. For all computations we employed the NAG solver, [Num02], under MPT 2.6, [KGBM04], interfaced via Yalmip, [Löf08].

Figure 11.3 depicts the complexity of the four controller types for all 20 random systems. Unfortunately, there is no control approach which is superior for all tested systems, the results are different depending on the system at hand. However, there are some trends. Apart from Systems 9 and 13, the controllers eMPCzWC are of relatively low complexity, while the controllers eMPCzAC tend to be of higher complexity. The controller complexities of the remaining two approaches differ, ranging from the least complex to the most complex solution.

In order to compare the control performance of the four approaches, 900 grid points, uniformly distributed over the hyperbox $\mathcal{X}$, were selected, which served as initial states for the closed-loop systems. The closed-loop systems were simulated for 50 time steps, while at each time step the scheduling parameter values were selected randomly in the parameter simplex. These simulations were repeated ten times with different scheduling parameter values (all control methods were confronted with the same scheduling parameter values, of course). During the simulations the actually experienced

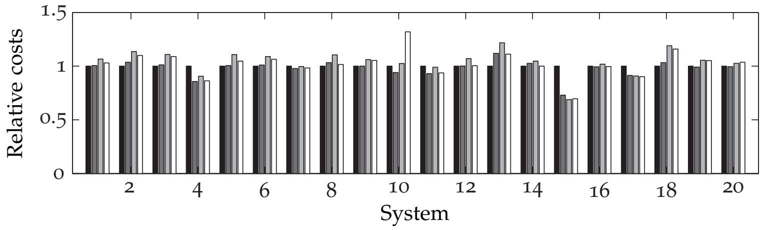


Figure 11.4: Average relative performance of the explicit control laws `eMPCzWC` (black), `eMPCzAC` (dark grey), `eMPCxWC` (light grey), and `eMPCxAC` (white) for 20 random example systems.

stage costs,

$$J_{\text{act}} = \sum_{k=1}^{50} \|Qx_k\|_\infty + \|Ru_k\|_\infty ,$$

were added up. After each run, the costs relative to the costs of `eMPCzWC` were determined. Finally the average of the relative cost values over all grid points and all simulations were taken. These average relative costs are displayed in Figure 11.4 for all 20 systems.

The results varied again. While in the majority of cases `eMPCzWC` outperforms the other approaches, the control performance is significantly worse than the remaining approaches for Systems 4, 15 and 17. The third approach `eMPCxWC` often yields the worst performance, with a few exceptions. The minimization of the average vertex costs in `eMPCzAC` and `eMPCxAC` often resulted in average relative costs in between the two worst-case approaches, with a small advantage for `eMPCzAC`.

What can be learnt from this series of examples? The approach `eMPCzWC` which was presented in Section (11.2) turns out to be the best option by means of complexity and control performance for many example systems. However, it is not always the best option, for some systems, the other considered methods are the better choice. `eMPCzAC` tends to be rather complex, while `eMPCxWC` tends to result in a worse control performance. The last approach `eMPCxAC` delivered a range of different results. We recommend to use `eMPCzWC` as a first choice, but to keep in mind that for some systems other approaches might be more successful.

Example 11.4 (Comparison of online computation times)

In the final example we want to demonstrate the actual motivation for using explicit LPV-MPC: The possible reduction of online computation times. As discussed in Section 11.2, the proposed explicit MPC scheme for LPV systems is not the optimal solution to the parametric optimization problem, but a suboptimal heuristic. Especially the last DP step has a large influence on complexity and performance of the resulting control law, and should be chosen with care. Hence we take the opportunity and use this example to test different approaches to explicit LPV-MPC.

The example system is again the LPV system from [PRSDM05], which was already described in Example 11.2. No terminal state constraints were enforced, and the weight matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 0.01\,, \qquad P = Q \tag{11.35}$$

and a prediction horizon of $N = 4$ of were chosen. The ∞-norm was minimized in the cost function (11.4).

Four methods are compared by means of control performance and online computational effort, when regulating the LPV system (11.32) to the origin.

1. The first method is the Quasi-Min-Max MPC method from Section 10.2. Since the system has a parameter-varying input matrix, the predicted future control laws were chosen to be independent of the scheduling parameter in order to keep the semi-definite program linear in the scheduling parameter. More information about Quasi-Min-Max MPC can be found in [LA00b].

2. For the second considered method, in order to avoid a parameter-varying input matrix, an input delay is introduced and the system is augmented to the LPV-A system

$$\begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} A(\theta) & B(\theta) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \tilde{u}_k\,. \tag{11.36}$$

   Afterwards the arising parametric optimization problem for the augmented system (11.36) is solved, resulting in a control law with 189 regions.

3. The computation of an explicit control law for the unaugmented LPV system (11.32) following the procedure in Section 11.2 led to an inappropriate high number of regions during the final DP step. Therefore the final DP step was adapted and in the third considered method the optimization problem (11.15) was solved instead of (11.17). In this alternative approach for the final DP step, instead of the worst-case cost, the average costs of the vertices of the parameter simplex are minimized. Pólya's relaxation is only needed for the constraint (11.15d). Including

the solution of (11.15), an explicit control law comprising 230 regions was computed.

4. As the last considered control method, the DP procedure is terminated before the last DP step, and the final DP step (11.14) is solved online instead.

All four control methods were tested in simulations by controlling the system from 400 initial points, uniformly distributed over the hyperbox $\|x\|_\infty \leq 10$, to the origin. To account for the varying scheduling parameter, these test were repeated ten times, each time with different random scheduling parameter values. During the simulations the actual quadratic costs,

$$J_{\text{act}} = \sum_{k=1}^{50} x_k^T Q x_k + u_k^T R u_k \,,$$

using the weight matrices (11.35), were collected to compute the actual closed-loop costs over a horizon of 50 time steps. Note that only the Quasi-Min-Max MPC approach minimizes a quadratic cost function, while all other tested approaches minimize the ∞-norm, i.e. the performance criterion (11.4) is in favour of the Quasi-Min-Max MPC approach. Additionally, the average computation times per step were taken.

Table 11.3 presents these data for initial points which were feasible under all control methods. The average actual costs during the simulations show the drawback of introducing an input delay to the LPV system: The controller cannot react as quickly to the variations in the scheduling parameter, leading to a performance degradation. This degradation will be even more severe in practice, since disturbances were not considered during the simulations, which would have to be compensated for with delay. Although the actual costs were measured in the quadratic form (11.4), the Quasi-Min-Max MPC has a worse performance than the Explicit LPV-MPC approaches.

This becomes more comprehensible, when one considers that the Quasi-Min-Max MPC scheme introduces some conservatism by considering a quadratic upper bound on the predicted cost, and moreover uses parameter-independent state feedback laws in the predictions. In this example this conservatism is on average more severe than the conservatism introduced by the explicit LPV-MPC schemes.

The computation times confirm the observations already made in Example 10.1 for the case of LPV-A systems: The evaluation of the control laws is orders of magnitude faster than the solution of the semi-definite programs in the Quasi-Min-Max MPC scheme. Computing the final step of the Explicit LPV-MPC scheme online reduces also the computation times compared to Quasi-Min-Max MPC, but with one order of magnitude not as significant as the completely explicit solution. The solution of the semi-definite programs

MPC controller	Quasi-Min-Max	Expl. (LPV-A)	Expl.	Semi-expl.
Avg. costs	376.14	445.03	366.23	363.58
t_{CPU} (mean)	148 ms	5.5 μs	4.8 μs	23.6 ms

Table 11.3: Average actual quadratic costs and online computation times for the four compared MPC approaches in Example 11.4.

within the Quasi-Min-Max MPC scheme was performed by SeDuMi, [Stu99], interfaced via Yalmip [Löf04].

Example 11.5 (A-priori stability guarantees)

In the second example we want to illustrate possible consequences of ensuring a-priori stability guarantees on the complexity of the explicit solution and on the size of the region of attraction. This example shall demonstrate that in some cases it is beneficial to verify stability a-posteriori instead. We consider an unstable LPV system with the system matrices

$$A_1 = \begin{bmatrix} 1.1 & 0 \\ 0.2 & 1.1 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} 1 \\ 0.8 \end{bmatrix}, \tag{11.37a}$$

$$A_2 = \begin{bmatrix} 1.1 & 0 \\ 0.4 & 1.1 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{11.37b}$$

under the constraints

$$x_k \in \mathcal{X} = \{x_k \in \mathbb{R}^2 \mid \|x_k\|_\infty \le 10\}, \quad u_k \in \mathcal{U} = \{u_k \in \mathbb{R} \mid |u_k| \le 1\}.$$

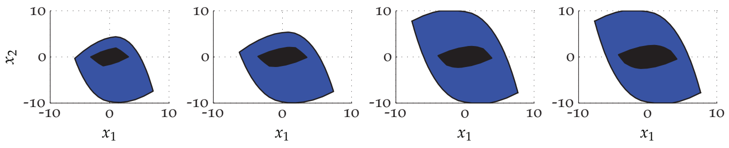
The weight matrices $Q = \operatorname{diag}([1 \;\; 1])$, $R = 0.01$, and the ∞-norm were employed in the cost function (10.5). The explicit control laws were computed for the prediction horizons $N = 2, \ldots, 5$, and the worst-case costs were minimized in the final DP step. In the first approach, denoted by `a-priori`, terminal cost and terminal constraints were determined by means of the procedure described in Section 11.3 in order to guarantee stability a-priori for all feasible states. In the second approach, denoted by `a-posteriori`, the terminal weight $P = Q$ and no terminal constraints were used, such that no a-priori stability guarantee can be given. Instead the region of attraction was determined afterwards by a reachability analysis.

The resulting complexities of the computed control laws are reported in Table 11.4. The number of regions for `a-priori` is higher than the number of regions for `a-posteriori`, which comes to no surprise, since terminal region constraints typically add to the solution complexity.

The size of the region of attraction for the different closed-loop systems is illustrated in Figure 11.5. In the approach with a-priori stability guarantees,

Prediction horizon N	2	3	4	5
No. of regions, `a-priori`	106	182	268	388
No. of regions, `a-posteriori`	69	115	149	204

Table 11.4: Number of controller regions with the approaches `a-priori` and `a-posteriori` for different prediction horizons in Example 11.5.



(a) $N = 2$. (b) $N = 3$. (c) $N = 4$. (d) $N = 5$.

Figure 11.5: Region of attraction of the closed-loop systems with the approaches `a-priori` (black) and `a-posteriori` (blue) for different prediction horizons in Example 11.5.

the region of attraction coincides with the set of feasible states. This set is indicated in black in Figure 11.5. The region of attraction increases with increasing prediction horizon, but only slowly. `A-posteriori` on the other hand results in larger regions of attraction. By omitting a-priori stability guarantees, the region of attraction is only a subset of the set of feasible states, but larger than the region of attraction of `a-priori`. Moreover one would require a much longer prediction horizon with `a-priori` to obtain similar regions of attraction, which typically results in even more controller regions.

11.5 Conclusions

In this chapter the computation of explicit MPC controllers for LPV systems was considered. A suboptimal approach incorporating an input parametrization and Pólya's relaxation was proposed to transform the closed-loop MPC problem to a series of parametric linear programs. This enables the advantages of explicit MPC – control under constraint satisfaction for systems with high sampling rate – for the class of LPV systems. In four examples different variations and parametrization possibilities were examined, and comparisons with other techniques such as Quasi-Min-Max MPC, robust MPC and nonlinear MPC were performed to demonstrate the application in different control situations and to illuminate the pros and cons of the different approaches.

12 (Sub-)Optimal Control of Constrained LPV Systems with Bounded Rate of Parameter Variation

'Do not anticipate trouble or worry about what may never happen. Keep in the sunlight.´

Benjamin Franklin

In the previous chapter it was shown how one can reformulate the MPC problem for LPV systems to a series of parametric linear programs by a closed-loop min-max MPC algorithm based on dynamic programming. A relaxation technique is employed to reformulate constraints which are polynomial in the scheduling parameters to parameter-independent constraints. The algorithm allows the computation of explicit control laws for LPV systems and enables the controller to exploit information about the scheduling parameter. This improves the control performance compared to a standard robust approach where no uncertainty knowledge is used, while keeping the benefits of fast online computations.

The presented algorithm is based on the assumption that the scheduling parameter - apart from being an element of the parameter simplex - can vary arbitrarily, and that the future scheduling parameter values can not be anticipated. In many applications however, the scheduling parameter varies only with a limited rate of variation, or even stays close to constant during the considered prediction horizon. Taking these limitations not into account can result in overly conservative control.

In this chapter we present an extension to the explicit LPV-MPC scheme in order to incorporate limits on the rate of parameter variation in the controller design. Taking these limits into account mitigates the mentioned conservatism since impossible trajectories of the scheduling parameter are not considered during control law computations.

12.1 Problem Statement

The class of systems we consider is the class of LPV systems, linear discrete-time systems with parameter-varying system matrices, which are defined by the state-update equation

$$x_{k+1} = A(\theta_k)x_k + B(\theta_k)u_k\,, \tag{12.1a}$$

with

$$A(\theta_k) = \sum_{j=1}^{n_\theta} A_j\theta_{k,j}\,, \qquad B(\theta_k) = \sum_{j=1}^{n_\theta} B_j\theta_{k,j}\,, \tag{12.1b}$$

and

$$\theta_k \in \Theta := \left\{ \theta_k \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_{k,j} = 1 \right\}. \tag{12.1c}$$

The discrete time is denoted by $k \in \mathbb{Z}$, whereas the variables $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $\theta_k \in \mathbb{R}^{n_\theta}$ denote the state, the control input, and the time-varying scheduling parameter, respectively. We assume that the state x_k is either measurable or observable. The system matrices $A(\theta_k)\colon \mathbb{R}^{n_\theta} \rightarrow \mathbb{R}^{n_x \times n_x}$ and $B(\theta_k)\colon \mathbb{R}^{n_\theta} \rightarrow \mathbb{R}^{n_x \times n_u}$ are known to lie in polytopes with the description (12.1b), where $A_j \in \mathbb{R}^{n_x \times n_x}, B_j \in \mathbb{R}^{n_x \times n_u}$ denote the jth vertices of the corresponding polytope. The scheduling parameter $\theta_k = [\theta_{k,1} \;\; \dots \;\; \theta_{k,n_\theta}]^T \in \mathbb{R}^{n_\theta}$ is constrained to the standard simplex (12.1c). This polytopic description is a common assumption in the LPV framework, see e.g. [AGB95].

Furthermore, the LPV system (12.1) is constrained, $u_k \in \mathcal{U}$ and $x_k \in \mathcal{X}$. The constraint sets $\mathcal{U}$ and $\mathcal{X}$ are assumed to be bounded polyhedra,

$$u_k \;\in\; \mathcal{U} = \{u_k \in \mathbb{R}^{n_u} \mid H_u u_k \leq \mathbf{1}\}\,, \tag{12.2a}$$
$$x_k \;\in\; \mathcal{X} = \{x_k \in \mathbb{R}^{n_x} \mid H_x x_k \leq \mathbf{1}\}\,, \tag{12.2b}$$

which contain the origin in their interiors, since we are interested in the regulator problem[1].

Remark 12.1 *For ease of notation, we restrict ourselves to separate constraints on the state and inputs in* (12.2). *It is straightforward to modify the presented algorithm in this chapter to the case of mixed polytopic constraints, i.e.* $E_x x + E_u u \leq f_{xu}$.

The essential assumption to LPV control is that the scheduling parameter is measured online and *known to the controller*. Future values are however only known to be constrained to the scheduling parameter and to the bound on the rate of variation.

[1]From a computational perspective the origin needs not to be included in these polytopic constraints.

Contrary to the previous chapters, we assume that the rate of variation $\Delta\theta_k \in \mathbb{R}^{n_\theta}$ is limited,

$$\Delta\theta_k = \theta_{k+1} - \theta_k \in \Gamma \subset \mathbb{R}^{n_\theta} \,. \tag{12.3}$$

The set $\Gamma \subset \mathbb{R}^{n_\theta}$ is assumed to be a polytope containing the origin in the interior, which denotes all possible variations of the scheduling parameter within one time step. The size of Γ is determined by the bound on the rate of parameter variation. In the limit the scheduling parameter stays constant over the prediction horizon (Scenario ($S3$)), such that Γ contains only the origin.

For the class of LPV systems with bounded rate of variation (BRV) we want to compute an explicit state feedback control law

$$u_k = \mu_\Gamma(x_k, \theta_k), \tag{12.4}$$

which makes use of the information of the current θ_k. The subscript Γ indicates that the control law takes the bounds on the rate of parameter variation into account. For the control problem to make sense[2], it is assumed that system (12.1) is controllable and observable for all admissible θ_k, [SM67, BBS03].

To compute this bounded rate of variation control law (12.4) within a model predictive control scheme, a finite-horizon cost function is to be minimized. We will optimize over finite-horizon *control policies*

$$\boldsymbol{\pi}_N = \{\mu_0, \mu_1, \dots, \mu_{N-1}\} \tag{12.5}$$

under the influence of the unknown *sequence of future scheduling parameters*

$$\boldsymbol{T}_k = \{\theta_{k+1}, \dots, \theta_{k+N-1}\} \,. \tag{12.6}$$

According to standard MPC, our cost function is defined as

$$J(\boldsymbol{\pi}_N; x_k, \theta_k, \boldsymbol{T}_k) = \|Px_{k+N}\|_p + \sum_{i=0}^{N-1} \|Qx_{k+i}\|_p + \|Ru_{k+i}\|_p \,, \tag{12.7}$$

where p denotes a polyhedral norm, e.g. the 1-norm or the ∞-norm. Polyhedral norms[3] enable a parametric solution to the stated problem using dynamic programming. For the minimization of the cost function (12.7) we have to consider the current as well as the unknown future parameter values, as the state trajectories are parameter-dependent.

[2]This assumption is not necessary for the actual computations of the proposed procedure.
[3]Quadratic cost functions are not possible since our procedure relies on epigraph reformulations, which would render the original problem a parametric quadratically constrained quadratic program, for which no efficient solution techniques are available.

In a closed-loop MPC approach (cf. Section 4.3), one would assume that the future control action u_{k+1} is calculated optimally over the horizon $N-1$ not until x_{k+1} and θ_{k+1} are available. But as the future values of the scheduling parameters are unknown, all possible cases must be regarded in order to accommodate for the worst-case scenario. This way it is assured that the actual cost function will be less or equal to the computed one, no matter how the scheduling parameters evolve. The optimization problem to solve in a closed-loop MPC approach is thus

$$\begin{aligned}
\mu_\Gamma^*(x_k,\theta_k) = \arg\min_{\mu_0}\max_{\theta_{k+1}}\min_{\mu_1}\cdots\max_{\theta_{k+N-1}}\min_{\mu_{N-1}} J(\boldsymbol{\pi}_N,\boldsymbol{T}_k;x_k,\theta_k) \qquad &(12.8a)\\
\text{s.t.}\quad \forall i\in\{0,\dots,N-1\}\qquad &\\
x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i},\theta_{k+i})\,, \qquad &(12.8b)\\
\mu_i(x_{k+i},\theta_{k+i})\in\mathcal{U}\,, \qquad &(12.8c)\\
x_{k+i+1}\in\mathcal{X}\,, \qquad &(12.8d)\\
\theta_{k+i}\in\Theta \qquad &(12.8e)\\
\Delta\theta_{k+i}\in\Gamma\,. \qquad &(12.8f)
\end{aligned}$$

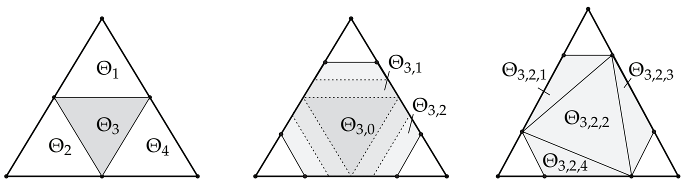
Remark 12.2 *It is straightforward to add terminal state constraints $x_{k+N}\in\mathcal{X}_T$ to the optimization problem* (12.8)*. However, for complexity reasons we recommend not to use terminal state constraints, but to verify stability a-posteriori (see the discussion in Section 11.3).*

12.2 Computation of Explicit MPC Controllers

In the following we will extend the results from the preceding chapters to incorporate limits on the rate of parameter variation in the controller computations. This extension is applicable to both the procedure in Chapter 10 for LPV-A systems and in Chapter 11 for LPV systems, respectively.

When the rate of parameter variation is limited, it follows that, starting from a certain initial scheduling parameter value, not all points in the parameter simplex can be attained in the succeeding time steps, but depending on time, some parts of the parameter simplex can be excluded. Which parts can be excluded depends on the value of the current scheduling parameter, such that the constraints of the future DP iterations depend on the value of the current scheduling parameter.

In order to solve the CPVOC problem (12.8) to optimality, one would have to respect the interconnection between the scheduling parameter values over time. The scheduling parameter values in earlier time instances would become parameters in the parametric optimization problems of later time



(a) Possible partition into four sub-simplices.

(b) Evolution of the simplex Θ_3 over two time-steps.

(c) Delaunay triangulation of polytope $\Theta_{3,2}$.

Figure 12.1: Determination of the sub-simplices $\Theta_{c,i,s}$ of the 2-dimensional parameter simplex Θ.

instances, when applying the dynamic programming procedures of Chapters 10 and 11. Unfortunately, the dependence on those scheduling parameters is nonlinear, preventing an efficient solution with a parametric linear programming solver. Moreover, the optimal solution, when computed, would not be defined over polytopes and thus is cumbersome to evaluate online. Hence we are proposing a *suboptimal* solution to the CPVOC problem (12.8) which is based on a relaxation of the constraint (12.8f).

