Diss. ETH No. 15785

Efficient Computation of Feedback Controllers for Constrained Systems

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY (ETH)
ZURICH

for the degree of
Doctor of Sciences

presented by

Pascal Grieder
Dipl. El.-Ing. ETH Zürich
born 28.2.1977
citizen of Basel, BS

accepted on the recommendation of

Prof. Dr. Manfred Morari, examiner
Prof. Dr. Komei Fukuda, co-examiner
Dr. Jan Maciejowski, co-examiner

2004

Acknowledgement

This section will undoubtedly be the most rigorously read of this thesis and hence I'll try to give proper credit to some of the people who helped me in getting to this point.

The most perfect working environment is worthless if you cannot retreat to a fulfilled private life. Since this aspect is of paramount importance to myself, I would first like to acknowledge the people who supported me outside of the office hours. I'd like to thank my girlfriend Claudia for always being there for me and making home a safe-haven from all types of stress. In addition, I am lucky enough to have excellent friends that I can truly rely on, whereby I'd like to especially mention Joni, Aline and Robin. I'd also like to thank 'Ibne'-Gültekin for being a good friend in and out of the office and I forgive him for never cleaning up his dirty plates. And last but not least, I'd like to thank my family (Mum, Dad and Phil) for backing me up all the way.

A special thank you goes to my supervisor Manfred Morari, who is both an outstanding supervisor and a great guy. He has always supported me to the fullest in all endeavors I have undertaken here at ETH and for that I am eternally grateful.

When I started my studies at the Automatic Control Lab, I was ignorant and had virtually no knowledge of even the most basic optimal control concepts. I'd here like to thank all those people who helped to remedy this situation to some extent (... I do not consider myself cured). I cannot thank my former office mates Fabio D. Torrisi and Francesco Borrelli enough for the time they spent in teaching me what I needed to know during my first months. If it weren't for them, it would have taken me significantly longer to get any type of research result. Special thanks also go to Eric Kerrigan for never being too busy to explain things, for giving me feedback on my papers and many useful discussions. I'd also like to express my gratitude to Mato Baotić, Sasa Raković, Johan Löfberg and Colin Jones whose insights have helped me a great deal and taught me a lot during these past years.

Special thanks to Michal Kvasnica for his relentless effort in making the MPT toolbox a piece of software to be proud of and the long work-hours he put into it. Without his enthusiasm, the toolbox would still be nothing more than a loose collection of functions. I'd also like to acknowledge Frank 'LaTeX' Christophersen for his invaluable feedback on just about anything I worked on. I am also indebted to Miroslav Barić, Pratik Biswas, Raphael Cagienard, Mayuresh Kothare, Marco Lüthi, Sasikanth Manipatruni, Pablo Parrilo, Anthony Rossiter, Raphael Suard, Kari Unneland and Zhaoyang Wan who worked with me on various projects and are responsible for some of the content of this thesis. I was lucky to be able to work on my Ph.D. in a great research and social environment and would like to especially mention Dominik, Eleonora, Gabi, George, Helfried, Kristian, Marc, Robert, Tobias (both of them) and Valentina for making the Automatic Control Lab a fun place to work at. Finally, I'd like to thank Andrea Gentilini, Alexandre Bayen and Claire Tomlin for showing me that control engineering is more fun than it sounds.

Abstract

One of the important problems in control theory is the computation of stabilizing controllers for dynamical systems subject to constraints on states and inputs. *Receding Horizon Control* (RHC), provides a very powerful framework to deal with this type of problem and has thus received great interest by both industry and academia. In RHC an objective function is specified and the input sequence which minimizes the objective and enforces constraint satisfaction over a finite prediction horizon is computed for the current state. Subsequently, only the first input of that sequence is applied to the system. At the next time step, the state is measured again and the procedure is repeated. One of the key problems with RHC is the inherent computational complexity of the optimization problem at hand which generally restricts the application of RHC to relatively slow processes.

This problem has been alleviated to some degree by the recent introduction of *multi-parametric programming* to control theory. Thereby, the RHC optimization problem is solved off-line for all possible initial states. In multi-parametric programming the analytical solution to this infinite-dimensional problem is obtained by solving a finite dimensional optimization problem. The solution then takes the form of a piecewise affine state feedback law which can be easily implemented on-line by the use of lookup tables. This scheme greatly decreases the cost of applying RHC to industrial systems and makes the on-line computation of the optimal input sequence significantly simpler. However, multi-parametric programming also suffers from a serious drawback: the size of the lookup table may grow exponentially with system size and complexity of the control objective. Therefore the application of multi-parametric programming in practice is restricted to cases where simple mathematical models of physical systems are available. The aim of this thesis is to mitigate this drawback.

When applying multi-parametric programming to control problems, three aspects,

which influence the overall complexity can be identified: the speed at which the lookup table is computed, the number of entries in the lookup table and the time which is required to find the correct entry in the lookup table. All three levers for complexity reduction are addressed in this thesis and the contributions herein enable the reduction of the overall complexity by orders of magnitude.

Specifically, various schemes to speed up the explicit controller computations are introduced. A combination of dynamic programming, infinite-time optimal control and efficient polytope reduction techniques yields controller computation algorithms which are significantly faster than prior schemes.

Novel methods to analyze PWA systems with a focus on stability and set invariance are presented. Special attention is placed on PWA systems which are subject to bounded additive disturbances. These analysis schemes are subsequently used in various complexity reduction schemes. The proposed methods yield controllers of very low complexity by imposing 'simple' control objectives.

In addition, various post-processing schemes are introduced to simplify the feedback controllers a posteriori. The proposed schemes reduce the necessary storage space and are able to significantly reduce the time which is required to perform the set membership test online.

Finally the MPT toolbox is presented. The MPT toolbox for MATLAB contains all of the algorithms presented in this thesis as well as a wide range of additional algorithms and tools developed by the academic community.

Zusammenfassung

Eines der wichtigsten Probleme der Regelungstechnik ist die Berechnung stabilisierender Regler für dynamische Systeme, dessen Zustände und Eingänge nur eine begrenzte Wertemenge annehmen dürfen. *Receding Horizon Control* (RHC) ist eine sehr mächtige Methode, um Probleme dieser Art zu lösen und hat folglich ein weitreichendes Interesse in Forschung und Industrie gefunden. Bei RHC wird eine Zielfunktion festgelegt und die Eingangssequenz, welche diese Funktion minimiert, wird für den momentanen Zustandswert berechnet. Danach wird der erste Wert der Eingangssequenz via Aktuator dem System auferlegt. Zum nächsten Zeitpunkt wird der Zustandswert neu eruiert, und die Prozedur wird wiederholt. Eines der Schlüsselprobleme mit RHC ist die inhärente Komplexität der notwendigen Berechnungen. Dies limitiert die Anwendungsmöglichkeit von RHC auf relativ langsam ablaufende Prozesse.

Dieser Nachteil wurde zu einem gewissen Grad mit der Einführung von *multi-parametric programming* in die Regelungstechnik wettgemacht. Es wurde ermöglicht, RHC Optimierungsprobleme für alle Initialzustände off-line zu lösen. In multi-parametric programming wird die unendlich dimensionale Lösung dieses Problems berechnet, indem ein endlich dimensionales Optimierungsproblem gelöst wird. Die Lösung zu diesem Problem ist eine abschnittsweise affine Zustandsrückführung, welche mit Hilfe eines 'Look-Up Tables' implementiert werden kann. Dieser Lösungsansatz macht den Einsatz von optimaler Regelung billiger und effizienter, als dies mittels on-line Optimierung der Fall wäre. Die Methode hat jedoch auch einen signifikanten Nachteil: die Grösse des Look-Up Tables wächst exponentiell mit der Komplexität des Regelproblems. Deswegen können multi-parametric programming Methoden nur angewendet werden, wenn ein relativ einfaches mathematisches Modell des Systems vorliegt. Das Ziel dieser Dissertation ist es, diesen Nachteil so weit wie möglich zu eliminieren.

Wenn multi-parametric programming im Rahmen der Regelungstechnik angewendet wird, gibt es drei Aspekte welche die Komplexität beeinflussen: die Geschwindigkeit mit der das Look-Up Table berechnet werden kann, die Anzahl der Einträge im Look-Up Table und die Zeit die nötig ist, um den korrekten Eintrag im Look-Up Table zu finden.

Alle drei Aspekte werden in dieser Dissertation behandelt, so dass dessen Kombination die Problemkomplexität um mehrere Grössenordnungen reduziert.

Konkret werden mehrere Methoden vorgestellt, um die Berechnung der Zustandsrückführungsregler zu beschleunigen. Im Vergleich zu bisherigen Methoden beschleunigt eine Kombination von *dynamic programming*, *infinite-time optimal control* und effizienter Umgang mit Polytopen die Reglerberechnung erheblich.

Neue Methoden um abschnittsweise affine Systeme zu analysieren werden vorgestellt. Der Fokus liegt hierbei auf Stabilitätsanalyse und Invarianz von Zustandsmengen. Die Analysemethoden werden benutzt um Regler von niedriger Komplexität zu berechnen. Die vorgeschlagenen Methoden liefern einfache Regler durch einfache Zielsetzungen.

Des weiteren werden zwei Methoden vorgestellt, um die Reglerkomplexität a posteriori zu verringern. Die beiden Methoden reduzieren den nötigen Speicherplatz und beschleunigen die Zeit, in der die optimale Zustandsrückführung identifiziert werden kann.

Am Schluss wird die MPT Toolbox präsentiert. Diese Toolbox für MATLAB beinhaltet sämtliche Algorithmen, welche in dieser Dissertation vorgestellt wurden sowie etliche Standardfunktionen der Forschungsgebiete Regelungstechnik und Computational Geometry.

Contents

Notation

Logic Operators and Functions

$A \Rightarrow B$	A implies B, i.e. if A *true* then B *true*
$A \Leftrightarrow B$	A implies B and B implies A, i.e. A *true* if and only if B *true*

Sets

$\mathbb{R}$ ($\mathbb{R}_+$)	Set of (non-negative) real numbers
$\mathbb{N}$	Set of non-negative integers
$\mathbb{R}^n$	Set of real vectors with n elements
$\mathbb{R}^{n \times m}$	Set of real matrices with n rows and m columns

Algebraic Operators

A^T	Transpose of matrix A
A^{-1}	Inverse of matrix A
$\det(A)$	Determinant of matrix A
$A(\succeq) \succ 0$	A positive (semi)definite matrix, $x^T Ax(\geq) > 0,\ \forall x \neq 0$
$A(\preceq) \prec 0$	A negative (semi)definite matrix, $x^T Ax(\leq) < 0,\ \forall x \neq 0$.
$A_{(i)}$	i-th row of matrix A
$x_{(i)}$	i-th element of the vector x
$\|x\|$	Element wise absolute value
$\|\|x\|\|$	Any vector norm of x
$\|\|x\|\|_2$	Euclidian norm of vector x
$\|\|x\|\|_1$	Sum of absolute elements of vector $x \in \mathbb{R}^n$, $\|\|x\|\|_1 := \sum_{i=1}^n \|x_{(i)}\|$
$\|\|x\|\|_\infty$	Largest absolute value of the vector $x \in \mathbb{R}^n$, $\|\|x\|\|_\infty := \max_{i \in \{1,\ldots,n\}} \|x_{(i)}\|$
$\|\|x\|\|_p$	p-norm of a vector $x \in \mathbb{R}^n$, $\|\|x\|\|_p := \sqrt[p]{\sum_{i=1}^n \|x_{(i)}\|^p}$

Set Operators and Functions

$\emptyset$	The empty set
$\mathcal{P} \cap \mathcal{Q}$	Set intersection $\mathcal{P} \cap \mathcal{Q} = \{x \mid x \in \mathcal{P} \text{ and } x \in \mathcal{Q}\}$
$\mathcal{P} \cup \mathcal{Q}$	Set union $\mathcal{P} \cup \mathcal{Q} = \{x \mid x \in \mathcal{P} \text{ or } x \in \mathcal{Q}\}$
$\bigcup_{r \in \{1,\dots,R\}} \mathcal{P}_r$	Union of R sets $\mathcal{P}_r$, i.e. $\bigcup_{r \in \{1,\dots,R\}} \mathcal{P}_r = \{x \mid x \in \mathcal{P}_0 \text{ or } \dots \text{ or } x \in \mathcal{P}_R\}$
$\mathcal{P}^c$	Complement of the set $\mathcal{P}$, $\mathcal{P}^c = \{x \mid x \notin \mathcal{P}\}$
$\mathcal{P} \setminus \mathcal{Q}$	Set difference $\mathcal{P} \setminus \mathcal{Q} = \{x \mid x \in \mathcal{P} \text{ and } x \notin \mathcal{Q}\}$
$\mathcal{P} \subseteq \mathcal{Q}$	The set $\mathcal{P}$ is a subset of $\mathcal{Q}$, $x \in \mathcal{P} \Rightarrow x \in \mathcal{Q}$
$\mathcal{P} \subset \mathcal{Q}$	The set $\mathcal{P}$ is a strict subset of $\mathcal{Q}$, $x \in \mathcal{P} \Rightarrow x \in \mathcal{Q}$ and $\exists x \in (\mathcal{Q} \setminus \mathcal{P})$
$\mathcal{P} \supseteq \mathcal{Q}$	The set $\mathcal{P}$ is a superset of $\mathcal{Q}$
$\mathcal{P} \supset \mathcal{Q}$	The set $\mathcal{P}$ is a strict superset of $\mathcal{Q}$
$\mathcal{P} \ominus \mathcal{Q}$	Pontryagin difference $\mathcal{P} \ominus \mathcal{Q} = \{x \mid x + q \in \mathcal{P},\ \forall q \in \mathcal{Q}\}$
$\mathcal{P} \oplus \mathcal{Q}$	Minkowski sum $\mathcal{P} \oplus \mathcal{Q} = \{x + q \mid x \in \mathcal{P},\ q \in \mathcal{Q}\}$
$\partial\mathcal{P}$	The boundary of $\mathcal{P}$
$\text{int}(\mathcal{P})$	The interior of $\mathcal{P}$, i.e. $\text{int}(\mathcal{P}) = \mathcal{P} \setminus \partial\mathcal{P}$

Dynamical Systems

$x(k)$	Measurement of state x at time k
x_k	Predicted value of state x at time k, given a measurement $x(0)$
x^+	Successor of vector x, i.e. if $x = x(k)$ then $x^+ = x(k+1)$
$\mathcal{F}_\infty$	Minimal robust positive invariant set
$\mathcal{O}_\infty$	Maximal robust positive invariant set
$\mathcal{C}_\infty$	Maximal robust control invariant set
$\mathcal{K}_\infty$	Maximal robust stabilizable set
$\mathcal{O}_\infty^{\text{LQR}}$	Maximal positive invariant set $\mathcal{O}_\infty$ for LTI systems subject to the Riccati LQR controller

Others

$\mathbf{I}$	Identity matrix
$\mathbf{1}$	Vector of ones, $\mathbf{1} = [1\ 1\ \dots\ 1]^T$
$\mathbf{0}$	Vector of zeros, $\mathbf{0} = [0\ 0\ \dots\ 0]^T$

Acronyms

ARE	Algebraic Riccati Equation
BMI	Bilinear Matrix Inequality
CFTOC	Constrained Finite Time Optimal Control
CITOC	Constrained Infinite Time Optimal Control
DP	Dynamic Program(ming)
LMI	Linear Matrix Inequality
LP	Linear Program(ming)
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
MILP	Mixed Integer Linear Program
MIQP	Mixed Integer Quadratic Program
MPC	Model Predictive Control
mp-LP	multi-parametric Linear Program
mp-QP	multi-parametric Quadratic Program
PWA	Piecewise Affine
PWP	Piecewise Polynomial
PWQ	Piecewise Quadratic
QP	Quadratic Program(ming)
RHC	Receding Horizon Control
SDP	Semi Definite Program(ming)

Seite Leer /
Blank leaf

Introduction

Outline

The focus of this thesis is on Receding Horizon Control (RHC) and Model Predictive Control (MPC) of discrete-time linear time invariant (LTI) and piecewise-affine (PWA) systems. PWA systems represent a powerful modelling tool to capture nonlinear and hybrid behavior of dynamical systems and have therefore received great interest in academia and industry. It is well known, that optimal state feedback controllers for these types of systems can be computed by applying multi-parametric programming techniques. The resulting controller then takes the form of a feedback law which is affine over polyhedral sets, such that the optimal input becomes a piecewise affine function of the current state. The necessary on-line effort thus reduces to identifying which polyhedral set contains the current state and evaluating the associated affine feedback law. The advantage of this scheme is that no time consuming on-line optimization is necessary and the control input can be computed with low hardware cost and small computation time. However, there is a drawback: the number of control laws grows exponentially with the size of the control problem and may quickly reach a prohibitive number of elements.

In this thesis, these complexity issues are investigated and numerous methods for reducing complexity are presented. Specifically, three levers which influence the overall complexity are investigated:

1. computation of control laws,
2. the formulation of alternate control problems which yield solutions of tractable complexity and
3. post-processing of the feedback solution such that on-line implementation of the control law can be performed more efficiently.

Improvements resulting from the application of all three levers are presented in this thesis, both for linear time-invariant (LTI) and piecewise affine (PWA) systems. Any method which is able to influence one of the three levers such that complexity is reduced, is referred to as *efficient*.

The thesis is subdivided into five parts, whereby each part is written to be self-contained. Hence, certain key theorems and definitions are stated more than once throughout the thesis.

In the first part of the thesis, the necessary background from the field of optimal control and computational geometry is summarized.

In Part II, various schemes for analysis of PWA systems are presented. Specifically, algorithms to compute robust positive invariant sets are given and sufficient conditions for finite time termination of these algorithms are derived. Furthermore, various computation schemes to analyze stability of PWA systems are given. It is shown how the search for a PWA Lyapunov function can be posed as an LP. In addition, various improvements and extensions to LMI based schemes for identifying piecewise quadratic Lyapunov functions are given. The results in this part are based on [GM03, GLPM03, GPM03, RG04a, RGK+04a, GRMM05].

In Part III, various schemes to obtain low complexity feedback control for discrete-time LTI systems are covered. Specifically, various algorithmic improvements to the multi-parametric programming solvers are presented, which serve to speed up the necessary off-line controller computation time. In a next step, it is shown how the stability analysis schemes from Part II can be utilized to obtain feedback controllers of very low complexity. Finally, two schemes for post-processing the explicit feedback solution are given. Both schemes reduce the necessary on-line effort and storage space. The results in this part are based on [SLG+04, GBTM03, GBTM04, GM03, GPM03, RG04b].

Part IV of this thesis deals with controller construction for PWA systems. First, a scheme to obtain stabilizing optimal controllers for PWA systems is derived. Then, a novel controller computation scheme based on dynamic programming is presented. The proposed algorithm computes the optimal controllers for PWA systems significantly more efficiently than previous methods. In a next step, it is shown how the stability analysis schemes from Part II can be utilized to formulate control problems

which yield feedback controllers of very low complexity. The results in this part are based on [BGBM05, GKBM04a]

In Part V, the Multi-Parametric Toolbox (MPT) is presented. The MPT toolbox for MATLAB contains all the algorithms presented in this thesis as well as a wide range of additional algorithms and tools developed by the academic community. This part of the thesis will introduce the reader to MPT, describe the software framework and provide examples. The content in this part is based on [KGB04].

Contributions

The main contribution of this thesis is a novel way of looking at the interaction between control objectives and the resulting controller complexity. In classic control schemes, the aim is to obtain controllers which guarantee constraint satisfaction, asymptotic stability and optimal performance. For systems subject to constraints, the disadvantage of these goals is the often prohibitive complexity of the resulting control laws. In order to obtain simpler controllers, it is necessary to pose simpler objectives.

Three approaches are conceivable: the requirement for constraint satisfaction can be dropped when constructing control laws and verified a posteriori [GLPM03]. Constraint satisfaction can be verified by checking if the controlled set is invariant. This approach is pursued in Chapter 7, where the computation of invariant sets for piecewise affine (PWA) systems subject to bounded additive disturbances [RG04a, RGK+04a, GRMM05] is investigated. The key contribution here is a set of sufficient criteria for finite time termination of the maximum robust invariant set computation.

The second approach is to use a performance objective that is different from standard optimal control [GM03, GPM03, GKBM04a]. Defining simpler performance objectives will yield simpler control laws. Specifically, 'minimal-time' and 'minimum-switching' control was investigated. This approach is pursued in Chapters 11 and 16 for LTI and PWA systems, respectively.

The third and most promising approach is to drop the requirement of asymptotic stability when constructing control laws [GKBM04b, GM04]. By checking asymptotic stability of the controller a posteriori instead of enforcing it during the controller

synthesis, very simple control laws can be obtained. For efficient stability analysis, several contributions are made in Chapter 8 [GKBM04b, GLPM03, BGLM05], where it is shown how to efficiently compute piecewise affine, piecewise quadratic and piecewise polynomial Lyapunov functions for discrete-time PWA systems. The control schemes to obtain control laws of very low complexity are presented in Chapters 11 and 16 for LTI and PWA systems, respectively.

The second key contribution during my Ph.D. studies is the co-development of the Multi-Parametric Toolbox which is covered in Part V [KGB04]. By developing such an extensive software tool, we have made the world of multi-parametric control accessible to a much larger audience than was previously the case. Hopefully, the easy access to the latest advances in multi-parametric theory will also lead to more practical applications in the future.

Finally, this thesis introduces a whole set of additional tools to further simplify the application of multi-parametric controllers.

- Section 10.2: Speed improvements of multi-parametric programming solvers by efficient removal of redundant hyper-planes in polytopes [SLG+04].
- Section 10.3: Computation of the constrained infinite-time linear quadratic regulator [GBTM03, GBTM04].
- Section 12.2: Controller post-processing scheme using interpolation to simplify feedback laws obtained via multi-parametric quadratic programming [RG04b].
- Section 12.3: Identification of the active feedback law in logarithmic time for controllers computed via multi-parametric linear programs [JGR04a].
- Chapter 14: Formulation of optimal control problems for PWA systems such that stability and constraint satisfaction is guaranteed [GKBM04a].
- Chapter 15: Efficient implementation of constrained finite-time optimal control of PWA systems with linear performance indices [BGBM05].

Note that all the results in this thesis have been obtained in close collaboration with various colleagues. I have tried to cite all results appropriately and would like

to point out that some sections of this thesis are based on papers of which I am not the first author. Hence, the exposition above merely states the contributions of this thesis and not my work as an individual. Conversely, not all of the results which were obtained during my graduate studies are contained in this thesis. Please see Appendix A for a full list of publications.

Part I

BACKGROUND

Seite Leer /
Blank leaf

1

Standard Optimization Problems

For the sake of completeness, some standard optimization problems and definitions will first be introduced. For a detailed reference, we refer the reader to the excellent book [BV04].

A generic optimization problem can be described by the following set of equations.

$$\begin{aligned} \min_{x} \quad & f_0(x) && && \text{(1.1a)} \\ \text{subj. to} \quad & f_i(x) \leq 0, && i = 1, \ldots, q, && \text{(1.1b)} \\ & g_j(x) = 0, && j = 1, \ldots, q_{eq}, && \text{(1.1c)} \end{aligned}$$

with an objective function $f_0 : \mathbb{R}^n \to \mathbb{R}$ and constraint functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $g_j : \mathbb{R}^n \to \mathbb{R}$. The variable x is the *optimization variable* and the solution x^* to optimization problem (1.1) is referred to as *optimizer*.

Definition 1.1.1 (Convex Function, [Wei]) *A convex function is a continuous function whose value at the midpoint of every interval in its domain does not exceed the average of its values at the ends of the interval. In other words, a function $f(x)$ is convex on an interval $[a, b]$ if for any two points x_1 and x_2 in $[a, b]$,*

$$f(\frac{1}{2}(x_1 + x_2)) \leq \frac{1}{2}(f(x_1) + f(x_2))$$

If $f(x)$ has a second derivative in $[a, b]$, then a necessary and sufficient condition for it to be convex on that interval is that the second derivative $f''(x) \geq 0$ for all x in $[a, b]$.

If the objective function f_0 and the constraint functions $f_i(x)$ are convex and the equality constraints g_j are all affine (i.e. $A_{eq}x = B_{eq}$), problem (1.1) is a convex optimization problem. Although general convex optimization problems can be solved relatively efficiently it is always advantageous to use dedicated solvers for specific problems. A number of specific convex optimization problems for which such solvers exist will be discussed in the following.

Linear Program (LP)

$$\begin{aligned} \min_x \quad & c^T x \\ \text{subj. to} \quad & Ax \leq B, \\ & A_{eq}x = B_{eq}. \end{aligned}$$

A practical algorithm to solve an LP with n variables and s constraints requires roughly $O((n^3 + n^2 s)\sqrt{s})$ operations on average (see Section 10.2, page 104). There are two fundamentally different types of algorithms for solving LPs: *simplex* and *interior-point* solvers [Van01]. The runtime for the simplex method is exponential in the worst case, while interior-point algorithms have a worst-case polynomial bound. However, this worst-case bound has little relevance for practical problems and both schemes are competitive in practice [AM85, Mit99, Mit04].

Quadratic Program (QP)

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{subj. to} \quad & Ax \leq B, \\ & A_{eq}x = B_{eq}. \end{aligned}$$

When referring to QPs it is generally assumed that $Q \succeq 0$, such that the resulting optimization problem is convex. QPs can be solved with roughly the same efficiency as LPs, but on average the solvers are approximately 5-times slower than LP solvers [Neu04, page 37].

Linear Matrix Inequality (LMI) The semidefinite cone $F(x) \succeq 0$ can be described with LMIs according to

$$F(x) = F_0 + \sum_{i=0}^{q} x_{(i)} F_i \succeq 0, \quad x \in \mathbb{R}^q,\ F_i = F_i^T \in \mathbb{R}^{n \times n},$$

where $x_{(i)}$ denotes the i-th element of the vector x. LMIs are generally used when searching for a matrix, for which some *linear* combination of the matrix is required to be positive definite, hence the term *Linear Matrix Inequality* (LMI). In control for example, LMIs are often used to obtain Lyapunov functions and stabilizing feedback laws [BGFB94]. Note that an LMI defines a feasible set and is not an optimization problem as (1.1).

Semidefinite Programming (SDP)

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{subj. to} \quad & F(x) \succeq 0, \\ & A_{eq}x = B_{eq}. \end{aligned}$$

Optimization over the semidefinite cone is called semidefinite programming. An introduction to SDPs and LMIs can be found in [WSV00, VB96] and a good overview of the application of SDPs to control problems is also given in [Löf03, BGFB94]. Although SDPs can be solved in polynomial time, the associated algorithms are roughly one order of magnitude slower than LP or QP solvers. The number of required iterations of the associated interior point methods is roughly the same, but much more work is required for each iteration when solving an SDP. Note that LPs and QPs can be solved via SDP.

Determinant Maximization (MAXDET)

$$\begin{aligned} \min_{x} \quad & c^T x - \log \det G(x) \\ \text{subj. to} \quad & F(x) \succeq 0 \\ & G(x) \succeq 0 \\ & A_{eq}x = B_{eq} \end{aligned}$$

The so called MAXDET problems have many applications in practice, since $G(x)$ can be chosen such that $\log \det G(x)$ is proportional to the volume of an ellipsoid. MAXDET problems are common in invariant set computations [Löf03].

Seite Leer /
Blank leaf

2

Polytopes

Polytopic (or, more general, polyhedral) sets are an integral part of most standard constrained control problems. For this reason we present some definitions and fundamental operations with polytopes. For additional details on polytope computation we refer the reader to [Zie94, Grü00, Fuk04c].

2.1 Definitions

Some basic definitions in computational geometry will be introduced in this section.

Definition 2.1.1 (Convex Set, [BV04]) *A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $x_1, x_2 \in C$ and any real scalar θ with $0 \leq \theta \leq 1$, we have $\theta x_1 + (1-\theta)x_2 \in C$.*

Definition 2.1.2 (Neighborhood, [Wei]) *The neighborhood of a point $x \in \mathbb{R}^n$ (also called an epsilon-neighborhood or infinitesimal open set) is the set of points inside an n-ball with center x and radius $\epsilon > 0$.*

Definition 2.1.3 (Closed Set, [Wei]) *A set S is closed if every point outside S has a neighborhood disjoint from S.*

Definition 2.1.4 (Bounded Set, [Wei]) *A set in $\mathbb{R}^n$ is bounded if it is contained inside some ball $\mathcal{B}_R = \{x \in \mathbb{R}^n \mid \|x\|_2 \leq R\}$ of finite radius R.*

Definition 2.1.5 (Compact Set, [Wei]) *A set in $\mathbb{R}^n$ is compact if it is bounded and closed.*

Definition 2.1.6 (Polyhedron, [Grü00]) *A convex set $\mathcal{S} \subseteq \mathbb{R}^n$ given as an intersection of a finite number of closed half-spaces*

$$\mathcal{S} = \{x \in \mathbb{R}^n \mid S^x x \leq S^o\}, \tag{2.1}$$

is called a polyhedron. Here, $S^o \in \mathbb{R}^q$, $S^x \in \mathbb{R}^{q \times n}$ where q denotes the number of half-spaces defining $\mathcal{S}$ and the operator $\leq$ denotes a element-wise comparison of two vectors.

Definition 2.1.7 (Polytope, [Grü00]) *A bounded polyhedron $\mathcal{P} \subset \mathbb{R}^n$*

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^o\}, \tag{2.2}$$

is called a polytope. Here, $P^o \in \mathbb{R}^q$, $P^x \in \mathbb{R}^{q \times n}$ where q denotes the number of half-spaces defining $\mathcal{P}$ and the operator $\leq$ denotes a element-wise comparison of two vectors.

A polytope defined by half-spaces is depicted in Figure 2.1(a).