One way to incorporate the change of the constraints in the DP algorithm is to consider different parts of the parameter simplex individually, which leads to the central point of this section: The adaptation of the dynamic programming procedures in Sections 10.2 and 11.2 for the computation of control laws $\mu_{\Theta_1}, \dots, \mu_{\Theta_{ns}}$ for separate subsets $\Theta_1, \dots, \Theta_{n_s}$ of the parameter simplex Θ. The resulting control law (12.4) is composed of these separate control laws,

$$u_k = \mu_\Gamma(x_k, \theta_k) = \begin{cases} \mu_{\Theta_1}(x_k, \theta_k)\,, & \theta_k \in \Theta_1\,, \\ \qquad\vdots & \\ \mu_{\Theta_{ns}}(x_k, \theta_k)\,, & \theta_k \in \Theta_{n_s}\,. \end{cases} \tag{12.9}$$

The adaptation of the presented procedures firstly involves the determination of appropriate subsets of the parameter simplex Θ for each time step of the DP procedure.

1. *Partition the parameter simplex Θ into n_s sub-simplices $\Theta_1, \dots \Theta_c, \dots, \Theta_{n_s}$.* A possible partition of a 2-dimensional parameter simplex into four simplices is depicted in Figure 12.1(a). Note that the partition is not required to be symmetric but can be adapted to the needs of the application at hand.

2. *Determine the evolution of each sub-simplex Θ_c over time, $\Theta_{c,0}, \dots, \Theta_{c,N-1}$.* Each simplex Θ_c represents one initial region $\Theta_{c,0}$ for the scheduling

parameter. Due to the limitation of the rate of parameter variation, the future scheduling parameters θ_{k+i}, starting from the simplex $\Theta_{c,0}$, have to lie within

$$\Theta_{c,i} = (\Theta_{c,i-1} \bigoplus \Gamma) \bigcap \Theta \,. \tag{12.10}$$

Figure 12.1(b) depicts the evolution of the simplex Θ_3 over two prediction steps.

3. *Triangulate each polytope $\Theta_{c,i}$ to obtain the simplices $\Theta_{c,i,1}, \dots, \Theta_{c,i,n_{s,c,i}}$.* The application of Pólya's relaxation as shown in Section 11.2 requires the scheduling parameter to live in the standard simplex. For this purpose the polytopes $\Theta_{c,i}$ at each prediction step are divided by a Delaunay triangulation, [Del34], resulting in a number $n_{c,i}$ of simplices. Afterwards each simplex $\Theta_{c,i,s}$ can be transformed to the standard simplex by a simple coordinate transformation. The Delaunay triangulation of polytope $\Theta_{3,2}$ is shown in Figure 12.1(c).

Remark 12.3 *Step 3 is not necessary when we are dealing with an LPV-A system with constant input matrix as in Section 10.2, since in that case the application of Pólya's relaxation simplifies to a vertex enumeration, which can be applied to arbitrary polytopes.*

With all subsets $\Theta_{c,i,s}$ of the parameter simplex Θ at hand – where the first subscript c denotes the considered control law, the second subscript i the prediction step and the third index s the simplex from the Delaunay triangulation – we can start the actual computation of the LPV-MPC control law. For every sub-simplex Θ_c a separate control law $\mu_{\Theta_c}(x_k, \theta_k)$ is computed using a modified version of the presented DP procedures. In the following we will focus on the required modifications for the normal DP steps using Pólya's relaxation starting from (11.11). These modifications are applied in an analogous fashion to the final DP step (11.17) and to the case of a vertex enumeration instead of Pólya's relaxation.

Instead of solving the semi-infinite optimization problem (11.11) at the $(N-i)$th step of the DP procedure, we propose to solve

$$J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \min_{\{\boldsymbol{U}_i, t_i\}} t_i(x_{k+i}) \tag{12.11a}$$

$$\text{s.t.} \quad x_{k+i+1} = A(\theta_{k+i})x_{k+i} + B(\theta_{k+i})\mu_i(x_{k+i}, \theta_{k+i})\,, \tag{12.11b}$$

$$\|R\mu_i(x_{k+i}, \theta_{k+i})\|_p + J_{i+1}^*(x_{k+i+1}) \le t_i(x_{k+i}) \quad \forall\, \theta_{k+i} \in \Theta_{c,i}\,, \tag{12.11c}$$

$$\mu_i(x_{k+i}, \theta_{k+i}) \in \mathcal{U} \quad \forall\, \theta_{k+i} \in \Theta_{c,i}\,, \tag{12.11d}$$

$$x_{k+i+1} \in \mathcal{X}_{i+1} \quad \forall\, \theta_{k+i} \in \Theta_{c,i}\,, \tag{12.11e}$$

$$x_{k+i} \in \mathcal{X}\,, \tag{12.11f}$$

$$\theta_{k+i} \in \Theta_{c,i}\,. \tag{12.11g}$$

for each sub-simplex Θ_c. Note that Θ is replaced by $\Theta_{c,i}$. In the case that $\Theta_{c,i}$ is not a simplex, each sub-simplex $\Theta_{c,i,s}$ from the Delaunay triangulation is treated separately,

$$J_i^*(x_{k+i}) = \|Qx_{k+i}\|_p + \min_{\{\mathcal{U}_i, t_i\}} t_i(x_{k+i}) \tag{12.12a}$$

$$\text{s.t.} \quad \forall s:$$

$$x_{k+i+1} = A(\theta_{k+i,s})x_{k+i} + B(\theta_{k+i,s})\mu_i(x_{k+i}, \theta_{k+i,s}), \tag{12.12b}$$

$$\|R\mu_i(x_{k+i}, \theta_{k+i,s})\|_p + J_{i+1}^*(x_{k+i+1}) \leq t_i(x_{k+i}) \quad \forall\, \theta_{k+i,s} \in \Theta_{c,i,s}, \tag{12.12c}$$

$$\mu_i(x_{k+i}, \theta_{k+i,s}) \in \mathcal{U} \quad \forall\, \theta_{k+i,s} \in \Theta_{c,i,s}, \tag{12.12d}$$

$$x_{k+i+1} \in \mathcal{X}_{i+1} \quad \forall\, \theta_{k+i,s} \in \Theta_{c,i,s}, \tag{12.12e}$$

$$x_{k+i} \in \mathcal{X}, \tag{12.12f}$$

$$\theta_{k+i,s} \in \Theta_{c,i,s}. \tag{12.12g}$$

This optimization problem is solved by transforming each simplex $\Theta_{c,i,s}$ to the standard simplex and applying Pólya's theorem.

Remark 12.4 *Note that the proposed procedure is conservative, it does not exclude all scheduling parameter trajectories which violate the constraint (12.8f). Nevertheless, the achieved exclusion of scheduling parameter trajectories can already lead to significant performance improvements as will be shown in Example 12.1.*

The online application of the computed control law now consists of the following steps:

(*i*) Measure or estimate the state x_k and the scheduling parameter θ_k.

(*ii*) Identify the sub-simplex Θ_c to which θ_k belongs.

(*iii*) Compute the uncontrolled successor state z_k.

(*iv*) Determine the control action u_k by evaluating the parametric solution μ_{Θ_c}.

Taking the rate of parameter variation into account results in an increase in complexity (approximately by a factor of the number of sub-simplices n_s). Hence the determination of an appropriate number n_s remains to be a tradeoff between control performance and complexity. Note however that the additional complexity is affecting the online computation time far less than the storage space. Step (*iv*) takes approximately the same amount of time as before, and only step (*ii*) is added to the case with unlimited rate of variation.

Remark 12.5 *The presented modification of the explicit LPV-MPC scheme can also be used to generalize the originally investigated case of unlimited rate of variation. It allows for applications where the scheduling parameter is not living in a standard* *simplex, but in a general polytope. Instead of increasing the dimension of the parameter simplex for each additional vertex of the polytope by one, one can partition the polytope into sub-simplices and apply the procedure above.*

Remark 12.6 *Performance benefits are not only due to considering the limitations on the rate of variation, but also to the richer structure of the resulting control law. Instead of being a continuous function of the scheduling parameter, the dependence of the control law is piecewise continuous, with discontinuities possible at the boundaries of the sub-simplices* Θ_c.

Remark 12.7 *The larger the number* n_s *of separately considered sub-simplices of the parameter simplex, the tighter the approximation of the actual possible future scheduling parameter values by the polytopes* $\Theta_{c,i}$, $i = 0, \dots, N-1$. *Moreover, the choice of the input parametrizations can also be adapted to the size of the polytopes* $\Theta_{c,i}$, *since the input parametrizations become more and more accurate the smaller the polytope.*

12.3 Numerical Example

Knowledge of the future scheduling parameter values can be of crucial importance for control performance. In order to demonstrate this statement, the following example was constructed.

Example 12.1 (Bounded rate of parameter variation)

Consider the LPV system

$$x_{k+1} = \begin{bmatrix} 1 & 0 \\ -1+2\theta_k & 0.99 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ -0.2+0.4\theta_k \end{bmatrix} u_k\,, \tag{12.13}$$

under the constraints

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq x_k \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad -1 \leq u_k \leq 1. \tag{12.14}$$

The parameter θ_k is assumed to vary only slowly within [0,1], with the rate of variation

$$\Gamma = \{\Delta\theta \ : \ -0.1 \leq \Delta\theta \leq 0.1\}\,. \tag{12.15}$$

This system can be seen as a disturbed double integrator, where the influence of the first integrator on the second depends on the scheduling parameter. Note that the scheduling parameter has a severe influence on the dynamics of the system.

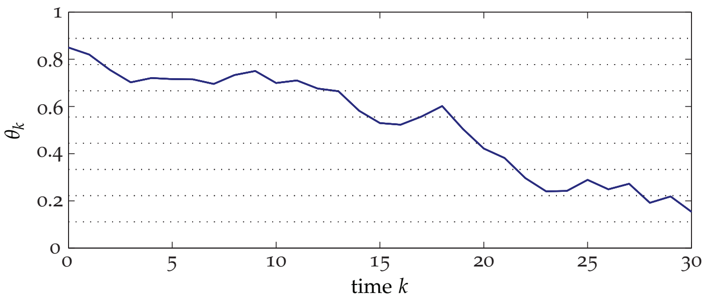


Figure 12.2: Scheduling parameter trajectory during the simulations. The dotted horizontal lines indicate the boundaries of the parameter simplex regions of `brv9`.

Four different controllers for this system were compared in terms of complexity and control performance. By considering the scheduling parameter as an unknown, bounded parametric uncertainty, an explicit robust controller was computed, in the following indicated by `rob`. This robust MPC scheme is presented in detail in [BBM03], where it is derived as the solution to the closed-loop constrained robust optimal control (CL-CROC) problem. The resulting control law is parameter-independent, $u_k = \mu(x_k)$.

Following Section 11.2, an explicit LPV-MPC controller with an affine input parametrization (11.6) in the scheduling parameter, making use of the uncontrolled successor state z_k and a Pólya degree of $N_p = 2$ was computed, in the following denoted by `eMPCzWC`. The obtained control law is parameter-dependent, $u_k = \mu(z_k, \theta_k)$. Finally two LPV-MPC controllers, which take the bounded rate of variation (12.15) into account, were computed as described in Section 12.2. The parameter interval $[0, 1]$ was uniformly divided in three subregions or in nine subregions, respectively; for each region, a parameter-affine control law was computed using a Pólya degree of $N_p = 2$ and the uncontrolled successor state, resulting in parameter-dependent control laws, $u_k = \mu_\Gamma(z_k, \theta_k)$. These controller are denoted by `brv3` and `brv9`.

The Multi-Parametric Toolbox (MPT) and Yalmip were used to compute the control laws, [KGBM04, Löf08]. The weight matrices

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 0.01, \quad P = Q \tag{12.16}$$

and a prediction horizon of $N = 4$ were chosen. The ∞-norm was used in the cost function (12.7). All control laws were tested in the task to regulate the system from the initial point $x_0 = [0.4, 0.5]$ to the origin, while the scheduling parameter evolves as shown in Figure 12.2.

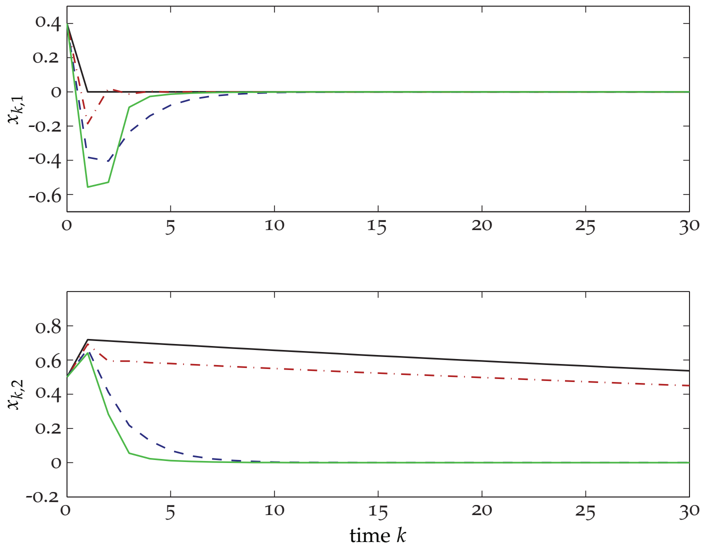


Figure 12.3: State trajectories of the LPV system (12.13) under the control of `rob` (—), `eMPCzWC` (– · –), `brv3` (– –) and `brv9`(—).

In Figure 12.3 the state trajectories of the system controlled by each controller is depicted. The strategy of the robust controller `rob`, which comprises 40 regions only, is simple. Since even the sign of the influence of the first integrator on the second is uncertain, it discharges the first integrator immediately to prevent a possibly counter-productive influence. Instead `rob` relies on the stability of the second 'integrator' to eventually converge to the origin. A similar strategy can be observed from `eMPCzWC`, which comprises 176 regions. In order to charge the first integrator appropriately, knowledge of the current scheduling parameter is not sufficient. The crucial point in this example is the anticipation of the future scheduling parameter values. When the effect of the first integrator on the second in the succeeding time step is conceivable, the first integrator can be charged accordingly. This allows `brv3` (which comprises 192 regions) and `brv9` (which comprises 584 regions) to outperform the other two controllers significantly and to steer the system to the origin within a few time steps.

In Figure 12.4, the control laws of the four compared MPC schemes at the initial point $x_0 = [0.4, 0.5]$ are plotted vs. the scheduling parameter θ_k. While `rob` is constant in the scheduling parameter, the remaining control laws are piecewise quadratic in the scheduling parameter. Furthermore, the discontinuities of the bounded rate of variation controllers at the boundaries between the intervals in $[0, 1]$ are visible.

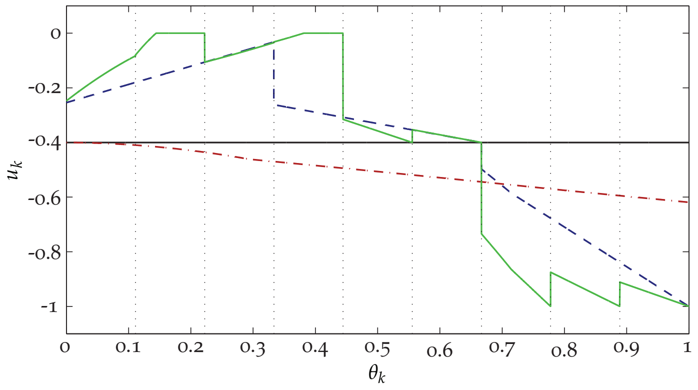


Figure 12.4: Control laws $\mu(x_0, \theta_k)$ at the initial point $x_0 = [0.4, 0.5]$ vs. scheduling parameter θ_k: `rob` (—), `eMPCzWC` (− · −), `brv3` (−−) and `brv9`(—). The dotted vertical lines indicate the boundaries of the parameter regions of `brv9`.

Controller:	`rob`	`eMPCzWC`	`brv3`	`brv9`	`brv27`	`brv81`
No. of regions:	40	176	192	584	1724	5093
Actual costs:	19.220	15.711	1.754	1.557	1.518	1.511

Table 12.1: Complexity and cumulated actual cost of the different MPC schemes.

Table 12.1 summarizes the complexity and the cumulated actual performance in terms of the cost function,

$$J_{\text{act}} = \sum_{k=0}^{30} \|Qx_k\|_\infty + \|Ru\|_\infty \,,$$

of the computed control laws. In order to point out the influence of the number of parameter regions on complexity and control performance, two more bounded rate of variation schemes were added, with 27 and with 81 subregions of the interval $[0, 1]$. Note that after 30 time steps the costs of the first two approaches are still growing, while the costs of the remaining approaches have converged.

12.4 Conclusions

In this chapter an extension to explicit LPV-MPC was proposed, which takes limitations on the rate of parameter variations into account. This enables the computation of less conservative controllers in the control scenarios ($S2$)

and (*S*3), since impossible scheduling parameter trajectories are excluded. Possible benefits of this extension were demonstrated in an example.

The current and preceding chapters provide the control engineer with an attractive novel approach to tackle the control of constrained nonlinear systems, involving the following steps: (*i*) embedding of the nonlinear system in an LPV model, (*ii*) computation of an explicit LPV-MPC controller, and (*iii*) the implementation as a look-up table in a microprocessor. If the entire behaviour of the nonlinear system is contained in the LPV model, guarantees on stability and constraint satisfaction of the LPV closed-loop system pass on to the nonlinear closed-loop system, and the cost function is an upper bound on the actual cost of the nonlinear closed-loop system.

13 Time-Optimal Control of Constrained LPV Systems

'Quaedam tempora eripiuntur nobis, quaedam subducuntur, quaedam effluunt. Turpissima tamen est iactura quae per neglegentiam fit.'

Lucius A. Seneca

Minimum-time MPC, involving the solution of the CTOC problem instead of the CFTOC problem, was shown previously to be a low-complexity alternative to explicit MPC for linear and hybrid systems. The complexity of the MPC control laws, which is the main obstacle to the application to systems of larger size, could be reduced significantly by a minimum-time approach. In this chapter we show how Pólya's relaxation can be employed to compute minimum-time MPC controllers for discrete-time LPV systems. Contrary to previous publications, our approach allows the use of parameter-varying input matrices. In a comparison over 20 random systems, explicit minimum-time MPC is compared to explicit LPV-MPC in terms of complexity and control performance.

13.1 Introduction

The model predictive control community has considered LPV systems in the past, and proposed methods which enable optimal control of LPV systems under state and input constraints, [LA00b]. Since MPC requires the solution of an optimization problem in each sampling interval (a semidefinite program in the case of LPV systems, [LA00b]), its application is limited to systems with sufficiently slow dynamics. A remedy for small systems is explicit LPV-MPC, the a-priori computation of the parameter-varying state-feedback control law.

In the previous chapters explicit MPC schemes for LPV systems were discussed, which allows for high-speed optimal control of constrained LPV systems. These scheme are based on parametric programming and thus participate in its shortcomings. One drawback of parametric programming is the inherent complexity of the resulting control laws, which grows starkly with the dimension of the parameter and the number of constraints. The application of Pólya's relaxation to a single polynomial constraint results in a number of parameter-independent constraints, potentially amplifying the complexity of the parametric program.

In order to mitigate these complexity issues, a common strategy is to formulate a simpler, yet similar parametric optimization problem, resulting in a less complex control law. Less complex control laws require less storage space and can be evaluated more quickly online. Therefore these approximate control schemes can be applied to systems of higher dimensions and/or with even faster dynamics. In this chapter we discuss such an approach which is based on *minimum-time MPC*. In minimum-time MPC, instead of minimizing an objective function, the number of time steps to converge to a terminal region is minimized. In [GM03] was shown for linear systems that explicit minimum-time MPC can be a low-complexity alternative to explicit MPC. The complexity of the control laws could be reduced by an order of magnitude, while yielding close-to-optimal control performance. Moreover, if the control applied inside the terminal region is stabilizing and renders the terminal region invariant, the controller obtained using minimum-time control is recursively feasible and stabilizing.

We consider the general class of discrete-time LPV systems, i.e. we do not restrict our attention to constant input matrices. The consequences of a varying input matrix are more extensive than one might expect, because it results in the closed-loop system to leave the class of polytopic systems. When the system at hand is an LPV-A system with constant input matrix, the involved computations simplify, and using an epigraph formulation instead of Pólya's relaxation is sufficient (compare Chapter 10).

13.2 Problem Statement

We consider again the class of LPV systems, linear discrete-time systems with parameter-varying system matrices, which are defined by the state-update equation

$$x_{k+1} = A(\theta_k)x_k + B(\theta_k)u_k\,, \tag{13.1a}$$

with

$$A(\theta_k) = \sum_{j=1}^{n_\theta} A_j\theta_{k,j}\,, \qquad B(\theta_k) = \sum_{j=1}^{n_\theta} B_j\theta_{k,j}\,, \tag{13.1b}$$

and

$$\theta_k \in \Theta := \left\{ \theta_k \in \mathbb{R}_+^{n_\theta} \;\middle|\; \sum_{j=1}^{n_\theta} \theta_{k,j} = 1 \right\}. \tag{13.1c}$$

The discrete time is denoted by $k \in \mathbb{Z}$, whereas the variables $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $\theta_k \in \mathbb{R}^{n_\theta}$ denote the state, the control input, and the time-varying scheduling parameter, respectively. We assume that the state x_k is either measurable or observable. The system matrices $A(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_x}$ and $B(\theta_k)\colon \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x \times n_u}$ are known to lie in polytopes with the description (13.1b), where $A_j \in \mathbb{R}^{n_x \times n_x}, B_j \in \mathbb{R}^{n_x \times n_u}$ denote the jth vertices of the corresponding polytope. The scheduling parameter $\theta_k = [\theta_{k,1} \quad \dots \quad \theta_{k,n_\theta}]^T \in \mathbb{R}^{n_\theta}$ is constrained to the standard simplex (13.1c). This polytopic description is a common assumption in the LPV framework, see e.g. [AGB95].

The considered LPV system (13.1) is subject to polytopic state and input constraints

$$x_k \in \mathcal{X} = \{x \in \mathbb{R}^{n_x} \mid H_x x \leq \mathbf{1}\}, \quad u_k \in \mathcal{U} = \{u \in \mathbb{R}^{n_u} \mid H_u u \leq \mathbf{1}\}\,, \tag{13.2}$$

containing the origin in their interiors.

Remark 13.1 *For ease of notation, we restrict ourselves to separate constraints on the state and inputs in* (13.2). *It is straightforward to modify the presented algorithm in this chapter to the case of mixed polytopic constraints, i.e.* $E_x x + E_u u \leq f_{xu}$.

The computation of minimum-time controllers for uncertain discrete-time systems, $u_k = \mu(x_k)$, was shown in [Bla92]. The objective of this chapter is to compute a stabilizing minimum-time MPC controller for the LPV system (13.1) which takes knowledge of the current scheduling parameter values into account,

$$u_k = \mu(x_k, \theta_k)\,. \tag{13.3}$$

For the control problem to make sense, it is assumed that the system (13.1) is controllable (and observable) for all $\theta_k \in \Theta$, see [SM67, BBS03].

13.3 Computation of Minimum-Time MPC Controllers

In the following section we provide a procedure for the computation of explicit minimum-time MPC controllers for LPV systems. The offline computations consist of three steps, in order to determine

(*i*) a terminal controller for the unconstrained system to be applied in a terminal region $\mathcal{X}_T$,

(*ii*) the shape and size of the terminal region $\mathcal{X}_T$ itself, which should be parameter invariant under the terminal control, and

(*iii*) the solution of the constrained time-optimal control (CTOC) problem as discussed in Section 4.4.

Unfortunately, taking the current scheduling parameter into account within the control law (13.3) renders the closed-loop system polynomially[1] dependent on the scheduling parameter, such that the set of states which can be mapped into a polytope under (13.3) is an intersection of infinitely many polytopes, thus convex, but not necessarily a polytope itself. In the following we propose to employ Pólya's relaxation (which is presented in Section 2.2) to approximate the maximum invariant and the maximum control invariant set of the LPV system (13.1) by polytopic subsets, and to use these polytopic sets within the minimum-time framework.

Terminal Region Control

The first step of the controller computation is concerned with the terminal region control, $u_k = \mu_T(x_k, \theta_k)$, of the unconstrained system, which is applied in a neighbourhood around the origin where no constraints are active. In principle, *any* control technique for unconstrained LPV systems can be employed here, assumed it is asymptotically stable and renders a set of states parameter invariant w.r.t. to the closed-loop system. Indeed, this flexibility allows one to employ minimum-time MPC as an add-on to an already existing control scheme, only acting when the system is affected by constraints.

A straightforward choice for the control of the unconstrained LPV system comprises the computation of LQR feedback matrices $K_j \in \mathbb{R}^{n_u \times n_x}$ for each vertex $j = 1, \ldots, n_\theta$ of the parameter simplex (13.1c), and the interpolation of the obtained feedback matrices,

$$u_k = \mu_T(x_k, \theta_k) = K(\theta_k)x_k = \sum_{j=1}^{n_\theta} K_j \theta_{k,j} x_k \,. \tag{13.4}$$

[1] Assumed (13.3) is an affine or polynomial function of the scheduling parameter.

In many cases this will result in a controller with satisfying stability and performance properties. On the other hand, a-priori guarantees can only be given for the vertices of the parameter simplex, while performance might decrease or even stability might be lost for scheduling parameter values in the relative interior of the simplex. Therefore we recommend to compute the state feedback matrices K_j in (13.4) by a more holistic approach similar to the computation of discrete-time LQR controllers . This approach is facilitated by a matrix-valued version of Pólya's relaxation, [SH04]. We consider the quadratic Lyapunov function

$$J(x_k) = x_k^T P x_k$$

with $P \in \mathbb{S}_+^{n_x}$ symmetric and positive semidefinite. We require that the costs decrease in each step at least by the quadratic stage costs,

$$x_{k+1}^T P x_{k+1} \leq x_k^T P x_k - x_k^T Q x_k - u_k^T R u_k \,, \tag{13.5}$$

with the weight matrices $Q \succ 0, R \succ 0$. Inserting the state-update equation (13.1) and the state-feedback equation (13.4) yields

$$\begin{aligned} x_k^T \Big[&(A(\theta_k) + B(\theta_k)K(\theta_k))^T P (A(\theta_k) + B(\theta_k)K(\theta_k)) \\ &-P + Q + K(\theta_k)^T R K(\theta_k) \Big] x_k \leq 0 \quad \forall \theta_k \in \Theta \,. \end{aligned} \tag{13.6}$$

For Equation (13.6) to be fulfilled, the matrix

$$\Big[(A(\theta_k) + B(\theta_k)K(\theta_k))^T P (A(\theta_k) + B(\theta_k)K(\theta_k)) - P + Q + K(\theta_k)^T R K(\theta_k) \Big] \tag{13.7}$$

must be $\preceq 0 \ \forall \theta_k \in \Theta$. This matrix inequality is not linear in the matrix variables $K(\theta_k)$ and P. Substituting $Y := P^{-1}$ and $F(\theta_k) := K(\theta_k)Y$, and multiplying from left and right with Y, [BEGFB94], yields

$$\begin{aligned} \Big[&(A(\theta_k)Y + B(\theta_k)F(\theta_k))^T Y^{-1} (A(\theta_k)Y + B(\theta_k)F(\theta_k)) \\ &-YPY + YQY + F(\theta_k)^T R F(\theta_k) \Big] \preceq 0 \quad \forall \theta_k \in \Theta \,. \end{aligned} \tag{13.8}$$

Applying Schur complement we obtain the linear matrix inequality (LMI)

$$\begin{bmatrix} -Y & A(\theta_k)Y + B(\theta_k)F(\theta_k) & O & O \\ (A(\theta_k)Y + B(\theta_k)F(\theta_k))^T & -Y & Y & F(\theta_k)^T \\ O & Y & -Q^{-1} & O \\ O & F(\theta_k) & O & -R^{-1} \end{bmatrix} \preceq 0 \quad \forall \theta_k \in \Theta \,, \tag{13.9}$$

which inherits the requirements $Y \succeq 0, Q \succ 0, R \succ 0$. In the semidefinite program incorporating the LMI we can minimize the trace or the largest eigenvalue of P, by solving

$$\min_{Y,F(\theta_k)} \quad \mathrm{trace}(X) \tag{13.10a}$$

$$\text{s.t.} \quad \begin{bmatrix} X & I \\ I & Y \end{bmatrix} \succeq 0 \quad \text{and} \quad (13.9)\,, \tag{13.10b}$$

with $X \in \mathbb{S}_+^{n_x}$, or

$$\min_{Y,F(\theta_k)} \quad -t \tag{13.11a}$$

$$\text{s.t.} \quad Y \succeq tI \quad \text{and} \quad (13.9)\,, \tag{13.11b}$$

with $t \in \mathbb{R}_+^{n_x}$, in order to obtain the cost function with the minimum expected or worst-case costs, respectively. Note the necessity of $t > 0$, to ensure that Y is invertible.

The LMI (13.9) is linear in the matrix variables Y and $F(\theta_k)$, but still depends polynomially on the scheduling parameter θ_k. Hence we apply Pólya's relaxation to obtain an LMI independent of the scheduling parameter θ_k. The Pólya degree N_p can be chosen high since there is a strong influence on the performance of $K(\theta)$, but only indirect influence on complexity through the size of the terminal region $\mathcal{X}_T$, which depends on the terminal region control (13.4). The solutions of both semidefinite programs (13.10) and (13.11) yield a stabilizing LPV gain-scheduling controller.

Proposition 13.1 *Due to the properties of Lyapunov functions, the existence of any $Y, F(\theta_k)$ which satisfy (13.9) for given $Q \succ 0, R \succ 0$ ensure asymptotic stability of the unconstrained closed-loop system*

$$x_{k+1} = A(\theta_k)x_k + B(\theta_k)F(\theta_k)Y^{-1}x_k$$

to the origin for all $\theta_k \in \Theta$.

Remark 13.2 *It is also possible not to consider a constant matrix Y, but a matrix $Y(\theta_k)$ polynomial in the scheduling parameter. This extension reduces the conservatism of the approach, but requires a modification of the equations above, in order to distinguish $Y(\theta_k)$ from $Y(\theta_{k+1})$. The modified version of LMI (13.9) would have to be satisfied for all possible θ_k and θ_{k+1}. Such an approach with a parameter-dependent Lyapunov function is usually applied to (but not restricted to) the case of a limitation of the rate of parameter variation (Scenarios (S2) and (S3)). The described modification leads to the controller $K(\theta_k) = F(\theta_k)Y(\theta_k)^{-1}$, being a rational function of the scheduling parameter.*

Terminal Region

In the second step the terminal region $\mathcal{X}_T$ is determined. The terminal region is required to be parameter invariant with regard to the LPV system (13.1) under the terminal region controller (13.4), and represents a set of states where neither state nor input constraints (13.2) are active. The terminal region $\mathcal{X}_T$ can be computed by the backpropagation Algorithm 2.1 on page 16 combined with Pólya's relaxation. Given the polytopic set $\mathcal{Z}_i = \{x \in \mathbb{R}^{n_x} \mid H_i x \leq \mathbf{1}\}$, its *preset* $\Omega(\mathcal{Z}_i)$ is defined as

$$\Omega(\mathcal{Z}_i) = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; \begin{array}{rcl} H_x x & \leq & \mathbf{1}, \\ H_u K(\theta) x & \leq & \mathbf{1}, \\ H_i(A(\theta) + B(\theta)K(\theta))x & \leq & \mathbf{1}, \\ \forall \theta & \in & \Theta \end{array} \right\}. \qquad (13.12)$$

The preset $\Omega(\mathcal{Z}_i)$ contains all states which are mapped by the LPV system (13.1) under the terminal region control (13.4) while satisfying the state and input constraints (13.2). Note that $\Omega(\mathcal{Z}_i)$ is a C-set, but in general not a polytope. Using Pólya's relaxation, the parameter-dependent constraints of (13.12) can be transformed into parameter-independent constraints, and we obtain a polytopic approximation of this preset, namely $\Omega_{N_p}(\mathcal{Z}_i) := \{x \in \mathbb{R}^{n_x} \mid H_{i+1} x \leq \mathbf{1}\}$.

Definition 13.1 (Polytopic Preset) *Let $\mathcal{Z}_i \subset \mathbb{R}^{n_x}$ be a polytope. A set $\Omega_{N_p}(\mathcal{Z}_i) \subset \mathbb{R}^{n_x}$ is called* polytopic preset *of $\mathcal{Z}_i$ for the LPV system* (13.1) *subject to the constraints* (13.2)*, if it contains all states which fulfil the polytopic constraints arising when applying Pólya's relaxation with degree N_p to the preset $\Omega(\mathcal{Z}_i)$.*

For $\Omega_{N_p}(\mathcal{Z}_i)$ again the polytopic preset $\Omega_{N_p}(\Omega_{N_p}(\mathcal{Z}_i))$ can be computed. Repeated computation of polytopic presets within Algorithm 2.1 eventually yields the *polytopic approximate maximum parameter invariant* set $\mathcal{Z}_\infty$. Note that the exact shape of $\mathcal{Z}_\infty$ depends on the Pólya degree N_p.

The polytopic approximate maximum parameter invariant sets $\mathcal{Z}_\infty$ can be taken as terminal region $\mathcal{X}_T$. The choice of the Pólya degree N_p reflects a tradeoff between the number of facets of $\mathcal{Z}_\infty$ and its size. It influences not only the shape of the terminal region $\mathcal{X}_T$, but also the complexity of the minimum-time control laws.

Note that the invariance property ensures that the state remains inside $\mathcal{X}_T$ once it has entered the terminal region, and that this property holds independent of the actual scheduling parameter values. By strengthening the invariance property of the terminal region with regard to the closed-loop system to λ-contractiveness with $\lambda \in [0, 1)$, asymptotic stability in the terminal region can be guaranteed (see Section 11.3). Hence the computation of the terminal region can also be used to verify stability of the terminal region controller, no matter how the state feedback matrices in Equation (13.4) were obtained.

Remark 13.3 *In the case of a rational terminal region controller,*

$$K(\theta) = F(\theta)Y(\theta)^{-1} = F(\theta)\frac{\operatorname{adj}(Y(\theta))}{\det(Y(\theta))},$$

where $\operatorname{adj}(Y(\theta))$ *and* $\det(Y(\theta))$ *denote the (parameter-dependent) adjugate and determinant of* $Y(\theta)$*, respectively, we simply multiply with* $\det(Y(\theta))$*, such that the preset* (13.12) *becomes*

$$\Omega(\mathcal{Z}_i) = \left\{ x \in \mathbb{R}^{n_x} \;\middle|\; \begin{array}{rcl} H_x x & \leq & \mathbf{1}, \\ H_u F(\theta) \operatorname{adj}(Y(\theta)) x & \leq & \mathbf{1} \det(Y(\theta)), \\ H_i(A(\theta) + B(\theta)F(\theta) \operatorname{adj}(Y(\theta)))x & \leq & \mathbf{1} \det(Y(\theta)), \\ \forall\theta & \in & \Theta \end{array} \right\}. \tag{13.13}$$

The resulting inequalities of (13.13) *are polynomials in the scheduling parameter, allowing the application of Pólya's relaxation for the determination of the terminal region. For procedures to compute the determinant or the inverse of polynomial matrices, see e.g. [GJV03], [JV05].*

Solution to the CTOC problem

After determining an invariant terminal region and terminal region control, the control laws $\mu_i(x, \theta)$, which steer the state in the minimal number of time steps to the terminal region $\mathcal{X}_T$, are computed. We solve the constrained time-optimal control (CTOC) problem, which was discussed in Section 4.4, for the LPV system (13.1). For this purpose the backpropagation Algorithm 2.1 on page 16 is applied, while Pólya's relaxation is employed for the computation of polytopic presets. In the iteration i we determine the set $\mathcal{Z}_i$ of all states which can be mapped to the terminal region $\mathcal{X}_T$ in i steps.

In the proposed procedure, the dependence of the input on the scheduling parameter has to be set beforehand. This dependence can be polynomial, but due to complexity reasons, we recommend to choose an affine dependence,

$$u_k = \mu_i(x_k, \theta_k) = \sum_{j=1}^{n_\theta} \theta_{k,j}\mu_{i,j}(x_k). \tag{13.14}$$

For notational simplicity we define $\boldsymbol{U}_i := [\mu_{i,1}, \ldots, \mu_{i,n_\theta}]$.

The control law $\mu_i(x, \theta)$, which steers the state from $\mathcal{Z}_{i+1}$ to $\mathcal{Z}_i$, is not unique, but can be selected from a set of control laws. For this selection the following parametric program can be employed,

$$\begin{aligned}
\min_{\boldsymbol{u}_i} \quad & J(u_k; x_k) && \text{(13.15a)}\\
\text{s.t.} \quad & H_x x_k \leq \mathbf{1}\,, && \text{(13.15b)}\\
& H_u u_k \leq \mathbf{1}\,, \quad \forall \theta_k \in \Theta\,, && \text{(13.15c)}\\
& H_i(A(\theta_k)x_k + B(\theta_k)u_k) \leq \mathbf{1}\,, \quad \forall \theta_k \in \Theta\,, && \text{(13.15d)}
\end{aligned}$$

where the objective function $J(x_k, u_k)$ determines the preferred selection. A possible objective function is

$$J(u_k; x_k) = \max_{\theta_k} \|Qx_{k+1}\|_p + \|Ru_k\|_p\,, \tag{13.16}$$

where p denotes a polyhedral norm. This objective function represents a minimization of the worst-case cost during a one-step-ahead prediction. Analogue to Chapter 11, an epigraph reformulation can be used to handle the maximization over the scheduling parameter.

It is possible, but not necessary, to minimize the worst case, since convergence to $\mathcal{X}_T$ is guaranteed, and the choice of the objective function $J(x_k, u_k)$ can focus entirely on control performance. When common operating points of the scheduling parameter $\{\bar{\theta}_1, \ldots, \bar{\theta}_{n_{\bar{\theta}}}\}$ (e.g. the vertices) are known, it is also possible to minimize the cost predictions for these operating points,

$$J(u_k; x_k) = \sum_{r=1}^{n_{\bar{\theta}}} (A(\bar{\theta}_r)x_k + B(\bar{\theta}_r)u_k)^T P (A(\bar{\theta}_r)x_k + B(\bar{\theta}_r)u_k) + u_k^T R u_k\,. \tag{13.17}$$

Since an epigraph formulation is not necessary, this objective function can be quadratic. The matrix P can be taken from the terminal region control, s.t. (13.17) reflects the infinite-horizon cost at the operating points under the assumption of unconstrained state evolution.

Pólya's relaxation can be employed to render the constraints of the parametric program (13.15) parameter-independent. The Pólya degree N_p is not only determining the size and number of constraints of the polytopic preset $\Omega_{N_p}(\mathcal{Z}_i)$, but also has a direct impact on the number of parameter-independent constraints, and thus on the complexity of the resulting control laws $\mu_i(x_k, \theta_k)$. Hence it should be selected rather low.

Remark 13.4 *It should be noted that the resulting control law is not a minimum-time control law in a strict sense, since there might be states in the preset, which do not belong to the polytopic preset, and thus are not mapped to the target region in the minimum number of steps.*

As described in Section 4.4, the collection of sets $\{\mathcal{Z}_i\}_{i=1}^{\infty}$ together with their corresponding control laws $\mu_i(x_k, \theta_k)$ are stored. Online a bisection procedure can be performed to solve the problem (4.19) on page 52 and thus to determine the set membership of the current state x_k. Afterwards the corresponding control law is evaluated.

Remark 13.5 *For higher-dimensional systems, the control laws $\mu_i(x_k, \theta_k)$ might become too complex for efficient storage and evaluation. In this situation it is still possible to compute the set collection $\{\mathcal{Z}_i\}_{i=1}^{\infty}$, by a set backpropagation (Algorithm 2.1). Online, we perform a set membership test and solve the optimization problem* (13.15) *with the cost function*

$$J(u_k; x_k, \theta_k) = (A(\theta_k)x_k + B(\theta_k)u_k)^T P(A(\theta_k)x_k + B(\theta_k)u_k) + u_k^T R u_k \quad (13.18)$$

for the current state x_k and scheduling parameter θ_k.

Comparison to Explicit LPV-MPC

Compared to the dynamic programming iterations in explicit LPV-MPC, which was presented in Chapter 11, the parametric program (13.15) comprises significantly fewer constraints, mainly because the objective function takes only the current and the succeeding step directly into account, and thus has a simpler structure. This is the main reason for the low complexity of minimum-time control compared to explicit LPV-MPC. Additionally, the future scheduling parameter values are unknown, what can lead to a broad variety of possible future trajectories, such that the exact future costs are simply not available. Hence it is questionable if much effort should be spend on a tradeoff between current and future costs, or if a simple approximation of the future costs as in (13.17) is not sufficient.

If the rate of parameter variations is limited (LPV scenario ($S2$) and ($S3$)), the procedure described in this chapter can be adapted. This adaptation can follow the lines of Chapter 12, i.e. partition the parameter simplex and compute a separate control law for each subset.

13.4 Numerical Examples

In order to justify claims about complexity and control performance, we compared the proposed minimum-time scheme to explicit LPV-MPC by means of 20 random examples. This comparison is by no means a proof, but can serve as an indicator of to be expected properties of the minimum-time scheme. For the computations we employed the NAG solver, [Num02], under MPT 2.6, [KGBM04], interfaced via Yalmip, [Löf08].

The example systems consist of two states and one input. The scheduling parameters live in standard 2-simplices, and the elements of the system matrices are uniformly distributed random numbers in the interval $[-1, 1]$, i.e. also unstable systems are considered. The state and input constraints were chosen to be

$$\mathcal{X} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 10\}, \quad \mathcal{U} = \{u \in \mathbb{R} \mid |u| \leq 1\}.$$

Two methods were compared, the first one being the minimum-time MPC scheme proposed in this chapter, denoted by `mt`. The terminal region control was computed by solving the LMI (13.9), while the cost function (13.17) penalizing the vertex predictions was minimized during the minimum-time iterations. The Pólya degree of the relaxations was $N_p = 10$ for the terminal region controller, and $N_p = 2$ for the terminal region computation and the solution of the CTOC problem.

The second method is the explicit LPV-MPC scheme from Chapter 11, denoted by `eMPCz`. The setup of the explicit LPV-MPC scheme was guided by the aim to achieve as much comparability as possible. In both methods the control inputs depend affinely on the scheduling parameter. The Pólya degree was set to $N_p = 2$, and the terminal cost matrix P from the minimum-time scheme was also used in the `eMPCz` scheme. In order to ensure a fair complexity comparison, the terminal region of the minimum-time controller was included as terminal region constraints in the `eMPCz` scheme. Thus both approaches face the same state and input constraints, and the set of feasible states is in principle the same, only limited by the methods itself. Unfortunately, it is not possible to employ quadratic cost functions in the explicit LPV-MPC approach, instead we are minimizing the maximum norm, $p = \infty$, of the finite-horizon predictions.

The MPC setup of both methods is as follows. The stage cost matrices are

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1. \tag{13.19}$$

The prediction horizon (and the number of minimum-time iterations, respectively) was set to $N = 3$. For all 20 systems, control laws were computed employing both mentioned control methods. The computation of `eMPCz` for Systems 13 and 19 was aborted after several hours[2]. The complexity of the remaining control laws is depicted in Figure 13.1. The complexity of `mt` varies from about half the complexity of `eMPCz`, up to an order of magnitude

[2]By changing the final DP approach to (11.15), it is possible to compute an explicit LPV-MPC controller. However, to keep the comparison consistent, we refrained from mixing different approaches, and sticked to the worst-case minimizations.

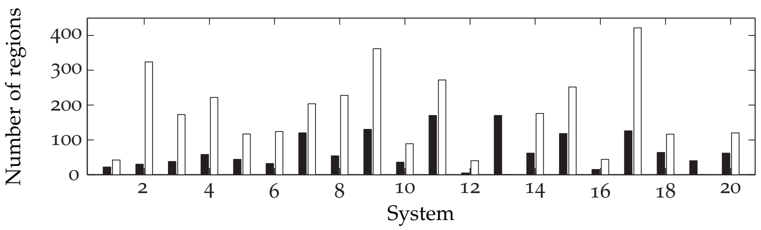


Figure 13.1: Number of regions of the control laws `mt` (black), `eMPCz` (white) for 20 random example systems.