Definition 2.1.8 (Dimension of Polytope) *A polytope $\mathcal{P} \subset \mathbb{R}^n$ is of dimension $d \leq n$, if there exists a d-dimensional ball with radius $\epsilon > 0$ contained in $\mathcal{P}$ and there exists no $(d+1)$-dimensional ball with radius $\epsilon > 0$ contained in $\mathcal{P}$.*

Definition 2.1.9 (Face,Vertex, Edge, Ridge, Facet, [Zie94]) *A linear inequality $a^T x \leq b$ is called valid for a polyhedron $\mathcal{P}$ if $a^T x \leq b$ holds for all $x \in \mathcal{P}$. A subset $\mathcal{F}$ of a polyhedron is called a face of $\mathcal{P}$ if it can be represented as*

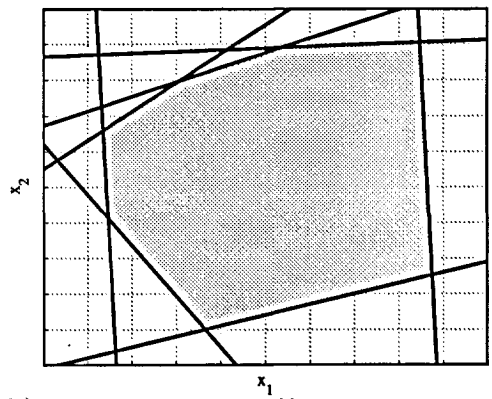
$$\mathcal{F} = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid a^T x = b\}, \tag{2.3}$$

for some valid inequality $a^T x \leq b$. The faces of a polyhedron $\mathcal{P}$ of dimension 0, 1, $(n-2)$ and $(n-1)$ are called vertices, edges, ridges and facets, respectively.

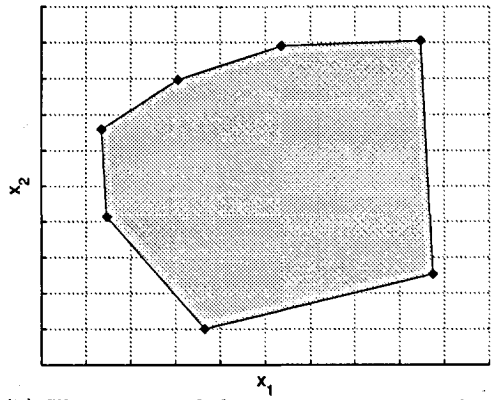
Note that $\emptyset$ and $\mathcal{P}$ itself are also faces of $\mathcal{P}$ [Fuk04c].

One of the fundamental properties of a polytope is that it can be described in half-space representation as in Definition 2.2 or in vertex presentation, as given below,

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{v_P} \alpha_i V_P^{(i)},\ 0 \leq \alpha_i \leq 1,\ \sum_{i=1}^{v_P} \alpha_i = 1\}, \tag{2.4}$$



(a) Illustration of the half-space representation of a polytope $\mathcal{P}$. The half-spaces $P^x_{(i)}x \leq P^o_{(i)}$, $i = 1, \ldots, 7$ are depicted in bold.



(b) Illustration of the vertex representation of a polytope $\mathcal{P}$. The vertices $V_P^{(1)}, \ldots, V_P^{(7)}$ are depicted in bold.

Figure 2.1: Illustration of a polytope $\mathcal{P}$ in half-space and vertex representation.

where $V_P^{(i)}$ denotes the i-th vertex of $\mathcal{P}$, and v_P is the total number of vertices of $\mathcal{P}$ (see Figure 2.1(b)).

It is obvious from the above definitions that every polytope represents a convex and compact set. We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^o\}$ is *full dimensional* if $\exists x \in \mathbb{R}^n$, $\epsilon \in \mathbb{R}$ such that $\epsilon > 0$ and $P^x(x+\delta) \leq P^o, \forall\, \delta \in \mathbb{R}^n$ subject to $\|\delta\| \leq \epsilon$, i.e. it is possible to fit a n-dimensional ball inside the polytope $\mathcal{P}$. A polytope is referred to as *empty* if $\nexists x \in \mathbb{R}^n$ such that $P^x x \leq P^o$. Furthermore, if $\|P^x_{(i)}\| = 1$, where $P^x_{(i)}$ denotes i-th row of a matrix P^x, we say that the polytope $\mathcal{P}$ is *normalized.*

Remark 2.1.10 *Note that the MPT toolbox (see Part V or [KGB04]) only deals with full dimensional polytopes. Polyhedra and lower dimensional polytopes are not considered, since they are not necessary to formulate realistic control problems, i.e. it is always possible to formulate the problems using full dimensional polytopic sets only.*

We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^o\}$ is in a *minimal representation* if the removal of any of the rows in $P^x x \leq P^o$ would change it (i.e., there are no redundant half-spaces). The computation of a minimal representation (henceforth referred to as *polytope reduction*) of polytopes is discussed in Section 10.2 and generally requires to solve one LP for each half-space defining the non-minimal representation of $\mathcal{P}$ [OSS95, Bon83]. It is straightforward to see that a normalized,

full dimensional polytope $\mathcal{P}$ has a *unique* minimal representation. This fact is very useful in practice. Normalized, full dimensional polytopes in a minimal representation allow us to avoid any ambiguity when comparing them and very often speed up other polytope manipulations.

Definition 2.1.11 (P-collection) *A P-collection is the (possibly non-convex) union of a finite number of R polytopes $\mathcal{R}_r$, i.e. $\mathcal{R} = \bigcup_{r\in\{1,\dots,R\}} \mathcal{R}_r$.*

Note that the polytopes $\mathcal{R}_r$ defining the P-collection $\mathcal{R}$ can be disjoint and/or overlapping.

Remark 2.1.12 *Algorithms for all operations and functions described in this chapter are contained in the MPT toolbox (see Part V or [KGB04]).*

2.2 Operations on Polytopes

In this section, some of the basic manipulations on polytopes will be defined.

Chebychev Ball: The Chebychev Ball of a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^o\}$ corresponds to the largest radius ball $\mathcal{B}_R(x_c) = \{x \in \mathbb{R}^n \mid \|x - x_c\|_2 \leq R\}$, such that $\mathcal{B}_R \subset \mathcal{P}$, see Figure 2.2(a). The center and radius of the Chebychev ball can be easily found by solving the following LP [BV04]

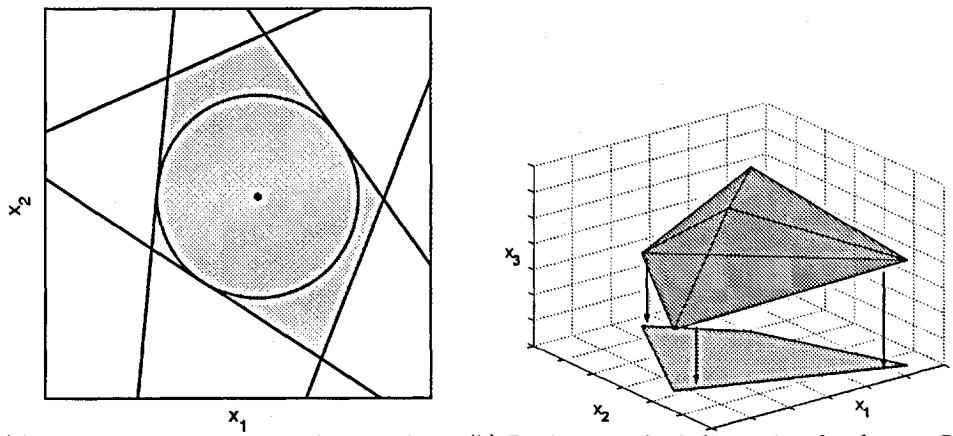
$$\max_{x_c,R} R \tag{2.5a}$$

$$\text{subj. to } P^x_{(i)} x_c + R\|P^x_{(i)}\| \leq P^o_{(i)}, \quad \forall i \in \{1,\dots,q\}. \tag{2.5b}$$

The subindex (i) in (2.5) denotes the i-th row of $P^x_{(i)}$ and $P^o_{(i)}$, respectively and $\mathcal{P}$ is defined by the intersection of q half-spaces. If the obtained radius $R = 0$, then the polytope is lower dimensional; if $R < 0$, then the polytope is empty. Note that the center of the Chebychev Ball is not unique, in general, i.e. there can be multiple solutions (e.g. for rectangles).

Projection: Given a polytope $\mathcal{P} = \{x \in \mathbb{R}^n, y \in \mathbb{R}^m \mid P^x x + P^y y \leq P^o\} \subset \mathbb{R}^{n+m}$ the orthogonal projection onto the x-space $\mathbb{R}^n$ is defined as

$$\text{proj}_x(\mathcal{P}) \triangleq \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m \text{subj. to } P^x x + P^y y \leq P^o\}. \tag{2.6}$$



(a) Chebychev ball contained in a polytope $\mathcal{P}$.

(b) Projection of a 3-dimensional polytope $\mathcal{P}$ onto a plane.

Figure 2.2: Illustration of the projection operation and the Chebychev ball.

An illustration of a projection operation is given in Figure 2.2(b). Current projection methods can be grouped into four classes: Fourier elimination [Cer63, KS90], block elimination [Bal98], vertex based approaches and wrapping-based techniques [JKM04]. For a good introduction to projection, we refer the reader to [JKM04] and the references therein.

Set-Difference: The Set-Difference of two polytopes $\mathcal{P}$ and $\mathcal{Q}$

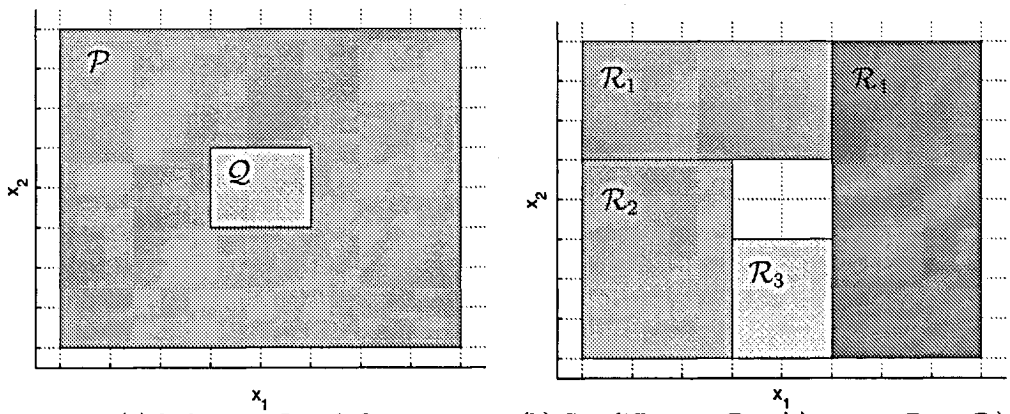
$$\mathcal{R} = \mathcal{P} \setminus \mathcal{Q} \triangleq \{x \in \mathbb{R}^n \mid x \in \mathcal{P}, x \notin \mathcal{Q}\}, \tag{2.7}$$

is a P-collection $\mathcal{R} = \bigcup_i \mathcal{R}_i$, which is easily computed by consecutively inverting the half-spaces defining $\mathcal{Q}$ as described in [BMDP02] (see Figure 2.3). The set difference between two P-collections $\mathcal{C}$ and $\mathcal{D}$ can be computed as described in [BT03, GKBM03, RKM03]. Checking whether $\mathcal{C} \subseteq \mathcal{D}$ is easily implemented since $\mathcal{C} \subseteq \mathcal{D} \Leftrightarrow \mathcal{C} \setminus \mathcal{D} = \emptyset$. Similarly $\mathcal{C} = \mathcal{D}$ is also easily verified since $\mathcal{C} = \mathcal{D} \Leftrightarrow (\mathcal{C} \setminus \mathcal{D} = \emptyset$ and $\mathcal{D} \setminus \mathcal{C} = \emptyset)$.

Remark 2.2.1 *The set difference of two closed sets $\mathcal{C}$ and $\mathcal{D}$ is an open set, if $\mathcal{C} \cap \mathcal{D} \neq \emptyset$. In this thesis, we will henceforth only consider the closure of $\mathcal{C} \setminus \mathcal{D}$.*

Convex Hull: The convex hull of a set of points $V = [v_1, \ldots, v_P]$ is defined as

$$\text{hull}(V) = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{v_P} \alpha_i v_i,\ 0 \leq \alpha_i \leq 1,\ \sum_{i=1}^{v_P} \alpha_i = 1\}. \tag{2.8}$$



(a) Polytopes $\mathcal{P}$ and $\mathcal{Q}$

(b) Set difference $\mathcal{R} = \bigcup_{i\in\{1,\dots,4\}} \mathcal{R}_i = \mathcal{P} \setminus \mathcal{Q}$.

Figure 2.3: Illustration of the set-difference operation.

The convex hull operation is used to switch between half-space and vertex representations. The convex hull of a union of polytopes (referred to as *Extended Convex Hull*, [FLL00]) $\mathcal{R}_r \subset \mathbb{R}^n$, $r = 1, \dots, R$, is a polytope

$$\text{hull}\left(\bigcup_{r=1}^{R} \mathcal{R}_r\right) \triangleq \{x \in \mathbb{R}^n \mid \exists x_r \in \mathcal{R}_r,\ x = \sum_{r=1}^{R} \alpha_r x_r,\ 0 \le \alpha_r \le 1,\ \sum_{r=1}^{R} \alpha_r = 1\}. \tag{2.9}$$

An illustration of the convex hull operation is given in Figure 2.4.

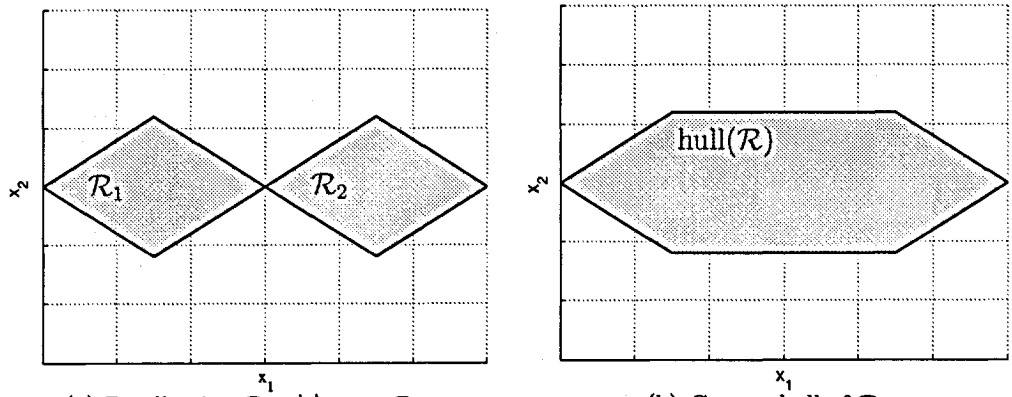
A convex hull problem is considered *non-degenerate* if $\nexists$ a valid hyperplane h in $\mathbb{R}^n$ containing more than n input vertices. If the convex hull problem is non-degenerate then the *Reverse Search* algorithm in [AF96, Avi00] is linear in the number of output facets. For many degenerate problems, the expected behavior of the *Double Description* method [FP96, Fuk04a, MRTT53] is polynomial. The same holds for the dual *Beneath-Beyond* method [BDH96]. Although other efficient methods exist (e.g. [Sei86]) the complexity of all convex hull computation schemes is exponential in the worst (i.e., degenerate) case.

Envelope: The envelope of two polyhedra $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \le P^o\}$ and $\mathcal{Q} = \{x \in \mathbb{R}^n \mid Q^x x \le Q^o\}$ is given by

$$\text{env}(\mathcal{P}, \mathcal{Q}) = \{x \in \mathbb{R}^n \mid \bar{P}^x x \le \bar{P}^o,\ \bar{Q}^x x \le \bar{Q}^o\}, \tag{2.10}$$

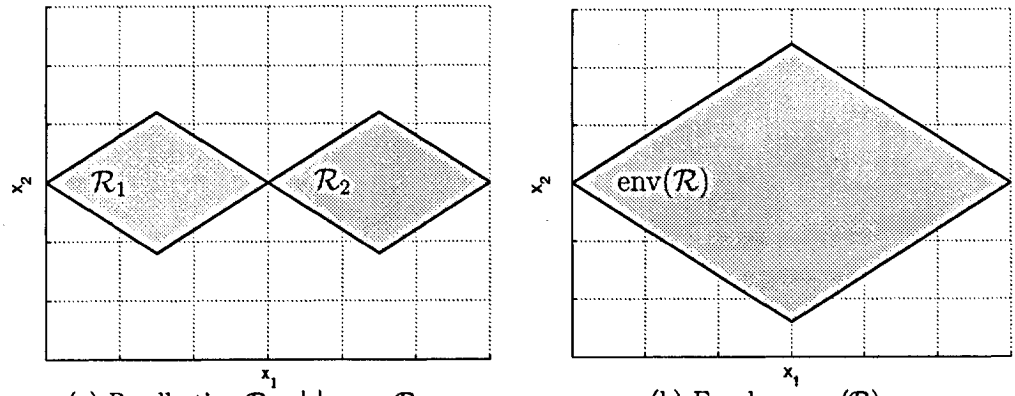
where $\bar{P}^x x \le \bar{P}^o$ is the subsystem of $P^x x \le P^o$ obtained by removing all the inequalities not valid for the polyhedron $\mathcal{Q}$, and $\bar{Q}^x x \le \bar{Q}^o$ are defined in

a similar way with respect to $Q^x x \leq Q^o$ and $\mathcal{P}$ [BFT01]. The envelope can analogously be computed for a P-collection or a complex. An illustration of the envelope operation is depicted in Figure 2.5. The envelope can be computed by solving $c \cdot d$ LPs where c is the number of input polytopes (here: $\mathcal{P}$ and $\mathcal{Q}$, i.e. $c = 2$) and d is the total number of facets [BFT01]. It holds that $\mathcal{P} \cup \mathcal{Q} \subseteq \text{env}(\mathcal{P}, \mathcal{Q})$ and that $\mathcal{P} \cup \mathcal{Q}$ is convex $\Leftrightarrow \mathcal{P} \cup \mathcal{Q} = \text{env}(\mathcal{P}, \mathcal{Q})$. Note that the envelope of several polytopes can be a polyhedral set or even $\mathbb{R}^n$, e.g. the envelope of a star shaped object is $\mathbb{R}^n$.



(a) P-collection $\mathcal{R} = \bigcup_{i \in \{1,2\}} \mathcal{R}_i$. (b) Convex hull of $\mathcal{R}$.

Figure 2.4: Illustration of the convex hull operation.



(a) P-collection $\mathcal{R} = \bigcup_{i \in \{1,2\}} \mathcal{R}_i$. (b) Envelope env($\mathcal{R}$).

Figure 2.5: Illustration of the envelope operation.

Vertex Enumeration: The operation of extracting the vertices of a polytope $\mathcal{P}$ given in half-space representation is referred to as vertex enumeration. This

operation is the dual to the convex hull operation and the algorithmic implementation is identical to a convex hull computation, i.e. given a set of points $V = [v_1, \ldots, v_P]$ it holds that $V = \text{vert}(\text{hull}(V))$, where the operator vert denotes the vertex enumeration.

A vertex enumeration problem is considered *non-degenerate*, if $\nexists$ a point $x \in \mathbb{R}^n$ lying on the boundary of more than n input half-spaces. If the vertex enumeration problem is non-degenerate then the *Reverse Search* algorithm in [AF96, Avi00] is linear in the number of input vertices. For many degenerate problems, the expected behavior of the *Double Description* method [FP96, Fuk04a, MRTT53] is polynomial. The same holds for the dual *Beneath-Beyond* method [BDH96]. Although other efficient methods exist (e.g. [Sei86]) the complexity of all vertex enumeration computation schemes is exponential in the worst case.

Pontryagin Difference: The Pontryagin difference (also known as Minkowski-Difference) of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a polytope

$$\mathcal{P} \ominus \mathcal{Q} \triangleq \{x \in \mathbb{R}^n \mid x + q \in \mathcal{P},\ \forall q \in \mathcal{Q}\}. \tag{2.11}$$

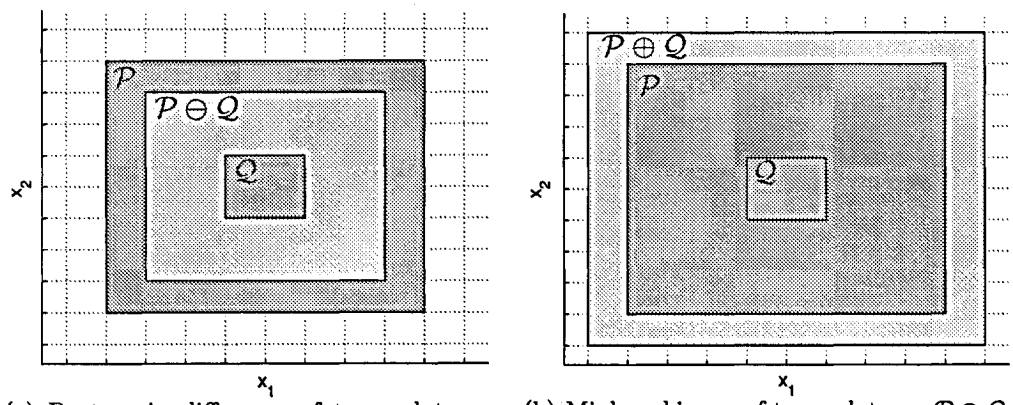
The Pontryagin difference can be computed by solving one LP for each half-space defining $\mathcal{P}$ [KG98]. For special cases (e.g. when $\mathcal{Q}$ is a hypercube), even more efficient computational methods exist [KM03]. An illustration of the Pontryagin difference is given in Figure 2.6(a).

Minkowski Sum: The Minkowski sum of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a polytope

$$\mathcal{P} \oplus \mathcal{Q} \triangleq \{x + q \in \mathbb{R}^n \mid x \in \mathcal{P},\ q \in \mathcal{Q}\}. \tag{2.12}$$

If $\mathcal{P}$ and $\mathcal{Q}$ are given in vertex representation, the Minkowski sum can be computed in time bounded by a polynomial function of input and output size [GS93, Fuk04b]. If $\mathcal{P}$ and $\mathcal{Q}$ are given in half-space representation, the Minkowski sum is a computationally expensive operation which requires either vertex enumeration and convex hull computation in n-dimensions or a projection from $2n$ down to n dimensions. The implementation of the Minkowsi sum via projection is described below.

$$P = \{y \in \mathbb{R}^n \mid P^y y \leq P^o\}, \qquad Q = \{z \in \mathbb{R}^n \mid Q^z z \leq Q^o\},$$



(a) Pontryagin difference of two polytopes $\mathcal{P} \ominus \mathcal{Q}$.

(b) Minkowski sum of two polytopes $\mathcal{P} \oplus \mathcal{Q}$.

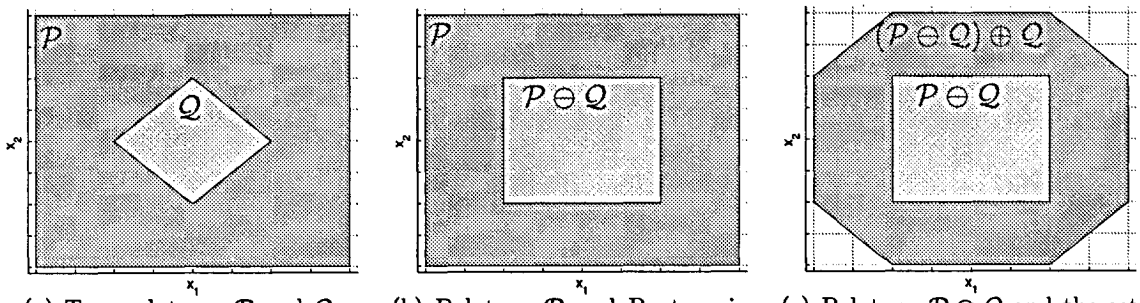
Figure 2.6: Illustration of the Pontryagin difference and Minkowski sum operations.

it holds that

$$\begin{aligned} W &= P \oplus Q \\ &= \left\{ x \in \mathbb{R}^n \mid x = y + z,\ P^y y \leq P^o,\ Q^z z \leq Q^o,\ y, z \in \mathbb{R}^n \right\} \\ &= \left\{ x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^n, \text{ subj. to } P^y y \leq P^o,\ Q^z (x - y) \leq Q^o \right\} \\ &= \left\{ x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^n, \text{ subj. to } \begin{bmatrix} 0 & P^y \\ Q^z & -Q^z \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} P^o \\ Q^o \end{bmatrix} \right\} \\ &= \text{proj}_x \left(\left\{ x, y \in \mathbb{R}^n \mid \begin{bmatrix} 0 & P^y \\ Q^z & -Q^z \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} P^o \\ Q^o \end{bmatrix} \right\} \right). \end{aligned}$$

Both the projection and vertex enumeration based methods are implemented in the MPT toolbox (see Part V or [KGB04]). An illustration of the Minkowski sum is given in Figure 2.6(b).

Remark 2.2.2 *The Minkowski sum is* not *the complement of the Pontryagin difference. For two polytopes $\mathcal{P}$ and $\mathcal{Q}$, it holds that $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q} \subseteq \mathcal{P}$. This is illustrated in Figure 2.7.*



(a) Two polytopes $\mathcal{P}$ and $\mathcal{Q}$.

(b) Polytope $\mathcal{P}$ and Pontryagin difference $\mathcal{P} \ominus \mathcal{Q}$.

(c) Polytope $\mathcal{P} \ominus \mathcal{Q}$ and the set $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q}$.

Figure 2.7: Illustration that $(\mathcal{P} \ominus \mathcal{Q}) \oplus \mathcal{Q} \subseteq \mathcal{P}$.

2.3 Operations on P-collections

This section covers some results and algorithms which are specific to operations with P-collections. P-collections are unions of polytopes and therefore the set of states contained in a P-collection can be represented in an infinite number of ways, i.e. the P-collection representation is not unique. For example, one can subdivide any polytope $\mathcal{P}$ into a number of smaller polytopes whose union is a P-collection which covers $\mathcal{P}$. Note that the complexity of all subsequent computations depends strongly on the number of polytopes representing a P-collection. The smaller the cardinality of a P-collection, the more efficient the computations.

The first two results given here show how the set difference of a P-collection and a P-collection (or polyhedron) may be computed:

Lemma 2.3.1 *Let* $\mathcal{C} \triangleq \bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j$ *be a P-collection, where all the* $\mathcal{C}_j$, $j \in \{1,\dots,J\}$, *are non-empty polyhedra. If* $\mathcal{S}$ *is a non-empty polyhedron, then* $\mathcal{C} \setminus \mathcal{S} = \bigcup_{j\in\{1,\dots,J\}} (\mathcal{C}_j \setminus \mathcal{S})$ *is a P-collection.*

Lemma 2.3.2 *Let the sets* $\mathcal{C} \triangleq \bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j$ *and* $\mathcal{D} \triangleq \bigcup_{y=1,\dots,Y} \mathcal{D}_y$ *be P-collections, where all the* $\mathcal{C}_j$, $j \in \{1,\dots,J\}$, *and* $\mathcal{D}_y$, $y \in \{1,\dots,Y\}$, *are non-empty polyhedra. If* $\mathcal{E}_0 \triangleq \mathcal{C}$ *and* $\mathcal{E}_y \triangleq \mathcal{E}_{y-1} \setminus \mathcal{D}_y$, $y \in \{1,\dots,Y\}$ *then* $\mathcal{C} \setminus \mathcal{D} = \mathcal{E}_Y$ *is a P-collection.*

The reader is referred to [RKM03] for proofs and comments on computational efficiency. That $\mathcal{C} \subseteq \mathcal{D}$ can be easily verified since $\mathcal{C} \subseteq \mathcal{D} \Leftrightarrow \mathcal{C} \setminus \mathcal{D} = \emptyset$, similarly $\mathcal{C} = \mathcal{D}$ is also easily verified since

$$\mathcal{C} = \mathcal{D} \Leftrightarrow (\mathcal{C} \setminus \mathcal{D} = \emptyset \text{ and } \mathcal{D} \setminus \mathcal{C} = \emptyset)$$

Next, an efficient algorithm for computing the Pontryagin difference of a P-collection and a polytope is presented. If $\mathcal{S}$ and $\mathcal{B}$ are two subsets of $\mathbb{R}^n$ it is known that $\mathcal{S} \ominus \mathcal{B} = [\mathcal{S}^c \oplus (-\mathcal{B})]^c$ (see for instance [Ser88, Ker00]), where $(\cdot)^c$ denotes the set complement. The following algorithm taken from [RGK+04a] implements the computation of the Pontryagin difference of a P-collection $\mathcal{C} \triangleq \cup_{j\in\{1,\ldots,J\}}\mathcal{C}_j$, where $\mathcal{C}_j, j \in \{1,\ldots,J\}$ are polytopes in $\mathbb{R}^n$, and a polytope $\mathcal{B} \subset \mathbb{R}^n$.

Algorithm 2.3.3 (Pontryagin Difference for P-collections, $\mathcal{C} \ominus \mathcal{B}$)

1. *Input: P-collection $\mathcal{C}$, polytope $\mathcal{B}$;*
2. $\mathcal{H} \triangleq \text{env}(\mathcal{C})$ *(or* $\mathcal{H} \triangleq \text{hull}(\mathcal{C})$*);*
3. $\mathcal{D} \triangleq \mathcal{H} \ominus \mathcal{B}$;
4. $\mathcal{E} \triangleq \mathcal{H} \setminus \mathcal{C}$;
5. $\mathcal{F} \triangleq \mathcal{E} \oplus (-\mathcal{B})$;
6. $\mathcal{G} \triangleq \mathcal{D} \setminus \mathcal{F}$;
7. *Output: P-collection $\mathcal{G} \triangleq \mathcal{C} \ominus \mathcal{B}$.*

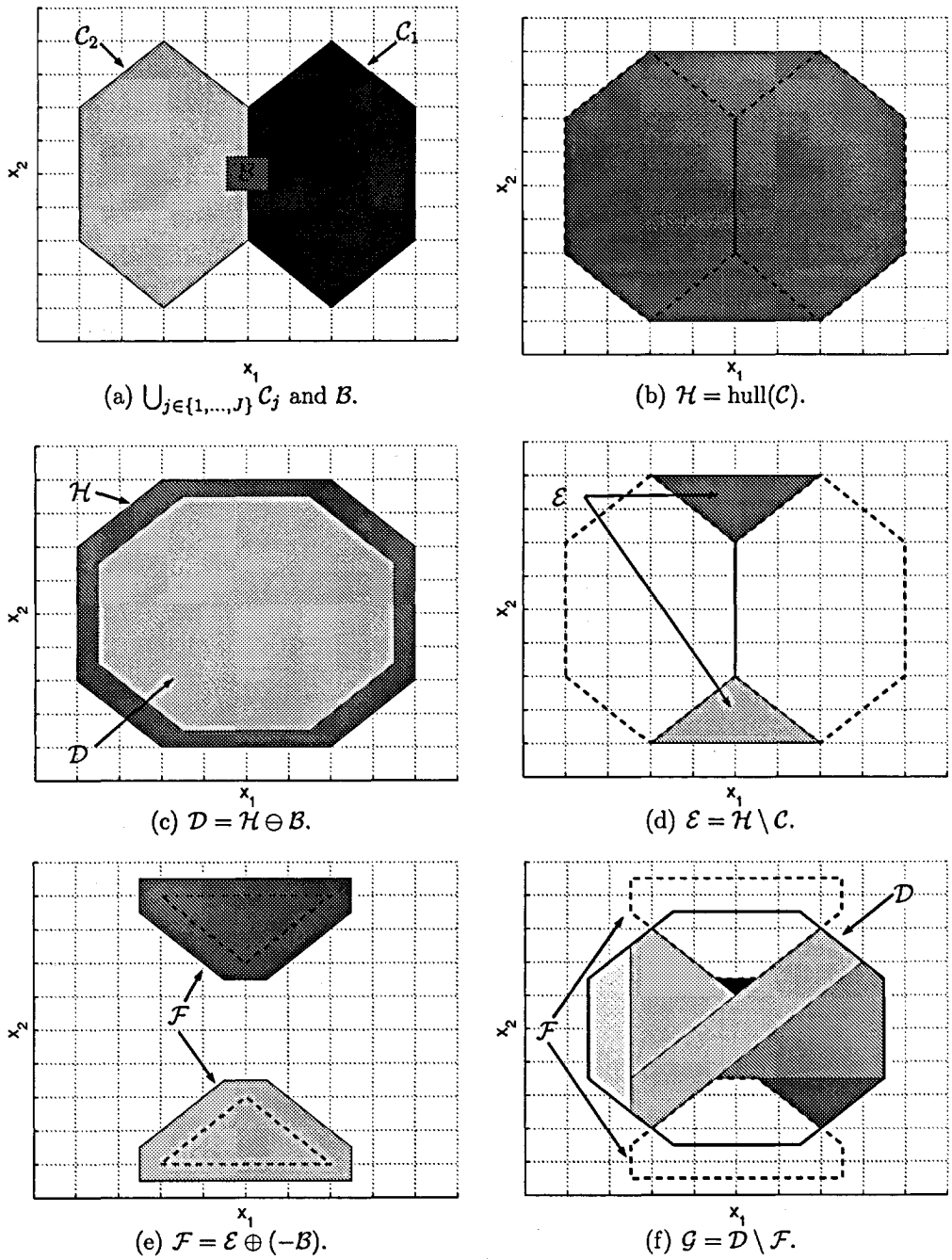
Remark 2.3.4 *Note that $\mathcal{H}$ in Step 2 of Algorithm 2.3.3 can be any convex set containing the P-collection $\mathcal{C}$. Furthermore, the computation of $\mathcal{H}$ is generally more efficient if the envelope operation is used instead of convex hull.*

Remark 2.3.5 *It is important to note that $(\bigcup_{j\in\{1,\ldots,J\}} \mathcal{C}_j) \ominus \mathcal{B} \neq \bigcup_{j\in\{1,\ldots,J\}}(\mathcal{C}_j \ominus \mathcal{B})$, where $\mathcal{B}$ and $\mathcal{C}_j$ are polyhedra; hence, the relatively high computational effort of computing the Pontryagin difference of a P-collection and a polytope.*

Theorem 2.3.6 (Computation of Minkowski Difference, [RGK+04a]) *For Algorithm 2.3.3, $\mathcal{G} = \mathcal{C} \ominus \mathcal{B}$.*

Proof It holds by definition that

$$\mathcal{D} \triangleq \mathcal{H} \ominus \mathcal{B} = \{x \mid x + w \in \mathcal{H},\ \forall w \in \mathcal{B}\},$$
$$\mathcal{E} \triangleq \mathcal{H} \setminus \mathcal{C} = \{x \mid x \in \mathcal{H} \text{ and } x \notin \mathcal{C}\}.$$



(a) $\bigcup_{j\in\{1,\dots,J\}} \mathcal{C}_j$ and $\mathcal{B}$.

(b) $\mathcal{H} = \text{hull}(\mathcal{C})$.

(c) $\mathcal{D} = \mathcal{H} \ominus \mathcal{B}$.

(d) $\mathcal{E} = \mathcal{H} \setminus \mathcal{C}$.

(e) $\mathcal{F} = \mathcal{E} \oplus (-\mathcal{B})$.

(f) $\mathcal{G} = \mathcal{D} \setminus \mathcal{F}$.

Figure 2.8: Graphical illustration of Algorithm 2.3.3.

By the definition of the Minkowski sum:

$$\mathcal{F} \triangleq \mathcal{E} \oplus (-\mathcal{B}) = \{x \mid x = z + w,\ z \in \mathcal{E}, w \in (-\mathcal{B})\}$$

$$= \{x \mid \exists w \in (-\mathcal{B}), \text{ s.t. } x + w \in \mathcal{E}\}.$$

By definition of the set difference:

$$\begin{aligned}\mathcal{D} \setminus \mathcal{F} &\triangleq \{x \mid x \in \mathcal{D} \text{ and } x \notin \mathcal{F}\} \\ &= \{x \in \mathcal{D} \mid \nexists\, w \in \mathcal{B} \text{ s.t. } x + w \in \mathcal{E}\} \\ &= \{x \in \mathcal{D} \mid x + w \notin \mathcal{E},\ \forall w \in \mathcal{B}\}.\end{aligned}$$

From the definition of the set $\mathcal{D}$:

$$\mathcal{D} \setminus \mathcal{F} = \{x \mid x + w \in \mathcal{H} \text{ and } x + w \notin \mathcal{E},\ \forall w \in \mathcal{B}\}$$

And from the definition of the set $\mathcal{E}$ and because $\mathcal{C} \subseteq \mathcal{H}$:

$$\begin{aligned}\mathcal{D} \setminus \mathcal{F} &= \{x \mid x + w \in \mathcal{H} \text{ and } (x + w \notin \mathcal{H} \text{ or } x + w \in \mathcal{C})\ \forall w \in \mathcal{B}\} \\ &= \{x \mid x + w \in \mathcal{C},\ \forall w \in \mathcal{B}\} \\ &= \mathcal{C} \ominus \mathcal{B}.\end{aligned}$$

□

Algorithm 2.3.3 is illustrated on a sample P-collection in Figures 2.8(a) to 2.8(f).

Remark 2.3.7 *It should be noted that Algorithm 2.3.3 for computation of the Pontryagin difference is conceptually similar to the one proposed in [Ser88, Ker00, KM02]. However, the envelope [BFT01] operation employed in step 2 significantly reduces (in general) the number of sets obtained at step 4, which in turn results in fewer Minkowski set additions. Since the computation of a Minkowski set addition is expensive, a runtime improvement can be expected. The necessary computations can be efficiently implemented by using standard computational geometry software such as [Ver03, KGB04].*

Seite Leer /
Blank leaf

3

Multi-Parametric Programming

In this chapter, the basics of multi-parametric programming will be summarized. For a review of standard optimization techniques, we refer the reader to [BV04]. An in-depth discussion of multi-parametric programs is given in [Bor03] and [Tøn00].

3.1 Definitions

Consider the following optimization problem

$$\begin{aligned} J_N^*(x) &= \min_{U_N} V(x, U_N) && \text{(3.1a)} \\ \text{subj. to} \quad & GU_N \leq W + Ex, && \text{(3.1b)} \end{aligned}$$

where $U_N \in \mathbb{R}^N$ is the optimization variable and $x \in \mathbb{R}^n$ is the parameter with $G \in \mathbb{R}^{q \times N}$, $W \in \mathbb{R}^q$ and $E \in \mathbb{R}^{q \times n}$. In multi-parametric programming, the objective is to obtain the optimizer U_N^* for a whole range of parameters x, i.e. to obtain $U_N^*(x)$ as an explicit function of the parameter x. The term *multi* is used to emphasize that the parameter x is a vector and not a scalar. Depending on whether the objective function $V(x, U_N)$ is linear or quadratic in the optimization variable U_N, the terminology *multi-parametric Linear Program* (mp-LP) or *multi-parametric Quadratic Program* (mp-QP) is used. In this chapter we will concentrate on the multi-parametric Quadratic Program. For a detailed description of mp-LPs, we refer the reader to [Gal95, BBM00a, BBM00b].

Consider the following quadratic program

$$\begin{aligned} J_N^*(x) &= \min_{U_N} \left\{ U_N^T H U_N + x^T F U_N \right\} && \text{(3.2a)} \\ \text{subj. to} \quad & GU_N \leq W + Ex, && \text{(3.2b)} \\ & H \succ 0, && \text{(3.2c)} \end{aligned}$$

where the column vector $U_N \in \mathbb{R}^N$ is the optimization vector. The number of constraints q corresponds to the number of rows of W, i.e. $W \in \mathbb{R}^q$. Henceforth, $U_N^*(x)$ will be used to denote the optimizer of (3.2) for a given parameter x. For any given x, it is possible to obtain the optimizer by solving a standard quadratic programming problem[1]. Before going further, we will introduce the following definitions.

Definition 3.1.1 (Feasible Set $\mathcal{X}_N$) *We define the feasible set $\mathcal{X}_N \subseteq \mathbb{R}^n$ as the set of states x for which the optimization problem (3.2) is feasible, i.e.*

$$\mathcal{X}_N = \{x \in \mathbb{R}^n | \exists U_N \in \mathbb{R}^N,\ GU_N \leq W + Ex\}. \tag{3.3}$$

The set $\mathcal{X}_\infty$ is defined accordingly by $\mathcal{X}_\infty \triangleq \lim_{N\to\infty} \mathcal{X}_N$. The set $\mathcal{X}_N$ can be computed via a projection operation as in (2.6).

Definition 3.1.2 (Polytopic/Polyhedral Partition) *A collection of polytopic (polyhedral) sets $\{\mathcal{P}_r\}_{r=1}^R = \{\mathcal{P}_1, \ldots, \mathcal{P}_R\}$ is a polytopic (polyhedral) partition[2] of a polytopic (polyhedral) set Θ if* (i) $\bigcup_{r=1}^R \mathcal{P}_r = \Theta$, (ii) $(\mathcal{P}_r \backslash \partial\mathcal{P}_r) \cap (\mathcal{P}_q \backslash \partial\mathcal{P}_q) = \emptyset$, *$\forall r \neq q$, where ∂ denotes the boundary.*

Definition 3.1.3 (PWA and PWQ) *Consider the function f over a polyhedral set $\mathcal{S}$.*
$f : \mathcal{S} \to \mathbb{R}^d$ with $d \in \mathbb{N}_+$ is piecewise affine (PWA), if a partition $\{\mathcal{P}_r\}_{r=1}^R$ of set $\mathcal{S}$ exists, such that $f(x) = L_r x + C_r$ if $x \in \mathcal{P}_r$.
$f : \mathcal{S} \mapsto \mathbb{R}$ is piecewise quadratic (PWQ), if a partition $\{\mathcal{P}_r\}_{r=1}^R$ of set $\mathcal{S}$ exists, such that $f(x) = x^T Q_r x + L_r x + C_r$ if $x \in \mathcal{P}_r$.

Definition 3.1.4 (Active Constraints $\mathcal{A}^N(x)$) *The set of active constraints $\mathcal{A}^N(x)$ at point x of problem (3.2) is defined as*

$$\mathcal{A}^N(x) = \{i \in \mathcal{J} \mid G_{(i)}U_N^*(x) - W_{(i)} - E_{(i)}x = 0\},\ \mathcal{J} = \{1, 2, \ldots, q\},$$

where $G_{(i)}$, $W_{(i)}$, and $E_{(i)}$ denote the i-th row of the matrices G, W, and E, respectively, and q denotes the number of constraints, i.e. $W \in \mathbb{R}^q$.

[1] The standing assumption here is that $H \succ 0$. The case $H \succeq 0$ is covered in [TJB03c].

[2] Note that a partition is a more general structure than a complex (see Definition 12.3.1). For partitions, more than $n-1$ full dimensional polytopes in n dimensions can touch a facet of another polytope. For a complex, at most $n-1$ full dimensional polytopes in n dimensions can touch a facet of another polytope.

Definition 3.1.5 (Linear Independence Constraint Qualification, [TJB03b]) *For an active set of constraints $\mathcal{A}^N$, we say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients are linearly independent , i.e. $G_{\mathcal{A}^N}$ has full row rank.*

3.2 Properties and Computation

As shown in [BMDP02, TJB01], we wish to solve problem (3.2) for all x within the polyhedral set of values $\mathcal{X}_N$, by considering (3.2) as a multi-parametric Quadratic Program (mp-QP).

Theorem 3.2.1 (Properties mp-QP, [BMDP02, Bor03]) *Consider the multi-parametric Quadratic Program (3.2). Then, the set of feasible parameters $\mathcal{X}_N$ is convex, the optimizer $U_N^* : \mathcal{X}_N \to \mathbb{R}^N$ is continuous and piecewise affine (PWA), i.e.*

$$U_N^*(x) = F_r x + G_r, \quad \text{if} \quad x \in \mathcal{P}_r = \{x \in \mathbb{R}^n | H_r x \leq K_r\}, \; r = 1, \ldots, R, \tag{3.4}$$

and the optimal value function $J^* : \mathcal{X}_N \to \mathbb{R}$ is continuous, convex and piecewise quadratic.

Definition 3.2.2 (Region) *Each polyhedron $\mathcal{P}_r$ of the polyhedral partition $\{\mathcal{P}_r\}_{r=1}^R$ is referred to as a region.*

For some mp-QP problem, the region partition $\{\mathcal{P}_r\}_{r=1}^R$ and PWQ value function $J^*(x)$ is depicted in Figures 3.1(a) and 3.2(a), respectively. Note that the evaluation of the PWA solution (3.4) of the mp-QP provides the same result as solving the quadratic program, i.e. for any given parameter x, the optimizer $U_N^*(x)$ in (3.4) is identical to the optimizer obtained by solving the quadratic program (3.2) for x.

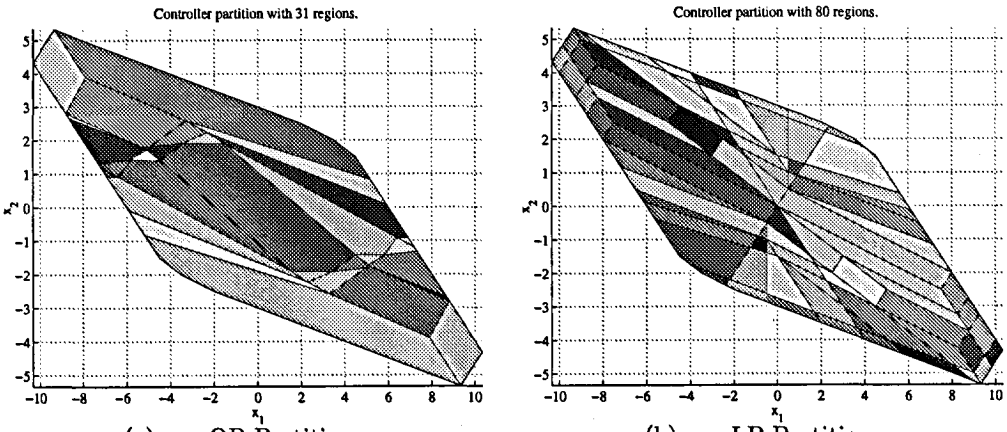
Problem (3.1) with an objective (3.1a) that is linear in the optimizer U_N can be stated as an mp-LP [BBM00b]. The properties of mp-LP solutions are stated below.

Theorem 3.2.3 (Properties mp-LP, [Bor03, Gal95]) *Consider the the optimization problem (3.1), with a linear objective $V(x, U_N) = x^T c_1^T U_N + c_2^T U_N$. Then, the set of feasible parameters $\mathcal{X}_N$ is convex, there exists an optimizer $U_N^* : \mathcal{X}_N \to \mathbb{R}^{Nm}$ which is continuous and piecewise affine (PWA), i.e.*

$$U_N^*(x) = F_r x + G_r, \quad \text{if} \quad x \in \mathcal{P}_r = \{x \in \mathbb{R}^n | H_r x \leq K_r\}, \; r = 1, \ldots, R,$$

and the value function $J_N^* : \mathcal{X}_N \to \mathbb{R}$ *is continuous, convex and piecewise affine.*

For some mp-LP problem, the region partition $\{\mathcal{P}_r\}_{r=1}^R$ and PWA value function $J^*(x)$ is depicted in Figures 3.1(b) and 3.2(b), respectively.



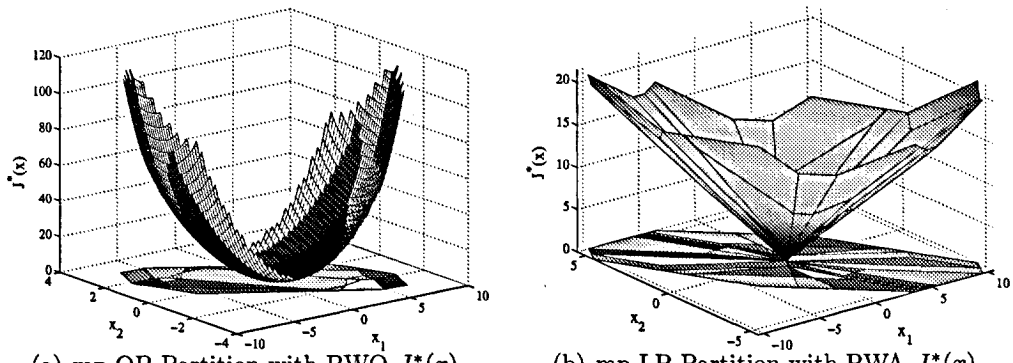
(a) mp-QP Partition. (b) mp-LP Partition.

Figure 3.1: Partition $\{\mathcal{P}_r\}_{r=1}^R$ for an mp-LP and an mp-QP problem. The constraints (3.1b) are identical for both problems. Therefore $\mathcal{X}_N$ is also identical for both problems.

Remark 3.2.4 *Assume that the origin is contained in the interior of the constraint polytope* (3.1b) *in* x-U_N *space. Because the value function for mp-QPs is PWQ, the origin is always contained in the interior of a single region. Specifically, the origin is always contained in the unconstrained region, i.e. the set of active constraints* $\mathcal{A}^N(x) = \emptyset$ *for* $x = 0$. *See Figure 3.1.*

Remark 3.2.5 *In the authors' experience, mp-LP solutions to control problems (see next chapter) generally comprise more regions than mp-QP problems subject to the same constraints (e.g., Figure 3.1). This may be because mp-LP problems in control are subject to more constraints and have optimizers of higher dimensions than their mp-QP counterparts (see Chapter 9). In addition mp-QPs are less susceptible to numerical problems since dual degeneracies cannot occur [Bor03].*

A brief outline of a generic mp-QP algorithm will be given next. For a detailed discussion of mp-QP algorithms we refer the reader to the literature [BMDP02, TJB03b, Bao02]. An mp-QP computation scheme consist of the following three steps:



(a) mp-QP Partition with PWQ $J^*(x)$. (b) mp-LP Partition with PWA $J^*(x)$.

Figure 3.2: Partition $\{\mathcal{P}_r\}_{r=1}^R$ and value function $J^*(x)$ for an mp-LP and an mp-QP problem.

1. **Active Constraint Identification:** A feasible parameter $\hat{x}$ is determined and the associated QP (3.2) is solved. This will yield the optimizer $U_N^*(\hat{x})$ and active constraints $\mathcal{A}^N(\hat{x})$ (see Definition 3.1.4).

2. **Region Computation:** The rows indexed by the active constraints $\mathcal{A}^N(\hat{x})$ are extracted from the constraint matrices G, W and S in (3.2b) to form the matrices $G_{\mathcal{A}_r^N}, W_{\mathcal{A}_r^N}$ and $S_{\mathcal{A}_r^N}$. The matrix S is derived from (3.2) by setting $S = E + GH^{-1}F^T$. Next, it is possible to use the Karush-Kuhn-Tucker [BV04] conditions to obtain an explicit representation of the optimizer $U_N^*(x)$ which is valid in some neighborhood of $\hat{x}$. Specifically,the optimizer $U_N^*(x)$ is defined by $U_N^* = F_r x + G_r$ (see (3.4)). If the LICQ holds (see Definition 3.1.5), then

$$F_r = H^{-1}G_{\mathcal{A}_r^N}^T(G_{\mathcal{A}_r^N}H^{-1}G_{\mathcal{A}_r^N}^T)^{-1}S_{\mathcal{A}_r^N} - H^{-1}F^T, \tag{3.5a}$$

$$G_r = H^{-1}G_{\mathcal{A}_r^N}^T(G_{\mathcal{A}_r^N}H^{-1}G_{\mathcal{A}_r^N}^T)^{-1}W_{\mathcal{A}_r^N}. \tag{3.5b}$$

For a discussion of how to deal with degenerate cases where the LICQ does not hold, we refer the reader to [Bor03, TJB03b].

In a next step, the set of states is determined where the optimizer $U_N^*(x)$ satisfies the constraints (3.2b) and is optimal. Specifically, the controller region $\mathcal{P}_r = \{x \in \mathbb{R}^n | H_r x \leq K_r\}$ is computed as in [BMDP02]

$$H_r = \begin{bmatrix} G(F_r + H^{-1}F^T) - S \\ (G_{\mathcal{A}_r^N}H^{-1}G_{\mathcal{A}_r^N}^T)^{-1}S_{\mathcal{A}_r^N} \end{bmatrix}, \tag{3.6a}$$

$$K_r = \begin{bmatrix} W - GG_r \\ -(G_{\mathcal{A}_r^N} H^{-1} G_{\mathcal{A}_r^N}^T)^{-1} W_{\mathcal{A}_r^N} \end{bmatrix}. \tag{3.6b}$$

3. **State Space Exploration:** Once the controller region is computed, the algorithm proceeds iteratively until the entire feasible state space $\mathcal{X}_N$ is covered with controller regions $\mathcal{P}_r$, i.e. $\mathcal{X}_N = \bigcup_{r=1,\dots,R} \mathcal{P}_r$.

Remark 3.2.6 *The number of rows in H_r is equal to the number of initial constraints* (3.1b), *i.e. H_r consists of q rows if $W \in \mathbb{R}^q$. Therefore, in order to obtain a non-redundant representation of $\mathcal{P}_r$, it is necessary to solve q LPs (see Chapter 2) for each region $r \in \{1, \dots, R\}$. In most cases one can increase the computational efficiency of multi-parametric solvers by computing the non-redundant representation of the original constraint polytope* (3.1b) *before solving the multi-parametric program.*

4

Optimal Control for Linear Time-Invariant Systems

Consider optimal control problems for discrete-time linear, time-invariant (LTI) systems

$$x(k+1) = Ax(k) + Bu(k), \tag{4.1}$$

with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times n}$. Let $x(k)$ denote the measured state at time k and x_k denote the predicted state at k steps ahead, given the state $x(0)$. Let u_k be the predicted input k steps ahead, given $x(0)$. In this chapter we will give a brief overview of optimal control problems for LTI systems discussed in the literature. The focus will be on quadratic optimization objectives. For an overview, we refer the reader to [LS95].

4.1 Unconstrained Finite-Time Optimal Control

Consider the unconstrained finite-time optimal control problem for system (4.1,

$$J_N^*(x(0)) = \min_{u_0,\ldots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_N^T Q_{x_N} x_N \right\}, \tag{4.2a}$$

$$Q_x \succeq 0, \quad Q_{x_N} \succeq 0, \quad Q_u \succ 0. \tag{4.2b}$$

The solution to (4.2) can be expressed by the optimal state-feedback control law [AM71, Ber95]

$$u_k^* = K_k x_k \quad k = 0, \ldots, N-1,$$

where the gain matrices K_k are given by the equation

$$K_k = -(B^T P_{k+1} B + Q_u)^{-1} B^T P_{k+1} A,$$

and where the symmetric positive semi-definite matrices P_k are given recursively by the algorithm

$$\begin{aligned} P_N &= Q_{x_N}, \\ P_k &= A^T(P_{k+1} - P_{k+1}B(B^TP_{k+1}B + Q_u)^{-1}BP_{k+1})A + Q_x. \end{aligned}$$

The optimal cost is

$$J_N^*(x(0)) = x(0)^TP_0x(0).$$

4.2 Unconstrained Infinite-Time Optimal Control Problem

If in (4.2) we set $N \to \infty$, we obtain the following optimal control problem

$$J_\infty^*(x(0)) = \min_{u_0,u_1,\dots} \left\{ \sum_{k=0}^{\infty} u_k^TQ_uu_k + x_k^TQ_xx_k \right\},$$
$$Q_x \succeq 0, \quad Q_u \succ 0.$$

If we assume (A, B) stabilizable, we can express the solution via the state feedback control law [AM71, Ber95]

$$u_k^* = Kx_k, \quad k = 0, \dots, +\infty,$$

where the gain matrix K is given by

$$K = -(B^TPB + Q_u)^{-1}B^TPA, \tag{4.3}$$

and P is the unique positive semidefinite symmetric solution of the Algebraic Riccati Equation (ARE)

$$P = A^T(P - PB(B^TPB + Q_u)^{-1}BP)A + Q_x. \tag{4.4}$$

The optimal cost is

$$J_\infty^*(x(0)) = x(0)^TPx(0). \tag{4.5}$$

The control law K in (4.3) is often simply referred to as LQR controller. This abbreviation is used since the dynamic system is **L**inear, the value function $J_\infty^*(x)$ is **Q**uadratic and the control objective is to **R**egulate the state to the origin.

Remark 4.2.1 *In order for the closed loop system to be asymptotically stable and the state to be driven to the origin, it is sufficient to select the weight matrix $Q_x \succ 0$. Alternatively, setting $Q_x = C^TC$ with (C, A) detectable is also sufficient.*

4.3 Constrained Finite-Time Optimal Control

Assume now that the states and the inputs of system (4.1) are subject to the following constraints

$$x(k) \in \mathbb{X} \subseteq \mathbb{R}^n, \qquad u(k) \in \mathbb{U} \subseteq \mathbb{R}^m, \qquad k \in \{0, \dots, N\}, \tag{4.6}$$

where $\mathbb{X}$ and $\mathbb{U}$ are polyhedral sets containing the origin in their interior[1]. Now consider the constrained finite-time optimal control (CFTOC) problem

$$\begin{aligned} J_N^*(x(0)) \;=\; & \min_{u_0,\dots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_N^T Q_{x_N} x_N \right\} && \text{(4.7a)} \\ \text{subj. to} \quad & x_k \in \mathbb{X},\ u_{k1} \in \mathbb{U}, \qquad \forall k \in \{1, \dots, N-1\}, && \text{(4.7b)} \\ & x_N \in \mathcal{T}_{\text{set}}, && \text{(4.7c)} \\ & x_{k+1} \;=\; Ax_k + Bu_k, \quad x_0 = x(0), && \text{(4.7d)} \\ & Q_u \succ 0, \quad Q_x \succeq 0, \quad Q_{x_N} \succeq 0. && \text{(4.7e)} \end{aligned}$$

The terminal set constraint (4.7c) is an additional constraint which is often added to obtain certain properties (i.e. stability and constraint satisfaction; See Chapter 5.2 for details). Henceforth, we will assume the terminal weight matrix Q_{x_N} to be equal to the ARE matrix P given in (4.4). The solution to problem (4.7) has been studied in [BMDP02]. We will briefly summarize the main results. By substituting $x_k = A^k x(0) + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$, problem (4.7) can be reformulated as a quadratic program (QP), i.e.

$$\begin{aligned} J_N^*(x(0)) \;=\; & x(0)^T Y x(0) + \min_{U_N} \left\{ U_N^T H U_N + x(0)^T F U_N \right\} && \text{(4.8a)} \\ \text{subj. to} \quad & GU_N \leq W + Ex(0), && \text{(4.8b)} \\ & H \succ 0, && \text{(4.8c)} \end{aligned}$$

where the column vector $U_N \triangleq [u_0^T, \dots, u_{N-1}^T]^T \in \mathbb{R}^{Nm}$ is the optimization vector and H, F, Y, G, W, E are easily obtained from Q_x, Q_u, Q_{x_N}, the system (4.1) and the constraints (4.6) (see [Mac02] for details[2]).

Remark 4.3.1 *The constraints $Q_u \succ 0$, $Q_x \succeq 0$ and $Q_{x_N} \succeq 0$ are imposed in* (4.7), *in order to guarantee that $H \succ 0$ in* (4.8).

[1] The extension to mixed constraints $C^x x + C^u u \leq C^o$ is straightforward and omitted here.

[2] For example, $Y = (A^N)^T Q_{x_N} A^N + \sum_{k=0}^{N-1} (A^k)^T Q_x A^k$.