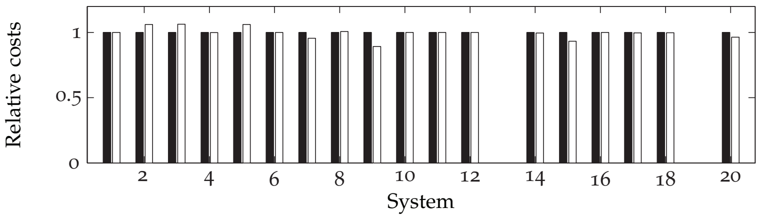


Figure 13.2: Relative average costs under the control laws `mt` (black), `eMPCz` (white) for 20 random example systems.

difference (System 2). For all systems, the offline computation time for `mt` was way shorter than for `eMPCz`.

In order to compare the control performance of the three methods, a grid of 900 uniformly distributed grid points was applied to the state space, and each grid point served as initial state. The closed-loop system was simulated for 40 time steps, and actually experienced stage costs were added up,

$$J_{\text{act}} = \sum_{k=0}^{40} \|Qx_k\|_\infty + \|Ru_k\|_\infty \, .$$

The scheduling parameter values at each time step were selected randomly in the parameter simplex. These simulations were repeated ten times with different scheduling parameter values (both control methods were confronted with the same scheduling parameter values, of course). Overall, each feedback system was tested on 9000 different trajectories, or 36000 sampling points, respectively.

Finally the average cost values over all grid points and all simulations were taken. Since the costs vary from system to system, a normalization with respect to the average costs of the minimum-time controller was performed. The resulting relative average costs are depicted in Figure 13.2. The control performance of `mt` is often similar to `eMPCz`, though there also exist systems, where one method outperforms the other.

13.5 Conclusions

What can be drawn as conclusions? Firstly that minimum-time LPV-MPC represents a low-complexity alternative to explicit LPV-MPC. Additionally it provides a-priori guarantees for stability. Minimum-time MPC controllers are not unique, and the cost functions of the optimization problems represent an additional degree of freedom for the selection of an appropriate minimum-time MPC controller. Contrary to explicit LPV-MPC, these "cost functions" can be quadratic. However, note that the true objective is to minimize the number of steps to reach a target set. Occasionally minimum-time LPV-MPC even outperforms explicit LPV-MPC, likely because the terminal cost matrices P were adapted to quadratic costs and because explicit LPV-MPC is optimizing worst-case costs instead of average costs. Consequently, the benefits of minimum-time MPC are even more compelling for LPV systems than for linear or hybrid systems. The application of minimum-time MPC for the high-speed close-to-optimal control of constrained LPV systems should also be considered when complexity is not the primary objective.

14 Autonomous Vehicle Steering (II)

'Knowing is not enough; we must apply. Willing is not enough; we must do.'

Johann Wolfgang von Goethe

Finally, we revisit the problem addressed in Chapter 8, the design of a highly performant controller for autonomous vehicle steering. In this chapter a novel control approach for autonomous steering of automobiles is presented. A nonlinear model of the lateral vehicle dynamics and the tire forces is approximated by an LPV model. Then explicit LPV-MPC is employed to derive a controller which requires only little online computations. The achieved control performance is compared to nonlinear MPC and explicit linear MPC in a double lane-change maneuver at different driving speeds. This project was supported by the European Commission research project FP6-IST-511368 *Hybrid Control* (HYCON).

14.1 Introduction

During the time of this doctorate, several extensions and improvements for the autonomous steering control were proposed. These include an improved vehicle model, which allows combined steering and braking, [FTB⁺07], and an experimentally validated stability analysis [FBT⁺08]. The issue of high computational complexity was also tackled by developing simplified prediction models that approximate the nonlinear model of the vehicle dynamics. In [FBA⁺07, FTB⁺07, FBT⁺08], a linear time-varying (LTV) model is employed, which is derived by linearizing the nonlinear model online. The LTV-MPC as well as the HPV-MPC procedure from Chapter 8 could reduce the computational burden and thus present schemes which are applicable on a test vehicle. Nevertheless, they still rely on solving the underlying optimization problem *online*, and require expensive computing equipment in the vehicle. The hybrid and the linear model on the other hand allow the computation of explicit controllers, and thus require significantly less online computations, but do not take the varying driving conditions into account. For a practical implementation however, both requirements are crucial. An autonomous steering system has to be able to cope with different driving situations, and should as well require only little online computations, since those can directly be translated into costs of the steering system.

The motivation for this work was twofold. On the one hand we wanted to propose a scheme for autonomous vehicle steering, which constitutes a reasonable compromise between both aforementioned requirements. A potential candidate for this task could be explicit MPC for linear parameter-varying (LPV) systems. On the other hand, it was interesting to verify explicit LPV-MPC by applying it to a realistic example, and to introduce a promising control method to the automotive community. As such, this chapter can be regarded as an real-life example for high-speed control of LPV systems under constraints.

The chapter is structured as follows. Section 14.2 shows how to derive a linear parameter-varying model and a linear model of the lateral vehicle dynamics, [JR03]. In Section 14.3, the model predictive control schemes are presented. Section 14.4 presents the application of the model predictive control schemes to the double lane-change maneuver.

14.2 Modelling

A nonlinear model of the lateral vehicle dynamics was presented in Section 8.2. This section describes the linearizations made to obtain a linear parameter-varying model and a purely linear model of the lateral dynamics.

In [BM08a], where Chapter 8 is based on, the derivation of a hybrid parameter-varying (HPV) model from (8.8) was presented. This HPV model accounts for the scheduling signals μ, s_f and the wind speed v_w. In the discussions following [BM08a], the difficulties to measure or estimate these scheduling signals during complicated driving maneuvers were emphasized. Hence, and contrary to [BM08a], we will not use the road friction μ as scheduling variable, but assume icy road conditions. We will also assume cruising conditions, i.e. no slip s_f. Instead we will investigate *gain-scheduling for different driving speeds* v_x. The hybridness of the HPV model was used to model saturation effects which appear at higher slip angles. By assuming the slip angles α_f, α_r small, it is possible to neglect the saturation and thus to simplify the HPV model to an LPV model. Note that from a theoretical point of view, it would be possible to consider more scheduling variables (e.g. the road friction, or to account for tire force saturation), though to the expense of quickly increasing complexity of the resulting control law. Since explicit LPV-MPC is based on dynamic programming, it also suffers from the curse of dimensionality, i.e. it requires the number of states n_x, the number of scheduling parameters n_θ and the number of prediction steps N to be relatively low.

In summary, the assumptions taken for the LPV model are:

(i) no slip $(s_f = s_r = 0)$,
(ii) icy road $(\mu = 0.1)$,
(iii) constant normal tire forces $F_{z,f}, F_{z,r}$,
(iv) small angles $(\psi \approx \delta_f \approx \alpha_f \approx \alpha_r \approx 0)$,
(v) front wheel steering $(\delta_r = 0)$.

With these assumptions, system (8.8) can be linearized to

$$\frac{d}{dt}\begin{bmatrix} Y \\ \dot{Y} \\ \psi \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{Y} \\ \frac{1}{m}(2F_{y,f} + 2F_{y,r}) \\ \dot{\psi} \\ \frac{1}{J}(2aF_{y,f} - 2bF_{y,r}) \end{bmatrix} . \tag{14.1}$$

where the lateral forces $F_{y,i}$ can be computed from the longitudinal and corner forces of the tires via the linearized force transformation (cf. (8.4))

$$F_{y,i} = F_{l,i}\delta_i + F_{c,i} , \quad i \in \{f, r\} . \tag{14.2}$$

These tire forces now need to be approximated. In the Pacejka tire force model, the longitudinal and corner forces are nonlinear functions of the slip

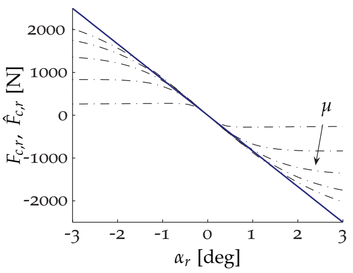


Figure 14.1: Cornering tire force of rear wheels $F_{c,r}$ for different road frictions μ ($- \cdot -$) and its approximation $\hat{F}_{c,r}$ (—) vs. slip angle α_r. The arrow indicates increasing road friction.

ratios s_i, the slip angles α_i and the friction coefficient μ. The slip angles α_i are depending in a nonlinear way on the state and input of the system, see (8.5) and (8.6). These relations can be linearized to

$$\alpha_f = -\,\delta_f - \psi + (\dot{Y} + a\dot{\psi})/v_x\,, \tag{14.3a}$$

$$\alpha_r = -\,\delta_r - \psi + (\dot{Y} - b\dot{\psi})/v_x\,, \tag{14.3b}$$

where the relation (8.7) between car coordinate $\dot{y}$ and global coordinate $\dot{Y}$ was linearized to

$$\dot{Y} = \dot{y} + v_x\psi\,. \tag{14.4}$$

From (14.1) to (14.3) follows that in order to obtain a model which depends on the state x and the control input δ_f in a linear manner, we have to approximate the longitudinal tire forces $F_{l,i}$ to be independent of the slip angles α_i. Therefore the longitudinal forces $F_{l,i}$ are approximated by evaluating the longitudinal Pacejka tire force for $\alpha_i = 0$,

$$\hat{F}_{l,i} = F_{l,i}(0, 0, 0.1) = 0, \quad i \in \{f, r\}\,. \tag{14.5}$$

The approximations of the corner forces $F_{c,f}$ and $F_{c,r}$ can depend on α_i in a linear way, since they are not multiplied by the control input δ_f in (14.2),

$$\hat{F}_{c,f}(\alpha_f) = k_f\alpha_f\,, \qquad \hat{F}_{c,r}(\alpha_r) = k_r\alpha_r\,, \tag{14.6}$$

where the coefficients k_f and k_r can be determined by a least squares approach for fixed values μ, s_i and small slip angles α_i. In Figure 14.1 the rear cornering force $F_{c,r}$ and its approximation vs. the slip angle α_r are depicted for different road frictions μ, while the slip s_r is zero. It can be seen that the linear approximation is reasonable for small slip angles, as long as the no-slip condition holds. The road friction μ determines the level at which the force saturates, but the slope of the force around zero is virtually unaffected.

The approximation of the front cornering force is carried out analogue to the rear force.

By replacing the tire forces in the linearized system (14.1) by its approximations (14.5) – (14.6), an LPV model with the following matrices is obtained:

$$A(\theta) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2(k_f+k_r)}{mv_x} & \frac{-2(k_f+k_r)}{m} & \frac{2(k_f a - k_r b)}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2(k_f a - k_r b)}{Jv_x} & \frac{-2(k_f a - k_r b)}{J} & \frac{2(k_f a^2 + k_r b^2)}{Jv_x} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ \frac{-2k_f}{m} \\ 0 \\ \frac{-2ak_f}{J} \end{bmatrix},$$
$$C = I_4, \quad D = O_4 . \tag{14.7}$$

The longitudinal velocity v_x of the vehicle is used as scheduling signal ρ. By defining the parameter $\theta = 1/v_x$, the dependence of the state transition matrix A on the parameter is affine. Note that B, C and D are constant matrices.

Finally, a linear model of the lateral vehicle dynamics is considered. This linear model will be used to point out the performance degradations caused by not considering varying vehicle speeds. A linear model can be obtained from the LPV model (14.7) by fixing the scheduling variable to a constant.

14.3 Controller Design

This section describes the setup of the four MPC schemes. MPC was introduced in Section 4.5. An Euler discretization with a sampling time of $T_s = 0.05\,\mathrm{s}$ is used to derive discrete-time models from the corresponding continuous-time models. In order to investigate the best performance achievable with MPC, the nonlinear model from Chapter 8 is employed for prediction in nonlinear model predictive control. In later stages the LPV model from Section 14.2 and finally the linear model is used for the prediction.

Nonlinear MPC

A nonlinear MPC (NMPC) controller for autonomous vehicle steering was presented in [KFB+06]. The standard MPC approach is applied with the cost function (8.27), i.e.

$$J(\boldsymbol{U}_k; x_k, \boldsymbol{X}_{\mathrm{ref},k}, p_k) = \sum_{i=1}^{N} \|x_{k+i} - x_{\mathrm{ref},k+i}\|_Q^2 + \sum_{i=1}^{N-1} \|s_{k+i}\|_{q_\alpha}^2 + \sum_{i=0}^{N_c-1} \|\Delta u_{k+i}\|_R^2 , \tag{14.8}$$

penalizing the difference between state vector and reference trajectory, and using a Δu*-formulation* with $\Delta u_{k+i} = u_{k+i} - u_{k+i-1}$. As in Chapter 8, we introduce the sequence of reference states $\boldsymbol{X}_{\text{ref},k} := \{x_{\text{ref},k+1}, \dots, x_{\text{ref},k+N}\}$ and the set of optimization variables $\boldsymbol{U}_k := \{\Delta u_k, \Delta u_{k+1}, \dots, \Delta u_{k+N_c}\}$ to simplify notation. The scheduling variables $\rho(k)$ are assumed to stay constant over the prediction horizon. A prediction horizon of $N = 4$ and a control horizon of $N_c = 2$ is chosen. The weights are chosen as

$$Q = \operatorname{diag}\left(\begin{bmatrix}1 & 0 & 40 & 0\end{bmatrix}\right), \quad q_\alpha = 1000 \quad \text{and} \quad R = 1\,.$$

As in [KFB+06], the control input u and the input rate Δu are constrained to

$$u_{\max/\min} = \pm 30\,\text{deg}\,, \quad \Delta u_{\max/\min} = \pm 20\,\text{deg/s}. \tag{14.9}$$

Additionally the front slip angle was constrained to $|\alpha_f| \leq 2\,\text{deg}$ by using *soft constraints*. For this chapter, the nonlinear optimizations were performed using MATLAB's internal solver `fmincon`.

Using the above stated cost function, we can formulate the constrained finite-time optimal control (CFTOC) problem, which is solved at each time step k, and employ a *receding horizon* control strategy, i.e. only the first input u_k is applied at time k, and another optimization is performed at the next time step.

Explicit LPV-MPC

Two explicit MPC controllers were computed for the LPV model presented in Section 14.2. The main issue in explicit MPC is the complexity of the resulting control laws. Driving forces for complexity are the number of scheduling parameters, the prediction horizon, and the dimension of the optimization problem. Hence several measures were taken to reduce the complexity of the problem.

The number of scheduling parameters is one, and in addition this parameter is only appearing in the state transition matrix A. This fact permits the usage of the LPV-A MPC scheme, which was described in Chapter 10. The dimension of the underlying parametric optimization problem is composed by the state, the number of reference signals and the number of inputs (to limit the input rate Δu). For the system at hand this would amount to a seven-dimensional parametric optimization problem. Fortunately the LPV model (14.7) can be decomposed into a three-dimensional subsystem containing the states $[\dot{Y},\, \psi,\, \dot{\psi}]$ and an integrator to derive Y. This allows us to use a cascade loop for control, containing the explicit LPV-MPC controller in the inner loop and a simple P-controller in the outer loop. The precise structure is depicted in Figure 14.2. Thus the controller computation simplifies to

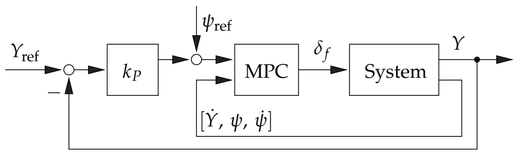


Figure 14.2: Cascade loop for Explicit LPV-MPC.

a five-dimensional parametric optimization problem with reduced complexity. While optimal performance is not guaranteed anymore, the satisfaction of constraints (14.9) is still preserved.

The following settings were used for the explicit LPV-MPC controllers: Instead of a quadratic cost function, the polytopic maximum norm was minimized,

$$J(\boldsymbol{U}_k, \boldsymbol{T}_k; x_k, x_{\text{ref},k}, \theta_k) = \sum_{i=1}^{N} \|Q(x_{k+i} - x_{\text{ref},k+i})\|_\infty + \sum_{i=0}^{N_c-1} \|Ru_{k+i}\|_\infty ,$$

where the sequence of control inputs and the sequence of unknown future scheduling parameters are denoted by $\boldsymbol{U}_k = \{u_k, u_{k+1}, \dots, u_{k+N-1}\}$ and $\boldsymbol{T}_k = \{\theta_{k+1}, \dots, \theta_{k+N-1}\}$, repectively.

The reference trajectory is assumed constant over the prediction horizon, $x_{\text{ref},k+i} = x_{\text{ref},k} \ \forall i \in \{0, \dots, N\}$. The prediction horizon was reduced to $N = N_c = 3$. The weight matrices are

$$Q = \text{diag}(\begin{bmatrix} 0.15 & 15 & 0 \end{bmatrix}) \quad \text{and} \quad R = 1 .$$

For the computation of the explicit LPV-MPC controller, the longitudinal speed v_x is assumed to vary within

$$10\,\text{m/s} \leq v_x \leq 20\,\text{m/s}.$$

The control input and control rate are constrained as in (14.9). The tire force approximations are only valid for sufficiently small slip angles (compare Figure 14.1). Restricting the slip angle α_f directly resulted in utterly conservative results. Therefore the slip angle is constrained indirectly by tightening the input constraints to $u_{\text{max/min}} = \pm 5\,\text{deg}$. State constraints, which are required in MPT, [KGBM04], were chosen large enough such that they do not effect the control actions during the simulations. Computations were performed using MPT via Yalmip, [KGBM04, Löf04]. The resulting controller (subsequently denoted by LPV-MPC(1)) comprises 2180 regions.

Explicit LPV-MPC with Bounded Rate of Parameter Variation

In the explicit LPV-MPC scheme presented in Chapter 10, it is assumed that the scheduling parameter θ varies arbitrarily within the parameter simplex Θ. If however the rate of parameter variation is bounded, this assumption is conservative, since impossible realizations of the future scheduling parameter are considered in the predictions. An extension of the explicit LPV-MPC scheme is proposed in Chapter 12, which reduces the conservatism, when bounds of the rate of parameter variation are known. The main idea of this extension is to partition the parameter simplex Θ into a number of subsets, and to compute for each subset a separate control law, only considering those future scheduling parameter values which can be obtained if the current scheduling parameter is contained in this subset. The choice of the number of subsets is then a tradeoff between complexity of the resulting control law and conservatism.

Control performance of the LPV-MPC scheme can be improved, if the rate of variation of the scheduling parameter is bounded. In order to verify this, we assumed the scheduling parameter θ to vary less than 0.1 per time step, and separated the parameter simplex in half, resulting in a controller with 4478 regions, denoted by LPV-MPC(2).

Explicit linear MPC

For the linear model from Section 14.2 and for the longitudinal speed $v_x = 10\,\mathrm{m/s}$, an explicit linear MPC controller was computed, [Bor03]. Besides the different prediction model, the setup of the linear MPC scheme is identical to the LPV-MPC schemes, to assure comparability. The resulting controller comprises 561 regions.

14.4 Simulations

This section presents the application of autonomous vehicle steering in a double lane-change maneuver on snow, [KFB+06]. In this control scenario, a car is driving with different constant longitudinal speeds $v_x = 10, 15, 16\,\mathrm{m/s}$ on a road with friction coefficient $\mu = 0.3$. The double lane-change shall be proceeded by following a reference trajectory in Y and ψ, assuming that this reference is supplied by a trajectory planning system. The control objective is to track the reference trajectory as closely as possible. The vehicle is assumed to cruise during the maneuver, no braking or accelerating is proceeded. For the simulation of the car in the control scenario, the nonlinear model (8.8) is used.

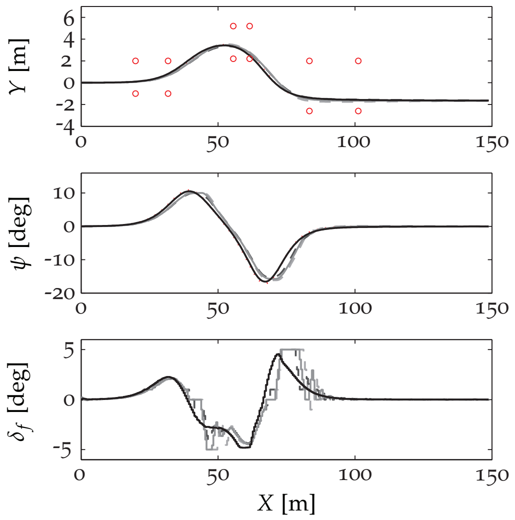


Figure 14.3: Double lane-change maneuver with $v_x = 10\,\mathrm{m/s}$. Reference trajectory ($\cdots$), NMPC (—), LPV-MPC(1) (– –), LPV-MPC(2) (—) and linear MPC (– –).

Figures 14.3, 14.4 and 14.5 show the lateral position, the yaw angle and the steering input for the four MPC schemes at the driving speeds 10 m/s, 15 m/s and 16 m/s, respectively. It can be seen that for 10 m/s, all controllers achieve good results. This is no surprise since the driving speed is relatively low and the linear MPC controller was designed for this speed. For larger speeds the performance deteriorates. At 15 m/s the NMPC and the LPV-MPC controllers can hold the vehicle within the lane, but the linear MPC can not. At 16 m/s, also LPV-MPC(1) is not able to keep the vehicle on lane and at higher speeds also LPV-MPC(2) and then the NMPC controller fail (not depicted). The difficulty of the maneuver increases with speed, and we see that LPV-MPC yields a better control performance than linear MPC, when the vehicle speed differs from the assumed 10 m/s. On the other hand, the control performance of the LPV-MPC schemes can not keep up with NMPC.