The optimizer of (4.8) will henceforth be denoted by $U_N^*(x)$. It follows from Theorem 3.2.1 that $U_N^*(x)$ is a PWA function of the state x, which we can obtain by solving problem 4.8 as an mp-QP (see Chapter 3 for details).

4.4 Constrained Infinite-Time Optimal Control

If in (4.7) we set $N \to \infty$ we obtain the constrained infinite-time optimal control (CITOC) problem:

$$J_\infty^*(x(0)) = \min_{u_0,u_1,\dots} \left\{ \sum_{k=0}^{\infty} u_k^T Q_u u_k + x_k^T Q_x x_k \right\} \tag{4.9a}$$

$$\text{subj. to} \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}, \qquad \forall k \geq 0, \tag{4.9b}$$

$$x_{k+1} = Ax_k + Bu_k, \qquad x_0 = x(0), \tag{4.9c}$$

$$Q_x \succeq 0,\ \ Q_u \succ 0. \tag{4.9d}$$

where the infinite dimensional vector $U_\infty \triangleq [u_0^T, u_1^T, \dots]$ is the optimization vector. We denote by U_∞^* the optimizer of (4.9). The computation of the CLQR will be covered in detail in Section 10.3. We will merely restate a fundamental theorem here:

Definition 4.4.1 (Maximal LQR Invariant Set $\mathcal{O}_\infty^{\text{LQR}}$) *For an LTI system (4.1) subject to the LQR control input $u = Kx$ (4.3), the set $\mathcal{O}_\infty^{LQR} \subseteq \mathbb{R}^n$ denotes the maximum invariant set of states which satisfies the constraints in (4.6) for all time, i.e.,*

$$\mathcal{O}_\infty^{LQR} = \{x(0) \in \mathbb{R}^n | x(k) \in \mathbb{X},\ Kx(k) \in \mathbb{U},\ x(k+1) = (A+BK)x(k),\ \forall k \geq 0\}. \tag{4.10}$$

The set $\mathcal{O}_\infty^{\text{LQR}}$ is positive invariant containing an open neighborhood of the origin [SD87], provided the origin is contained in the interior of the set described by (4.6)[3]. The following theorem (derived from [CM96,SR98]) summarizes the key point of this section:

Theorem 4.4.2 (Finite Dimensional Infinite Horizon Optimal Control) *For any given initial state $x(0)$, the solution to (4.7) with $\mathcal{T}_{set} = \mathbb{X}$ and the ARE terminal weight $Q_{x_N} = P$ (see (4.4)), is equal to the initial segment of the infinite-time*

[3] If the origin is not contained in (4.6), no solution to (4.9) exists, since $J_\infty^*(x(0))$ is infinite.

solution (4.9) if the terminal state x_N of (4.7) lies in the maximal LQR invariant set $\mathcal{O}_\infty^{LQR}$ ($x_N \in \mathcal{O}_\infty^{LQR}$).

Seite Leer /
Blank leaf

5

Receding Horizon Control

In the previous chapter it was shown how to solve constrained optimal control problems for LTI systems. Since it is generally desirable to have feedback control for all time as well as optimal performance, the infinite-time controllers would seem to be the preferred choice. However, since the infinite-time optimal control problem is often too complex to be computationally tractable, it has become common practice to approximate the infinite-time solution by solving a sequence of finite time optimal control problems. This strategy is referred to as *Receding Horizon Control* (RHC) and is the focus of this chapter. For a more detailed discussion of RHC, we refer the reader to the review paper [MRRS00]. For in-depth insights, we recommend the publications [Mac02, Löf03].

5.1 State Feedback Control of Constrained Dynamical Systems

Clearly, the most widely applied method of optimal feedback control for dynamical systems is *Receding Horizon Control* (RHC). The RHC policy has become standard practice in modern control applications and besides numerous PhD theses [Mig02, Bor03, Löf03, Ker00, Tøn00, Sak04] and survey papers [QB97, MRRS00, BM99b, ABQ+99, GPM89, May01, ML99, Raw00], several textbooks [Mac02, Ros03, CB99, KC01] have been published.

In RHC, a finite-time optimal control problem is solved at each time step to obtain the optimal input sequence U_N^*. Subsequently, only the first element of that sequence is applied to the system. At the next time step, the state is measured and the procedure is repeated from the beginning. The input sequence can be computed by solving an optimization problem (e.g. (4.8)) on-line at each time step. Alternatively,

it is possible to solve the optimization problem off-line as a multi-parametric program. Then, the on-line effort reduces to finding the correct feedback law entry in a lookup table. The RHC scheme is depicted in Figure 5.1.

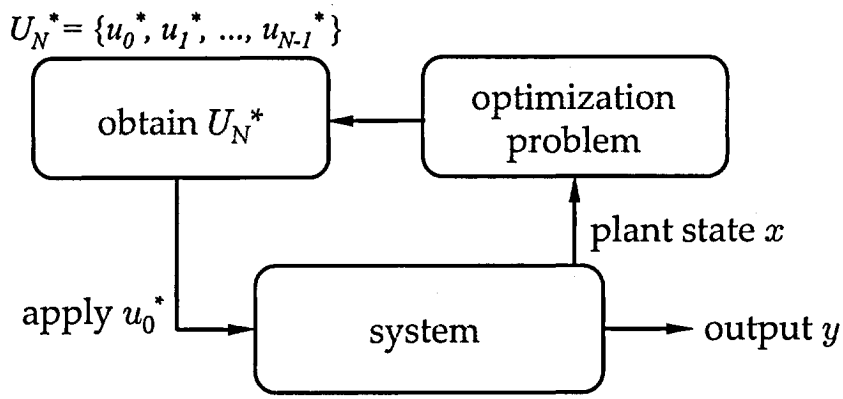


Figure 5.1: Illustration of the Receding Horizon Control (RHC) scheme.

RHC obviously provides a control law for all time. However, since only finite-time optimal control problems are being solved, constraint satisfaction cannot be guaranteed for all time, i.e. it is possible that the RHC policy will drive the state outside of the feasible set $\mathcal{X}_N$. In addition, RHC does not guarantee stability, unless a special structure is imposed on the optimization problem. The next section will show how to formulate a finite time optimal control problem, such that the associated RHC guarantees stability and feasibility.

5.2 Stability and Feasibility of Receding Horizon Control

The concepts of feasibility (constraint satisfaction) and stability of RHC are often misunderstood and some basic facts are often stated in an ambiguous manner. This section will attempt to make the distinction clear. We will first address the issue of constraint satisfaction.

Definition 5.2.1 (Feasibility) *A system state x is feasible for an optimization problem of type* (3.1)*, if there exists an input sequence U_N which satisfies the constraints* (3.1b).

Hence, an optimization problem of type (3.1) is feasible if and only if $x \in \mathcal{X}_N$.

Definition 5.2.2 (Infinite-Time Feasibility) *A system state $x(0)$ subject to receding horizon control is infinite-time feasible, if feasibility of the optimization problem (3.1) for $x(0)$ implies feasibility of the optimization problem for all time, i.e. (3.1) is feasible for all $x(k)$, $k \geq 0$.*

Hence, a RHC controller for systems of type (4.1) (subject to constraints (4.6)) can only be infinite-time feasible if (and only if) $x(0)$ is contained in any control invariant subset $\mathcal{S}_{\text{inv}}$ of the feasible set $\mathcal{X}_N$,

$$\mathcal{S}_{\text{inv}} = \{x(0) \in \mathcal{X}_N \mid \forall x(0) \in \mathcal{S}_{\text{inv}},\ \exists u(0) \in \mathbb{U}, \text{s.t. } Ax(0) + Bu(0) \in \mathcal{S}_{\text{inv}}\}.$$

Note that invariance of the subset $\mathcal{S}_{\text{inv}}$ may or may not be enforced by RHC, i.e. $x(0) \in \mathcal{S}_{\text{inv}}$ only implies infinite-time feasibility of RHC, if additional measures are taken when formulating the control problem (4.7). For instance, the additional constraint $Ax(0) + Bu(0) \in \mathcal{S}_{\text{inv}}$ in (4.7), would trivially imply infinite-time feasibility. However, the most common approach to guarantee infinite-time feasibility is to add the constraint $x_N \in \mathcal{O}_\infty^{\text{LQR}}$ (see Definition 4.4.1) to the problem formulation (4.7), i.e. $\mathcal{T}_{\text{set}} = \mathcal{O}_\infty^{\text{LQR}}$. If a feasible sequence $U_N = [u_0^T, \ldots, u_{N-1}^T]^T$ is obtained at time k, then it follows from the terminal set constraint $x_N \in \mathcal{O}_\infty^{\text{LQR}}$, that $U_N = [u_1^T, \ldots, u_{N-1}^T, Kx_N]^T$ is a feasible sequence[1] at time $k+1$. Therefore feasibility is guaranteed for all time.

Remark 5.2.3 *In order to guarantee infinite-time feasibility, it is not necessary for the terminal set $\mathcal{T}_{set}$ to be equal to the set $\mathcal{O}_\infty^{LQR}$ in Definition 4.4.1. If $\mathcal{T}_{set}$ corresponds to any control invariant set (see Chapter 7), then infinite-time feasibility is guaranteed.*

Remark 5.2.4 *The terminal set $\mathcal{T}_{set}$ (4.7c) has a significant impact on the size of the feasible set $\mathcal{X}_N$. If $\mathcal{T}_{set}$ is control invariant (e.g., $\mathcal{T}_{set} = \mathcal{O}_\infty^{LQR}$), then $\mathcal{X}_N \subseteq \mathcal{X}_{N+1}$. On the other hand, if $\mathcal{T}_{set} = \mathbb{X}$, then $\mathcal{X}_N \supseteq \mathcal{X}_{N+1}$. The impact of $\mathcal{T}_{set}$ on $\mathcal{X}_N$ has been investigated by numerous authors (e.g., [BCLK03, CKD03, WK03a]).*

Note that infinite-time feasibility *does not* imply exponential stability. There is no guarantee that the state will ever enter the terminal set $\mathcal{O}_\infty^{\text{LQR}}$ if RHC is applied, since the input is recomputed at each time step. The following Theorem is based on [Vid93, p. 267]:

[1] Here, K is used to denote the Riccati LQR feedback law (4.3).

Theorem 5.2.5 (Exponential Stability) *The origin $x = 0$ is exponentially stable if there exists a function $V(x)$ and scalar coefficients $\alpha > 0$, $\beta > 0$, $\rho > 0$ and $p > 1$ such that $\beta\|x\|^p \geq V(x) \geq \alpha\|x\|^p$ and $V(x^+) - V(x) \leq -\rho\|x\|^p$, $\forall x \in \mathcal{X}_N$. Here x^+ denotes the successor state of the dynamical system and $\|\cdot\|$ denotes a vector norm.*

There are several methods for enforcing exponential stability of RHC by modifying the open-loop optimal control problem (4.7). The most commonly used scheme to guarantee exponential stability of RHC, is to impose both an invariant terminal set constraint (e.g. $\mathcal{O}_\infty^{\text{LQR}}$) as well as a terminal cost $x_N^T P x_N$ which corresponds to a local exponential Lyapunov function, i.e. the decay rate of the Lyapunov function must be bounded by the stage cost (see (5.2)). This approach was motivated by results in [SD87] and [KG88]. It follows from (4.5) that if $x \in \mathcal{O}_\infty^{\text{LQR}}$ and the input $u = Kx$ (4.3) is applied to system (4.1), then

$$x_N^T P x_N = x_N^T Q_x x_N + u_N^T Q_u u_N + x_{N+1}^T P x_{N+1}. \tag{5.1}$$

For $Q_x \succ 0$, it directly follows from (4.7a) and (5.1) that $\exists \rho > 0$ such that

$$J_N^*(x_1) - J_N^*(x_0) = \left(\sum_{k=1}^{N} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_{N+1}^T P x_{N+1} \right) \tag{5.2a}$$

$$- \left(\sum_{k=0}^{N-1} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_N^T P x_N \right) \tag{5.2b}$$

$$= \underbrace{-x_0^T Q_x x_0 - u_0^T Q_u u_0}_{\leq -\rho\|x_0\|_2^2} + \underbrace{x_{N+1}^T P x_{N+1} + x_N^T Q_x x_N + u_N^T Q_u u_N - x_N^T P x_N}_{=0}. \tag{5.2c}$$

Therefore, if the terminal cost is chosen as the solution of the ARE (4.4) and the terminal set constraint $x_N \in \mathcal{O}_\infty^{\text{LQR}}$ is added to (4.7), the function $J_N^*(x)$ is a Lyapunov function according to Theorem 5.2.5 and the closed-loop system is exponentially stable.

Remark 5.2.6 *In this section we assume that RHC is applied for quadratic performance objectives. If the control objective is linear, asymptotic stability of RHC can be guaranteed by selecting the terminal weight matrix P such that*

$$-\|Px\|_p + \|PAx\|_p + \|Qx\|_p \leq 0.$$

Here, the subindex p denotes a linear norm (e.g. $p = 1$ or $p = \infty$) and P must be of full column rank. Details on the theoretical background and computation of P for

linear RHC are given in [Bor03, Section 3.2.3].

Generic conditions on the terminal set constraint $x_N \in \mathcal{T}_{\text{set}}$, the feedback law $\kappa(x)$ (for $x \in \mathcal{T}_{\text{set}}$) and the terminal cost $V(x_N)$ which guarantee exponential stability of RHC for general dynamical systems are given in [MRRS00] and will be restated here:

A1) Constraint Satisfaction: $\mathcal{T}_{\text{set}} \subseteq \mathbb{X}$, $\kappa(x) \in \mathbb{U}, \forall x \in \mathcal{T}_{\text{set}}$.

A2) Invariance: $x \in \mathcal{T}_{\text{set}} \Rightarrow x^+ \in \mathcal{T}_{\text{set}}$.

A3) Stability: $\exists \rho > 0$ such that $V(x^+) - V(x) \leq -\ell(x,u)$, where $\ell(\cdot)$ denotes the stage cost (here $\ell(x,u) = x^T Q_x x + u^T Q_u u$).

The second most widely used approach to guarantee stability is based on *contraction constraints*. These approaches are based on results in [PY93,dM00,Bla93,Bla94, Bla95] and add a constraint to the open-loop problem (4.7) which enforces that the state decreases in some norm (e.g., $\|x_{k+1}\| \leq \|x_k\|$). If the constraint is chosen appropriately, exponential stability can be guaranteed. However, a contraction constraint does not guarantee infinite-time feasibility.

Note that the conditions which guarantee exponential stability are merely sufficient, i.e., the closed-loop RHC system may be exponentially stable without satisfying any of the previously mentioned constraints.

It should be noted that both terminal set- and contraction-based approaches rely on 'artificial' (user-defined) constraints in order to provide stability guarantees. The added constraints are *not* system inherent. Since artificial constraints are added, the controllable set of states $\mathcal{X}_N$ is generally only a subset of the maximum controllable set of states. Techniques of avoiding this problem are discussed in Section 10.3, Chapter 11 and Chapter 16.

Example 5.2.7 *Consider the double integrator*

$$x(k+1) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u(k).$$

The task is to regulate the system to the origin while fulfilling the input and state constraints

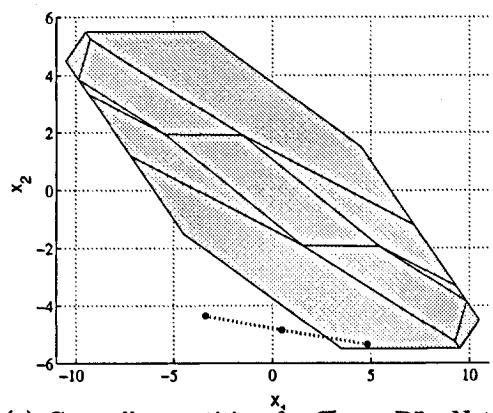
$$-1 \leq u(k) \leq 1, \qquad \forall k \geq 0,$$
$$\begin{pmatrix} -5 \\ -5 \end{pmatrix} \leq x(k) \leq \begin{pmatrix} 5 \\ 5 \end{pmatrix}, \ \forall k \geq 1.$$

We consider the quadratic optimization problem (4.7) *for a prediction horizon* $N = 2$. *The weight matrices are set to* $Q_x = I$ *and* $Q_u = 1$ *and* P *follows from the ARE* (4.4).

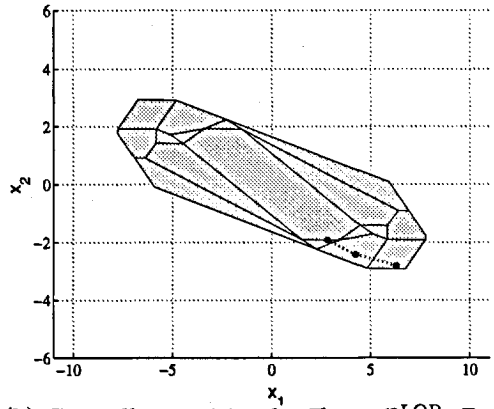
Two open-loop trajectories obtained for $\mathcal{T}_{\text{set}} = \mathbb{X}$ and $\mathcal{T}_{\text{set}} = \mathcal{O}_{\infty}^{\text{LQR}}$ are shown in Figure 5.2.

It can be seen that the solution for $\mathcal{T}_{\text{set}} = \mathbb{X}$ may yield input sequences which lead to infeasibility ($x_1 \notin \mathcal{X}_N$, see Figure 5.2(a)). In Figure 5.3(a), all initial states which are infinite-time feasible are depicted, i.e. we have removed all states from Figure 5.2(a), whose closed-loop trajectory exits $\mathcal{X}_N$. One can also see that most feasible states depicted in Figure 5.2(a) (i.e. all states in the partition) are also infinite-time feasible. Although this is often the case for $\mathcal{T}_{\text{set}} = \mathbb{X}$, no a priori guarantees can be given.

If we use the terminal set constraint $\mathcal{T}_{\text{set}} = \mathcal{O}_{\infty}^{\text{LQR}}$ as in Figure 5.2(b), infinite-time feasibility is guaranteed. However, the feasible set of states $\mathcal{X}_N$ is relatively small for $N = 2$. If we extract all infinite-time feasible states from Figure 5.2(b), we obviously obtain the same set again, as is depicted in Figure 5.3(b).

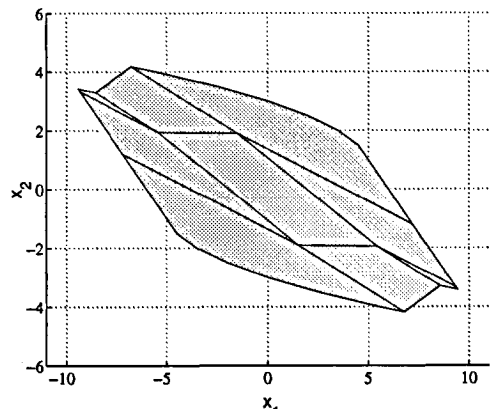


(a) Controller partition for $\mathcal{T}_{\text{set}} = \mathbb{R}^n$. Not all states in the partition are infinite-time feasible.

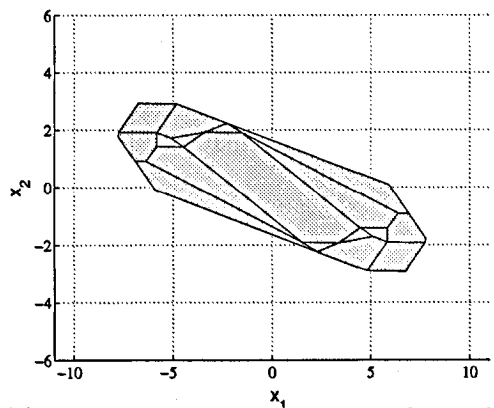


(b) Controller partition for $\mathcal{T}_{\text{set}} = \mathcal{O}_{\infty}^{\text{LQR}}$. By construction, all states in the partition are infinite-time feasible.

Figure 5.2: Various system trajectories if the optimal input sequence $U_N^*(x(0)) = [u_0 \;\; u_1]$ is applied in open-loop. The input sequences were obtained for Example 5.2.7 and $N = 2$ using the terminal set constraints $\mathcal{T}_{\text{set}} = \mathbb{R}^n$ and $\mathcal{T}_{\text{set}} = \mathcal{O}_{\infty}^{\text{LQR}}$, respectively.



(a) Invariant subset of the partition depicted in Figure 5.2(a), which was obtained for $\mathcal{T}_{\text{set}} = \mathbb{X}$. All states in the partition are infinite-time feasible.



(b) Invariant subset of the partition depicted in Figure 5.2(b), which was obtained for $\mathcal{T}_{\text{set}} = \mathcal{O}_{\infty}^{\text{LQR}}$. By construction, the invariant subset is identical to the original partition depicted in Figure 5.2(b).

Figure 5.3: The maximal invariant subsets of the controller partitions in Figures 5.2(a) and 5.2(b), i.e. all states in the depicted controller partition are infinite-time feasible.

Seite Leer /
Blank leaf

Part II

ANALYSIS OF PWA SYSTEMS

Seite Leer /
Blank leaf

6

Problem Description

Piecewise affine (PWA) systems have attracted much interest in the research community since they provide a useful modelling framework for a large class of hybrid systems. Discrete-time PWA systems are equivalent to interconnections of linear systems and finite automata [Son96], to linear complementary systems [HSB01] and also hybrid systems in the mixed logical dynamical (MLD) form [BM99a]. The MLD form encompasses a large class of hybrid systems including linear hybrid dynamical systems, hybrid automata and some classes of discrete event systems. Software for MLD modelling is available from [TB04] and an algorithm to transform an MLD system into a discrete-time PWA system is given in [BFTM00, Bem04]. PWA systems are also a powerful tool for approximating non-linear systems [Son81]. Furthermore, LTI systems subject to linear or quadratic optimal control correspond to a PWA system (see Remark 6.1.1 below).

The following two chapters will deal with the analysis of PWA systems. Chapter 7 presents algorithms for computing invariant sets for PWA systems and Chapter 8 introduces various methods for computing Lyapunov functions for PWA systems. The analysis schemes introduced here will be used in the context of controller synthesis in subsequent chapters.

In the following, we will deal with two different types of PWA systems, namely, autonomous PWA systems and PWA systems subject to external inputs. We will assume that both systems are also subjected to an additive disturbance $w(k)$ as well as the constraints

$$x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad w(k) \in \mathbb{W}, \quad \forall\, k \geq 0. \tag{6.1}$$

The sets $\mathbb{X}$, $\mathbb{U}$ and $\mathbb{W}$ are compact and polytopic and contain the origin in their interior.

An autonomous PWA system subject to an additive disturbance $w(k)$ can be described by

$$x(k+1) = f_a(x(k), w(k)) = \tilde{A}_r x(k) + \tilde{g}_r + w(k), \qquad \text{if } x(k) \in \mathcal{P}_r,\ r \in \mathcal{R}, \quad (6.2)$$

where the currently active dynamic r is defined by the polyhedron $\mathcal{P}_r$ and the index set $\mathcal{R} \triangleq \{1, 2, \ldots, R\}$, where R denotes the number of different dynamics. We will denote the set of states over which the PWA system (6.2) is defined as $\mathcal{S}_{\text{PWA}} = \bigcup_{r \in \mathcal{R}} \mathcal{P}_r$, where $\mathcal{S}_{\text{PWA}}$ is a P-collection (see Chapter 2).

Remark 6.1.1 *Note that the autonomous PWA dynamics* (6.2) *can result from a constrained LTI system* (4.1) *subject to linear or quadratic optimal control* ($u(k) = F_r x(k) + G_r$). *For these systems* $\tilde{A}_r \triangleq A + BF_r$, $\tilde{g}_r \triangleq BG_r$ *and* $\mathcal{S}_{PWA} = \mathcal{X}_N$ *as defined in Theorems 3.2.1 and 3.2.3.*

In addition to the autonomous PWA system (6.2), we will consider PWA systems subject to the input $u(k)$ and the disturbance $w(k)$,

$$\begin{aligned} x(k+1) &= f_{\text{PWA}}(x(k), u(k), w(k)) && (6.3a)\\ &= A_r x(k) + B_r u(k) + g_r + w(k), && (6.3b)\\ &\quad \text{if } [x(k)^T \ u(k)^T]^T \in \mathcal{P}_r, \quad r \in \mathcal{R}, && (6.3c) \end{aligned}$$

whereby the dynamics (6.3b) are valid in the polyhedral set $\mathcal{P}_r$ defined in (6.3c). With slight abuse of notation, we will use $f_{\text{PWA}}(x(k), u(k), \mathbb{W})$ to denote the set of states which is reachable for any $w(k) \in \mathbb{W}$, i.e.

$$f_{\text{PWA}}(x(k), u(k), \mathbb{W}) \triangleq \{A_r x(k) + B_r u(k) + g_r + w(k) \in \mathbb{R}^n \mid w(k) \in \mathbb{W}, \\ [x(k)^T \ u(k)^T] \in \mathcal{P}_r\}.$$

Standing assumption for both PWA system classes is that $\mathbb{X} \subseteq \bigcup_{r \in \mathcal{R}} \mathcal{P}_r$. We furthermore assume that the interiors $\text{int}(\mathcal{P}_r)$ of the partition $\{\mathcal{P}_r\}_{r=1}^R$ are disjoint.

7

Computation of Invariant Sets for Piecewise Affine Systems

This chapter[1] will deal with the computation of invariant sets for PWA systems of type (6.2) and (6.3) subject to bounded disturbances and is a summary of [RGK+04a, RG04a, GRMM05]. We refer the interested reader to [Ker00] for an excellent overview of set-invariance in control.

As stated in Section 5.2, set invariance implies infinite-time feasibility of RHC. Therefore invariant sets are of great importance when dealing with control of constrained systems. Although computation of invariant sets has garnered great interest in the control community [GT91, Bla99, KG98, Ker00, Aub91, Bit88], only few results for obtaining invariant sets for PWA systems have been published, e.g. [ALQ+02]. This is especially true for PWA systems subject to bounded disturbances. The results in this chapter are based on the results for linear systems in [GT91, KG98, MS97, GM03] as well as recent extensions to PWA systems in [KM02, GKBM04a]. Additional references include [RKKM03, RKM03].

In this chapter, an algorithm for computing the maximal robust positive invariant set $\mathcal{O}_\infty$ is described and sufficient conditions for finite-time termination of this algorithm are given. It will subsequently be shown how the set $\mathcal{O}_\infty$ can be used to initialize an iterative computation scheme which converges to the maximal robust stabilizable set $\mathcal{K}_\infty(\mathcal{O}_\infty)$. A similar scheme is applied to obtain the maximal robust control invariant set $\mathcal{C}_\infty$.

[1] Note that the content of this chapter is the result of a collaboration with Sasa Raković who was the primary contributor.

7.1 Definitions

We will first introduce some basic notation and definitions before defining the invariant sets that we wish to compute.

For any integer k, $\mathbf{w}_k$ denotes the sequence $\{w(0), w(1), \ldots, w(k-1)\}$, i.e. $\mathbf{w}_k \in \mathbb{W}^k$, and $\phi(k; x(0), \mathbf{w}_k)$ denotes the solution of $x(k+1) = f_a(x(k), w(k))$ at time k if the initial state is $x(0)$ and the disturbance sequence is $\mathbf{w}_k$. For the autonomous PWA system (6.2), we will denote the k-step reachable set for initial states x contained in the set $\mathcal{S}$ as

$$\text{Reach}(k; \mathcal{S}, \mathbb{W}) \triangleq \{\phi(k; x(0), \mathbf{w}_k) \in \mathbb{R}^n \mid x(0) \in \mathcal{S}, \mathbf{w}_k \in \mathbb{W}^k\}.$$

Furthermore,

$$\begin{aligned} \text{Pre}(\mathcal{S}, \mathbb{W}) \triangleq &\{x \in \mathbb{X} \mid \exists u \in \mathbb{U} \text{ s.t. } f_{\text{PWA}}(x, u, \mathbb{W}) \subseteq \mathcal{S}\} && \text{(7.1a)} \\ = &\{x \in \mathbb{X} \mid \exists u \in \mathbb{U} \text{ s.t. } f_{\text{PWA}}(x, u, 0) \subseteq \mathcal{S} \ominus \mathbb{W}\} && \text{(7.1b)} \end{aligned}$$

will define the set of states which can be robustly driven into the target set $\mathcal{S}$ in one time step for the PWA system (6.3).

Two different types of sets are being considered in this chapter: *invariant sets* and *control invariant sets*. We will first discuss invariant sets. The invariant sets are computed for an autonomous system which is not subject to external inputs. These types of sets are useful to answer questions such as: "For a *given* linear feedback controller K ($u = Kx$), find the set of states whose trajectory will never violate the system constraints". The following definitions, derived from [Ker00, Bla99, BR71, Ber71, KG98], introduce the different types of invariant sets.

Definition 7.1.1 (Robust Positive Invariant Set) *A set $\mathcal{O}$ is said to be a robust positive invariant set for the autonomous PWA system in* (6.2) *subject to the constraints in (6.1), if* $\text{Reach}(1; \mathcal{O}, \mathbb{W}) \subseteq \mathcal{O}$.

Definition 7.1.2 (Minimal Robust Positive Invariant Set $\mathcal{F}_\infty$) *The set $\mathcal{F}_\infty$ is the minimal robust invariant set*[2] *of the autonomous PWA system (6.2) subject to the constraints in (6.1), if $0 \in \mathcal{F}_\infty$, $\mathcal{F}_\infty$ is robust invariant and $\mathcal{F}_\infty$ is a subset of all robust invariant sets that contain the origin.*

[2] Also known as infinite-time disturbance response set.

Definition 7.1.3 (Maximal Robust Positive Invariant Set $\mathcal{O}_\infty$) *The set $\mathcal{O}_\infty$ is the maximal robust invariant set of the autonomous PWA system (6.2) subject to the constraints in (6.1), if $0 \in \mathcal{O}_\infty$, $\mathcal{O}_\infty$ is robust invariant and $\mathcal{O}_\infty$ contains all robust invariant sets that contain the origin.*

Remark 7.1.4 *The condition that $\mathcal{F}_\infty$ and $\mathcal{O}_\infty$ must contain the origin is added because PWA systems may have multiple equilibrium points and thus multiple robust invariant sets which are disconnected (i.e. $\mathcal{F}_\infty = \emptyset$). Furthermore, if these set are used as a target sets in control problems, they should only contain one equilibrium point in order to get predictable closed-loop behavior.*

Remark 7.1.5 *The maximal (robust) invariant sets defined here, are often referred to as 'maximal admissible sets' or 'maximal output admissible sets' in the literature (e.g. [GT91]), depending on whether the system state or output is constrained.*

We will now discuss control invariant sets. Control invariant sets are defined for systems subject to external inputs, i.e. for PWA systems as in (6.3). These types of sets are useful to answer questions such as: "Find the set of states for which *there exists* a controller such that the system constraints are never violated". The following definitions, derived from [Ker00, Bla99, BR71, Ber71, KG98], introduce the different types of control invariant sets.

Definition 7.1.6 (Robust Control Invariant Set) *A set $\mathcal{C} \subseteq \mathbb{X}$ is said to be a robust control invariant set for the PWA system in* (6.3) *subject to the constraints in (6.1), if for every $x(k) \in \mathcal{C}$ there exists a $u(k) \in \mathbb{U}$ such that $f_{PWA}(x(k), u(k), \mathbb{W}) \subseteq \mathcal{C}$.*

Definition 7.1.7 (Maximal Robust Control Invariant Set $\mathcal{C}_\infty$) *The set $\mathcal{C}_\infty$ is said to be the maximal robust control invariant set for the PWA system in* (6.3) *subject to the constraints in* (6.1), *if it is robust control invariant and contains all robust control invariant sets contained in $\mathbb{X}$.*

For all states contained in the maximal control invariant set $\mathcal{C}_\infty$ there exists a control law, such that the system constraints are never violated. This does not imply that there exists a control law which can drive the state into a user-specified target set. This issue is addressed in the following by introducing the concept of stabilizable sets.

Definition 7.1.8 (N-Step Robust Stabilizable Set $\mathcal{K}_N(\mathcal{O})$) *For a robust control invariant target set $\mathcal{O} \subseteq \mathbb{X}$, the N-step robust stabilizable set $\mathcal{K}_N(\mathcal{O})$ of the PWA system* (6.3) *subject to the constraints* (6.1) *is defined as:*

$$\mathcal{K}_N(\mathcal{O}) \triangleq \mathrm{Pre}(\mathcal{K}_{N-1}(\mathcal{O}), \mathbb{W}), \quad \mathcal{K}_0(\mathcal{O}) = \mathcal{O}, \qquad N \in \mathbb{N}_+.$$

Definition 7.1.9 (Maximal Robust Stabilizable Set $\mathcal{K}_\infty(\mathcal{O})$) *For a robust control invariant set $\mathcal{O} \subseteq \mathbb{X}$, the maximal robust stabilizable set $\mathcal{K}_\infty(\mathcal{O})$ for the PWA system* (6.3) *subject to the constraints in* (6.1) *is the union of all N-step robust stabilizable sets contained in $\mathbb{X}$ ($N \in \mathbb{N}$).*

The set $\mathcal{K}_\infty(\mathcal{O})$ contains all states which can be robustly steered into the robust control invariant set $\mathcal{O}$ and hence $\mathcal{K}_\infty(\mathcal{O}) \subseteq \mathcal{C}_\infty$.

7.2 The Maximal Robust Positive Invariant Set

We now address the computation of the maximal robust positive invariant set $\mathcal{O}_\infty$ for PWA systems around the origin, see Definition 7.1.3. Assume that the origin is an equilibrium of the nominal system $x(k+1) = f_a(x(k), 0)$, where $f_a(\cdot)$ is defined as in (6.2), and that $g_r = 0$ for all $r \in \mathcal{R}_0$, where the set of different system dynamics $\mathcal{R}_0 \subseteq \mathcal{R} \triangleq \{1, \ldots, R\}$, is such that

$$\mathcal{R}_0 \triangleq \{r \in \mathcal{R} \mid 0 \in \mathcal{P}_r\} \tag{7.2}$$

where 0 is the origin of the state space. We furthermore define

$$\mathbb{X}_0 \triangleq \left(\bigcup_{r \in \mathcal{R}_0} \mathcal{P}_r \right) \cap \mathbb{X}. \tag{7.3}$$

Remark 7.2.1 *Note that the assumptions above are made in order to obtain a proof for finite termination of the algorithm which is presented in this section. The subsequent computations may be applied even if these assumptions do not hold. However, there will not be any a priori guarantee of finite termination.*

For the autonomous PWA system (6.2), we use $\mathrm{Pre}_a(\mathcal{S})$ to denote the set of the states that robustly evolves to $\mathcal{S} \subseteq \mathbb{X}_0$ in one step:

$$\mathrm{Pre}_a(\mathcal{S}, \mathbb{W}) \triangleq \{x \in \mathbb{X}_0 \mid f_a(x, w) \in \mathcal{S}, \ \forall w \in \mathbb{W}\}. \tag{7.4}$$

The set $\mathrm{Pre}_a(\mathcal{S}, \mathbb{W})$ can be efficiently computed, as shown in the proof of the following lemma.

Lemma 7.2.2 *Let $\mathcal{S} \subseteq \mathbb{X}_0$ be a P-collection and let $f_a(x, w)$ be an autonomous PWA system* (6.2)*, then the set* $\mathrm{Pre}_a(\mathcal{S}, \mathbb{W})$*, defined in* (7.4)*, is a P-collection.*

Proof Since $\mathcal{S}$ is a P-collection by properties of the Pontryagin difference, $\mathcal{S}^* = \mathcal{S} \ominus \mathbb{W}$ is also a P-collection so that $\mathcal{S}^* = \bigcup_{y=1,\ldots,Y} \mathcal{S}_y^*$ for some finite integer Y. It trivially follows from the definition of $f_a(x, w)$ that $\mathrm{Pre}_a(\mathcal{S}, \mathbb{W}) = \bigcup_{(y,j)\in\{1,\ldots,Y\}\times\mathcal{R}_0} \mathcal{S}_{y,j}^+$, where $\mathcal{S}_{y,j}^+ \triangleq \{x \in \mathbb{X} \mid A_j x \in \mathcal{S}_y^*\}$. Since each $\mathcal{S}_{y,j}^+$ is polyhedral, the set $\mathrm{Pre}_a(\mathcal{S}, \mathbb{W})$ is a P-collection. □

The following algorithm provides a procedure for computing the maximal robust positive invariant subset of $\mathbb{X}_0$ [Aub91, Ber71, Ker00].

Algorithm 7.2.3 (Computation of $\mathcal{O}_\infty$)

1. $\Omega_0 = \mathbb{X}_0$
2. $\Omega_{k+1} = \mathrm{Pre}_a(\Omega_k, \mathbb{W})$
3. *If* $\Omega_{k+1} = \Omega_k$*, return; Else, set* $k = k + 1$ *and goto 2.*

The algorithm generates the set sequence $\{\Omega_k\}$ satisfying $\Omega_{k+1} \subseteq \Omega_k, \forall k \in \mathbb{N}$ and it terminates if $\Omega_{k+1} = \Omega_k$ so that Ω_k is the maximal robust positive invariant set $\mathcal{O}_\infty$. Otherwise $\mathcal{O}_\infty = \bigcap_{k\geq 0} \Omega_k$. If $\Omega_k = \emptyset$ for some integer k then the simple conclusion is that $\mathcal{O}_\infty = \emptyset$ [Aub91, Ber71, Ker00, JvdS02].

7.3 Finite Termination of the Computation of the Maximal Robust Positive Invariant Set

In this section, we isolate a set of conditions that are sufficient to guarantee finite-time termination of Algorithm 7.2.3. It is very difficult to derive this proof for PWA systems directly, such that we approximate the PWA system with a switched system, which we here refer to as 'augmented system'. PWA systems are a subclass of the

augmented systems considered here, such that all proofs derived for the augmented systems will also hold for PWA systems.

Our first step is to introduce the set-valued augmented system

$$x(k+1) \in f_a^{\text{aug}}(x(k), w(k)) \tag{7.5a}$$
$$\triangleq \{A^{\text{aug}}x(k) + w(k) \mid A^{\text{aug}} \in \{\tilde{A}_r,\ r \in \mathcal{R}_0\}\}. \tag{7.5b}$$

This augmented system $f_a^{\text{aug}}(x, w)$ corresponds to the autonomous PWA system (6.2) with $\mathcal{P}_i = \mathbb{R}^n$, i.e. any dynamic $r \in \mathcal{R}_0$ may be active at any time step, whereby r is random. This system type is often referred to as *switched system* subject to arbitrary switching, i.e. the active dynamics cannot be influenced. Let $\phi^{\text{aug}}(k; x_0, \mathbf{w}_k)$ denote the set of states which is reachable from the initial state x_0 in k steps for $x(k+1) = f_a^{\text{aug}}(x(k), w(k))$ and for the disturbance sequence $\mathbf{w}_k$. The k-step robust reachable set for the augmented system is then given by

$$\text{Reach}^{\text{aug}}(k; \mathcal{S}, \mathbb{W}) \triangleq \{\phi^{\text{aug}}(k; x_0, \mathbf{w}_k) \in \mathbb{R}^n \mid x_0 \in \mathcal{S},\ \mathbf{w}_k \in \mathbb{W}^k\}$$

Definition 7.3.1 (Disturbance Response Set $\mathcal{F}_k$) *Let the set $\mathcal{F}_k$ ($\mathcal{F}_k^{\text{aug}}$) be the k-step disturbance response for the (augmented) system defined in* (6.2) *((7.5)) so that*

$$\mathcal{F}_k = \text{Reach}(k; 0, \mathbb{W}),$$
$$\mathcal{F}_k^{\text{aug}} = \text{Reach}^{\text{aug}}(k; 0, \mathbb{W}),$$

and let the set $\mathcal{F}_\infty$ ($\mathcal{F}_\infty^{\text{aug}}$) be the infinite-time disturbance response set, i.e. $\mathcal{F}_\infty = \lim_{k\to\infty} \mathcal{F}_k$ ($\mathcal{F}_\infty^{\text{aug}} = \lim_{k\to\infty} \mathcal{F}_k^{\text{aug}}$).

Note that the infinite-time disturbance response set $\mathcal{F}_\infty$ corresponds to the minimal robust invariant set in Definition 7.1.2. Methods for approximating this set have been published in [RKKM03] for LTI and in [RG04a] for PWA systems. It follows trivially, from the definitions of the corresponding sets, that $\text{Reach}^{\text{aug}}(k; \mathcal{S}, 0) \supseteq \text{Reach}(k; \mathcal{S}, 0)$, $\text{Reach}^{\text{aug}}(k; \mathcal{S}, \mathbb{W}) \supseteq \text{Reach}(k; \mathcal{S}, \mathbb{W})$, $\mathcal{F}_k^{\text{aug}} \supseteq \mathcal{F}_k$ and $\mathcal{F}_\infty^{\text{aug}} \supseteq \mathcal{F}_\infty$. As shown in [RG04a], the set $\mathcal{F}_\infty^{\text{aug}}$ is bounded by a compact set $\mathcal{F}$ (i.e. $\mathcal{F}_k \subseteq \mathcal{F}_k^{\text{aug}} \subset \mathcal{F},\ \forall k > 0$), if the nominal system $f_a^{\text{aug}}(x, 0)$ in (7.5) is asymptotically stable. Furthermore, if $\lim_{k\to\infty} \mathcal{F}_k^{\text{aug}} \to \mathcal{F}$, then $\mathcal{F}$ is robust positive invariant for the augmented system (7.5) and the autonomous PWA system (6.2) [RG04a].

Theorem 7.3.2 (Finite-Time Termination of Algorithm 7.2.3, [RGK+04a]) *Suppose that there exists a compact set Υ such that $\mathcal{F}_\infty^{\text{aug}} \subseteq \Upsilon \subseteq \text{int}(\mathbb{X}_0)$ and that the nominal augmented system (7.5) with $\mathbb{W} = \{0\}$ is asymptotically stable; Then, Algorithm 7.2.3 terminates in finite time.*

Proof First note that the following set inclusion holds [RG04a],

$$\text{Reach}(k; \mathbb{X}_0, \mathbb{W}) \subseteq \text{Reach}^{\text{aug}}(k; \mathbb{X}_0, 0) \oplus \mathcal{F}_k^{\text{aug}} = \text{Reach}^{\text{aug}}(k; \mathbb{X}_0, \mathbb{W}).$$

It follows from asymptotic stability of the nominal augmented system (7.5) that there exists a finite time k^* such that $\text{Reach}^{\text{aug}}(k^* + 1; \mathbb{X}_0, 0) \subseteq \mathbb{X}_0 \ominus \Upsilon$. Let the set $\Omega_{k^*} \subseteq \mathbb{X}_0$, denote the k^*-th term of the set sequence $\{\Omega_k\}$ generated by Algorithm 7.2.3, so that Ω_{k^*} denotes the set of states which satisfy $\text{Reach}(k^*; \Omega_{k^*}, \mathbb{W}) \subseteq \mathbb{X}_0$. Since $\Omega_{k^*} \subseteq \mathbb{X}_0$, it follows that $\text{Reach}^{\text{aug}}(k^* + 1; \Omega_{k^*}, 0) \subseteq \mathbb{X}_0 \ominus \Upsilon$ which implies $\text{Reach}(k^* + 1; \Omega_{k^*}, 0) \subseteq \mathbb{X}_0 \ominus \Upsilon$. Hence we have established that $\text{Reach}(k^*; \Omega_{k^*}, \mathbb{W}) \subseteq \mathbb{X}_0$ and $\text{Reach}(k^* + 1; \Omega_{k^*}, \mathbb{W}) \subseteq \mathbb{X}_0$, which implies that $\Omega_{k^*} \subseteq \Omega_{k^*+1}$.

Since the sequence $\{\Omega_k\}$ satisfies $\Omega_{k+1} \subseteq \Omega_k$ for all $k \in \mathbb{N}$ it must hold that $\Omega_{k^*} = \Omega_{k^*+1}$. This directly implies that Ω_{k^*} is the maximal robust invariant set. □

Note that the finite-time termination conditions of Theorem 7.3.2 are less restrictive than they may seem and are automatically satisfied in various control problems, e.g. see Chapter 14.

Corollary 7.3.3 *Suppose that the nominal autonomous system defined in (6.2) is asymptotically stable with $\mathbb{W} = \{0\}$ and $\mathbb{X}$ is a compact set that contains the origin in its interior. Then Algorithm 7.2.3 computes the maximal positive invariant set in finite time.*

Remark 7.3.4 *A detailed overview of the properties of the disturbance response set $\mathcal{F}_\infty$ ($\mathcal{F}_\infty^{\text{aug}}$) as well as algorithms to compute Υ such that $\Upsilon \supseteq \mathcal{F}_\infty^{\text{aug}}$ are given in [RG04a].*

Remark 7.3.5 *Note that finite-time termination of the proposed algorithms is important, but should not be overrated. For PWA systems, the computational requirements grow exponentially (in the worst case) from iteration to iteration such that it is not possible to compute the proposed invariant sets for all types of systems, even when finite termination can be guaranteed. Even if the algorithm were to terminate*

after 50 iterations, the necessary computation power may be beyond current hardware. This is not so much an issue for LTI systems, since the computational demand for those cases remains (relatively) limited because of convexity.

7.4 Maximal Robust Control Invariant Set

This subsection shows how computation of $\mathcal{O}_\infty$ permits the computation of $\mathcal{K}_\infty(\mathcal{O}_\infty)$ by using projection methods, e.g. [JKM04,KS90,FLL00]. A similar procedure can be employed for the computation of $\mathcal{C}_\infty$. The algorithm below describes the computation scheme for $\mathcal{K}_\infty(\mathcal{O}_\infty)$ or $\mathcal{C}_\infty$, depending on the choice of initial target set $\mathcal{S}_0$:

Algorithm 7.4.1 (Computation $\mathcal{K}_\infty(\mathcal{O}_\infty)$ or $\mathcal{C}_\infty$)

1. *Define a target set $\mathcal{S}_0$ and set $k = 0$.*
2. *Compute $\mathcal{S}_{k+1} = \mathrm{Pre}(\mathcal{S}_k, \mathbb{W})$ as in (7.1).*
3. *If $\mathcal{S}_{k+1} = \mathcal{S}_k$, return; Else, set $k = k + 1$ and goto step 2.*

The sets $\mathcal{S}_k$ are P-collections (see (7.1) and Lemma 7.2.2), making Algorithm 7.4.1 computationally demanding. At each time k, the target set $\mathcal{S}_k \ominus \mathbb{W} = \bigcup_{l \in L_k} \mathcal{S}_k^l$ is a P-collection, where the set L_k has a finite cardinality that changes with time k, so that

$$\mathrm{Pre}(\mathcal{S}_k, \mathbb{W}) = \bigcup_{l \in L_k} \mathrm{Pre}(\mathcal{S}_k^l, 0). \tag{7.6}$$

where $\mathcal{S}_k^l$ is a polytopic set. Therefore, $\mathrm{Pre}(\mathcal{S}_k, \mathbb{W})$ can be computed via a sequence of projection operations (see Chapter 2).

Theorem 7.4.2 (Computation of $\mathcal{C}_\infty$, [RGK+04b]) *Suppose that $\mathcal{S}_0 = \mathbb{X}$ and that there exists a $k^* \in \mathbb{N}$ such that $\mathcal{S}_{k^*} = \mathcal{S}_{k^*+1}$. Then, Algorithm 7.4.1 terminates and $\mathcal{C}_\infty = \mathcal{S}_{k^*}$.*

Proof It holds that $\mathcal{S}_0 = \mathbb{X}$ is the largest feasible set and $\mathcal{S}_k \supseteq \mathcal{S}_{k+1}$. If $\mathcal{S}_{k^*} = \mathcal{S}_{k^*+1}$ then $\mathcal{S}_{k^*}$ is a fixed point of Algorithm 7.4.1 and it is the maximal robust control invariant set contained in $\mathbb{X}$, i.e. $\mathcal{C}_\infty = \mathcal{S}_{k^*}$. □

Theorem 7.4.3 (Computation of $\mathcal{K}_\infty(\mathcal{O}_\infty)$, [RGK+04b]) *Suppose that $\mathcal{S}_0 = \mathcal{O}_\infty$ and that there exists a $k^* \in \mathbb{N}$ such that $\mathcal{S}_{k^*} = \mathcal{S}_{k^*+1}$. Then, Algorithm 7.4.1 terminates and $\mathcal{K}_\infty(\mathcal{O}_\infty) = \mathcal{S}_{k^*}$.*

Proof It holds that $\mathcal{S}_0$ is a robust positive invariant set by construction and $\mathcal{S}_k \subseteq \mathcal{S}_{k+1}$. If $\mathcal{S}_{k^*} = \mathcal{S}_{k^*+1}$, it follows that there does not exist a state $x \notin \mathcal{S}_{k^*}$ such that $f_{\text{PWA}}(x, u, w) \in \mathcal{S}_{k^*}$ for any $u \in \mathbb{U}$ and $\forall w \in \mathbb{W}$. Therefore $\mathcal{K}_\infty(\mathcal{O}_\infty) = \mathcal{S}_{k^*}$ if $\mathcal{S}_{k^*} = \mathcal{S}_{k^*+1}$. □

Remark 7.4.4 *Note that $\mathcal{K}_\infty(\mathcal{O}_\infty)$ and/or $\mathcal{C}_\infty$ may not be finitely determined. It may therefore be necessary to abort Algorithm 7.4.1 after a predefined maximum number of iterations or after the state space of interest has been covered.*

7.5 Numerical Results

In order to illustrate the proposed procedure we consider two second order PWA systems.

Example 7.5.1 *Our first example is the following 2-dimensional problem adopted from [MR03]:*

$$x(k+1) = A_r x(k) + B_r u(k) + g_r + w(k) \tag{7.7}$$

where $r = 1$ if $x_{(1)}(k) \leq 1$ and $r = 2$ if $x_{(1)}(k) > 1$,

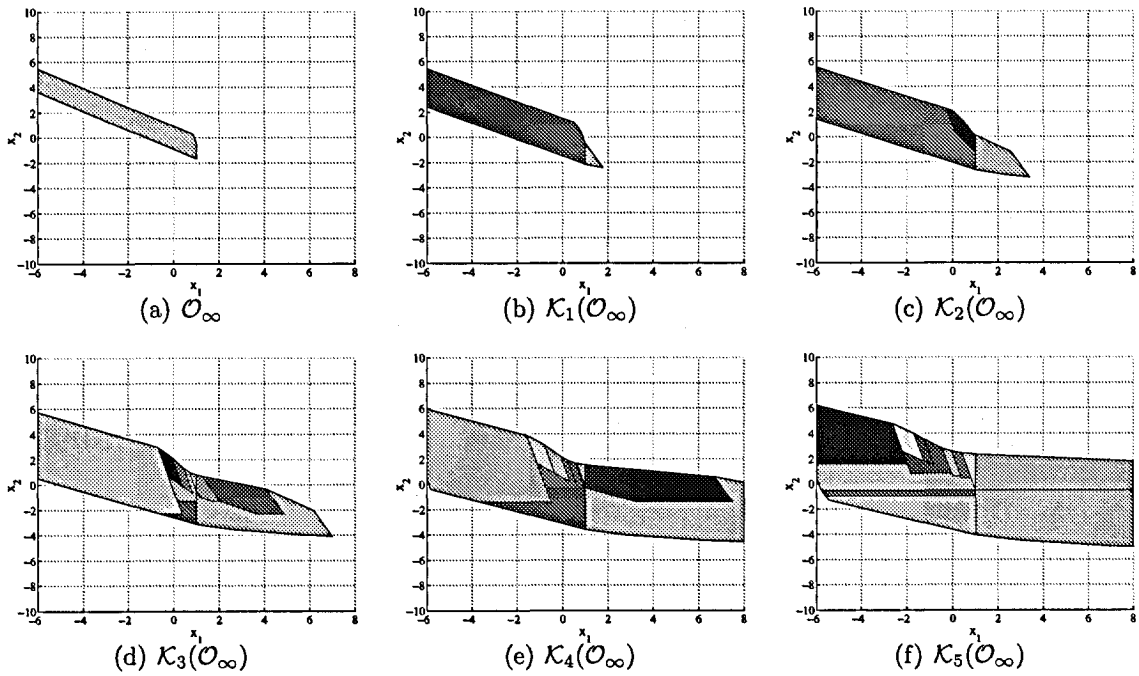
$$A_1 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad g_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$A_2 = \begin{bmatrix} 0.5 & 0.2 \\ 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}.$$

and the additive disturbance $w(k)$ is bounded:

$$w(k) \in \{w \in \mathbb{R}^2 \mid \|w\|_\infty \leq 0.1\}.$$

Here, the subindex in parenthesis is used to denote specific elements of the state vector x. The system is subject to the constraints $-x_{(1)}(k) + x_{(2)}(k) \leq 15$, $-3x_{(1)}(k) - x_{(2)}(k) \leq 25$, $0.2x_{(1)}(k) + x_{(2)}(k) \leq 9$, $x_{(1)}(k) \geq -6$, $x_{(1)}(k) \leq 8$, and $-1 \leq u(k) \leq 1$.

We applied Algorithm 7.4.1 to the PWA system (7.7) to obtain the set $\mathcal{K}_\infty(\mathcal{O}_\infty)$. The target set $\mathcal{O}_\infty$ was obtained by computing the maximal invariant set for the Riccati LQR feedback controller with the algorithm in [KG98] (see Figure 7.1(a)). The first iterations of the algorithm are shown in Figure 7.1 and the final result is depicted in Figure 7.2.



(a) $\mathcal{O}_\infty$ (b) $\mathcal{K}_1(\mathcal{O}_\infty)$ (c) $\mathcal{K}_2(\mathcal{O}_\infty)$

(d) $\mathcal{K}_3(\mathcal{O}_\infty)$ (e) $\mathcal{K}_4(\mathcal{O}_\infty)$ (f) $\mathcal{K}_5(\mathcal{O}_\infty)$

Figure 7.1: The first iterations of Algorithm 7.4.1 applied to Example 7.5.1.

In the following, we apply the presented algorithm to a switched system. Note that switched systems are a special class of PWA systems, where each dynamic set $\mathcal{P}_r = \mathbb{R}^n$ and the active dynamic r can be selected as an external input. The previously presented algorithms can be directly applied to this system class. In fact, most computations become easier and some of the conservative assumptions/conditions made in the previous section become less restrictive. A detailed discussion of invariant set computation for switched systems is given in [GRMM05]. We will illustrate the application of the proposed algorithms on the following numerical example.

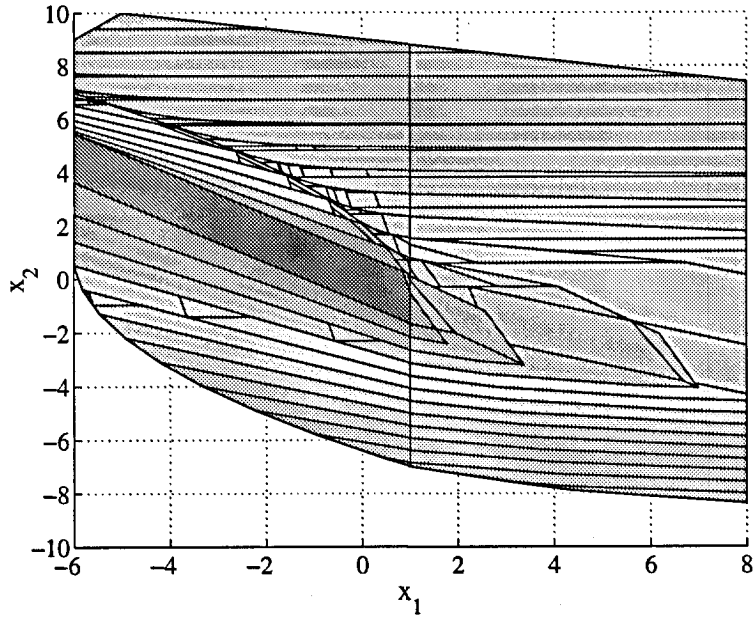


Figure 7.2: $\mathcal{K}_\infty(\mathcal{O}_\infty) = \mathcal{K}_{15}(\mathcal{O}_\infty)$ for Example 7.5.1.

Example 7.5.2 *Assume a switched system with the following dynamics*

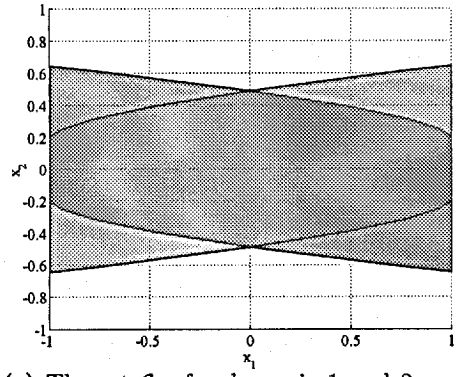
$$\tilde{A}_1 = \begin{bmatrix} 0.8 & -1 \\ 0 & 0.8 \end{bmatrix}, \qquad \tilde{g}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\tilde{A}_2 = \begin{bmatrix} 0.8 & 1 \\ 0 & 0.8 \end{bmatrix}, \qquad \tilde{g}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

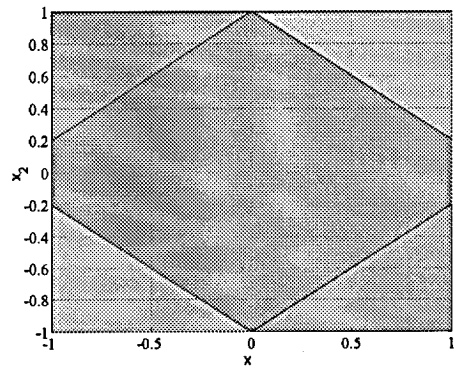
The system is subject to the state constraints $\mathbb{X} = \{x \in \mathbb{R}^n | \ \|x\|_\infty \leq 1\}$. *The objective of the controller is to satisfy these constraints for all time.*

We will initially consider no additive disturbance, i.e. $\mathbb{W} = \{0\}$. The maximal positively invariant set for each of the dynamics is depicted in Figure 7.3(a). Figure 7.3(b) depicts the partition of a switching controller. It is clear from the figures that an appropriate switching scheme will enlarge the set of controllable states. Let us now assume that the system is subject to additive disturbance bounded by $\mathbb{W} = \{w \in \mathbb{R}^n | \ \|w\|_\infty \leq 0.1\}$. For this case, there is no robust invariant subset contained inside the target box, if we assume no switching occurs. Hence, there is a clear need to consider dynamic switches. The maximal robust switched invariant set is depicted in Figure 7.3(c), if we allow for switches.

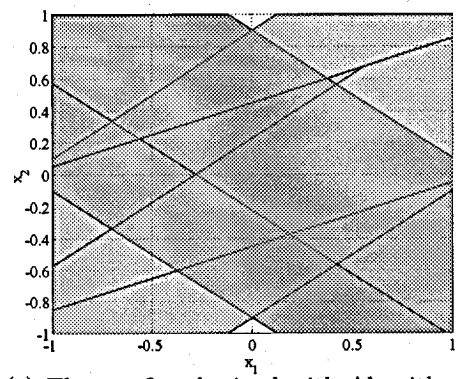
Next, we use the maximum robust positive invariant set in Figure 7.3(c) as a target set for a minimum time controller, i.e. the control objective is to drive the state into the target set in minimum time. In order to obtain a larger controller partition, the system constraints were increased to $\mathbb{X} = \{x \in \mathbb{R}^n | \ \|x\|_\infty \leq 10\}$. Figure 7.4 depicts two different minimum-time state trajectories for the initial state $x(0) = [0 \ \ 9.5]^T$ as well as the associated switching sequence. Note that the minimum-time switching sequence is not unique, i.e. both switching in Figure 7.4 yield minimum-time trajectories.