What limits the performance of the LPV-MPC schemes compared to NMPC? There are three causes for the introduction of prediction errors and conservatism in the LPV-MPC schemes: (*i*) the assumptions of an icy road ($\mu = 0.1$) and the small angles assumption ($\psi \approx \delta_f \approx \alpha_f \approx \alpha_r \approx 0$) are not precise; (*ii*) the LPV-MPC schemes take future variations of the vehicle speed into account, while the vehicle speed remains constant; (*iii*) the cascade loop, which was selected to decrease the complexity of the explicit solvers, is accompa-

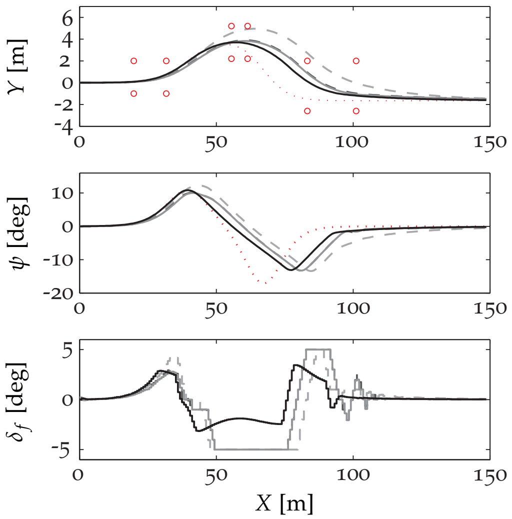


Figure 14.4: Double lane-change maneuver at $v_x = 15\,\mathrm{m/s}$. Reference trajectory $(\cdots)$, NMPC (—), LPV-MPC(1) (– –), LPV-MPC(2) (—) and linear MPC (– –).

nied with some conservatism due to the imposed control structure and the suboptimal tuning of k_P.

Besides the control performance, the online computational effort plays a major role, as it determines the applicability and the costs of the control system. Table 14.1 summarizes the average and maximum CPU-times of the controllers during the simulations, which were performed on a 3 GHz Pentium 4 Processor under Matlab. The online computational burden of explicit LPV-MPC and explicit linear MPC is comparable, while both require orders of magnitude less time than NMPC. It should be noted that no effort was spent to implement the explicit MPC controllers efficiently. A further reduction of the evaluation times of the explicit control laws is to be expected via the computation of a binary search tree, evaluated under C (compare Example 11.4 on page 194).

14.5 Conclusions

Summarizing the results of the simulations, we conclude that the gain-scheduling with respect to the vehicle speed enables explicit LPV-MPC to outperform the explicit linear MPC controller, when the vehicle speed does not

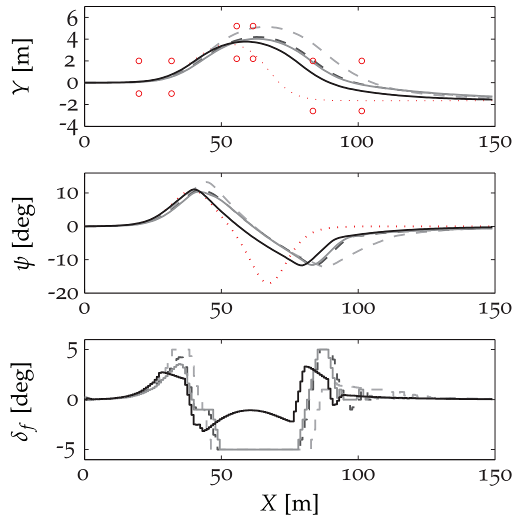


Figure 14.5: Double lane-change maneuver at $v_x = 16\,\text{m/s}$. Reference trajectory $(\cdots)$, NMPC (—), LPV-MPC(1) (– –), LPV-MPC(2) (—) and linear MPC (– –).

v_x	10 m/s	15 m/s	16 m/s
NMPC	66/156 ms	97/248 ms	113/778 ms
LPV-MPC(1)	2.8/3.8 ms	2.6/2.8 ms	2.6/3.3 ms
LPV-MPC(2)	2.5/3.3 ms	2.7/3.5 ms	2.7/4.0 ms
linear MPC	3.1/3.5 ms	3.1/3.4 ms	3.1/3.9 ms

Table 14.1: Online computation times (Average/Maximum).

match the predicted one. At the same time the online computational effort remains similar, i.e. orders of magnitude lower than for the nonlinear MPC scheme, and enables an efficient implementation on an electronic control unit.

In future work, an extension of the LPV model should be considered. The slip ratio could be included as an additional scheduling signal. When reliable information about the road friction coefficient becomes available, it could be used to adapt the steering system to different road frictions. Finally, the proposed control scheme should be implemented and tested in an experimental setup.

Part IV

Appendix

A Notation

THE FOLLOWING NOTATION is used throughout this thesis. In the application-based Chapters 7, 8 and 14 our notation follows the established notation in mechanical engineering. Apart from those chapters, we use lowercase italic letters, x, to denote scalars and vectors, and uppercase italic letters, X, to denote matrices. Spaces are denoted by blackboard-bold letters, $\mathbb{X}$. Calligraphic letters are used for uncountable sets, $\mathcal{X}$, while bold italic capital letters are used for sets with a finite number of elements $\boldsymbol{X} = \{x_1, \dots, x_n\}$. We use i, j, k as indices and n_x to denote the dimension of x. $\mathbf{1}$ denotes a column vector with ones, $\mathbf{1} = [1, \dots, 1]^T$, O a zero matrix of appropriate dimensions. e_j denotes the jth standard basis vector. Element-wise inequalities are denoted by $\leq$, while $\preceq$ denotes matrix inequalities. A $\star$ in a symmetric matrix denotes the transposed complement.

Spaces

$\mathbb{N}$	natural numbers (including zero)
$\mathbb{N}_+$	natural numbers (excluding zero)
$\mathbb{N}^n$	space of n-dimensional vectors of natural numbers
$\mathbb{Z}$	integers
$\mathbb{R}$	real numbers
$\mathbb{R}_+$	nonnegative real numbers
$\mathbb{R}^n$	space of n-dimensional real vectors
$\mathbb{R}^{n_1 \times n_2}$	space of real matrices with n_1 rows and n_2 columns
$\mathbb{R}[\theta]$	polynomial ring on $\mathbb{R}$
$\mathbb{S}^n$	space of symmetric matrices in $\mathbb{R}^{n \times n}$
$\mathbb{S}^n_+$	space of symmetric, positive semidefinite matrices in $\mathbb{R}^{n \times n}$

Sets

$\mathcal{B}$	ball
r, x_c	radius and centrepoint of a ball

$\mathcal{H}$	hyperplane
$\mathcal{P}$	polyhedron
$\mathcal{E}$	ellipsoid
$\mathcal{F}$	face of a polyhedron
Δ, Δ^n	simplex, standard simplex of dimension n
v	vertex of a polyhedron
h, k, H, k	matrices, vectors defining hyperplanes or polyhedra
$\tilde{H}$	scaled constraint matrix

Set Operators

$\mathcal{B}_b(\mathcal{P})$	bounding box of polytope $\mathcal{P}$
$\mathcal{B}_c(\mathcal{P})$	Chebyshev ball of polytope $\mathcal{P}$
$\dim(\mathcal{X})$	dimension of set $\mathcal{X}$
$\mathrm{int}(\mathcal{X})$	strict interior of set $\mathcal{X}$
$\mathrm{relint}(\mathcal{X})$	relative interior of set $\mathcal{X}$
$\bar{\mathcal{X}}$	closure of set $\mathcal{X}$
$\delta\mathcal{X}$	boundary of set $\mathcal{X}$
$\mathrm{affh}(\mathcal{X})$	affine hull of set $\mathcal{X}$
$\mathrm{convh}(\mathcal{X})$	convex hull of set $\mathcal{X}$
$\mathrm{cone}(\mathcal{X})$	conic hull of set $\mathcal{X}$

Function Terminology

f	function
$\tilde{f}$	extension of function f
$\mathrm{dom}(f)$	domain of function f
$\mathrm{dom}_{x_1}(f)$	partial domain of function f regarding variable x_1
$\mathcal{D}$	domain of a function
$\mathrm{epi}(f)$	epigraph of function f
$\|\cdot\|$	norm
$\psi_{\mathcal{X}}$	Minkowski function induced by set $\mathcal{X}$, gauge function
d	distance function
$\det(A)$	determinant of matrix A
λ_i, v_i	ith eigenvalue, eigenvector of a matrix

Polynomials and Pólya's Theorem

α	multi-index
θ^α	monomial
$\deg(\theta^\alpha), \lvert\alpha\rvert$	degree of a monomial
p	polynomial
$\text{supp}(p)$	support of a polynomial p
$d, \deg(p)$	degree of a polynomial p
N_p	Pólya degree
p_{N_p}	polynomial extended by a Pólya relaxation of degree N_p
p^*	minimum of a polynomial over the standard simplex
$c_{\max}$	maximum of the scaled coefficients of a polynomial
$\mathbf{C}_{N_p}$	coefficients of the extended polynomial p_{N_p}

Control Systems

t	time
k	discrete time
T_s	sampling period
T_{sim}	simulation period
$u(t)$, u_k	input vector, continuous-time or discrete-time
$x(t)$, x_k	state vector, continuous-time or discrete-time
$y(t)$, y_k	output vector, continuous-time or discrete-time
$w(t), w_k$	uncertainty vector, continuous-time or discrete-time
f	dynamic system, state-update equation of a dynamic system
f_{DYN}	state-update equation of dynamic system
f_{PWA}	piecewise affine state-update equation
g	output equation of a dynamic system
A	state transition matrix
B	input matrix
C	output matrix
D	feed-through matrix
f	affine term
$\bar{x}$	equilibrium point
V	Lyapunov function
P	Lyapunov matrix
$\mathcal{D}$	domain of attraction

λ	contraction ratio
$(\cdot)_{\mathrm{ref}}$	reference value

Optimization

J	cost function, objective function
J^*	optimal value (function)
u	optimization variable
u^*	solution
$\bar{u}$	locally optimal point
$\mathcal{U}$	feasible set, set of candidate solutions
$\mathcal{U}^*$	optimal set
x	parameter
$\mathcal{X}$	set of parameters
$\mathcal{X}_f$	set of feasible parameters
g_i	ith constraint
$\boldsymbol{A}, g_{\boldsymbol{A}}$	set of active constraints
$\boldsymbol{I}, g_{\boldsymbol{I}}$	set of inactive constraints
$\mathcal{R}$	critical region
n_R	number of critical regions
t	epigraph variable
$(\cdot)_c, (\cdot)_b$	continuous variable, binary variable
n_{uc}, n_{ub}	number of continuous variables, number of binary variables

Constrained Optimal Control

i	prediction step
N	prediction horizon
N_c	control horizon
N^*	minimum number of steps to reach a target set
J	cost function
J_i	truncated cost function, cost-to-go at ith prediction step
J_{act}	actually sustained cost
L_i	stage costs at ith prediction step
L_N	terminal cost

Q, R, P	weight matrices
s	slack variable for soft constraints
q_s	weight of slack variable
Δu	input rate
μ	control law
μ_T	terminal controller
$\boldsymbol{U}_k$	sequence of control actions
$\boldsymbol{U}_k^*$	optimal control sequence
$\mathcal{U}$	set of admissible inputs
$\boldsymbol{\pi}$	control policy
$\boldsymbol{\pi}^*$	optimal policy
Π	set of admissible control policies
$\boldsymbol{T}_k$	sequence of future scheduling parameters
$\boldsymbol{W}_k$	sequence of uncertainties
$\mathcal{W}$	uncertainty set
$\bar{x}$	operating point
$\tilde{x}$	difference to operating point, $x - \bar{x}$
$\boldsymbol{X}_{\text{ref}}$	sequence of future reference states
$\mathcal{X}$	set of admissible states
$\mathcal{X}_f$	set of feasible states
$\mathcal{X}_i$	set of feasible states at ith prediction step
$\mathcal{X}_T$	terminal set
$\mathcal{R}$	controller region
$\mathcal{P}$	controller partition
F_r, g_r	state-feedback matrix and affine term in rth controller region
n_r	number of regions

Piecewise Affine Systems

$\mathcal{D}$	domain of a PWA system
n_D	number of polyhedral regions of a PWA system
P_x, P_u, p_0	constraints describing domains in state-input space
z_k, δ_k	continuous and binary auxiliary variables of the MLD form.
E_i, e_i	constraints of the MLD formulation
s	switching sequence
n_s	number of switching sequences

n_p	number of partitions
m_i	mode of the system at the ith prediction step
$(\cdot)_j$	related to the jth dynamic

Regions Removal

ΔJ	difference of cost functions
A, b, c	quadratic cost function coefficients
$\Delta A, \Delta b, \Delta c$	difference of cost function coefficients
n_p	number of partitions
n_p^r	number of regions in the pth partition
n_r	number of regions of a control law
$(\cdot)_r^p$	related to region r in partition p
α_k	angle of sine-cosine system

Linear Parameter-Varying Systems

$\rho(t), \rho_k$	scheduling signal, continuous-time or discrete-time
$\bar{\rho}$	constant scheduling signal
f_ρ	scheduling function
$\theta(t), \theta_k$	scheduling parameter, continuous-time or discrete-time
$\theta_{k,j}$	jth element of the scheduling parameter at time step k
$\Delta\theta_k$	rate of parameter variation
Θ	parameter simplex
Θ_c	cth subset of parameter simplex
$\Theta_{c,i,s}$	sth subsimplex of the successor of Θ_c at prediction step i
Γ	bounds on the rate of parameter variation
$n_{c,i}$	number of subsimplices of $\Theta_{c,i}$
μ_Γ	control law considering the rate bounds Γ
μ_{Θ_c}	control law in subset Θ_c of parameter simplex
$\mu_{i,j}$	control law at the ith prediction step at the jth vertex of Θ
$\boldsymbol{U}_i$	set of vertex control laws at ith prediction step
$A(\theta), B(\theta)$	parameter-varying system matrices
A_j, B_j	system matrices at the $j$the vertex of the parameter simplex
A_0, B_0	nominal system matrices
$\Delta A, \Delta B$	difference to nominal system matrices
$f(\theta)$	parameter-varying affine term

Backlash

α	half size of backlash gap
θ	shaft angle
θ_b	backlash position angle
$\Delta\theta$	total shaft displacement
ω	rotational speed of shaft
$\omega_{l,\mathrm{ref}}$	reference value of load shaft speed
$\Delta\omega$	shaft speed difference
$\Delta\omega_{\max}$	maximal shaft speed difference
x_{aug}	augmented state
$\Delta y(t)$	output estimation error
b	damping coefficient
c	spring coefficient
I	moment of inertia
K	driving motor gain
M_1, M_2	driving motor, disturbance motor
T	torque
$(\cdot)_l, (\cdot)_m$	related to load shaft, related to motor shaft
$(\cdot)_{sh}$	related to central shaft element
$(\cdot)_{bl}, (\cdot)_{co}$	in backlash mode, in contact
$\hat{\cdot}$	estimated variable

Autonomous Vehicle Steering

X, Y	road coordinate system
x, y	vehicle coordinate system
v	velocity
ψ	yaw angle
δ	steering angle
α	slip angle
s	slip ratio
μ	road friction coefficient
v_x	vehicle speed
a, b	distance between centre of mass and front/rear wheels
F	force
I	vehicle inertia

m	vehicle mass
M	moment
c_i, k_i	coefficients
J_{cum}	cumulated cost
$(\cdot)_l$, $(\cdot)_c$	in longitudinal wheel direction, in cornering wheel direction
$(\cdot)_f$, $(\cdot)_r$	front wheels, rear wheels
$(\cdot)_x$, $(\cdot)_y$	in longitudinal vehicle direction, in lateral vehicle direction
$(\cdot)_w$	wind
$\hat{\cdot}$	approximated saturated variable
$\tilde{\cdot}$	approximated variable

Acronyms

CFTOC	constrained finite-time optimal control
CROC	constrained robust optimal control
OL-CROC	open-loop constrained robust optimal control
CL-CROC	closed-loop constrained robust optimal control
CPVOC	constrained parameter-varying optimal control
CTOC	constrained time-optimal control
DC	direct current
DP	dynamic programming
EKF	extended Kalman filter
HPV	hybrid parameter-varying
KKT	Karush-Kuhn-Tucker
LICQ	linear inequality constraint qualification
LP	linear program
LPV	linear parameter-varying
LPV-A	linear parameter-varying state transition matrix
LQ	linear quadratic
LQR	linear quadratic regulator
LTI	linear time-invariant
LTV	linear time-varying
MHE	moving horizon estimation
MILP	mixed-integer linear program
MINLP	mixed-integer nonlinear program
MIQP	mixed-integer quadratic program
MPC	model predictive control
MPT	multi-parametric toolbox
NP	nonlinear programming, non-deterministic polynomial-time
PID	proportional-integral-derivative
pLP	parametric linear program
pMILP	parametric mixed-integer linear program

pMIQP	parametric mixed-integer quadratic program
pQP	parametric quadratic program
PWA	piecewise affine
QP	quadratic program
SDP	semidefinite program

B Mathematical Definitions

The material of this thesis is based on a number of mathematical notions, definitions and concepts. This appendix was added for the sake of completeness. Instead of being read linearly, it is intended to be used by the reader to look up forgotten or unclear definitions. The advanced reader may safely skip this chapter.

Of course it is not possible to cover all mentioned topics in their adequate depth, such that we will give references to textbooks for further reading. If not stated otherwise, we will restrict ourselves to the Euclidean space $\mathbb{R}^n$.

B.1 Set Terminology

The first section states basic set notions which are commonly used throughout this thesis. Starting with set algebra and basic topological concepts, this section covers also some types of sets such as affine sets, convex sets and conic sets. Finally, polyhedra are covered. For further reading see [Dev93, Cro97] or [Roc70].

Set Algebra

Definition B.1 (Set operations) *Given the two sets $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^n$, we denote the following algebraic operations:*

$$\begin{aligned}
\mathcal{X} \cup \mathcal{Y} &:= \{z \mid (z \in \mathcal{X}) \vee (z \in \mathcal{Y})\} && \textit{(Union)}\\
\mathcal{X} \cap \mathcal{Y} &:= \{z \mid (z \in \mathcal{X}) \wedge (z \in \mathcal{Y})\} && \textit{(Intersection)}\\
\mathcal{X}^c &:= \{z \mid z \notin \mathcal{X}\} && \textit{(Complement)}\\
\mathcal{X} \setminus \mathcal{Y} &:= \{z \mid (z \in \mathcal{X}) \wedge (z \notin \mathcal{Y})\} && \textit{(Difference)}
\end{aligned}$$

$$\begin{aligned}
\mathcal{X} \oplus \mathcal{Y} &:= \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\} && \textit{(Minkowski sum)}\\
\mathcal{X} \ominus \mathcal{Y} &:= \{z \mid (z + y \in \mathcal{X}) \quad \forall y \in \mathcal{Y}\} && \textit{(Pontryagin difference)}
\end{aligned}$$

Minkowski sum is sometimes called *dilatation*, Pontryagin difference *erosion* or *Minkowski difference*.

Topological Concepts

Definition B.2 (n-dimensional Ball) *Let $d\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}_+$ be a distance function. Then an n-*dimensional open ball *around (or a* neighbourhood *of) a centre point x_c with radius $r > 0$ is defined as*

$$\mathcal{B}(x_c, r) := \{x \in \mathbb{R}^{n_x} \mid d(x_c, x) < r\}.$$

A closed ball *is defined accordingly by replacing the strict inequality by an inequality.*

Example B.1 (Euclidean ball) *An* Euclidean ball *is a ball defined by the Euclidean distance,*

$$\mathcal{B}_e(x_c, r) = \{x \in \mathbb{R}^{n_x} \mid \|x - x_c\|_2 < r\}.$$

Example B.2 (Unit ball) *An* open unit ball *is an open ball with radius 1 around the origin, based on a norm $\|\cdot\|_p\colon \mathbb{R}^{n_x} \to \mathbb{R}_+$,*

$$\mathcal{B}_p = \{x \in \mathbb{R}^{n_x} \mid \|x\|_p < 1\}.$$

A closed unit ball *is defined accordingly by replacing the strict inequality by an inequality. Figure B.2 shows the closed unit balls induced by common vector norms.*

Definition B.3 (Dimension) *Let $\mathcal{X}$ be a set in $\mathbb{R}^{n_x}$. By* $\dim(\mathcal{X})$ *we denote the* dimension *of $\mathcal{X}$, defined as the dimension of the affine hull of $\mathcal{X}$. The dimension of an affine set is defined as the dimension of the subspace parallel to it.*

Definition B.4 (Full-dimensional set) *A set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is called* full-dimensional set *if* $\dim(\mathcal{X}) = n_x$. *Otherwise it is called* lower-dimensional.

Definition B.5 (Interior) *The* interior *of a set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is defined as*

$$\operatorname{int}(\mathcal{X}) := \{x \in \mathbb{R}^{n_x} \mid \exists \varepsilon > 0 : \mathcal{B}(x, \varepsilon) \subseteq \mathcal{X}\}.$$

Definition B.6 (Relative interior) *The* relative interior *of a set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is defined as*

$$\operatorname{relint}(\mathcal{X}) := \{x \in \mathbb{R}^{n_x} \mid \exists \varepsilon > 0 : (\operatorname{affh}(\mathcal{X}) \cap \mathcal{B}(x, \varepsilon)) \subseteq \mathcal{X}\}.$$

Definition B.7 (Open and closed set) *A set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is called* open *if* $\operatorname{relint}(\mathcal{X}) = \mathcal{X}$. *A set is called* closed *if $\mathcal{X}^c$ is open.*

Definition B.8 (Closure) *The smallest closed set containing a set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is called the* closure *of $\mathcal{X}$,*

$$\bar{\mathcal{X}} := \{x \in \mathbb{R}^{n_x} \mid (\mathcal{B}(x, \varepsilon) \cap \mathcal{X}) \neq \emptyset \quad \forall \varepsilon > 0\}\,.$$

Definition B.9 (Boundary) *The* boundary *$\partial\mathcal{X}$ of a set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the difference between its closure and its relative interior,*

$$\partial\mathcal{X} := \bar{\mathcal{X}} \setminus \operatorname{relint}(\mathcal{X})\,.$$

Definition B.10 (Bounded set) *A set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is called* bounded *if it is contained in a ball of finite radius, i.e. if $\exists x_c \in \mathbb{R}^{n_x}, r < \infty : \mathcal{X} \subseteq \mathcal{B}(x_c, r)$.*

Definition B.11 (Compact set) *A set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is called* compact *if every sequence $\{x_i\}_{i \in \mathbb{N}}$ with $x_i \in \mathcal{X}$ $\forall i \in \mathbb{N}$ contains a subsequence that converges to a point $\hat{x} \in \mathcal{X}$.*

Theorem B.1 (Heine-Borel) *A set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is compact if and only if it is a closed and a bounded set.*

Definition B.12 (Connected set) *A set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is called* connected *if it cannot be partitioned into two nonempty subsets such that each subset has no common points with the closure of the other set. Otherwise it is called* disconnected.

Definition B.13 (Collection of sets) *A finite number of sets $\{\mathcal{X}_i\}_{i=1}^{n} \subset \mathbb{R}^{n_x}$ is called* collection of sets *or* family of sets.

Definition B.14 (Partition) *Let $\mathcal{X} \subset \mathbb{R}^{n_x}$ be a set and $\mathcal{P} = \{\mathcal{X}_i\}_{i=1}^{n} \subset \mathbb{R}^{n_x}$ be a collection of sets. $\mathcal{P}$ is called a* partition *of $\mathcal{X}$, if the following conditions hold:*

1. $\displaystyle\bigcup_{i=1}^{n} \mathcal{X}_i = \mathcal{X}$ *(Collectively exhaustive)*,
2. $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset \quad \forall i \neq j$ *(Mutally exclusive)*.

Remark B.1 *With an abuse of notation, we will call a collection of sets also a* partition (in a broader sense)*, if it is collectively exhaustive and the relative interiors are mutually exclusive, i.e.*

$$\operatorname{relint}(\mathcal{X}_i) \cap \operatorname{relint}(\mathcal{X}_j) = \emptyset \quad \forall i \neq j\,.$$

Affine Sets

Definition B.15 (Affine combination) *An* affine combination *of a finite number of points $x_1, x_2, \dots, x_n \in \mathbb{R}^{n_x}$ is a point $x \in \mathbb{R}^{n_x}$ which can be expressed by*

$$x = \sum_{i=1}^{n} \lambda_i x_i \quad \textit{with some coefficients} \quad \lambda_i \in \mathbb{R} : \sum_{i=1}^{n} \lambda_i = 1 \,.$$

Definition B.16 (Affine set) *A set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is called* affine *if for any points $x_1, \dots, x_n \in \mathcal{X}$ all affine combinations are contained in the set, i.e.*

$$\sum_{i=1}^{n} \lambda_i x_i \in \mathcal{X} \quad \forall \lambda_i \in \mathbb{R} : \sum_{i=1}^{n} \lambda_i = 1 \,.$$

Definition B.17 (Affine hull) *The* affine hull *of a set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the set of all affine combinations of its elements:*

$$\operatorname{affh}(\mathcal{X}) := \left\{ \sum_{i=1}^{n} \lambda_i x_i \,\middle|\, x_i \in \mathcal{X},\, \lambda_i \in \mathbb{R} : \sum_{i=1}^{n} \lambda_i = 1, n \in \mathbb{N} \right\} .$$

Convex Sets

The convexity of a set is a property of paramount importance to all kinds of mathematical operations. This section states basic definitions of convexity, [Roc70].

Definition B.18 (Convex combination) *A* convex combination *of a finite number of points $x_1, x_2, \dots, x_n \in \mathbb{R}^{n_x}$ is a point $x \in \mathbb{R}^{n_x}$ which can be expressed by*

$$x = \sum_{i=1}^{n} \lambda_i x_i \quad \textit{with some coefficients} \quad \lambda_i \in \mathbb{R}_+ : \sum_{i=1}^{n} \lambda_i = 1 \,.$$

Definition B.19 (Convex set) *A set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is called* convex *if for any points $x_1, \dots, x_n \in \mathcal{X}$ all convex combinations are contained in the set, i.e.*

$$\sum_{i=1}^{n} \lambda_i x_i \in \mathcal{X} \quad \forall \lambda_i \in \mathbb{R}_+ : \sum_{i=1}^{n} \lambda_i = 1 \,.$$

Definition B.20 (Convex hull) *The* convex hull *of a set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ denotes the smallest convex set that contains $\mathcal{X}$,*

$$\operatorname{convh}(\mathcal{X}) = \left\{ \sum_{i=1}^{n} \lambda_i x_i \,\middle|\, x_i \in \mathcal{X},\, \lambda_i \in \mathbb{R}_+ : \sum_{i=1}^{n} \lambda_i = 1,\, n \in \mathbb{N} \right\} .$$

Definition B.21 (C-set) *A set $\mathcal{X} \subset \mathbb{R}^{n_x}$ is called* C-set *if it is a convex and compact set containing the origin as an interior point.*

Conic Sets

Definition B.22 (Conic combination) *A* conic combination *of a finite number of points $x_1, x_2, \ldots, x_n \in \mathbb{R}^{n_x}$ is a point $x \in \mathbb{R}^{n_x}$ which can be expressed by*

$$x = \sum_{i=1}^{n} \lambda_i x_i \quad \textit{with some coefficients} \quad \lambda_i \in \mathbb{R}_+ .$$

Definition B.23 (Cone) *A set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is called* cone, *if $\lambda x \in \mathcal{X}$ for all $x \in \mathcal{X}$ and for all non-negative $\lambda \in \mathbb{R}_+$. A cone is called* proper *if it is closed, convex, solid (nonempty interior) and pointed ($x, -x \in \mathcal{X}$ implies $x = 0$).*

A proper cone $\mathcal{X}$ induces a *partial ordering* of the elements in $\mathbb{R}^{n_x}$. We write $x_1 \succeq x_2$ if and only if $x_1 - x_2 \in \mathcal{X}$. Analogue, we write $x_1 \succ x_2$ if and only if $x_1 - x_2 \in \operatorname{int}(\mathcal{X})$.

Example B.3 *The set of symmetric positive semidefinite matrices forms a proper cone, denoted by $\mathbb{S}^n_+$.*

Definition B.24 (Conic hull) *The* conic hull *of a set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ denotes the smallest conic set that contains $\mathcal{X}$,*

$$\operatorname{cone}(\mathcal{X}) = \left\{ \sum_{i=1}^{n} \lambda_i x_i \;\middle|\; x_i \in \mathcal{X},\, \lambda_i \in \mathbb{R}_+,\, n \in \mathbb{N} \right\}.$$

Example B.4 (Different hulls of two points) *Figure B.1 depicts the affine hull,* $\operatorname{affh}(\boldsymbol{X})$*, the convex hull,* $\operatorname{convh}(\boldsymbol{X})$*, and the conic hull,* $\operatorname{cone}(\boldsymbol{X})$*, of a set $\boldsymbol{X}$ consisting of two points, $\boldsymbol{X} = \{x_1, x_2\}$.*

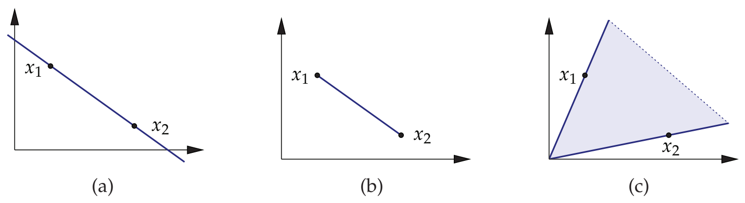


Figure B.1: Affine hull (a), convex hull (b) and conic hull (c) of two points x_1 and x_2 in $\mathbb{R}^2$.

Polyhedra

Polyhedra constitute the mostly used sets in this thesis. In the following we cover some common basic concepts. More on polyhedra can be found e.g. in [Zie95] or [Grü03].

Definition B.25 (Hyperplane) *A* hyperplane $\mathcal{H} \subset \mathbb{R}^{n_x}$ *is a set of the form*

$$\mathcal{H} := \{x \in \mathbb{R}^{n_x} \mid h^T x = k\}$$

with some $h \in \mathbb{R}^{n_x}, h \neq 0$ *and* $k \in \mathbb{R}$. h *is the* normal vector *of the hyperplane.*

Definition B.26 (Supporting hyperplane) *A hyperplane* $\mathcal{H} \subset \mathbb{R}^{n_x}$ *is said to* support *a set* $\mathcal{X} \subset \mathbb{R}^{n_x}$ *if*

(i) *the hyperplane contains at least one point of the set,* $\mathcal{H} \cap \mathcal{X} \neq \emptyset$,

(ii) *the set lies only on one side of the hyperplane,* $\nexists x_1, x_2 \in \mathcal{X} : (h^T x_1 < k) \wedge (h^T x_2 > k)$.

Definition B.27 (Half-space) *An* open half-space *is the set* $\{x \in \mathbb{R}^{n_x} \mid h^T x < k\}$ *with some* $h \in \mathbb{R}^{n_x}, h \neq 0$ *and* $k \in \mathbb{R}$. *A* closed half-space *is the set* $\{x \in \mathbb{R}^{n_x} \mid h^T x \leq k\}$. h *is the* outward normal vector *of the half-space.*

Definition B.28 (Polyhedron) *A* polyhedron $\mathcal{P} \subset \mathbb{R}^{n_x}$ *is the intersection of finitely many halfspaces, i.e.*

$$\mathcal{P} = \{x \in \mathbb{R}^{n_x} \mid Hx \leq k\} = \{x \in \mathbb{R}^{n_x} \mid h_i^T x \leq k_i,\ i = 1, \ldots, n_c\}$$

with $H = [h_1, \ldots, h_{n_c}]^T \in \mathbb{R}^{n_c \times n_x}, k \in \mathbb{R}^{n_c}$.

Definition B.29 (Polyhedral set) *A set* $\mathcal{X} \subset \mathbb{R}^{n_x}$ *is called a* polyhedral set *if its closure is a polyhedron,*

$$\bar{\mathcal{X}} = \{x \in \mathbb{R}^{n_x} \mid Hx \leq k\}.$$

If a polyhedron $\mathcal{P}$ contains the origin in its interior, the elements of the right-hand side vector k are positive, $k \in \mathbb{R}_+^{n_c}$, and it is possible to rewrite the polyhedron as $\mathcal{P} = \{x \in \mathbb{R}^{n_x} \mid \tilde{H}x \leq \mathbf{1}\}$ where $\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^{n_c}$ denotes the vector of ones.

Definition B.30 (Redundant inequality) *An inequality describing a polyhedron is called* redundant *if adding or removing it does not change the polyhedron.*

Definition B.31 (Minimal representation of a polyhedron) *A representation of a polyhedron is called* minimal *if it does not contain redundant inequalities.*

Definition B.32 (Polytope) *A* polytope *is a compact polyhedron.*

An equivalent definition of a polytope is the convex hull of finitely many points in $\mathbb{R}^{n_x}$. A polytope can thus either be specified by its half-spaces (*half-space or H-representation*) or by its vertices (*vertex or V-representation*). The translation from one representation to another can be a difficult task, in the worst case with exponential complexity.

Definition B.33 (Full-dimensional polytope) *We call a polytope $\mathcal{P} \subset \mathbb{R}^{n_x}$* full-dimensional, *if* $\dim(\mathcal{P}) = n_x$. *Otherwise we call it* lower-dimensional.

Definition B.34 (Face) *A* face *of an n-dimensional polyhedron $\mathcal{P}$ is the intersection of $\mathcal{P}$ with some of its supporting hyperplanes. Depending on the dimension d of the face, we call a face: polyhedron ($d = n$),* facet *($d = n - 1$),* ridge *($d = n - 2$),* edge *(d = 1),* vertex *(d = 0) or empty set.*

The following proposition states some basic properties of faces of polytopes.

Proposition B.2 *Let $\mathcal{P} \subset \mathbb{R}^{n_x}$ be a polytope and let $\mathcal{F}$ be one of its faces. Then the following statements hold:*

1. *$\mathcal{F}$ is a polytope.*
2. *The intersection of faces of $\mathcal{P}$ is a face of $\mathcal{P}$.*
3. *The faces of $\mathcal{P}$ contained in $\mathcal{F}$ are exactly the faces of $\mathcal{F}$.*
4. *The vertices of $\mathcal{P}$ contained in $\mathcal{F}$ are exactly the vertices of $\mathcal{F}$.*
5. *$\mathcal{P}$ is the convex hull of its vertices.*

Definition B.35 (Simplex) *An n-dimensional* simplex *is a polytope with $n + 1$ linear independent vertices $x_1, x_2, \ldots, x_{n+1} \in \mathbb{R}^n$,*

$$\Delta := \left\{ x \in \mathbb{R}^n \;\middle|\; x = \sum_{i=1}^{n+1} \lambda_i x_i \quad with \quad \lambda_i \in \mathbb{R}_+ : \sum_{i=1}^{n+1} \lambda_i = 1 \right\} .$$

Example B.5 (Standard simplex) *The* standard n-simplex *is a simplex spanned by the unit vectors,*

$$\Delta^n = \left\{ x = [x_1, \ldots, x_{n+1}]^T \in \mathbb{R}^{n+1} \;\middle|\; x_i \in \mathbb{R}_+ : \sum_{i=1}^{n+1} x_i = 1 \right\} .$$

Definition B.36 (Polyhedral partition) *A* polyhedral partition *is a partition of a set into finitely many subsets such that the closure of each subset is a polyhedron.*

Definition B.37 (Bounding box) *The* bounding box $\mathcal{B}_b(\mathcal{P}) \in \mathbb{R}^{n_x}$ *of a polytope $\mathcal{P}$ is the smallest axis-aligned, hyperrectangle containing $\mathcal{P}$, and is determined by the maximal and minimal values of $\mathcal{P}$ in each dimension.*

$$\mathcal{B}_b(\mathcal{P}) := \left\{x = [x_1, \ldots, x_{n_x}]^T \in \mathbb{R}^{n_x} \mid \forall x_i \exists \hat{x}_j \forall j \neq i : [\hat{x}_1 \ldots x_i \ldots \hat{x}_{n_x}]^T \in \mathcal{P}\right\}$$

Definition B.38 (Chebyshev ball) *The largest Euclidean ball inside a polytope $\mathcal{P}$ is the* Chebyshev ball*:*

$$\mathcal{B}_c(\mathcal{P}) := \mathcal{B}_e(x_c, r_c) \text{ with } r_c = \max_{\{r, x_c\}} \{r : \mathcal{B}_e(x_c, r) \subset \mathcal{P}\} .$$

The centre x_c is called Chebyshev centre.

The computation of a Chebyshev ball is a computationally inexpensive operation requiring only the solution of a single linear program. The Chebyshev centre need not to be unique.

B.2 Function Terminology

Functions define relations between sets. This section describes function terminology from real analysis, and covers properties of norms and matrices. For more insight into these topics see a textbook on mathematics, e.g. [Str86, Str93, Str91].

Real Analysis

Definition B.39 (Domain) *Given a function $f \colon \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ we denote the* domain *by*

$$\text{dom}(f) := \{x \in \mathbb{R}^{n_x} \mid f(x) \text{ exists }\} .$$

Given a function of multiple variables $f \colon \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \cdots \times \mathbb{R}^{n_n} \to \mathbb{R}^{n_y}$ we denote the (partial) domain *by*

$$\text{dom}_{x_1}(f(x_1, \ldots, x_n)) := \{x_1 \in \mathbb{R}^{n_1} \mid f(x_1, x_2, \ldots, x_n) \text{ exists }\} .$$

Note that the partial domain of a function is set-valued map of the remaining variables. We will often refer to the domain of a function by $\mathcal{D}$, and this domain will also be used in the definition of a function, e.g. $f \colon \mathcal{D} \to \mathbb{R}^{n_y}$.

Definition B.40 (Extension of a function) *Given a function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with the domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ we denote the* extension *of f by*

$$\tilde{f}(x) = \begin{cases} f(x), & x \in \operatorname{dom}(f) \\ \infty, & x \notin \operatorname{dom}(f) \end{cases}$$

With a slight abuse of notation we will use the same symbol for the function f and its extension.

Definition B.41 (Epigraph of a function) *The* epigraph *of a function $f\colon \mathcal{D} \to \mathbb{R}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is defined as*

$$\operatorname{epi}(f) := \{(x,t) \in \mathbb{R}^{n_x+1} \mid f(x) \leq t\}.$$

Definition B.42 (Definite function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with the domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called*

- positive definite *if $f(0) = 0$ and $f(x) > 0\ \forall x \in \mathcal{D} \setminus 0$,*
- positive semidefinite *if $f(0) = 0$ and $f(x) \geq 0\ \forall x \in \mathcal{D} \setminus 0$,*
- negative definite *if $f(0) = 0$ and $f(x) < 0\ \forall x \in \mathcal{D} \setminus 0$,*
- negative semidefinite *if $f(0) = 0$ and $f(x) \leq 0\ \forall x \in \mathcal{D} \setminus 0$,*
- indefinite *if $f(0) = 0$ and $\exists x_1, x_2 \in \mathcal{D} : f(x_1) > 0 \wedge f(x_2) < 0$.*

Definition B.43 (Continuous function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* continuous *at a point $\bar{x} \in \mathcal{D}$ if*

$$\forall \varepsilon > 0\ \exists \delta > 0 : \|f(x) - f(\bar{x})\| < \varepsilon \quad \forall x \in \mathcal{B}(\bar{x}, \delta).$$

A function is called continuous *if it is continuous at each point $x \in \mathcal{D}$. Otherwise it is called* discontinuous*.*

Definition B.44 (Partial derivative) *The* partial derivative *of a function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ in the direction x_i is defined as*

$$\frac{\partial f}{\partial x_i}(x) = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_{n_x}) - f(x_1, \ldots, x_i, \ldots, x_n)}{h}.$$

Definition B.45 (Differentiable function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* differentiable *at a point $\bar{x} \in \mathcal{D}$ if the partial derivatives in all directions exist. A function is called* differentiable *if it is differentiable at each point $x \in \mathcal{D}$.*

Definition B.46 (Gradient) *The* gradient *of a scalar-valued function $f\colon \mathcal{D} \to \mathbb{R}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is the vector*

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_{n_x}}(x) \right) .$$

Definition B.47 (Continuously differentiable function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* continuously differentiable, *if it is differentiable and its derivative is continuous.*

Definition B.48 (Convex function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* convex *if $\mathcal{D}$ is convex and if the condition*

$$f(\sum_{i=1}^{n} \lambda_i x_i) \leq \sum_{i=1}^{k} \lambda_i f(x_i) \quad \textit{holds} \quad \forall \lambda_i \in \mathbb{R}_+ : \sum_{i=1}^{n} \lambda_i = 1, n \in \mathbb{N}$$

and for any points $x_1, x_2, \dots, x_n \in \mathcal{D}$. A function f is called strictly convex, *if the condition holds with strict inequality.*

A function is convex if and only if its epigraph, $\text{epi}(f)$, is a convex set. If the epigraph is a polyhedron, we use the term *polyhedral function*.

Definition B.49 (Concave function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with a convex domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* concave *if $-f$ is convex, and* strictly concave *if $-f$ is strictly convex.*

Definition B.50 (Affine function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* affine *if it is convex and concave.*

Definition B.51 (Piecewise affine function) *A function $f_{PWA}\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with the domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is* piecewise affine *(PWA), if $\{\mathcal{D}_i\}_{i=1}^n$ is a partition of $\mathcal{D}$ and*

$$f_{PWA}(x) = A_i x + b_i \quad \forall x \in \mathcal{D}_i$$

with $A_i \in \mathbb{R}^{n_y \times n_x}$ and $b_i \in \mathbb{R}^{n_y}$ and $i \in \{1, \dots, n\}$. Moreover, a function is called polyhedral piecewise affine *(PPWA) or* piecewise affine over polyhedra, *if $\{\mathcal{D}_i\}_{i=1}^n$ is a polyhedral partition of $\mathcal{D}$.*

Definition B.52 (Polyhedral function) *A function $f\colon \mathcal{D} \to \mathbb{R}^{n_y}$ with the domain $\mathcal{D} \subseteq \mathbb{R}^{n_x}$ is called* polyhedral function, *if its epigraph is a polyhedron,*

$$\text{epi}(f) = \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \in \mathbb{R}^{n_x+1} \middle| H \begin{pmatrix} x \\ t \end{pmatrix} \leq k \right\} ,$$

with $H = [h_1, \dots, h_{n_c}]^T \in \mathbb{R}^{n_c \times n_x+1}, k \in \mathbb{R}^{n_c}$.

Norms

Definition B.53 (Norm) *A function* $\|\cdot\|\colon \mathbb{R}^{n_x} \to \mathbb{R}$ *is called a* norm *if for any* $x_1, x_2 \in \mathbb{R}^{n_x}$ *and any* $\lambda \in \mathbb{R}$ *it fulfils the conditions:*

1. $\|x_1\| \geq 0$ *(Nonnegativity)*,
2. $\|x_1\| = 0 \Leftrightarrow x_1 = 0$ *(Definiteness)*,
3. $\|\lambda x_1\| = |\lambda|\|x_1\|$ *(Absolute homogeneity)*,
4. $\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|$ *(Subadditivity)*.