(a) The set $\mathcal{C}_\infty$ for dynamic 1 and 2, respectively ($\mathbb{W} = \{0\}$, no switching).

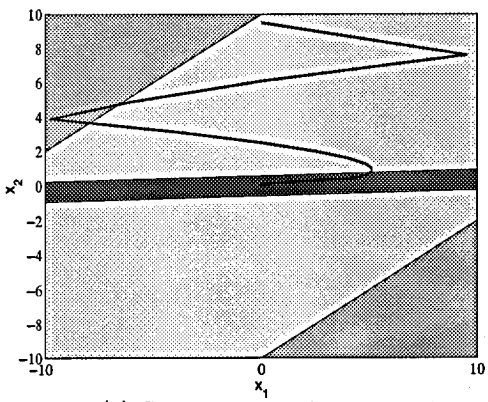


(b) The set $\mathcal{C}_\infty$ obtained with Algorithm 7.4.1 ($\mathbb{W} = \{0\}$, with switching).

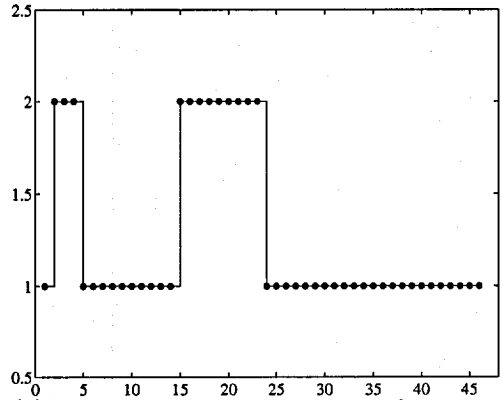


(c) The set $\mathcal{C}_\infty$ obtained with Algorithm 7.4.1 ($\mathbb{W} = \{w \in \mathbb{R}^n | \ ||w||_\infty \leq 0.1\}$, with switching).

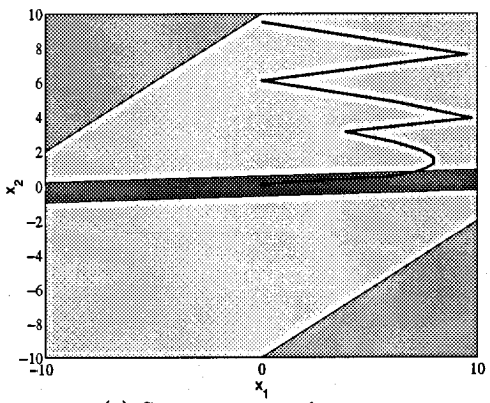
Figure 7.3: Example 7.5.2: Controllable state space with and without switching, contained in $\mathbb{X} = \{x \in \mathbb{R}^n | \ \|x\|_\infty \leq 1\}$. The system subject to disturbances $\mathbb{W} = \{w \in \mathbb{R}^n | \ ||w||_\infty \leq 0.1\}$ is not controllable without switching. The various shadings in the figures correspond to the feasible dynamics ($i = 1$, $i = 2$ or $i \in \{1, 2\}$).



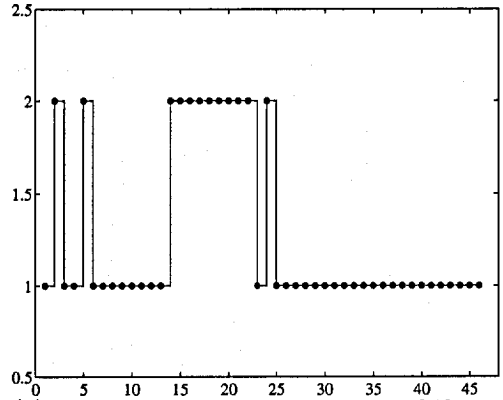
(a) State space trajectory.



(b) Switching law. Dynamics are kept constant if possible.



(c) State space trajectory.



(d) Switching law. Dynamic 1 is used if possible.

Figure 7.4: Example 7.5.2: Minimum-time trajectory for different switching regimes and initial state $x(0) = [0 \;\; 9.5]^T$ ($\mathbb{W} = \{0\}$, $\mathbb{X} = \{x \in \mathbb{R}^n | \; \|x\|_\infty \leq 10\}$). The target set for the minimum-time controller is depicted in Figure 7.3(c).

8

Stability Analysis of Piecewise Affine Systems

8.1 Introduction

In this chapter we will demonstrate how Lyapunov functions which guarantee asymptotic stability can be constructed for autonomous PWA systems of type (6.2). For a good additional reference on this topic, we refer the reader to [Joh02].

Throughout this chapter we assume that the autonomous PWA system does not contain overlapping regions $\mathcal{P}_i$, i.e. $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for $i \neq j$. However, if the system dynamics are continuous, it is possible for two regions $\mathcal{P}_i$ and $\mathcal{P}_j$ to share a common facet. These conditions are needed to ensure that the state-update equation is uniquely defined for all states. Furthermore, we assume the PWA partition to be invariant, since the notion of stability has no practical relevance if the state trajectory exits the defined state space $\bigcup_{r\in\mathcal{R}} \mathcal{P}_r$, $\mathcal{R} \triangleq \{1,\dots,R\}$.

For instance, assume ellipsoidal level sets of a Lyapunov function and a feasible state space corresponding to a hypercube. The hypercube may not be invariant, hence, successful computation of a Lyapunov function defined over the hypercube will not imply that all states in the hypercube converge to the origin.

Therefore all following computations will be performed on the maximum robust invariant subset $\mathcal{S}^{inv}_{\text{PWA}}$ of $\mathcal{S}_{\text{PWA}} = \bigcup_{r\in\mathcal{R}} \mathcal{P}_r$ of the autonomous PWA system (6.2), i.e.

$$\mathcal{S}^{inv}_{\text{PWA}} = \{x(0) \in \mathcal{S}_{\text{PWA}} \mid x(k) \in \mathcal{S}_{\text{PWA}}, \forall k \geq 0,\ x(k+1) = f_a(x(k), w(k)),\ \forall w(k) \in \mathbb{W}\}.$$

The previously stated assumptions can be summarized as:

Assumption 8.1.1 *The assumptions throughout this chapter can be summarized as follows:*

- $\mathcal{P}_i \cap \mathcal{P}_j$ *($i \neq j$, $\mathcal{P}_i, \mathcal{P}_j \subseteq \mathbb{R}^n$) is an empty set or lower dimensional.*
- $\mathcal{S}_{PWA}^{inv} = \mathcal{S}_{PWA} = \bigcup_{r \in \mathcal{R}} \mathcal{P}_r$.

Next we will introduce some basic definitions that will be used in this chapter.

Definition 8.1.2 (Stability, [Kha96]) *The equilibrium point $x = 0$ of a dynamic system is*

- *stable, if for each $\epsilon > 0$, there is $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(0)\| < \delta \Rightarrow \|x(k)\| < \epsilon, \quad \forall k \geq 0.$$

- *unstable, if not stable.*
- *asymptotically stable if it is stable and δ can be chosen such that*

$$\|x(0)\| < \delta \Rightarrow \lim_{k \to \infty} \|x(k)\| = 0.$$

It directly follows that a *stable* PWA system will have $g_r = 0, \forall \mathcal{P}_r$ containing 0. It is not necessary that $0 \in \text{int}(\mathcal{P}_r)$ for some r.

The following theorems are based on [Vid93, p. 267] and have been adapted to deal with the special case of constrained autonomous PWA systems of type (6.2) (see also [FTCMM02] for details):

Definition 8.1.3 (Class K Function, [Vid93]) *A function $\alpha(x) : \mathbb{R}^n \to \mathbb{R}_+$ is of class K if it is continuous, strictly increasing and $\alpha(0) = 0$.*

Theorem 8.1.4 (Lyapunov Stability) *The origin $x = 0$ is* stable *for the autonomous PWA system (6.2) if there exists a function $V(x) : \mathcal{S}_{PWA} \to \mathbb{R}_+$ with $0 \in \mathcal{S}_{PWA}$ such that there exists a class K function $\alpha(x)$ such that $\alpha(x) \leq V(x)$ (with $V(0) = 0$) and $V(f_a(x, 0)) - V(x) \leq 0$. Here, it is assumed $\mathbb{W} = \{0\}$.*

Theorem 8.1.5 (Asymptotic Stability) *The origin $x = 0$ is* asymptotically stable *for the autonomous PWA system (6.2) if there exists a function $V(x) : \mathcal{S}_{PWA} \to \mathbb{R}_+$ with $0 \in \mathcal{S}_{PWA}$ and scalar coefficients $\alpha > 0$, $\beta > 0$, $\rho > 0$ such that $\beta\|x\| \geq V(x) \geq \alpha\|x\|$ and $V(f_a(x, 0)) - V(x) \leq -\rho\|x\|$, $\forall x \in \mathcal{S}_{PWA}$. Here, $\|\cdot\|$ denotes a vector norm and it is assumed $\mathbb{W} = \{0\}$.*

Theorem 8.1.6 (Exponential Stability) *The origin $x = 0$ is* exponentially stable *for the autonomous PWA system (6.2) if there exists a function $V(x) : \mathcal{S}_{PWA} \rightarrow \mathbb{R}_+$ with $0 \in \mathcal{S}_{PWA}$ and scalar coefficients $\alpha > 0$, $\beta > 0$, $\rho > 0$ and $p > 1$ such that $\beta||x||^p \geq V(x) \geq \alpha||x||^p$ and $V(f_a(x,0)) - V(x) \leq -\rho||x||^p$, $\forall x \in \mathcal{S}_{PWA}$. Here, $||\cdot||$ denotes a vector norm and it is assumed $\mathbb{W} = \{0\}$.*

In the following sections, we will introduce various methods to construct Lyapunov functions which prove asymptotic and/or exponential stability of autonomous PWA systems (6.2). In Section 8.5, methods for analyzing robust convergence of PWA systems subject to additive uncertainty will be presented. In Section 8.7, an extensive case study is given, where the various stability analysis techniques are compared with respect to likelihood of successful analysis and runtime.

8.2 Computation of PWA Lyapunov Functions for PWA Systems

It will be shown in the following how to formulate the search for a PWA Lyapunov function guaranteeing asymptotic stability for nominal autonomous PWA systems as a linear program (LP). The results in this section are based on [GKBM04b]. Specifically, we will consider nominal autonomous PWA systems (6.2) with $\mathbb{W} = \{0\}$, i.e.

$$x(k+1) = \tilde{A}_r x(k) + \tilde{g}_r, \qquad \text{if } x(k) \in \mathcal{P}_r,\ r \in \mathcal{R}.$$

The search for a PWQ Lyapunov function is conservative, since the associated SDP formulation utilizes the S-procedure, which is not lossless for the cases considered [BGFB94]. This issue will be discussed in detail in Section 8.3. Therefore, instead of searching for a PWQ Lyapunov function via SDP, we show here how to construct a PWA Lyapunov function via LP. The proposed scheme is based on results for continuous time systems which were published in [Joh02].

The computation scheme for the PWA Lyapunov function is non-conservative (i.e. if a PWA Lyapunov function over the given partition exists, it will be found) such that it may succeed when no PWQ Lyapunov function can be found with the schemes in [FTCMM02, Fen02, GLPM03].

Remark 8.2.1 *Note that the scheme is non-conservative for a* given *partition. If no function is found, there may still exist a PWA Lyapunov function which is defined*

over a different partition than the partition defining the PWA system.

Since we are searching for a PWA function $V(x)$, the explicit representation of the decay rate $V(x(k+1)) - V(x(k))$ depends on the regions $\mathcal{P}_i$, $\mathcal{P}_j$ which contain $x(k)$ and $x(k+1)$, respectively. Therefore, a region transition map needs to be created in order to formulate the LP problem. For computational efficiency, this reachability computation is split into two parts: First, the feasible transitions from region i to j are identified and subsequently the set of states $\mathcal{P}_{ij}$ which actually undergo such a transition are computed.

Specifically, a transition map T is first created $\forall i, j \in \{1, \ldots, R\}$ according to

$$T(i,j) = \begin{cases} 1, & \text{if } \exists x \in \text{int}(\mathcal{P}_i), \text{ s.t. } \tilde{A}_i x + \tilde{g}_i \in \mathcal{P}_j, \\ 0, & \text{otherwise,} \end{cases}$$

where $\text{int}(\cdot)$ denotes the strict interior of a set. The matrix T is then used to construct the set $\mathcal{T} \triangleq \{i, j \in \mathcal{R} \mid T(i,j) = 1\}$. Only the interior of the dynamic $\mathcal{P}_i$ is considered in the transition map in order to guarantee that the transition set $\mathcal{P}_{ij}$ is full dimensional.

Remark 8.2.2 *In principle, one LP needs to be solved for each element of the transition map $\mathcal{T}$, i.e. a total of R^2 LPs, where R denotes the total number of system dynamics. However, instead of solving LPs directly, it is advisable to first compute bounding boxes (hyper-rectangles) for each region $\mathcal{P}_r$ ($r \in \mathcal{R}$). In addition, a bounding box of the affine map of the region $\mathcal{P}_r^+ = \{\tilde{A}_r x + \tilde{g}_r \in \mathbb{R}^n | \ x \in \mathcal{P}_r\}$ needs to be computed. The number of LPs which need to be solved in order to compute the bounding box is linear in the number of regions R and state space dimension n. This computation is tractable even for very complex partitions. The bounding boxes can be efficiently checked for intersections, such that certain transitions $i \to j$ can be ruled out. In our experience, the bounding box implementation is the most effective way to compute $\mathcal{T}$ for complex region partitions.*

In a second step, the transition sets $\mathcal{P}_{ij}$ for system (6.2) are explicitly computed for all $i, j \in \mathcal{T}$:

$$\mathcal{P}_{ij} = \{x \in \mathbb{R}^n | \ x \in \mathcal{P}_i, \ \tilde{A}_i x + \tilde{g}_i \in \mathcal{P}_j\} \tag{8.1a}$$

$$= \{x \in \mathbb{R}^n | \ H_{ij}\, x \leq K_{ij}\}. \tag{8.1b}$$

If $T(i,j) = 0$, then $\mathcal{P}_{ij} = \emptyset$. In addition, the vertices of the transition sets ($\text{vert}(\mathcal{P}_{ij})$) and the original PWA sets ($\text{vert}(\mathcal{P}_i)$) are computed. The problem of finding a PWA

Lyapunov function,

$$\mathrm{PWA}_r(x) = V_r^{(1)}x + V_r^{(0)}, \qquad \text{if } x \in \mathcal{P}_r,\ r \in \mathcal{R},$$

for the autonomous PWA system (6.2) such that the conditions in Theorem 8.1.5 are satisfied can now be stated as

$$\beta||x||_1 \geq \mathrm{PWA}_r(x) \geq \quad \alpha||x||_1,\ \alpha,\beta > 0,\ \forall x \in \mathrm{vert}(\mathcal{P}_r),\ \forall r \in \mathcal{R}, \tag{8.2a}$$

$$\mathrm{PWA}_j(\tilde{A}_i x + \tilde{g}_i) - \mathrm{PWA}_i(x) \leq \ -\rho||x||_1,\ \rho > 0,\ \forall x \in \mathrm{vert}(\mathcal{P}_{ij}),\ \forall i,j \in \mathcal{T}. \tag{8.2b}$$

Since the vertices of all sets $\mathcal{P}_i$ and $\mathcal{P}_{ij}$ are known, the resulting problem is linear in $V_r^{(1)}, V_r^{(0)}$ and can therefore be solved as an LP. Note that it is possible to replace the 1-norm in (8.2) with any other linear norm, e.g. the ∞-norm.

Theorem 8.2.3 (Asymptotic Stability Guarantee via LP, [GKBM04b]) *If the LP* (8.2) *associated with the autonomous PWA system* (6.2) *is feasible, then this system is asymptotically stable.*

Proof First note that any linear norm is convex. Since the function $\mathrm{PWA}_r(x)$, $r \in \mathcal{R}$ is piecewise affine, it follows that satisfaction of (8.2a) for all vertices of $\mathcal{P}_r$ implies that the inequalities in (8.2a) will also hold $\forall x \in \mathcal{P}_r$. Furthermore, if (8.2b) holds for all vertices of $\mathcal{P}_{ij}$, it follows from linearity of the system dynamics (6.2) that the inequality will hold for all states $x \in \mathcal{P}_{ij}$. Since the partition $\mathcal{S}_{\mathrm{PWA}}$ is invariant, it follows that $\mathcal{S}_{\mathrm{PWA}} = \bigcup_{r\in\mathcal{R}} \mathcal{P}_r = \bigcup_{i,j\in\mathcal{T}} \mathcal{P}_{ij}$. Therefore, the inequalities in (8.2a) and (8.2b) hold $\forall x \in \mathcal{S}_{\mathrm{PWA}}$ such that the conditions in Theorem 8.1.5 are satisfied, i.e. feasibility of (8.2) implies asymptotic stability of the autonomous PWA system (6.2). □

It should be noted that the required computation time may become large due to the extensive reachability analysis, vertex enumeration and size of the final LP. Specifically, the LP (8.2) introduces one constraint for each vertex of each region $\mathcal{P}_r$, $\forall r \in \mathcal{R}$ (see (8.2a)) and one constraint for each vertex of each $\mathcal{P}_{ij}$, $\forall i,j \in \mathcal{T}$ (see (8.2b)). The number of variables is $(n+1)R$, where R denotes the number of dynamics and n the state space dimension.

However, in the authors experience, the computational effort for constructing PWA Lyapunov functions via LP is comparable to the required effort for constructing PWQ Lyapunov functions via SDP [FTCMM02, Fen02, GLPM03] (see Section 8.7).

Remark 8.2.4 *It follows from the constraints* (8.2a) *(i.e.* $V(0) = 0$*) that the PWA Lyapunov function will have no offset term for all regions containing the origin, i.e.* $V_i^{(0)} = 0,\ \forall i \in \mathcal{R}_0$. *Since the Lyapunov function is PWA for the remainder of the state space, there will always exist a parameter* β *bounding the Lyapunov function from above. Hence, the 'upper bound' constraint* $\beta||x||_1 \geq PWA_i(x)$ *does not need to be enforced when solving the LP* (8.2).

Remark 8.2.5 *Note that it is not possible to find a PWA Lyapunov function if there exists a region* $\mathcal{P}_r$ *such that* $0 \in \text{int}(\mathcal{P}_r)$, *where* $\text{int}(\cdot)$ *denotes the strict interior of a set. For this region, the function would have to be strictly linear (Remark 8.2.4) such that it would assume negative values in some neighborhood of the origin, thus violating* (8.2a). *Note that mp-QP partitions always contain regions containing the origin in their interior, provided the system constraints* (6.1) *contain the origin in their interior*[1]. *On the other hand, the equivalent mp-LP solution will never comprise regions containing the origin in their interior.*

Remark 8.2.6 *A standing assumption throughout this section is that* $\mathcal{S}_{PWA} = \mathcal{S}_{PWA}^{inv} = \bigcup_{i,j\in\mathcal{T}} \mathcal{P}_{ij}$. *This assumption may not hold in all cases. If the partition is not invariant, it may still be desirable to refrain from computing* $\mathcal{S}_{PWA}^{inv}$ *for computational reasons (see Chapter 7). It then holds that* $\mathcal{S}_{PWA} = \bigcup_{i\in\mathcal{R}} \mathcal{P}_i$ *and* $\mathcal{S}_{PWA} \supseteq \bigcup_{i,j\in\mathcal{T}} \mathcal{P}_{ij}$, *such that conservativeness is introduced to the formulation* (8.2). *However, if the LP analysis* (8.2) *over* $\mathcal{S}_{PWA}$ *is feasible, this directly implies stability of the set* $\mathcal{S}_{PWA}^{inv}$. *It does not, however, imply invariance of the set* $\mathcal{S}_{PWA}$.

8.3 Computation of PWQ Lyapunov Functions for PWA Systems

It was shown how to use SDPs to construct PWQ Lyapunov functions for continuous-time systems in [JR98, Joh02] and for discrete-time systems in [FTCMM02, Fen02].

[1] It would be possible to artificially split the region containing the origin, such that a PWA Lyapunov function can be constructed. However, it is not obvious how select a suitable splitting scheme.

The contribution of this section is based on [GLPM03] and consists of a modification to the stability analysis method in [FTCMM02,Fen02], which makes the computation less conservative.

Specifically, we will consider nominal autonomous PWA systems (6.2) with $\mathbb{W} = \{0\}$, i.e

$$x(k+1) = \tilde{A}_r x(k) + \tilde{g}_r, \qquad \text{if } x(k) \in \mathcal{P}_r,\ r \in \mathcal{R}.$$

Since we are searching for a PWQ function $V(x)$, the explicit representation of the decay rate $V(x(k+1)) - V(x(k))$ depends on the regions $\mathcal{P}_i$, $\mathcal{P}_j$ which contain $x(k)$ and $x(k+1)$, respectively. Therefore, a region transition map needs to be created in order to formulate the subsequent SDP problem. The construction of the transition set $\mathcal{T}$ and the reachable sets $\mathcal{P}_{ij}$ is described in Section 8.2.

The problem of finding a PWQ Lyapunov function, such that exponential stability according to Definition 8.1.6 is guaranteed can now be formulated as an SDP by applying the S-Procedure as done in [FTCMM02, Fen02, JR98]. In each polyhedral cell $\mathcal{P}_r$, the function PWQ(x) will be defined by $\text{PWQ}_r(x) = x^T V_r^{(2)} x + x^T V_r^{(1)} + V_r^{(0)}$. It should be pointed out that the PWQ Lyapunov function is allowed to be discontinuous and/or non-convex, since we are dealing with discrete-time systems.

The following constraints are now imposed on the function PWQ(x) in order to obtain a PWQ Lyapunov function:

$$\beta x^T x \geq \text{PWQ}_r(x) \geq \alpha x^T x, \qquad \alpha, \beta > 0,\ \forall x \in \mathcal{P}_r,\ \forall r \in \mathcal{R}, \tag{8.3a}$$

$$\text{PWQ}_j(\tilde{A}_i x + \tilde{g}_i) - \text{PWQ}_i(x) \leq -\rho x^T x, \qquad \rho > 0,\ \forall x \in \mathcal{P}_{ij},\ \forall\, i,j \in \mathcal{T}. \tag{8.3b}$$

Although the formulation in (8.3) is similar to the procedure applied in [FTCMM02, Fen02], there is a subtle but important difference. Specifically, the transitions sets $\mathcal{P}_{ij}$ were not considered in [FTCMM02, Fen02] and the constraint (8.3b) was required to hold for all $x \in \mathcal{P}_i$, if $T(i,j) = 1$ for any $j \in \mathcal{R}$. This is always more conservative than requiring the constraint to hold only for all $x \in \mathcal{P}_{ij}$, if $T(i,j) = 1$, as is proposed here, since $\mathcal{P}_{ij} \subseteq \mathcal{P}_i$.

Problem (8.3) can be formulated as an SDP as will be shown in the following. Let $G_{ij}(x) = K_{ij} - H_{ij}x$ (recall the notation in (8.1)) and $\Delta V_{ij}(x) = \text{PWQ}_j(A_i x + g_i) - \text{PWQ}_i(x)$. By applying the S-procedure[2] [BGFB94], we can *conservatively*

[2] $f(x) \geq 0\ \forall x : g_i(x) \geq 0$ is conservatively replaced with the sufficient condition

approximate (8.3b) with

$$\exists N_{ij} \geq 0 : \Delta V_{ij}(x) \leq -\rho x^T x - G_{ij}^T(x) N_{ij} G_{ij}(x), \tag{8.4}$$

where $\rho > 0$ and N_{ij} is an arbitrary symmetric matrix consisting of non-negative elements only. With $\bar{x} = [x \;\; 1]^T$ and $x \in \mathcal{P}_{ij}$, we arrive at the following inequality from (8.3b) [JR98, FTCMM02]:

$$\begin{aligned}
\Delta V_{ij}(x) &= \mathrm{PWQ}_j(\tilde{A}_i x + \tilde{g}_i) - \mathrm{PWQ}_i(x), && \text{(8.5a)}\\
&= \bar{x}^T \begin{bmatrix} \Delta V_{ij}^{(2)} & \Delta V_{ij}^{(1)}, \\ \Delta (V_{ij}^{(1)})^T & \Delta V_{ij}^{(0)} \end{bmatrix} \bar{x} && \text{(8.5b)}\\
&\leq \bar{x}^T \left(- \begin{bmatrix} -H_{ij}^T \\ K_{ij}^T \end{bmatrix} N_{ij} [-H_{ij} \;\; K_{ij}] - \rho \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \bar{x}, && \text{(8.5c)}\\
&\leq -\rho x^T x, && \text{(8.5d)}
\end{aligned}$$

where $\Delta V_{ij}^{(2)}, \Delta V_{ij}^{(1)}, \Delta V_{ij}^{(0)}$ are given by

$$\begin{aligned}
\Delta V_{ij}^{(2)} &= \tilde{A}_i^T V_j^{(2)} \tilde{A}_i - V_i^{(2)}, \\
\Delta V_{ij}^{(1)} &= \frac{1}{2} \left(2 \tilde{A}_i^T V_j^{(2)} \tilde{g}_i + \tilde{A}_i^T V_j^{(1)} - V_i^{(1)} \right), \\
\Delta V_{ij}^{(0)} &= \tilde{g}_i^T V_j^{(2)} \tilde{g}_i + V_j^{(0)} + \tilde{g}_i^T V_j^{(1)} - V_i^{(0)}.
\end{aligned}$$

The S-procedure is applied in (8.5c). The matrix N_{ij} consists of arbitrary non-negative elements only and the matrices H_{ij}, K_{ij} are defined by (8.1). Note that the term in (8.5c) is smaller than the term in (8.5d), if the state x is inside the set $\mathcal{P}_{ij}$. This follows from (8.1).

Remark 8.3.1 *Ideally we would want $\Delta V_{ij}(x) \leq -\rho x^T x$, $\forall x \in \mathcal{P}_{ij}$ and $\Delta V_{ij}(x)$ arbitrary for $x \notin \mathcal{P}_{ij}$. Since this constraint is non-convex, we relax this condition by imposing that $\Delta V_{ij}(x) \leq -\rho x^T x$ for all x in a quadratic surface containing $\mathcal{P}_{ij}$ in* (8.5c). *This constraint can be made convex by applying the S-procedure [BGFB94]. Since the only constraint on N_{ij} in* (8.5c) *is the non-negativity of its elements, the shape of this quadratic surface can be (almost) arbitrarily chosen.*

$\exists \lambda_i \geq 0 : f(x) \geq \sum \lambda_i g_i(x)$.

It is now possible to pose the SDP associated with (8.3):

$$\begin{aligned}
&\text{find} \quad \text{PWQ}_r,\ N_r,\ N_{ij},\ \rho,\ \epsilon, \quad \text{s.t.} \quad \forall r \in \mathcal{R},\ \forall i,j \in \mathcal{T}, \quad \text{subj. to} \\
&\begin{bmatrix} -\Delta V_{ij}^{(2)} - \rho I & -\Delta V_{ij}^{(1)} \\ -\Delta (V_{ij}^{(1)})^T & -\Delta V_{ij}^{(0)} \end{bmatrix} \succeq \begin{bmatrix} -H_{ij}^T \\ K_{ij}^T \end{bmatrix} N_{ij} \begin{bmatrix} -H_{ij} & K_{ij} \end{bmatrix}, \qquad (8.6a) \\
&\begin{bmatrix} V_r^{(2)} - \epsilon I & \frac{1}{2} V_r^{(1)} \\ \frac{1}{2} (V_r^{(1)})^T & V_r^{(0)} \end{bmatrix} \succeq \begin{bmatrix} -H_r^T \\ K_r^T \end{bmatrix} N_r \begin{bmatrix} -H_r & K_r \end{bmatrix}, \qquad (8.6b) \\
&N_{ij} \geq 0, \quad N_r \geq 0, \qquad \rho > 0, \quad \epsilon > 0, \qquad (8.6c) \\
&N_r = N_r^T, N_r \in \mathbb{R}^{d_r \times d_r}, \qquad N_{ij} = N_{ij}^T, N_{ij} \in \mathbb{R}^{d_{ij} \times d_{ij}} \\
&V_q^{(0)} = 0, \quad V_q^{(1)} = 0 \in \mathbb{R}^n, \quad \forall\, q \in \mathcal{R}_0, \qquad \mathcal{R}_0 \triangleq \{r \in \mathcal{R} \mid 0 \in \mathcal{P}_r\}. \qquad (8.6d)
\end{aligned}$$

It follows from (8.5) that (8.6a) induces $\Delta V_{ij}(x) \leq -\rho x^T x$. Inequality (8.6b) ascertains that the PWQ Lyapunov function is bounded from below by a quadratic function and (8.6c) ensures that all elements of N_r and N_{ij} are nonnegative while d_r and d_{ij} denote the number of rows of H_r and H_{ij}, which are defined by $\mathcal{P}_r = \{x \in \mathbb{R}^n \mid H_r\, x \leq K_r\}$ and (8.1). As elaborated in Remark (8.2.4) for PWA functions, the quadratic upper bound on the PWQ function does not need to be enforced here, since the Lyapunov function is quadratic around the origin (see (8.6d)) and PWQ on the rest of the state space.

Note that equation (8.6a) is sufficient (not necessary) for $\Delta V_{ij}(x) \leq -\rho x^T x$ as follows from (8.5). Hence, the SDP formulation is still conservative and may not yield a solution even if it exists. The scalar parameters ϵ and ρ are arbitrarily small and greater than zero in order to enforce a strictly positive PWQ function and exponential stability, respectively.