Example B.6 (Vector p-norm) *For a* $p \in [1, \infty)$ *the* vector p-norm *of* $x \in \mathbb{R}^{n_x}$ *is defined as*

$$\|x\|_p := \left(\sum_{i=1}^{n_x} |x_i|^p \right)^{\frac{1}{p}} .$$

The most common choices for are $p = 1, p = 2$ *and* $p \to \infty$*, resulting in the norms*

$$\|x\|_1 = \sum_{i=1}^{n_x} |x_i| \qquad \textit{(Taxicab norm)},$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n_x} x_i^2} \qquad \textit{(Euclidean norm)},$$

$$\|x\|_\infty = \max_i (|x_1|, |x_2|, \dots, |x_{n_x}|) \qquad \textit{(Maximum norm)}.$$

Definition B.54 (Minkowski function) *Given a C-set* $\mathcal{X} \subset \mathbb{R}^{n_x}$*, the corresponding* Minkowski function *(or* gauge function*) is defined as*

$$\psi_{\mathcal{X}}(x) := \inf_{\lambda} \{\lambda \in \mathbb{R}_+ \mid x \in \lambda \mathcal{X}\} .$$

A Minkowski function is convex, continuous and the unit ball of the Minkowski function is the set $\mathcal{X}$. Moreover, a Minkowski function $\psi_{\mathcal{X}}$ satisfies all properties of a gauge function:

1. $\psi_{\mathcal{X}}(x_1) \geq 0$ (Nonnegativity),
2. $\psi_{\mathcal{X}}(x_1) = 0 \Leftrightarrow x_1 = 0$ (Definiteness),
3. $\psi_{\mathcal{X}}(\lambda x_1) = \lambda \psi_{\mathcal{X}}(x_1)\ \forall \lambda \in \mathbb{R}_+$ (Positive homogeneity),
4. $\psi_{\mathcal{X}}(x_1 + x_2) \leq \psi_{\mathcal{X}}(x_1) + \psi_{\mathcal{X}}(x_2)$ (Subadditivity).

Gauge functions are generalizations of norms with similar properties, but with unit balls not necessarily symmetric to the origin. If a C-set $\mathcal{X}$ is symmetric with respect to the origin, the induced Minkowski function is a norm.

If moreover the C-set $\mathcal{X}$ is a polytope/polyhedron, the induced norm is called *polytopic/polyhedral norm*.

Definition B.55 (Polyhedral norm) *We call a norm* polyhedral norm *if the closed unit ball based on this norm is a polyhedron.*

Example B.7 *Figure B.2 shows the closed unit balls of the taxicab norm, $\mathcal{B}_1$, the Euclidean norm, $\mathcal{B}_2$, and the maximum norm, $\mathcal{B}_\infty$, in $\mathbb{R}^2$. The taxicab norm and the maximum norm are both polyhedral norms, while the Euclidean norm is not a polyhedral norm.*

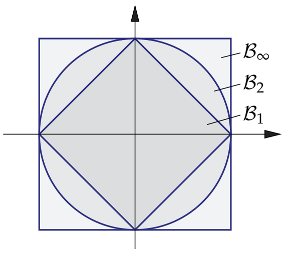


Figure B.2: Closed unit balls of the taxicab norm, the Euclidean norm, and the maximum norm.

Proposition B.3 (Distance function) *Each norm* $\|\cdot\|\colon \mathbb{R}^{n_x} \to \mathbb{R}$ *induces a* distance function *or* metric $d\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}_+$ *by*

$$d(x_1, x_2) = \|x_2 - x_1\| \, .$$

Matrices

Let $\mathbb{S}^n \subset \mathbb{R}^{n\times n}$ denote the space of real symmetric square matrices of size $n \times n$. We denote the space of positive semidefinite matrices by $\mathbb{S}^n_+$. $\mathbb{S}^n_+$ forms a convex cone in $\mathbb{R}^{n\times n}$, while the interior of this cone, $\operatorname{int}(\mathbb{S}^n_+)$, is the space of positive definite matrices. For ease of notation we write $A_1 \succeq A_2$ for $A_1 - A_2 \in \mathbb{S}^n_+$ and $A_1 \succ A_2$ for $A_1 - A_2 \in \operatorname{int}(\mathbb{S}^n_+)$.

Definition B.56 (Eigenvalues, eigenvectors) *The* eigenvalues λ_i, $i = 1, \dots, n$, *of a square matrix $A \in \mathbb{R}^{n\times n}$ are the solutions to the characteristic equation*

$$\det(A - \lambda I) = 0 \, .$$

The (right) eigenvector *v_i corresponding to the ith eigenvalue λ_i is a non-trivial solution to*

$$(A - \lambda_i I)v_i = 0\,.$$

Definition B.57 (Singular value decomposition) *Any matrix $A \in \mathbb{R}^{n_1 \times n_2}$ can be decomposed into*

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

with an orthogonal matrix $U \in \mathbb{R}^{n_1 \times n_1}$, an orthogonal matrix $V \in \mathbb{R}^{n_1 \times n_1}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{n_1 \times n_2}$ with nonnegative diagonal elements σ_i in descending order. This decomposition is called singular value decomposition *(SVD), and the diagonal elements σ_i are called the* singular values *of A.*

Definition B.58 (Definite matrix) *Let $A \in \mathbb{S}^{n_x}$ be a real symmetric square matrix and let $x \in \mathbb{R}^{n_x}$ denote a non-zero vector. A is called*

- positive definite *if $x^T A x > 0 \; \forall x \neq 0$,*
- positive semidefinite *if $x^T A x \geq 0 \; \forall x \neq 0$,*
- negative definite *if $x^T A x < 0 \; \forall x \neq 0$,*
- negative semidefinite *if $x^T A x \leq 0 \; \forall x \neq 0$,*
- indefinite *if $\exists x_1, x_2 \in \mathcal{D} : x_1^T A x_1 > 0 \wedge x_2^T A x_2 < 0$.*

The definiteness of a matrix is related to other properties of the matrix, e.g. its eigenvalues. The following proposition summarizes some properties of a positive semidefinite matrix. Similar propositions can be formulated for the other types of definiteness.

Proposition B.4 *Let $A \in \mathbb{S}^{n_x}$ be a real symmetric square matrix. The following statements are equivalent:*

(i) A is positive semidefinite.

(ii) $x^T A x \geq 0 \quad \forall x \in \mathbb{R}^{n_x}$.

(iii) All eigenvalues of A are nonnegative, $\lambda_i(A) \geq 0 \quad \forall i = 1, \ldots, n$.

(iv) $A = U^T U$ for some $U \in \mathbb{R}^{n \times n}$.

(v) $A = \sum_{i=1}^{n} \lambda_i a_i a_i^T$ for some $a_i \in \mathbb{R}^n$ and $\lambda_i \geq 0$.

Proposition B.5 (Schur complement formula) *Consider the square matrix $M \in \mathbb{R}^{n \times n}$ comprising the four blocks $A \in \mathbb{R}^{n_1 \times n_1}, B \in \mathbb{R}^{n_1 \times n_2}, C \in \mathbb{R}^{n_2 \times n_1}$ and $D \in \mathbb{R}^{n_2 \times n_2}$. The following statements are equivalent:*

1. $$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \prec 0$$

2. $D \prec 0 \quad \wedge \quad A - BD^{-1}C \prec 0$
3. $A \prec 0 \quad \wedge \quad D - CA^{-1}B \prec 0$

The term $A - BD^{-1}C$ is called the Schur complement *of block D of matrix M, and the term $D - CA^{-1}B$ is called the Schur complement of block A of matrix M.*

Bibliography

[AA98] P. Apkarian and R.J. Adams, *Advanced Gain-Scheduling Techniques for Uncertain Systems*, IEEE Transactions on Control Systems Technology **6** (1998), no. 1, 21 – 32.

[AB06] A. Alessio and A. Bemporad, *Feasible Mode Enumeration and Cost Comparison for Explicit Quadratic Model Predictive Control of Hybrid Systems*, Preprints of the 2nd Conf. on Analysis and Design of Hybrid Systems (Alghero, Italy), June 2006.

[ABGH96] P. Apkarian, G. Becker, P. Gahinet, and H.Kajiwara, *LMI Techniques in Control Engineering from Theory to Practice*, Workshop Notes from the IEEE Conference on Decision and Control (Kobe, Japan), December 1996.

[AC01] A. Alessandri and P. Colleta, *Design of Luenberger Observers for a Class of Hybrid Systems.*, Proceedings of the International Workshop on Hybrid Systems: Computation and Control (Rome, Italy), March 2001.

[Ack99] J. Ackermann, *Topics in Control and Its Applications*, ch. Robust Control for Car Steering, pp. 1 – 15, Springer Verlag, 1999.

[Ack01] J. Ackermann, *The Human Reaction Time - What Can Control Do to Overcome It?*, Keynote lecture, 25 Years of Nonlinear Control at Ecole des Mines de Paris, March 2001.

[AG95] P. Apkarian and P. Gahinet, *A Convex Characterization of Gain-Scheduled $\mathcal{H}_\infty$ Controller*, IEEE Transactions on Automatic Control **40** (1995), no. 5, 853 – 864.

[AGB95] P. Apkarian, P. Gahinet, and G. Becker, *Self-Scheduled $\mathcal{H}_\infty$ Control of Linear Parameter-Varying Systems: A Design Example*, Automatica **31** (1995), no. 9, 1251 – 1261.

[AGS$^+$95] J. Ackermann, J. Guldner, W. Sienel, R. Steinhauser, and V. Utkin, *A Robust Nonlinear Control Approach to Automatic Path Tracking of a Car*, IEEE Transactions on Control Systems Technology **3** (1995), no. 1, 132 – 143.

[Alt07] F. Althaus, *Identification and Control of a Mechanical System with Backlash*, Semester thesis, Automatic Control Laboratory, ETH Zurich, February 2007.

[AP97] J. Acevedo and E.N. Pistikopoulos, *A Multiparametric Programming Approach for Linear Process Engineering Problems under Uncertainty*, Industrial & Engineering Chemistry Research **36** (1997), no. 3, 717 – 728.

[ÅW95] K. J. Åström and B. Wittenmark, *Adaptive Control*, second ed., Addison-Wesley, 1995.

[AW09] H.S. Abbas and H. Werner, *An Instrumental Variable Technique for Open-Loop and Closed-Loop Identification of Input-Output LPV Models*, Proceedings of the European Control Conference (Budapest, Hungary), August 2009, pp. 2646 – 2651.

[Axe08] D. Axehill, *Integer Quadratic Programming for Control and Communication*, Ph.D. thesis, Linköpings Universitet, Linköping, Sweden, 2008.

[Bao02] M. Baotic, *An Efficient Algorithm for Multiparametric Quadratic Programming*, Tech. report, Automatic Control Laboratory, ETH Zurich, Switzerland, April 2002.

[Bao05] ______, *Optimal Control of Piecewise Affine Systems – A Multi-parametric Approach*, Ph.D. thesis, Automatic Control Laboratory, ETH Zurich, March 2005.

[Bar83] B.R. Barmish, *Stabilization of Uncertain Systems via Linear Control*, IEEE Transactions on Automatic Control **28** (1983), no. 8, 848 – 850.

[BBBM05] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari, *Dynamic Programming for Constrained Optimal Control of Discrete-Time Linear Hybrid Systems*, Automatica **41** (2005), no. 10, 1709 – 1721.

[BBM00] A. Bemporad, F. Borelli, and M. Morari, *Explicit Solution of LP-Based Model Predictive Control*, Proceedings of the IEEE Conference on Decision & Control (Sydney, Australia), December 2000.

[BBM02] A. Bemporad, F. Borrelli, and M. Morari, *Model Predictive Control Based on Linear Programming – The Explicit Solution*, IEEE Transactions on Automatic Control **47** (2002), no. 12, 1974 – 1985.

[BBM03] ______, *Min-Max Control of Constrained Uncertain Discrete-Time Linear Systems*, IEEE Transactions on Automatic Control **48** (2003), no. 9, 1600 – 1606.

[BBS03] G. Balas, J. Bokor, and Z. Szabó, *Invariant Subspaces for LPV Systems and Their Applications*, IEEE Transactions on Automatic Control **48** (2003), no. 11, 2065 – 2069.

[BDRK00] G.I. Bara, J. Daafouz, J. Ragot, and F. Kratz, *State Estimation for Affine LPV Systems*, Proceedings of the IEEE Conference on Decision & Control (Sydney, Australia), 2000, pp. 4565–4570.

[BEGFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, Pennsylvania, 1994.

[Bel57] R.E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[Bem98] A. Bemporad, *Reducing Conservativeness in Predictive Control of Constrained Systems with Disturbances*, Proceedings of the IEEE Conference on Decision & Control (Tampa, USA), December 1998, pp. 1384–1391.

[Bem04] A. Bemporad, *Efficient Conversion of Mixed Logical Dynamical Systems into an Equivalent Piecewise Affine Form*, IEEE Transactions on Automatic Control **49** (2004), no. 5, 832 – 838.

[Ber95] D. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 1995.

[Ber03] ______, *Convex Analysis and Optimization*, Athena Scientific, 2003.

[Bes07] G. Besonçon, *Nonlinear Observers and Applications*, Lecture Notes in Control and Information Sciences, Springer, 2007.

[BGK$^+$82] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer, *Non-Linear Parametric Optimization*, Akademie-Verlag, Berlin, 1982.

[BIP09] D. Bertsimas, D.A. Iancu, and P.A. Parillo, *Optimality of Affine Policies in Multi-stage Robust Optimization*, Proceedings of the IEEE Conference on Decision & Control (Shanghai, China), December 2009.

[BIRS98] J.R. Banga, R. Irizarry-Rivera, and W.D. Seider, *Stochastic Optimization for Optimal and Model-Predictive Control* , Computers & Chemical Engineering **22** (1998), no. 4 – 5, 603 – 612.

[Bla92] F. Blanchini, *Minimum-Time Control for Uncertain Discrete-Time Systems*, Proceedings of the IEEE Conference on Decision & Control (Tucson, USA), December 1992, pp. 2629 – 2634.

[Bla94] ______, *Ultimate Boundedness Control for Uncertain Discrete-Time Systems via Set-Induced Lyapunov Functions*, IEEE Transactions on Automatic Control **39** (1994), no. 2, 428 – 433.

[Bla95] F. Blanchini, *Nonquadratic Lyapunov Functions for Robust Control*, Automatica **31** (1995), no. 3, 451 – 461.

[BLM08] T. Besselmann, J. Löfberg, and M. Morari, *Explicit Model Predictive Control for Systems with Parameter-Varying State Transition Matrix*, IFAC World Congress (Seoul, Korea), July 2008.

[BLM09] ______, *Explicit LPV-MPC with Bounded Rate of Parameter Variation*, IFAC Symposium on Robust Control Design (Haifa, Israel), 2009.

[BM99] A. Bemporad and M. Morari, *Control of Systems Integrating Logics, Dynamics, and Constraints*, Automatica **35** (1999), no. 3, 407 – 427.

[BM03] F. Blanchini and S. Miani, *Stabilization of LPV systems: State Feedback, State Estimation and Duality*, Proceedings of the IEEE Conference on Decision & Control (Maui, USA), December 2003, pp. 1492 – 1497.

[BM08a] T. Besselmann and M. Morari, *Hybrid Parameter-Varying Model Predictive Control for Autonomous Vehicle Steering*, European Journal of Control **14** (2008), no. 5, 418 – 431.

[BM08b] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, Birkhäuser, Boston, 2008.

[BMC$^+$09] A.G. Beccuti, S. Mariéthoz, S. Cliquennois, S. Wang, and M. Morari, *Explicit Model Predictive Control of DC-DC Switched Mode Power Supplies with Extended Kalman Filtering*, IEEE Trans. on Industrial Electronics **56** (2009), no. 6, 1864 – 1874.

[BMDP02] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, *The Explicit Linear Quadratic Regulator for Constrained Systems*, Automatica **38** (2002), no. 1, 3–20.

[BMS07] F. Blanchini, S. Miani, and C. Savorgnan, *Stability Results for Linear Parameter Varying and Switching Systems*, Automatica **43** (2007), no. 10, 1817 – 1823.

[BNP87] E. Bakker, L. Nyborg, and H.B. Pacejka, *Tyre Modelling for Use in Vehicle Dynamics Studies*, paper 870421, Society of Automotive Engineers, 1987.

[BO97] A.M. Burke and O.A. Olatunbosun, *Static Tyre/Road Interaction Modelling*, Meccanica **32** (1997), 473 – 479.

[Bor03] F. Borrelli, *Constrained Optimal Control of Linear and Hybrid Systems*, Lecture Notes in Control and Information Sciences, vol. 290, Springer-Verlag, 2003.

[BRBM08] M. Barić, S. Raković, T. Besselmann, and M. Morari, *Max-Min Optimal Control of Constrained Discrete-Time Systems*, IFAC World Congress, July 2008, pp. 8803 – 8808.

[BTGGN04] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, *Adjustable Robust Solutions of Uncertain Linear Programs*, Mathematical Programming **99** (2004), no. 2, 351 – 376.

[BV04] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[CFF02] A. Casavola, D. Famularo, and G. Franzè, *A Feedback Min-Max MPC Algorithm for LPV Systems Subject to Bounded Rates of Change of Parameters*, IEEE Transactions on Automatic Control **47** (2002), no. 7, 1147 – 1153.

[CFZ03] L. Chisci, P. Falugi, and G. Zappa, *Gain-scheduling MPC for Nonlinear Systems*, International Journal of Robust and Nonlinear Control **13** (2003), no. 3, 295 – 308.

[CGKM07] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, *Move Blocking Strategies in Receding Horizon Control*, Journal of Process Control **17** (2007), no. 6, 563–570.

[Chu08] K. Chung, *Robustness of Hybrid MPC Controllers for a Mechanical System with Backlash*, Semester thesis, Automatic Control Laboratory, ETH Zurich, May 2008.

[CPL94] CPLEX Optimization Inc., Incline Village, USA, *Using the CPLEX Callable Library and the CPLEX Mixed Integer Library*, 1994.

[Cro97] P.R. Cromwell, *Polyhedra*, Cambridge University Press, Cambridge, UK, 1997.

[CW06] S.S. Chughtai and H. Werner, *Synthesis of Low Order Controllers for LPV Systems Using LMIs and Evolutionary Search*, Proceedings of the IEEE Conference on Decision & Control (San Diego, USA), December 2006, pp. 5784 – 5789.

[Dak65] R. J. Dakin, *A Tree Search Algorithm for Mixed Integer Programming Problems*, The Computer Journal **8** (1965), no. 3, 250 – 255.

[Dan98] G. Dantzig, *Linear Programming and Extensions*, 11th ed., Princeton Landmarks in Mathematics and Physics, Princeton University Press, New Jersey, USA, 1998.

[DBK00] J. Daafouz, G.I. Bara, and F. Kratz, *State Observers for Discrete-Time LPV Systems: An Interpolation Based Approach*, Proceedings of the IEEE Conference on Decision & Control (Sydney, Australia), December 2000, pp. 4571 – 4572.

[DBP02] V. Dua, N.A. Bozinis, and E.N. Pistikopoulos, *A Multiparametric Programming Approach for Mixed-Integer Quadratic Engineering Problems*, Computers & Chemical Engineering **26** (2002), no. 4-5, 715–733.

[Del34] B.N. Delaunay, *Sur la sphère vide*, Bulletin of Academy of Sciences of the USSR **7** (1934), no. 6, 793 – 800.

[Dev93] K. Devlin, *The Joy of Sets*, Springer Verlag, 1993.

[DP00] V. Dua and E.N. Pistikopoulos, *An Algorithm for the Solution of Multiparametric Mixed Integer Linear Programming Problems*, Annals of Operations Research **99** (2000), no. 1 – 4, 123 – 139.

[FBA$^+$07] P. Falcone, F. Borrelli, J. Asgari, H.E. Tseng, and D. Hrovat, *Predictive active steering control for autonomous vehicle systems*, IEEE Transactions on Control Systems Technology **15** (2007), no. 3, 566 – 580.

[FBT$^+$08] P. Falcone, F. Borrelli, H.E. Tseng, J. Asgari, and D. Hrovat, *Linear Time-Varying Model Predictive Control and Its Application to Active Steering Systems: Stability Analysis and Experimental Validation*, International Journal of Robust and Nonlinear Control **18** (2008), no. 8, 862 – 875.

[Flo95] C.A. Floudas, *Nonlinear and Mixed-Integer Optimization*, Oxford University Press, 1995.

[FMM02] G. Ferrari-Trecate, D. Mignone, and M. Morari, *Moving Horizon Estimation for Hybrid Systems*, IEEE Transactions on Automatic Control **47** (2002), no. 10, 1663 – 1676.

[FTB$^+$07] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H.E. Tseng, *A Linear Time Varying Model Predictive Control Approach to the Integrated Vehicle Dynamics Control Problem in Autonomous Systems*, Proceedings of the IEEE Conference on Decision & Control (New Orleans, USA), December 2007.

[FW04] A. Farag and H. Werner, *A Ricatti - Genetic Algorithms Approach of Fixed-Structure Controller Synthesis*, Proceedings of the American Control Conference (Boston, USA), vol. 3, July 2004, pp. 2799 – 2804.

[FW06] ______, *Fixed-structure mu-synthesis – an evolutionary approach*, Proceedings of the American Control Conference (Minneapolis, USA), June 2006.

[Gal95] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics*, 2nd ed., de Gruyter, Berlin, 1995.

[GG97] T. Gal and H.J. Greenberg (eds.), *Advances in Sensitivity Analysis and Parametric Programming*, International Series in Operations Research & Management Science, vol. 6, Kluwer Academic Publishers, 1997.

[GJV03] P. Giorgi, C.-P. Jeannerod, and G. Villard, *On the Complexity of Polynomial Matrix Computations*, Proceedings of the International Symposium on Symbolic and Algebraic Computation (Philadelphia, USA), August 2003, pp. 135 – 142.

[GM03] P. Grieder and M. Morari, *Complexity Reduction of Receding Horizon Control*, Proceedings of the IEEE Conference on Decision & Control (Maui, Hawaii, USA), December 2003, pp. 3179 – 3184.

[GMSW] P.E. Gill, W. Murray, M.A. Sanders, and M.H. Wright, *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming*, Stanford Business Software Inc.

[GN72] T. Gal and J. Nedoma, *Multiparametric Linear Programming*, Management Science **18** (1972), no. 7, 406 – 422.

[GPM89] C. Garcia, D. Prett, and M. Morari, *Model Predictive Control: Theory and Practice – a Survey*, Automatica **25** (1989), no. 3, 335 – 348.

[GPM09] T. Geyer, G. Papafotiou, and M. Morari, *Model Predictive Direct Torque Control - Part I: Concept, Algorithm and Analysis*, IEEE Trans. on Industrial Electronics **56** (2009), no. 6, 1894–1905.

[Grü03] B. Grünbaum, *Convex Polytopes*, 2nd ed., Springer, 2003.

[GS03] A.H. Glattfelder and W. Schaufelberger, *Control Systems with Input and Output Constraints*, Advanced Textbooks in Control and Signal Processing, Springer, 2003.

[GT01] N.I.M. Gould and P.L. Toint, *A Quadratic Programming Bibliography*, Tech. report, Numerical Analysis Group, Rutherford Appleton Laboratory, February 2001.

[HAF00] D. Hrovat, J. Asgari, and M. Fodor, *Mechatronics Systems, Techniques and Applications*, ch. Automotive Mechatronics Systems, pp. 1 – 98, Gordon and Breach Science Publishers, 2000.

[HdSB01] W.P.M.H. Heemels, B. de Schutter, and A. Bemporad, *Equivalence of Hybrid Dynamical Models*, Automatica **37** (2001), no. 7, 1085 – 1091.

[HLP52] G. Hardy, J.E. Littlewood, and G. Pólya, *Inequalities*, 2nd ed., Cambridge University Press, Cambridge, UK, 1952.

[Hén76] M. Hénon, *A Two-Dimensional Mapping with a Strange Attractor*, Communications in Mathematical Physics **50** (1976), no. 1, 69–77.

[IS01] T. Iwasaki and G. Shibata, *LPV System Analysis via Quadratic Separator for Uncertain Implicit System*, IEEE Transactions on Automatic Control **46** (2001), no. 8, 1195 – 1208.

[JHW07] A.L. Juloski, W.P.M.H. Heemels, and S. Weiland, *Observer Design for a Class of Piecewise Linear Systems*, International Journal of Robust and Nonlinear Control **17** (2007), no. 15, 1387 – 1404.

[JMK07] C.N. Jones, J.M. Maciejowski, and E.C. Kerrigan, *Lexicographic Perturbation for Multiparametric Linear Programming with Applications to Control*, Automatica **43** (2007), no. 10, 1808–1816.

[JR03] R. Johansson and A. Rantzer (eds.), *Nonlinear and Hybrid Systems in Automotive Control*, Springer, 2003.

[JV05] C.-P. Jeannerod and G. Villard, *Essentially Optimal Computation of the Inverse of Generic Polynomial Matrices*, Journal of Complexity **21** (2005), no. 1, 72 – 86.

[Kar39] W. Karush, *Minima of Functions of Several Variables with Inequalities as Side Conditions*, Master's thesis, Department of Mathematics, University of Chicago, 1939.

[Kar84] N. Karmarkar, *A new Polynomial-Time Algorithm for Linear Programming*, Combinatorica **4** (1984), no. 4, 375 – 395.

[KBM96] M. V. Kothare, V. Balakrishnan, and M. Morari, *Robust Constrained Model Predictive Control Using Linear Matrix Inequilities*, Automatica **32** (1996), no. 10, 1361–1379.

[KFB$^+$06] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat, *Predictive Control Approach to Autonomous Vehicle Steering*, Proceedings of the American Control Conference (Minneapolis, USA), June 2006.

[KG02] V. Kapila and K.M. Grigoriadis (eds.), *Actuator Saturation Control*, Control Engineering, CRC Press, 2002.

[KGBM04] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari, *Multi-Parametric Toolbox (MPT)*, Proceedings of the International Workshop on Hybrid Systems: Computation and Control (Philadelphia, USA), March 2004, pp. 448–462.

[Khaoo] H.K. Khalil, *Nonlinear Systems*, 3rd ed., Pearson Education International Inc., Michigan State University, USA, 2000.

[KM00] E.C. Kerrigan and J.M. Maciejowski, *Soft Constraints and Exact Penalty Functions in Model Predictive Control*, The UKACC International Conference on Control (Cambridge, UK), September 2000.

[KP03] M. Kon-Popovska, *On Some Aspects of the Matrix Data Perturbation in Linear Program*, Yugoslav Journal of Operations Research **13** (2003), no. 2, 153 – 164.

[KT51] H.W. Kuhn and A.W. Tucker, *Nonlinear Programming*, 2nd Berkeley Symposium (Berkeley, CA) (1951), 481 – 492.

[Kwi07] A. Kwiatkowski, *LPV Modelling and Application of LPV Controllers to SI Engines*, Ph.D. thesis, Institute of Control Systems, Hamburg University of Technology, December 2007.

[LA99] Y. Lu and Y. Arkun, *A Scheduling Quasi-Min-Max MPC for LPV Systems*, Proceedings of the American Control Conference (San Diego, USA), June 1999.

[LA00a] Y. Lu and Y. Arkan, *A Quasi-Min-Max MPC Algorithm for Linear Parameter Varying Systems with Bounded Rate of Change of Parameters*, Proceedings of the American Control Conference (Chicago, USA), June 2000, pp. 3234–3238.

[LA00b] Y. Lu and Y. Arkun, *Quasi-Min-Max MPC Algorithms for LPV Systems*, Automatica **36** (2000), no. 4, 527 – 540.

[Lag01] A. Lagerberg, *A Literature Survey on Control of Automotive Powertrains with Backlash*, Tech. Report R013/2001, Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, December 2001.

[Lag04] ______, *Control and Estimation of Automotive Powertrains with Backlash*, Ph.D. thesis, Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, 2004.

[LaS86] J.P. LaSalle, *The Stability and Control of Discrete Processes*, Applied Mathematical Sciences, vol. 62, Springer Verlag, 1986.

[LAW08] N. Lachhab, H.S. Abbas, and H. Werner, *A Neural-Network Based Technique for Modelling and LPV Control of an Arm-Driven Inverted Pendulum*, Proceedings of the IEEE Conference on Decision & Control (Cancun, Mexico), December 2008, pp. 3860 – 3865.

[Laz06] M. Lazar, *Model Predictive Control of Hybrid Systems: Stability and Robustness*, Ph.D. thesis, Technische Universiteit Eindhoven, Eindhoven, the Netherlands, 2006.

[LD60] A.H. Land and A.G. Doig, *An Automatic Method of Solving Discrete Programming Problems*, Econometrica **28** (1960), no. 3, 497 – 520.

[LE03a] A. Lagerberg and B. Egardt, *Backlash Gap Position Estimation in Automotive Powertrains*, Proceedings of the European Control Conference (Cambridge, UK), September 2003.

[LE03b] ______, *Estimation of Backlash with Application to Automotive Powertrains*, Proceedings of the IEEE Conference on Decision & Control (Maui, USA), December 2003.

[LE05] ______, *Model Predictive Control of Automotive Powertrains with Backlash*, IFAC World Congress (Prague, Czech Republic), July 2005.

[Lju06] L. Ljung, *System Identification Toolbox, User's Guide, Version 6*, The MathWorks, Inc., 2006.

[Löf03] J. Löfberg, *Minimax Approaches to Robust Model Predictive Control*, Linköping studies in science and technology. thesis no 812, Linköping universitet, April 2003.

[Löf04] J. Löfberg, *YALMIP : A Toolbox for Modelling and Optimization in MATLAB*, CCA/ISIC/CACSD (Taipei, Taiwan), September 2004.

[Löf08] ______, *Modelling and Solving Uncertain Optimization Problems in YALMIP*, IFAC World Congress (Seoul, Korea), July 2008, pp. 1337 – 1341.

[Los07] T. Loser, *Hybrid Estimation for a Mechanical System with Backlash*, Semester thesis, Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland, December 2007.

[LW06] S. Lee and S. Won, *Model Predictive Control for Linear Parameter Varying Systems Using a New Parameter Dependent Terminal Weighting Matrix*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E89-A** (2006), no. 8, 2166 – 2172.

[MA91] D.L. Margolis and J. Asgari, *Multipurpose Models of Vehicle Dynamics for Controller Design*, SAE Technical Papers (1991).

[Mac01] J. M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, June 2001.

[MBB03] M. Morari, M. Baotic, and F. Borrelli, *Hybrid Systems Modelling and Control*, European Journal of Control **9** (2003), no. 2-3, 177–189.

[MBM09] U. Maeder, F. Borrelli, and M. Morari, *Linear Offset-Free Model Predictive Control*, Automatica **45** (2009), no. 10, 2214 – 2222.

[ML99] M. Morari and J.H. Lee, *Model Predictive Control: Past, Present and Future*, Computers & Chemical Engineering **23** (1999), no. 4, 667 – 682.

[MRRS00] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, *Constrained Model Predictive Control: Stability and Optimality*, Automatica **36** (2000), no. 6, 789 – 814.

[Mur83] K.G. Murty, *Linear Programming*, John Wiley & Sons, 1983.

[NG02] N. Nordin and P.-O. Gutman, *Controlling Mechanical Systems with Backlash - A Survey*, Automatica **38** (2002), no. 10, 1633–1649.

[NGG97] M. Nordin, J. Galic, and P.-O. Gutman, *New Models for Backlash and Gear Play*, International Journal of Adaptive Control and Signal Processing **11** (1997), no. 1, 49–63.

[NN94] Y.E. Nesterov and A.S. Nemirovski, *Interior Point Polynomial Algorithms in Convex Programming*, SIAM Publications, 1994.

[NP97] V. Nevistic and J.A Primbs, *Constrained Finite Receding Horizon Linear Quadratic Control*, Proceedings of the IEEE Conference on Decision & Control (San Diego, USA), December 1997, pp. 3196–3201.

[Num02] Numerical Algorithms Group Ltd., Oxford, UK, *NAG Foundation Toolbox for Matlab 6*, 2002, http://nag.co.uk/.

[NW06] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed., Springer Verlag, Berlin, New York, 2006.

[OH55] W. Orchard-Hays, *Notes on Linear Programming (Part 6): The Rand Code for the Simplex Method (SX4)*, Technical report 1440, Rand Corporation, 1955.

[PDB$^+$00] E.N. Pistikopoulos, V. Dua, N.A Bozinis, A. Bemporad, and M. Morari, *On-line Optimization via Off-line Parametric Optimization Tools*, Proceedings of the International Symposium on Symbolic and Algebraic Computation (Keystone, USA), July 2000, pp. 183–188.

[PJ04] P. Park and S.C. Jeong, *Constrained RHC for LPV Systems with Bounded Rate of Parameter Variations*, Automatica **40** (2004), no. 5, 865 – 872.

[Pól28] G. Pólya, *Über positive Darstellung von Polynomen*, Vierteljahrschrift d. naturforschenden Gesellschaft in Zürich **73** (1928), 141 – 145, reprinted in: Collected Papers, Volume 2, 309 – 313, Cambridge: MIT Press, 1974.

[PL09] D. Petersson and J. Löfberg, *Optimization Based LPV-Approximation of Multi-Model Systems*, Proceedings of the European Control Conference, August 2009, pp. 3172–3177.

[PR01] V. Powers and B. Reznick, *A New Bound for Pólya's Theorem with Applications to Polynomial Positive on Polyhedra*, Journal of Pure and Applied Algebra **164** (2001), no. 1 – 2, 221 – 229.

[PR06] ———, *A Quantitative Pólya's Theorem with Corner Zeros*, Proceedings of the International Symposium on Symbolic and Algebraic Computation (New York) (J.-G. Dumas, ed.), ACM Press, 2006, pp. 285 – 290.

[PRSDM05] B. Pluymers, J.A. Rossiter, J.A.K. Suykens, and B. De Moor, *Interpolation Based MPC for LPV Systems Using Polyhedral Invariant Sets*, Proceedings of the American Control Conference (Portland, OR, USA), June 2005.

[PSSU80] P.C. Parks, W. Schaufelberger, C. Schmid, and H. Unbehauen, *Methods and Applications in Adaptive Control*, vol. 24/1980, ch. Applications of Adaptive Control Systems, pp. 161 – 198, Springer, 1980.

[PV91] P.M. Pardalos and S.A. Vavasis, *Quadratic Programming with One Negative Eigenvalue is NP-Hard*, Journal of Global Optimization **1** (1991), no. 1, 15 – 22.

[QB03] S.J. Qin and T.A. Badgwell, *A Survey of Industrial Model Predictive Control Technology*, Control Engineering Practice **11** (2003), no. 7, 733 – 764.

[Raj06] R. Rajamani, *Vehicle Dynamics and Control*, Springer, 2006.

[Ra000] C. Rao, *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*, Ph.D. thesis, University of Wisconsin-Madison, 2000.

[RBB$^+$07] P. Rostalski, T. Besselmann, M. Baric, F. van Belzen, and M. Morari, *A Hybrid Approach to Modelling, Control and State Estimation of Mechanical Systems with Backlash*, International Journal of Control **80** (2007), no. 11, 1729 – 1740.

[RM09] J.B. Rawlings and D.Q. Mayne, *Model Predictive Control: Theory and Design*, Nob Hill Publishing, 2009.

[Roc70] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[Ros09] P. Rostalski, *Algebraic Moments - Real Root Finding and Related Topics*, Ph.D. thesis, Automatic Control Laboratory, ETH Zurich, 2009.

[SA90] J.S. Shamma and M. Athans, *Analysis of Gain Scheduled Control for Nonlinear Plants*, IEEE Transactions on Automatic Control **35** (1990), no. 8, 898 – 907.

[SA91] ______, *Guaranteed Properties of Gain Scheduled Control for Linear Parameter-Varying Plants*, Automatica **27** (1991), no. 3, 559 – 564.

[Sah00] N.V. Sahinidis, *BARON - Branch And Reduce Optimization Navigator*, University of Illinois at Urbana-Champaign, Urbana, USA, 4.0 ed., June 2000.

[Sch86] A. Schrijver, *Theory of Linear and Integer Programming*, John Wileys & Sons, 1986.

[Sch96] C.W. Scherer, *Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ Control for Time-Varying and Linear Parametrically-Varying Systems.*, International Journal of Robust and Nonlinear Control **6** (1996), no. 9 – 10, 929 – 952.

[Sch01] ______, *LPV Control and Full Block Multipliers*, Automatica **37** (2001), no. 3, 361 – 375.

[SG54] T. Saaty and S. Gass, *Parametric Objective Function (Part 1)*, Journal of the Operations Research Society of America **2** (1954), no. 3, 316 – 319.

[SH04] C.W. Scherer and C.W.J. Hol, *Asymptotically Exact Relaxations for Robust LMI Problems based on Matrix-Valued Sum-of-Squares*, Proceedings of the Mathematical Theory of Networks and Systems (Leuven, Belgium), July 2004.

[SL91] J. Slotine and W. Li, *Applied Nonlinear Control*, Prentice Hall, New Jersey, USA, 1991.

[SM67] L.M. Silverman and H.E. Meadows, *Controllability and Observability in Time-Variable Linear Systems*, SIAM Journal on Control **5** (1967), no. 1, 64 – 73.

[SMR99] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings, *Suboptimal Model Predictive Control (Feasibility Implies Stability)*, IEEE Transactions on Automatic Control **44** (1999), no. 3, 648 – 654.

[Son81] E.D. Sontag, *Nonlinear Regulation: The Piecewise Linear Approach*, IEEE Transactions on Automatic Control **26** (1981), no. 2, 346 – 358.

[SP05] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed., John Wileys & Sons Ltd., 2005.

[SS06] H. Suzuki and T. Sugie, *MPC for LPV Systems with Bounded Parameter Variation Using Ellipsoidal Set Prediction*, Proceedings of the American Control Conference (Minneapolis, USA), June 2006, pp. 5251 – 5256.

[Str86] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, 1986.

[Str91] ______, *Calculus*, Wellesley-Cambridge Press, 1991.

[Str93] ______, *Introduction to Linear Algebra*, Wellesley-Cambridge Press, 1993.

[STS04] S. Suryanarayanan, M. Tomizuka, and T. Suzuki, *Design of Simultaneously Stabilizing Controllers and Its Application to Fault-Tolerant Lane-Keeping Controller Design for Automated Vehicles*, IEEE Transactions on Control Systems Technology **12** (2004), no. 3, 329 – 339.

[Stu99] J. Sturm, *Using SeDuMi 1.02, a Matlab Toolbox for Optimization over Symmetric Cones*, Optimization Methods and Software **11–12** (1999), 625–653.

[TBB$^+$02] F.D. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and D. Mignone, *HYSDEL - User Manual*, Tech. report, Automatic Control Lab., ETH Zurich, August 2002.

[THVdH07] R. Tóth, P.S.C. Heuberger, and P.M.J. Van den Hof, *(In)equivalence of Discrete Time LPV State-Space and Input/Output Representations*, Proceedings of the Benelux Meeting on Systems and Control (Lommel, Belgium), March 2007, p. 34.

[TJB01] P. Tøndel, T.A Johansen, and A. Bemporad, *An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions*, Proceedings of the IEEE Conference on Decision & Control (Orlando, USA), December 2001, pp. 1199 – 1204.

[TJB03] P. Tøndel, T.A. Johansen, and A. Bemporad, *Evaluation of Piecewise Affine Control via Binary Search Tree*, Automatica **39** (2003), no. 5, 945 – 950.

[Tóto8] R. Tóth, *Modelling and Identification of Linear Parameter-Varying Systems - An Orthonormal Basis Function Approach*, Ph.D. thesis, Delft Center for Systems and Control, Technische Universiteit Delft, Delft, the Netherlands, 2008.

[TTT99] K.C. Toh, M.J. Todd, and R.H. Tütüncü, *SDPT3 — A Matlab Software Package for Semidefinite Programming*, Optimization Methods and Software **11** (1999), 545–581.

[Vin97] T.L. Vincent, *Control Using Chaos*, IEEE Control Systems Magazine **17** (1997), no. 6, 65 – 76.

[VRS87] O.V. Volkovich, V.A. Roshchin, and I.V. Sergienko, *Models and Methods of Solution of Quadratic Integer Programming Problems*, Cybernetics **23** (1987), no. 2, 289 – 305.

[WGP00] F. Wu, K.M. Grigoriadis, and A. Packard, *Anti-Windup Controller Design Using Linear Parameter-Varying Control Methods*, International Journal of Control **73** (2000), no. 12, 1104 – 1114.

[Wit68] H.S. Witsenhausen, *A Minimax Control Problem for Sampled Linear Systems*, IEEE Transactions on Automatic Control **13** (1968), no. 1, 5 – 21.

[WSV00] H. Wolkowicz, R. Saigal, and L. Vanderberghe (eds.), *Handbook of Semidefinite Programming – Theory, Algorithms, and Applications*, International Series in Operations Research and Management Science, vol. 27, Kluwer, February 2000.

[YF05] L. Yepremyan and J.E. Falk, *Delaunay Partitions in $\mathbb{R}^n$ Applied to Non-Convex Programs and Vertex/Facet Enumeration Problems.*, Computational Optimization and Applications (2005), no. 32, 793 – 812.

[ZD98] K. Zhou and J.C. Doyle, *Essentials of Robust Control*, Prentice Hall, 1998.

[ZDG96] K. Zhou, J.C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice Hall, New Jersey, USA, 1996.

[Zie95] G.M. Ziegler, *Lectures on Polytopes*, 1st ed., Springer, 1995.

List of Publications

The following list includes all publications written during the author's doctorate at the Automatic Control Laboratory, ETH Zurich.

Constrained Time-Optimal Control of Linear Parameter-Varying Systems. T. Besselmann, J. Löfberg and M. Morari. IEEE Conference on Decision and Control, Shanghai, China, 2009.

Autonomous Vehicle Steering Using Explicit LPV-MPC. T. Besselmann and M. Morari. European Control Conference, Budapest, Hungary, 2009.

Explicit LPV-MPC with Bounded Rate of Parameter Variation. T. Besselmann, J. Löfberg and M. Morari. IFAC Symposium on Robust Control Design, Haifa, Israel, 2009.

Explicit MPC for LPV Systems. T. Besselmann, J. Löfberg and M. Morari. IEEE Conference on Decision and Control, Cancun, Mexico, 2008.

Hybrid Parameter-Varying MPC for Autonomous Vehicle Steering. T. Besselmann and M. Morari. European Journal of Control, vol. 14, no. 5, pp. 418 - 431, 2008.

Explicit MPC for Systems with Linear Parameter-Varying State Transition Matrix. T. Besselmann, J. Löfberg and M. Morari. IFAC World Congress, Seoul, Korea, 2008.

Max-Min Optimal Control of Constrained Discrete-Time Systems. M. Barić, S.V. Rakovic, T. Besselmann and M. Morari. IFAC World Congress, Seoul, Korea, 2008.

Removal of Regions for Quadratic Norm Optimal Control of Piecewise Affine Systems. T. Besselmann and M. Morari. Technical Report, Automatic Control Laboratory, ETH Zurich, vol. AUT07-12, 2007.

A Hybrid Approach to Modeling, Control and State Estimation of Mechanical Systems with Backlash. P. Rostalski, T. Besselmann, M. Barić, F. von Belzen and M. Morari. International Journal of Control, vol. 80, no. 11, pp. 1729-1740, Special Issue on "Automotive Systems and Control", 2007.

Hybrid Parameter-Varying Model Predictive Control for Lateral Vehicle Stabilization. T. Besselmann, P. Rostalski and M. Morari. European Control Conference, Kos, Greece, 2007.

Explicit MPC for Systems with Linear Parameter-Varying State Transition Matrix. T. Besselmann, J. Löfberg and M. Morari. Technical Report, Automatic Control Laboratory, ETH Zurich, vol. AUT07-14, 2007.

Curriculum Vitae

Thomas Besselmann
born on the 6th of May 1980, in Thuine, Germany.

2006 – 2010	Doctorate at the Automatic Control Laboratory, ETH Zurich, Switzerland (Dr. sc. ETH Zurich)
2003 – 2005	Diploma studies in Mechatronics, Hamburg University of Technology, Germany (Dipl.-Ing.)
2005	Internship at Robert Bosch GmbH, Schwieberdingen, Germany
2004	Exchange semester at the Control Systems Centre, University of Manchester, UK
2000 – 2003	Undergraduate studies in General Engineering Science, Hamburg University of Technology, Germany (B.Sc.)
1993 – 1999	Franziskus Gymnasium Lingen, Germany (Abitur, High school)