Theorem 8.3.2 (Exponential Stability Guarantee via SDP, [GM04]) *If the SDP* (8.6) *associated with an autonomous PWA system of type* (6.2) *is feasible, then this system is exponentially stable.*

Proof The conditions in (8.3) are sufficient for exponential stability according to Definition 8.1.6, since $\mathcal{S}_{\text{PWA}}^{inv} = \bigcup_{i,j \in \mathcal{T}} \mathcal{P}_{ij} = \bigcup_{i \in \mathcal{R}} \mathcal{P}_i$ according to Assumption 8.1.1. We therefore need to show that (8.6) implies (8.3). It follows from (8.5) that (8.6a) implies (8.3b). Furthermore (8.6b) implies that there exists a lower quadratic bound on the PWQ Lyapunov function. A quadratic upper bound exists automatically due to (8.6d). Hence, (8.6) implies (8.3). □

When computing a common quadratic Lyapunov function $x^T V^{(2)} x$ the problem formulation (8.6) can be drastically simplified. Specifically it is sufficient to impose

$$\text{find} \quad V^{(2)} \succ 0, \ \rho > 0, \tag{8.7a}$$

$$-\begin{bmatrix} A_r^T V^{(2)} A_r - V^{(2)} + \rho I & A_r^T V^{(2)} g_r \\ (A_r^T V^{(2)} g_r)^T & g_r^T V^{(2)} g_r \end{bmatrix} \succeq \begin{bmatrix} -H_r^T \\ K_r^T \end{bmatrix} N_r \begin{bmatrix} -H_r & K_r \end{bmatrix}, \ \forall r \in \mathcal{R}, \tag{8.7b}$$

$$N_r \geq 0, \quad N_r = N_r^T, \quad N_r \in \mathbb{R}^{d_r \times d_r}, \tag{8.7c}$$

where H_r and K_r are defined by the controller region r (i.e. $\mathcal{P}_r = \{x \in \mathbb{R}^n \mid H_r x \leq K_r\}$) and (8.7c) is used to enforce that each element of the matrix N_r is non-negative. In (8.7), the number of constraints is linear in the number of regions R while they are quadratic in (8.6), when searching for a PWQ Lyapunov function.

8.4 Computation of Piecewise Polynomial Lyapunov Functions for PWA Systems

It will be shown in the following how to formulate the search for a polynomial or piecewise polynomial Lyapunov function guaranteeing asymptotic stability for autonomous PWA systems by using sum-of-squares (SOS) methods [Par03]. This problem has been investigated for continuous time PWA systems in [PP03].

Before describing the use of SOS for Lyapunov functions, a brief introduction to SOS theory is in order. A multivariate polynomial $p(x)$ is a sum of squares if there exist polynomials $p_1(x) \dots p_m(x)$ such that $p(x) = \sum_{i=1}^m p_i^2(x)$. Equivalently,

$$p(x) = Z(x)^T Q Z(x), \tag{8.8}$$

where $Z(x)$ is a vector of monomials (e.g. $x \in \mathbb{R}^2$ and $Z(x)$ of order $k = 2$ implies $Z(x) = [1 \ x_1 \ x_2 \ x_1 x_2 \ x_1^2 \ x_2^2]^T$) and Q is a positive semi-definite matrix. Being a sum of squares immediately implies non-negativity of $p(x)$, a condition that otherwise is very

hard to prove[3]. The computation of an SOS decomposition can be through a semi-definite program, which can be solved efficiently. These properties lend themselves very conveniently to the construction of Lyapunov functions. As we move to higher order polynomials, there are more degrees of freedom to choose for the Lyapunov function and this implies that there is a higher probability of finding a Lyapunov function, if one exists. It is also possible to use SOS techniques when applying the S-procedure, e.g. it is possible to replace each element in the matrix N_r in (8.6b) with an SOS function. Higher order functions allow better approximations of the polytopic regions over which the Lyapunov function constraints are imposed, hopefully leading to a further reduction in conservativeness.

Specifically, we aim to find a piecewise polynomial (PWP) Lyapunov function $\mathrm{PWP}(x)$ of degree k, where k is a positive even number, defined by polynomials $\mathrm{PWP}_r(x)$ over each polytopic region $\mathcal{P}_r$. In the same vein as for the piecewise quadratic case, define $\Delta V_{ij} = \mathrm{PWP}_j(A_i x + g_i) - \mathrm{PWP}_i(x)$. For a stability certificate, we need

$$\mathrm{PWP}_r(x) \geq \alpha x^T x \quad \forall x \in \mathcal{P}_r, \forall r \in \mathcal{R}, \tag{8.9a}$$

$$\Delta V_{ij}(x) \leq -\rho x^T x, \ \forall x \in \mathcal{P}_{ij}, \forall\, i, j \in \mathcal{T}. \tag{8.9b}$$

In the following, let $G(x) = K - Hx$ and $G_{(i)}(x)$ denote the ith row of $G(x)$. Here, H and K define a polytopic set $\mathcal{P} = \{x \in \mathbb{R}^n \mid Hx \leq K\}$. As in the piecewise quadratic case, we can use the S-procedure to eliminate the polytopic regions, i.e. add terms of the type $G_{(i)}(x) N_{(ij)} G_{(j)}(x)$ to the constraints. However, nothing prevents us from using higher order multipliers $N_{(ij)}$, i.e. parameterize the elements $N_{(ij)}(x)$ as positive polynomials. To allow for even more degrees of freedom, we can also add terms of the form $G_{(i)}(x)G_{(j)}(x)G_{(k)}(x)G_{(l)}(x)$ and so on.

To calculate a piecewise polynomial Lyapunov function, we apply a higher order S-procedure to (8.9) and replace non-negativity constraints with SOS constraints. The SOS program will be

$$\mathrm{PWP}_r(x) - \alpha x^T x = S_r(x) \quad \forall r \in \mathcal{R}, \tag{8.10a}$$

$$-\rho x^T x - \Delta V_{ij}(x) = S_{ij}(x), \quad \forall i, j \in \mathcal{T}. \tag{8.10b}$$

[3]Non-negativity does however not imply that the polynomial can be written as a SOS. It is only a sufficient condition for non-negativity.

The S-procedure terms $S_q(x)$ are defined by

$$S_q(x) = F_0^q(x) + \sum_{i_1=1}^{m}\sum_{i_2=1}^{m} F_{i_1 i_2}^q(x) G_{(i_1)}(x) G_{(i_2)}(x) + \dots$$
$$+ \sum_{i_1=1}^{m}\sum_{i_2=1}^{m} \dots \sum_{i_k=1}^{m} F_{i_1 i_2 \dots i_k}^q(x) G_{(i_1)}(x) G_{(i_2)}(x) \dots G_{(i_k)}(x), \tag{8.11}$$

where $F_0(x)$ is an SOS polynomial of degree k, $F_{i_1 i_2}(x)$ of degree $k-2$ and so on and the functions $G_{i_k}(x)$ are defined by the sets $\mathcal{P}_r$ and $\mathcal{P}_{ij}$ respectively. By constraining all functions $F^q(x)$ to be SOS, we can ensure that $S_q(x)$ is non-negative if $x \in \mathcal{P}_q$. This is a more powerful condition compared to the SDP based S-procedure described in the previous section.

The SOS problem for a common polynomial function can be formulated along the same lines as the common quadratic function scheme described in Section 8.3. We will refrain from a detailed discussion here since the modifications to (8.7) are straightforward.

Regarding complexity, each SOS condition of degree k involves a vector of monomials $Z(x)$ (see (8.8)) from degree 1 to d, where $d = \frac{k}{2}$. For an n dimensional problem, the total number of monomials is $\binom{n+d}{d}$. This translates to solving an SDP of size $\binom{n+d}{d} \times \binom{n+d}{d}$ [Par03]. Every positivity constraint for a region (8.10a) or decay constraints between two regions (8.10b) is a single such SOS constraint of degree k.

Furthermore, each SOS multiplier condition $F^q(x)$ (see (8.11)) of degree $l = 0, 2 \dots k$ adds a semi-definite constraint with the size determined by l. Consider a constraint of the type (8.10a) or (8.10b) over a polytope defined by m half-spaces. There would be $\binom{m}{k-l}$ SOS multipliers of degree l for this single constraint corresponding to different combinations of $G_{i_1}(x) G_{i_2}(x) \dots G_{i_l}(x)$, each of which corresponds to a semi-definite constraint of size $\binom{n+l}{l} \times \binom{n+l}{l}$. Note that there is no benefit in choosing the S-procedure terms to be of higher order than the Lyapunov function $V(x)$. Ideally, they are of equal order. The rapid growth in problem size places a practical limit on the order of Lyapunov functions (8.10) and the order of the S-procedure terms (8.11) which can be computed for medium sized PWA systems (i.e. several hundred regions).

8.5 Robust Convergence of Piecewise Affine Systems Subject to Bounded Disturbances

The contribution of this section is a method for testing robust convergence of PWA systems to the minimal robust invariant set. To the authors' knowledge, there are no previous results in the literature which address the topic of robust convergence analysis of PWA systems.

Specifically, we will consider the autonomous PWA systems (6.2), i.e

$$x(k+1) = f_a(x(k), w(k)) = \tilde{A}_r x(k) + \tilde{g}_r + w(k), \qquad \text{if } x(k) \in \mathcal{P}_r,\ r \in \mathcal{R},$$

with $w(k) \in \mathbb{W}$, where $\mathbb{W}$ is a polytope.

8.5.1 Conditions for Robust Convergence

This section will introduce the definitions and the key theorem which will be applied in the subsequent section.

Since it is not possible for *any* dynamical system subject to additive uncertainty to exhibit robust asymptotic stability, we aim to show that all feasible states $x \in \mathcal{X}_N$ converge to the minimal robust invariant set $\mathcal{F}_\infty$ (see Definition 7.1.2) for all possible disturbance sequences. This behavior is here referred to as *Robust Convergence* and is defined by the following:

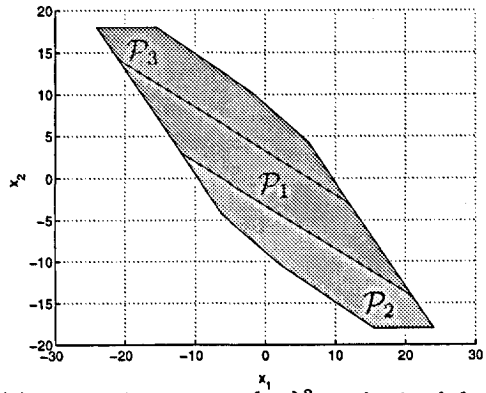
Definition 8.5.1 (Robust Convergence) *The autonomous PWA system (11.4) is robustly convergent on the set $\mathcal{X}_N \subseteq \mathbb{R}^n$, if $x_0 \in \mathcal{X}_N$ implies $d_H(x_N, \mathcal{F}_\infty) \to 0$ for $N \to \infty$. Here, $d_H(\cdot,\cdot)$ denotes the Hausdorff distance and $\mathcal{F}_\infty$ the minimal robust invariant set of system (11.4).*

Definition 8.5.2 (Hausdorff Distance) *If Ω and Φ are two non-empty, compact sets in $\mathbb{R}^n$, then the* Hausdorff Distance *is defined as*

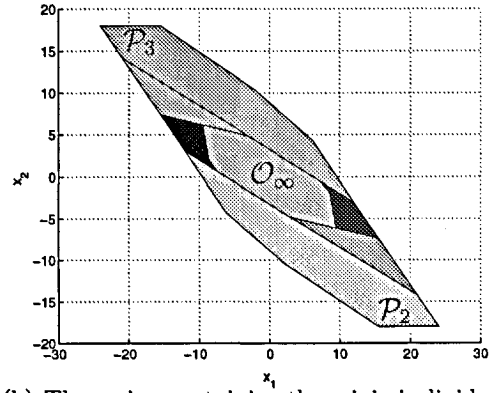
$$d_H^p(\Omega, \Phi) \triangleq \max\{\sup_{\phi\in\Phi} d(\phi, \Omega), \sup_{\omega\in\Omega} d(\omega, \Phi)\}$$

where

$$d(x, \mathcal{A}) \triangleq \inf_{y\in\mathcal{A}} ||x - y||_p$$



(a) Original partition $\{\mathcal{P}_r\}_{r=1}^{3}$ obtained for Example 11.4.1 in Section 11.4 by solving an mp-QP for $N = 1$.



(b) The region containing the origin is divided into the maximal robust control invariant set $\mathcal{O}_\infty \subseteq \mathcal{P}_1$ and a finite number of other convex sets into a new partition $\mathcal{X}_N \setminus \mathcal{O}_\infty = \{\mathcal{P}_r\}_{r=1}^{8}$ (total of 8 regions).

Figure 8.1: Procedure of extracting a robust invariant set from the partition. The partition was obtained from Example 11.4.1 in Section 11.4 for additive noise $w \in \mathbb{R}^2$, $\|w\|_\infty \leq 1.6$.

In order to prove robust convergence, we will show that it is sufficient to find a function $V(x)$ which satisfies the following constraints,

$$\beta\|x\|^p \geq V(x) \geq \alpha\|x\|^p, \qquad \alpha, \beta > 0,\ p \geq 1,\ \ \forall x \in \mathcal{X}_N \setminus \mathcal{O}, \tag{8.12a}$$

$$V(f_a(x,w)) - V(x) \leq -\rho\|x\|^p, \qquad \forall x \in \mathcal{X}_N \setminus \mathcal{O},\ \forall w \in \mathbb{W}, \tag{8.12b}$$

where $\mathcal{O}$ is a robust invariant set with $0 \in \text{int}(\mathcal{O})$. Since $\mathcal{X}_N$ is bounded and robust invariant, the existence of such a function will directly imply that all states $x \in \mathcal{X}_N$ enter the robust invariant set $\mathcal{O}$ in finite time. Note that (8.12) is defined only over $\mathcal{X}_N \setminus \mathcal{O}$, because the condition in (8.12b) cannot be satisfied for all $x \in \mathcal{O}$, e.g. assume $x = 0$ and $w \neq 0$. Hence, when searching for $V(x)$ it is necessary to remove the set of states for which (8.12b) cannot be satisfied. In practice, we recommend to apply the algorithm in [KG98] to remove the maximal robust invariant set $\mathcal{O}_\infty \subseteq \mathcal{P}_1$ around the origin from the partition $\mathcal{X}_N = \bigcup_{r=1,\ldots,R} \mathcal{P}_r$, as is depicted in Figure 8.1.

Remark 8.5.3 *There is no guarantee that the maximal robust invariant set contained in $\mathcal{P}_1$ is non-empty. If $\mathcal{O}_\infty \subseteq \mathcal{P}_1$ is empty, one can attempt to find invariant sets which are subsets of multiple regions indexed by $\mathcal{R}$, i.e. find $\mathcal{O}_\infty \subseteq \bigcup_{r\in\mathcal{R}} \mathcal{P}_r$*

with the methods in Chapter 7. If this is not possible for any index set $\mathcal{R}$, no robust invariant set around the origin exists and hence the system is unstable.

Theorem 8.5.4 (Robust Convergence Conditions, [GM04]) *Consider an autonomous PWA system* (11.4). *Assume $\mathcal{X}_N$ is bounded and robust invariant and a function $V(x)$ exists such that* (8.12) *holds. Then there exists a finite time k^* such that $x(0) \in \mathcal{X}_N \Rightarrow x(k^*) \in \mathcal{O}$. Furthermore, if the system is nominally stable on the robust invariant set $\mathcal{O}$ with $0 \in \text{int}(\mathcal{O})$, then the system is robust convergent.*

Proof It follows from (8.12b) and $0 \in \text{int}(\mathcal{O})$ that the decay rate in the function $V(x)$ is bounded by $-\rho\|x\|^p$ $(p \geq 1)$ and is thus finitely determined, i.e. its absolute value cannot be arbitrarily small. Since the robust invariant set $\mathcal{X}_N$ and the function $V(x)$ are bounded, it follows that the state must enter $\mathcal{O}$ in a finite number of steps. Finally, if the nominal system is asymptotically stable on $\mathcal{O}$, it follows from [KG98] (see also Theorem 7.3.2) that the state will converge to the minimal robust invariant set $\mathcal{F}_\infty$. □

8.5.2 Robust Convergence via Quadratic Functions

In this section we will show how to construct a quadratic function $V(x)$ such that the constraints in (8.12) are satisfied.

The constraints on the decay function $V(x)$ in (8.12) are infinite dimensional, i.e. it is necessary to impose them for all possible combinations of uncertainties and disturbances. In such cases it is common to only consider the vertices of the disturbance/uncertainties (e.g. [KBM96]). However, in order for such a formulation to imply that the conditions in (8.12) hold for all disturbance/uncertainty combinations, it is necessary for $V(x)$ to be convex. Hence, it is not possible to search for a PWQ function $V(x)$ as in Section 8.3. Therefore, we will show here how a convex quadratic function $V(x) = x^T P_L x$ which satisfies the conditions in (8.12) can be found by solving an SDP.

Since the quadratic function $V(x)$ is globally defined, no reachability analysis needs

to be performed and we can directly formulate the SDP:

$$\text{find } P_L \succ 0 \tag{8.13a}$$

$$\text{s.t. } x_1^T P_L x_1 - x_0^T P_L x_0 \leq -\rho \|x_0\|^2, \tag{8.13b}$$

$$x_1 = f_a(x_0, w),\ \forall x_0 \in \mathcal{X}_N \setminus \mathcal{O},\ \forall w \in \text{vert}(\mathbb{W}),\ \rho > 0. \tag{8.13c}$$

Theorem 8.5.5 (Robust Convergence via Quadratic Function, [GM04]) *Consider an autonomous PWA system* (11.4). *Assume $\mathcal{X}_N$ is bounded and robust invariant and a quadratic matrix P_L satisfying (8.13) is found. Then there exists a finite time k^* such that $x(0) \in \mathcal{X}_N \Rightarrow x(k^*) \in \mathcal{O}$. If the system is nominally stable on the robust invariant set $\mathcal{O}$ with $0 \in \text{int}(\mathcal{O})$ then the system is robust convergent and all states will enter the set $\mathcal{O}$ in finite time.*

Proof We only need to show here that (8.13) implies (8.12).

The function $V(x) = x^T P_L x$ is guaranteed to decrease for all extreme disturbance/uncertainty combinations and for all $x \in \mathcal{X}_N$. Furthermore, the function $V(x)$, the system dynamics (11.1) as well as all constraints are convex. Therefore, each state will remain in the convex hull of the extreme disturbance / uncertainty combination considered in (8.13) at the next time step. Hence, (8.13b) is sufficient for (8.12b). Furthermore, condition (8.13a) is trivially sufficient for the positivity constraint (8.12a). Therefore the rest of the proof follows directly from Theorem 8.5.4. □

Obviously, the use of a quadratic function $V(x)$ is very restrictive. However, if a quadratic function $V(x)$ satisfying (8.12) exists, formulation (8.13) is guaranteed to yield a solution. Furthermore, it is straightforward to extend the stability analysis scheme presented here to search for general convex polynomial functions $V(x)$ by applying sum-of-squares methods [Par03].

Remark 8.5.6 *If the SDP* (8.13) *is infeasible, it is advisable to re-solve the problem for a different robust invariant set $\mathcal{O}$. Slight modifications to the set $\mathcal{O}$ may make the subsequent stability analysis feasible.*

8.6 Tuning Parameters

As stated in the previous sections, the complexity of the various Lyapunov function computation schemes can be prohibitive for large partitions. This will also be illustrated by the case study in Section 8.7. Hence, this section will discuss modifications to the previously introduced problem formulations which make the associated computations more efficient.

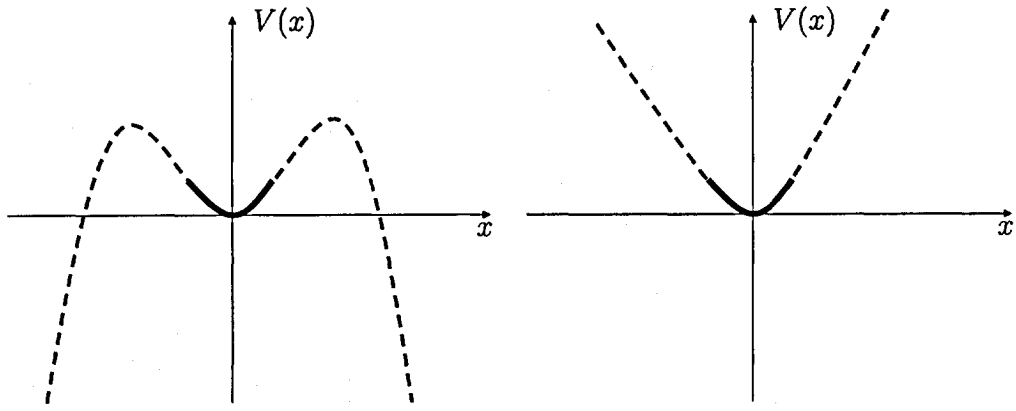
Fixed Exterior Ellipsoids: The standard S-procedure described in Section 8.3 achieves the objective of ensuring positivity over a particular region by approximating that region with a quadratic surface (see (8.5c)). This version of the S-procedure is not lossless [BGFB94], but the shape of the quadratic surface is a degree of freedom in the solution. As an alternative, it is possible to fix the surface a priori, e.g. by selecting it to be the minimal volume ellipsoid containing the region. The associated S-procedure is lossless and the degrees of freedom in the SDP are significantly reduced. The new problem formulation is given by

$$V(x) - \tau\big(1 - (x - x_0)^T E(x - x_0)\big) \geq 0$$

where the minimum volume exterior ellipsoid is defined by $(x-x_0)^T E(x-x_0) \leq 1$. Here, there is just one free variable τ, compared to the $\frac{m(m-1)}{2}$ free variables in (8.5c), where m is the number of half-spaces defining the polytope.

However, as stated in [Joh02], using the exterior ellipsoid is always more conservative than using the S-procedure in (8.5c). Therefore, for simple partitions, the ellipsoidal approach may be more of a liability due to the effort required for the calculation of the exterior ellipsoid and the increased conservativeness. The computational advantages are more discernible for partitions with a large number of regions, for which a significant reduction in the number of decision variables is achieved.

Upper Bound Constraints: The upper bound constraints ($V(x) \leq \beta\|x\|^p$) in Theorems 8.1.5 and 8.1.6 can be omitted by enforcing a certain structure upon the Lyapunov function around the origin. For example, when searching for PWQ functions it is sufficient to enforce that the function has no linear and offset terms ($L_r = 0$, $C_r = 0$) for the regions containing the origin ($0 \in \mathcal{P}_r$)



(a) Unstable system: The decay rate constraint is not sufficient to imply existence of lower bound.

(b) Stable system: The decay rate constraint is sufficient to imply existence of lower bound.

Figure 8.2: Illustration of two feasible solutions (dashed lines) according to Theorem 8.1.6, if no lower bound constraint is imposed on the Lyapunov function. Note that positivity around the origin is always enforced (constraint depicted in bold).

and for PWA functions it is sufficient to enforce that the function has no offset terms ($C_r = 0$) for the regions containing the origin (see Remark 8.2.4). If this structure is imposed, it follows trivially that an upper bound on $V(x)$ exists, and the associated constraints can be omitted.

Lower Bound Constraints: In many practical cases, it is advisable to completely discard the lower bound constraint ($\alpha\|x\|^p \leq V(x)$) for regions that do not contain the origin [Joh02]. Fewer constraints will result in fewer S-procedure variables and faster runtime. For asymptotically stable systems, the decay constraint will directly imply that a lower bound exists (see Figure 8.2(b)). Since this is not true for unstable systems (see Figure 8.2(a)), it is still necessary to check the existence of a lower bound, i.e. after solving the Lyapunov SDP, a second SDP needs to be solved to verify that a lower bound exists. Since the complexity of SDP solvers is polynomial this 'divide-and-conquer' approach will typically result in faster runtime.

Degree of SOS Multipliers: In SOS-schemes, the degree of the polynomial functions is a crucial influence on the likelihood of finding a Lyapunov function. In order for the constraints in (8.10) not to be too conservative, it is advisable to select the Lyapunov polynomial to be of the same order as the associated S-procedure terms. However, as we increase the degree of the SOS functions, the number of variables in the associated SDP problem increases fast[4], placing a practical limit on tractable problem sizes. Hence, it may sometimes be advisable to select the S-procedure multipliers of a lower order than the associated Lyapunov function, in order to keep the degrees of freedom limited.

8.7 Case Study

8.7.1 Problem Setup

The aim of this case study is to gain insights on the efficiency of the various stability analysis schemes. To this end, it is necessary to consider a broad class of PWA systems in our analysis. We will describe the PWA systems which were investigated and then motivate their selection towards the end of this section. All results in this section are taken from the survey [BGLM05].

The systems considered in our case study are constrained LTI and PWA systems subject to optimal PWA state feedback control. The LTI systems were selected as open-loop stable and unstable systems of order 2 and 3 with one input. The selected PWA systems were of order 2 with one input. The PWA systems were created by assuming random dynamic matrices[5] defined over four random non-overlapping polytopes, whose union covers the feasible state space. The elements of the dynamic matrices for LTI and PWA systems were assigned random values between -2 and $+2$. For both LTI and PWA systems the system inputs and states were constrained to

$$\|u(k)\|_\infty \leq 1 \quad \text{and} \quad \|x(k)\|_\infty \leq 10, \ \forall k \geq 0. \tag{8.14}$$

In a second step, these systems were subjected to optimal PWA feedback control

[4] Depending on the specific problem formulation the complexity is either roughly $(\frac{1}{2(d!)^2} - \frac{1}{(2d)!})n^{2d}$ or $\frac{1}{(2d)!}n^{2d}$ [Par03].

[5] Analyzing the stability of generic PWA systems is an NP-hard problem [BGT00]. Hence, it is impossible to construct random open-loop stable and unstable PWA systems.

(see Remark 6.1.1 or Chapter 3 for details), such that an autonomous PWA system,

$$x(k+1) = \tilde{A}_r x(k) + \tilde{g}_r, \qquad \text{if } x(k) \in \mathcal{P}_r,\ r \in \mathcal{R},$$

is obtained. The control objective was defined by

$$J^*(x(k)) = \sum_{k=0}^{N-1} \min_u \|Q_x x(k+1)\|_p + \|Q_u u(k)\|_p \tag{8.15}$$

using both the standard squared Euclidean norm ($p = 2$) and linear norms ($p = 1$ and $p = \infty$). To make things interesting, the weights in (8.15) were set to $Q_u = 10I$ and $Q_x = I$, such that the expensive control action may easily lead to unstable closed-loop behavior.

In order to ascertain that the closed-loop system is invariant, the control schemes in [GM03, GKBM04a] (see Chapter 11 and 16, respectively) were applied. In [GM03, GKBM04a], control invariance is achieved by posing a receding horizon control problem with an invariance constraint on the first state, i.e. the state at time $k+1$ is restricted to be contained inside the maximal control invariant set.

Although the resulting PWA partitions are guaranteed to be invariant, there is no guarantee of asymptotic stability. Therefore, the design schemes in [GM03, GKBM04a] rely heavily on the stability analysis of PWA systems investigated here and the stability results are of practical relevance.

The PWA partitions[6] considered here were obtained for prediction horizons $N = 1, 3, 5$. The partitions consisted of 9 to 201 regions with 9 to 515 associated transitions. We chose relatively small systems since this allowed us to perform the case study on a large number of systems within a reasonable amount of time.

All computations were carried out on Pentium IV driven PCs, running MATLAB, the Multi-Parametric Toolbox [KGB04], YALMIP [Löf04] and SOSTools [PPSP04]. Note that YALMIP and SOSTools are currently the only available solvers for SOS problems.

Finally, we will now motivate our selection of systems which we analyzed. The choice of systems was mainly driven by two objectives: the stability analysis must have practical relevance and the PWA partition must be invariant (Assumption 8.1.1).

[6] All of the systems considered here can be downloaded from [KGB04].

Both of these objectives are naturally met by the controller partitions considered here. Finally, the scheme in [GM03, GKBM04a] yields PWA systems of relatively low complexity. Therefore, the systems used in this case study are a good choice for the stated reasons of invariance, practical relevance and low complexity. Note that it is *not* possible to consider random bounded PWA partitions directly since these will not be invariant, in general.

8.7.2 Numerical Results - Specific Systems

Before presenting the results of the random system case study in Section 8.7.3, we will focus on specific systems which exhibit certain properties that we wish to highlight.

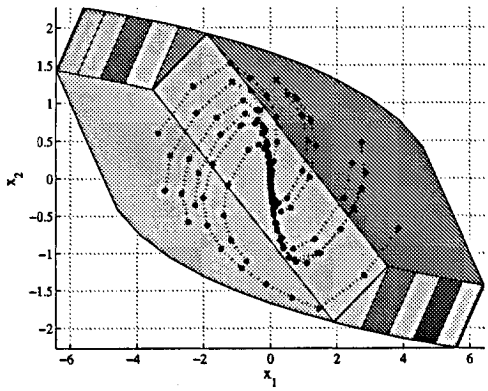
Example 8.7.1 *Consider the unstable* 2^{nd} *order system with one input defined by*

$$x(k+1) = \begin{bmatrix} 1.2 & 1.2 \\ 0 & 1.2 \end{bmatrix} x(k) + \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix} u(k). \tag{8.16}$$

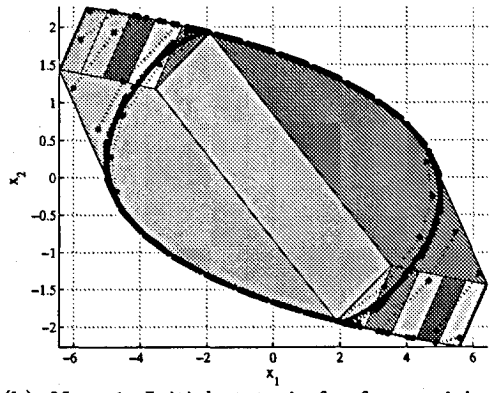
The system is subject to the constraints $\|x(k)\|_\infty \leq 5$ *and* $\|u(k)\|_\infty \leq 1$, $\forall k \geq 1$. *The control objective in* (8.15) *is defined by the* 2*-norm and the weights* $Q_x = I$ *and* $Q_u = 1$.

If we apply an optimal controller as in [GM03] with prediction horizon $N = 1$ to Example 8.7.1, all stability analysis schemes considered here (PWA, PWQ, piecewise SOS up to fourth order) fail. When simulating closed loop trajectories, one can observe that the system converges to the origin if the initial state is close to the origin, see Figure 8.3(a). However, if the initial state is further away, the system reaches a limit cycle, as is depicted in Figure 8.3(b). Hence, the system is indeed not asymptotically stable.

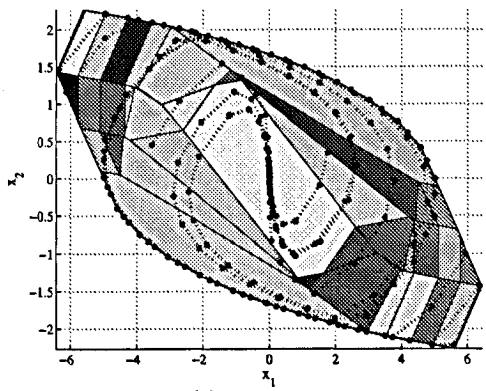
If we increase the prediction horizon to $N = 3$, the convergent closed-loop trajectories in Figure 8.3(c) are obtained. However, none of the techniques considered here succeeds in finding a Lyapunov function. If the prediction horizon is increased to $N = 5$, the system is stable as can be seen from the trajectories in Figure 8.3(d). For the resulting partition, it is not possible to find common quadratic or common quartic Lyapunov functions, while piecewise quadratic and piecewise quartic functions can be constructed.



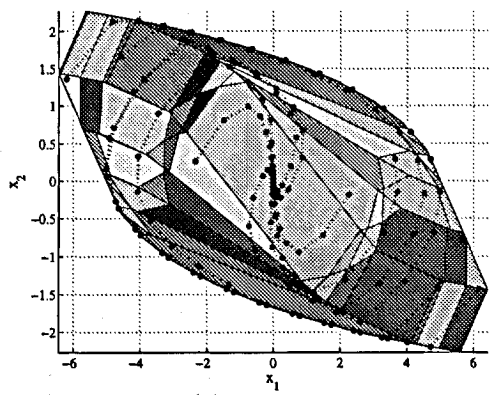
(a) $N = 1$, Initial state is close to origin: Convergent Trajectories.



(b) $N = 1$, Initial state is far from origin: Limit Cycle Trajectories.



(c) $N = 3$



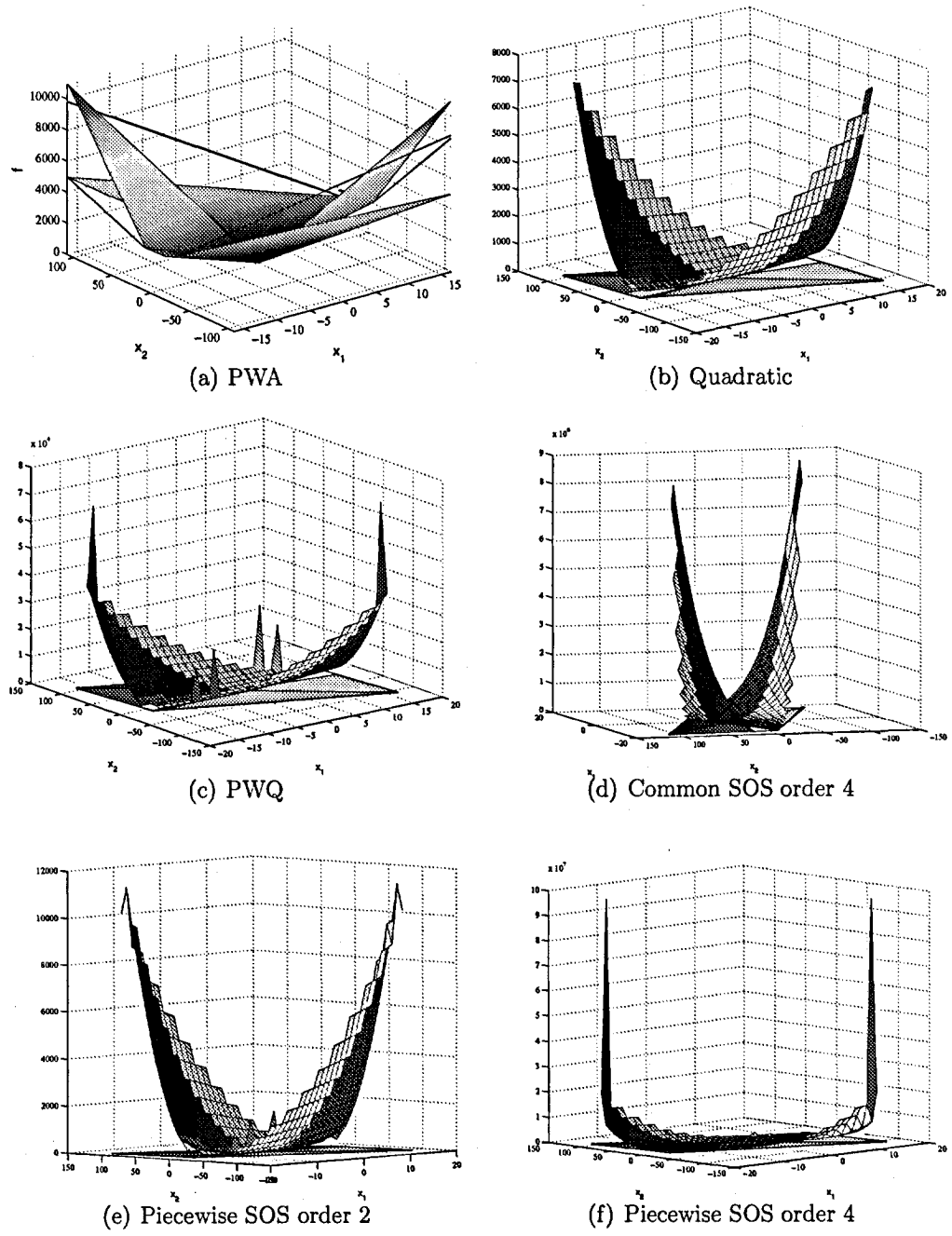
(d) $N = 5$

Figure 8.3: Closed-loop trajectories for Example 8.7.1 for various prediction horizons N.

This simple example clearly illustrates the conservativeness of certain types of Lyapunov functions as well as the impact of the controller prediction horizon N on stability of the closed-loop system.

Figure 8.4 shows different Lyapunov functions for the PWA partition which is obtained when applying the control scheme in [GM03] with prediction horizon $N = 1$ to the following stable LTI system:

$$x(k+1) = \begin{bmatrix} 0.4734 & 0.6756 \\ 0.7353 & -0.1321 \end{bmatrix} x(k) + \begin{bmatrix} 0.4776 \\ 0.4459 \end{bmatrix} u(k). \tag{8.17}$$



(a) PWA (b) Quadratic

(c) PWQ (d) Common SOS order 4

(e) Piecewise SOS order 2 (f) Piecewise SOS order 4

Figure 8.4: Different Lyapunov functions for same controller.

8.7.3 Numerical Results - Random Systems

The results in this section were obtained by considering 100 random PWA systems and 200 random LTI systems subject to the control scheme in [GM03, GKBM04a] for prediction horizons $N = 1, 3$ and 5. Specifically, the likelihood of successfully computing a Lyapunov function as well as the associated computation time are given in Tables 8.1–8.3.

The solution time corresponds to the time spent in computing the solution to the problems as described in the previous sections. The setup time refers to the time required to do the pre-processing (e.g. reachability analysis, vertex enumeration) and actual setup of the constraints. All the SOS data provided in the following tables were obtained with YALMIP [Löf04]. See the discussion in Section 8.7.4 for details on the choice of solvers. Note that we have also used the SOS based code to construct both common quadratic and PWQ Lyapunov functions, for verification reasons. The computation times and success rates were consistent with the results obtained with the methods in Section 8.3 and are therefore not restated here.

If no Lyapunov function could be found with any method, we analyzed the corresponding partition through exhaustive simulation. For all LTI systems which exhibited convergent trajectories, we were able to construct Lyapunov functions. For the PWA systems however, we were able to find Lyapunov functions for only slightly more than 90% of all systems that exhibited convergent trajectories. An intuitive explanation for this behavior is the fact that PWA systems subject to control generally exhibit more complex closed-loop vector fields than their LTI equivalents. In the cases considered here, the PWA systems even exhibit discontinuous behavior across the dynamic boundaries. Hence, it is to be expected that Lyapunov functions are harder to find for PWA systems.

8.7.4 Discussion of Results

Construction of Lyapunov Function

- When the control scheme in [GM03] was applied to LTI systems (Tables 8.2 and 8.3), the stability analysis of the resulting PWA systems was always successful.

- The stability analysis of the PWA systems generated by using the control

Partitions obtained for 2^{nd} order PWA systems, control scheme [GKBM04a], $N = 1$						
	50 Systems, ∞ norm objective			50 Systems, 1 norm objective		
Method	Success Rate	Solution Time	Setup Time	Success Rate	Solution Time	Setup Time
Quadratic	22/45	1.0 sec.	0.6 sec.	9/46	1.2 sec.	0.7 sec.
Piecewise Affine	42/45	0.9 sec.	7.7 sec.	40/46	1.9 sec.	13.1 sec.
Piecewise Quadratic	43/45	5.4 sec.	9.5 sec.	38/46	7.0 sec.	10.7 sec.
Common SOS order 4	22/45	6.9 sec.	4.1 sec.	8/46	7.7 sec.	4.3 sec.
Piecewise SOS order 4	36/45	54.7 sec.	25.7 sec.	22/46	57.5 sec.	27.2 sec.

Table 8.1: The number of regions were between 29 and 201 with 63-515 transitions. 'Success' denotes the number of Lyapunov functions found out of the total number of systems with convergent trajectories, the 'Solution Time' is the average cpu-time required to solve the associated optimization problem and 'Setup Time' is the average time needed to pre-process the problem.

Partitions obtained for 3^{rd} order LTI systems, 2 norm objective, control scheme [GM03]						
	50 Stable Systems, $N = 1$			50 Unstable Systems, $N = 1$		
Method	Success Rate	Solution Time	Setup Time	Success Rate	Solution Time	Setup Time
Quadratic	49/50	0.6 sec.	0.2 sec.	47/50	0.6 sec.	0.02 sec.
Piecewise Quadratic	50/50	2.1 sec.	1.1 sec.	50/50	3.2 sec.	1.8 sec.
Common SOS order 4	50/50	2.7 sec.	1.4 sec.	47/50	3.8 sec.	2.0 sec.
Piecewise SOS order 4	39/50	8.5 sec.	4.0 sec.	13/50	22.9 sec.	9.2 sec.

Table 8.2: The number of regions were between 9 and 15 with 9-47 transitions. 'Success' denotes the number of Lyapunov functions found out of the total number of systems with convergent trajectories, the 'Solution Time' is the average cpu-time required to solve the associated optimization problem and 'Setup Time' is the average time needed to pre-process the problem.

Partitions obtained for unstable 2^{nd} order LTI systems, ∞ norm objective						
	50 Systems, $N = 3$			50 Systems, $N = 5$		
Method	Success Rate	Solution Time	Setup Time	Success Rate	Solution Time	Setup Time
Quadratic	28/28	1.1 sec.	0.9 sec.	17/23	2.1 sec.	1.4 sec.
Piecewise Affine	28/28	4.8 sec.	22.6 sec.	17/23	10.3 sec.	45.5 sec.
Piecewise Quadratic	28/28	6.8 sec.	12.2 sec.	17/23	10.6 sec.	23.9 sec.
Common SOS order 4	28/28	5.7 sec.	4.2 sec.	17/23	8.7 sec.	7.2 sec.
Piecewise SOS order 4	24/28	47.9 sec.	27.7 sec.	16/23	109.1 sec.	63.0 sec.

Table 8.3: For $N = 3$ the number of regions was between 40 and 72 with 56-154 transitions. For $N = 5$ the number of regions was between 70 and 184 with 100-363 transitions. 'Success' denotes the number of Lyapunov functions found out of the total number of systems with convergent trajectories, the 'Solution Time' is the average cpu-time required to solve the associated optimization problem and 'Setup Time' is the average time needed to pre-process the problem.

scheme in [GKBM04a] for PWA systems has a higher possibility of failure (Table 8.1). The failure rate of common quadratic and higher order polynomial approaches is the highest. It is much easier to find a Lyapunov function using piecewise techniques. This behavior coincides with our expectations, since the likelihood of finding a common Lyapunov function over completely different dynamics is relatively low.

- As expected, there were some cases where the PWA approach failed but the PWQ approach succeeded. On the other hand, we have also observed cases where the PWA approach succeeded while the PWQ approach failed.

- It is interesting to observe that the number of convergent closed-loop systems resulting from unstable LTI systems is much lower for linear performance objectives, even though the associated prediction horizons are much larger than for the quadratic objectives (see Tables 8.2 and 8.3). The cause of this may be the fact that the terminal weight Q_{x_N} was selected as the infinite horizon Riccati solution for quadratic objectives while $Q_{x_N} = Q_x$ for linear objectives.

Thus, the terminal cost was much higher for quadratic objectives.

- The large number of convergent closed-loop systems which were obtained by controlling PWA systems (see Table 8.1) is attributable to the fact that not all of the random PWA systems were unstable. Note that it is impossible to generate *random* PWA systems that are guaranteed to be unstable, since there is no easy way of confirming instability of a PWA system[7], apart from exhaustive simulation.

- The large number of constraints involving strict inequalities (e.g. $\rho > 0$) result in severe numerical problems for all analysis schemes. While it is relatively easy to 'patch' the standard LP or SDP approaches (e.g. PWA or PWQ Lyapunov functions) to deal with this issue by adding strictly positive slack variables to the constraints, this is not possible for SOS based approaches. This is mainly due to the fact that it is not obvious how to pose problem constraints which provide a result that remains an SOS function, even if the individual solution parameters are slightly perturbed, i.e. it is not obvious how to to add reasonable slack variables which ensure a solution in the *strict* interior of the feasible solution space.

- Theoretically, a higher order S-Procedure for piecewise Lyapunov functions as illustrated in (8.11) should have a positive influence on the likelihood of successful analysis, since it allows for higher order approximations of the polytopic regions. In practice, however, higher order multipliers result in more frequent numerical problems and the likelihood of successful analysis is decreased. The numerical problems associated with the SOS approaches occur regardless of SOS [Löf04, PPSP04] and SDP [Stu99, TTT99] solver. In hundreds of simulations we have not been able to find a PWA partition where the piecewise SOS approaches outperform the lower order piecewise SDP schemes. This observation does not hold for common Lyapunov functions.

[7] If each individual dynamic of a PWA system is unstable, this does not imply that the PWA system as a whole is unstable. The objective in this chapter is to find simple ways of ascertaining stability of PWA systems. In general, this problem is NP-hard [BGT00].

- Using the minimum volume exterior ellipsoids to reduce the number of variables has a surprisingly strong negative impact on the likelihood of successful analysis, i.e. it is *much* harder to find Lyapunov functions.

- Overall, we are able to give the following recommendations for the analysis of PWA systems: first try to construct a common quadratic Lyapunov function, since the associated computation is very cheap. Second, attempt to construct a PWQ function and finally, if the previous approaches fail, try constructing a PWA function.

Computation Time

- The overall computation time correlates directly with the number of regions and, more importantly, with the number of transitions, which occur between regions. In general, unstable LTI systems result in more complex partitions such that the associated stability analysis is more time consuming.

- In general, the linear cost objectives generate partitions comprising more regions than those obtained for quadratic cost objectives. Hence, the associated analysis is more time consuming.

- It was observed that YALMIP [Löf04] has much shorter setup times than SOSTools [PPSP04] and is equally reliable in terms of numerical robustness. Hence, YALMIP was used for the case study.

- Although the runtime of all schemes grows polynomially with the partition size, the order of growth is very large. Hence, the limit of applicability is quickly reached for the current software implementations. We have been able to construct PWQ Lyapunov functions for 4^{th} order PWA systems with 400 regions. The associated computations took several hours.

- Neglecting the lower bound constraints in the problem formulation as described in Section 8.6 leads to large speedups, especially in the SOS based cases.

- Exterior Ellipsoids: For the partition sizes considered here, no runtime benefit was obtained by computing the exterior ellipsoids a priori. A benefit may result for larger partitions.

8.8 Conclusion

An extensive survey of various methods of constructing Lyapunov functions for discrete-time PWA systems was presented in this chapter. First, the basic building blocks (e.g. reachability analysis) and assumptions (e.g. set invariance) were established. Second, it was shown how to construct PWA, PWQ and higher order piecewise polynomial Lyapunov functions for discrete-time systems. Subsequently, it was shown how to analyze robust convergence properties of autonomous PWA systems subject to additive disturbances.

Finally, the results of an extensive case study are given. The case study illustrates that simple Lyapunov functions (i.e. quadratic, PWA or PWQ) are generally sufficient for analyzing discrete-time PWA systems of the type considered here. Furthermore, we did not find a single PWA partition where the higher order piecewise SOS Lyapunov functions succeeded and the other methods failed.

All tools as well as the random systems considered in the case study can be downloaded from [KGB04].

Seite Leer /
Blank leaf

Part III

EFFICIENT CONTROL OF CONSTRAINED LINEAR SYSTEMS

Seite Leer /
Blank leaf

9

Problem Description

This part of the thesis will address the topic of efficient feedback control of discrete-time, linear, time-invariant (LTI) systems subject to constraints. Here, the term *efficient* refers to any scheme which is able to simplify or speed-up the process of controller design and/or application, compared to current techniques. A general introduction to feedback control of constrained LTI systems was given in Chapter 4. We will briefly recap the key issues, before giving an overview of the content of the next chapters.

Consider the LTI system

$$x(k+1) = Ax(k) + Bu(k), \tag{9.1}$$

subject to the constraints

$$x(k) \in \mathbb{X} \subseteq \mathbb{R}^n, \qquad u(k) \in \mathbb{U} \subseteq \mathbb{R}^m, \qquad k \geq 0. \tag{9.2}$$

Remark 9.1.1 *For ease of notation, we restrict ourselves to separate constraints on state and input in* (9.2)*. It is straightforward to modify all algorithms in this chapter to deal with systems subject to mixed constraints, i.e.* $C^x x(k) + C^u u(k) \leq C^c$, $\forall k \geq 0$.

We are interested in two types of regulation problems: problems with linear and quadratic objectives. The linear regulation problem

$$J_N^*(x(0)) = \min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(||Q_u u_k||_{1,\infty} + ||Q_x x_k||_{1,\infty} \right) + ||Q_{x_N} x_N||_{1,\infty}, \tag{9.3a}$$

$$\text{subj. to} \quad x_k \in \mathbb{X},\ u_{k-1} \in \mathbb{U}, \qquad \forall k \in \{1,\ldots,N\}, \tag{9.3b}$$

$$x_N \in \mathcal{T}_{\text{set}}, \tag{9.3c}$$

$$x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0), \tag{9.3d}$$

with the optimizer $U_N = [u_0^T, \ldots, u_{N-1}^T]^T$ can be recast as an LP [BBM00b, Bor03, RR00, ZM95, GN93, CM86] by substituting $x_k = A^k x(0) + \sum_{j=0}^{k-1} A^k B u_{k-1-j}$. Remark 5.2.6 provides sufficient conditions on the selection of the terminal weight matrix Q_{x_N} such that (9.3) applied in Receding Horizon Control (RHC) yields asymptotically stable trajectories.

For the $\|\cdot\|_\infty$ norm,

$$
\begin{aligned}
J_N^*(x(0)) \;=\; & \min_{\substack{U_N,\ \epsilon_0,\ldots,\epsilon_{N-1},\\ \delta_0,\ldots,\delta_{N-1},\gamma}} \sum_{k=0}^{N-1} (\epsilon_k + \delta_k) + \gamma && \text{(9.4a)}\\
\text{subj. to} \quad & GU_N \leq W + Ex(0), && \text{(9.4b)}\\
& Q_u u_k \leq \mathbf{1}\epsilon_k, \quad -Q_u u_k \leq \mathbf{1}\epsilon_k, \quad k = 0, \ldots, N-1, && \text{(9.4c)}\\
& Q_x x_k \leq \mathbf{1}\delta_k, \quad -Q_x x_k \leq \mathbf{1}\delta_k, \quad k = 0, \ldots, N-1, && \text{(9.4d)}\\
& Q_{x_N} x_N \leq \mathbf{1}\gamma, \quad -Q_{x_N} x_N \leq \mathbf{1}\gamma. && \text{(9.4e)}
\end{aligned}
$$

Constraint (9.4b) corresponds to (9.2) and constraints (9.4c)-(9.4e) are used to describe the linear objective function.

Quadratic regulation problems such as

$$
\begin{aligned}
J_N^*(x(0)) \;=\; & \min_{u_0,\ldots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} (u_k^T Q_u u_k + x_k^T Q_x x_k) + x_N^T Q_{x_N} x_N \right\} && \text{(9.5a)}\\
\text{subj. to} \quad & x_k \in \mathbb{X},\ u_{k-1} \in \mathbb{U}, \qquad k \in \{1, \ldots, N\}, && \text{(9.5b)}\\
& x_N \in \mathcal{T}_{\text{set}}, && \text{(9.5c)}\\
& x_{k+1} \;=\; Ax_k + Bu_k, \quad x_0 = x(0), && \text{(9.5d)}\\
& Q_x \succeq 0, \quad Q_{x_N} \succeq 0, \quad Q_u \succ 0, && \text{(9.5e)}
\end{aligned}
$$

can be reformulated as a QP by substituting $x_k = A^k x(0) + \sum_{j=0}^{k-1} A^k B u_{k-1-j}$ so that

$$
\begin{aligned}
J_N^*(x(0)) \;=\; & x(0)^T Y x(0) + \min_{U_N} \left\{ U_N^T H U_N + x(0)^T F U_N \right\} && \text{(9.6a)}\\
\text{subj. to} \quad & GU_N \leq W + Ex(0). && \text{(9.6b)}
\end{aligned}
$$

It was shown that the optimal feedback controller for problems of type (9.4) and (9.6) is PWA and defined over convex polyhedra which will henceforth be referred to as regions (see Theorems 3.2.1 and 3.2.3). See Chapter 5 for a discussion on sufficient conditions on (9.5), such that the associated RHC yields asymptotically

stable trajectories.

The quadratic control problem has found more widespread application than the linear equivalent because of its natural analogy to energy functions and because closed-loop stability can easily be enforced for quadratic RHC [SR98, Mac02, Bor03, BMDP02, CM96, GBTM03]. Note that tracking problems can easily be recast as regulation problems (e.g. see [PK03] for offset-free tracking strategies).

The input sequence for (9.3) and (9.5) can be obtained by solving an optimization problem (i.e. linear or quadratic program) on-line at each time step or by evaluating the optimal piecewise affine (PWA) feedback law, which can be pre-computed off-line. The explicit feedback solution of linear optimal control for constrained linear systems was introduced in [BBM00b, BBM00a] (see Section 3 on multi-parametric programming) and the quadratic counterpart was presented in [BMDP02]. The results were later extended to tackle robustness for linear cost objectives in [BBM03, KM04a] and quadratic cost objectives in [KM03].

We will now discuss in detail what we mean by *efficient* control of constrained LTI systems. The *complexity* of the control schemes given here can be subdivided into three components: the runtime required to compute the controller, the size of the resulting controller partition (number of regions) and the time required to apply the controller in real-time. Here we refer to these three components as the 'three levers' (see Figure 9.1) for complexity reduction, i.e. to arrive at efficient controllers.

Lever 1 - Controller Computation: The aim here is to reduce the time required to compute a feedback controller for constrained LTI systems, i.e. to speed-up solvers for multi-parametric programs. Specifically, the aim here is to reduce the computation time that is needed to construct a single controller region. This topic is covered in Chapter 10.

Lever 2 - Partition Complexity: The aim here is to reduce the partition complexity of PWA state feedback controllers, i.e. to reduce the number of controller regions. There are two ways to achieve this: either formulate the control problem such that the resulting controller is simple or process complex partitions a posteriori to reduce the number of regions. Lever 2 only deals with

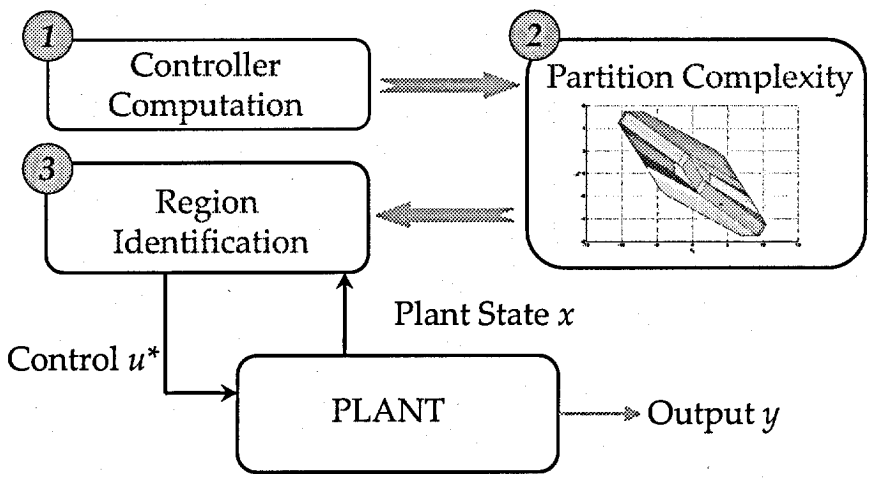


Figure 9.1: Illustration of the three levers for complexity reduction in receding horizon control.

the first aspect. All post-processing schemes are part of Lever 3. Lever 2 complexity reduction is covered in Chapter 11.

Lever 3 - Region Identification: The aim here is to reduce the runtime necessary to find the active feedback law for a *given* controller partition, i.e. all of the schemes investigated here process an existing controller partition such that the necessary on-line effort is reduced. This topic is covered in Chapter 12.

10

Efficient Computation of Multi-Parametric Programs in Control

This chapter will address the first lever for complexity reduction. Namely, the efficient computation of explicit control laws.

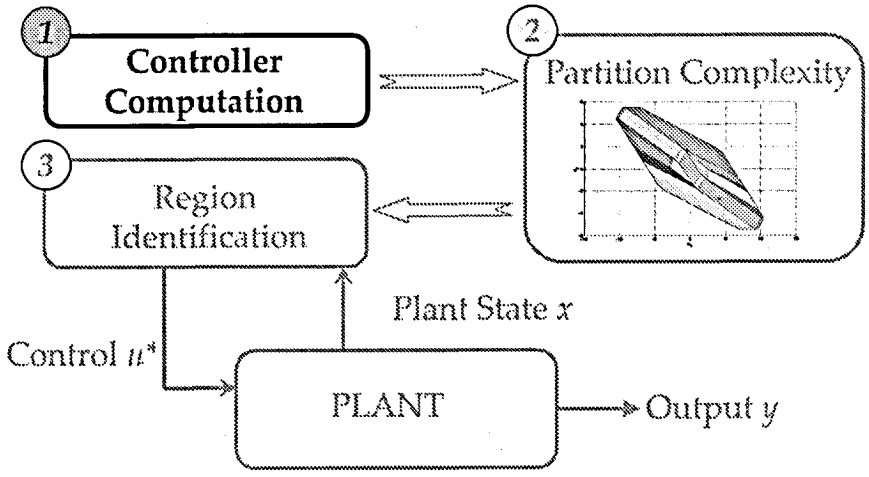


Figure 10.1: Illustration of the three levers for complexity reduction in receding horizon control. The first lever, which is the focus of this chapter, is highlighted.

We will first analyze the complexity of multi-parametric solvers in Section 10.1. Based on the gained insights, we will introduce improvements to multi-parametric solvers in Sections 10.2 and 10.3.

10.1 Analysis of Multi-Parametric Programs in Control

As stated in Chapter 3, many different algorithms for solving multi-parametric programs have been proposed [TJB03a, Bao02, BMDP02, BBM00b]. Despite their differences, the key aspects which determine solver runtime remain the same for all approaches. Specifically, all algorithms work along the following lines (see Section 3.2 for details)

1. Determine the active constraints $\mathcal{A}_N(x)$ for a given state x and determine the optimizer $U_N^*(x)$ as an explicit function of x.
2. Compute the polyhedral set $\mathcal{P}_r$ of states where $U_N^*(x)$ is optimal and satisfies constraints.
3. Proceed iteratively until the entire feasible state space $\mathcal{X}_N$ is covered.

Since the basic structure is the same for all solvers, the runtime of a generic multi-parametric solver can be given by

$$\textit{Total Runtime} \;=\; \#\ \textit{Regions} \;\times\; \textit{Runtime per Region} \;+\; \textit{Overhead.}$$

These three key aspects, which determine the total off-line controller computation time, are discussed in the following:

 # Regions: The number of regions R in the solution partition is clearly the single most important factor influencing the time that is required to solve a multi-parametric program. The number of regions correlates closely with the number of facets c of the original constraint polytope $GU_N \leq W + Ex$ (see (9.4b) or (9.6b)) in x-U_N-space. Specifically, an upper bound on the number of regions can be given by $R \leq \sum_{i=0}^{c} \binom{c}{i} = 2^c$ [BMDP02]. Note that this theoretical upper bound can only be reached if the dimension m_U of the optimizer U_N is equal to the number of constraints c. This condition is very unrealistic for practical control problems (typically $m_U \ll c$), such that this upper bound is very conservative. Instead of the number of constraints c, it is the dimension m_U of the optimizer U_N which is the key influence on the number of active constraint combinations which occur in realistic control problems. Assuming

non-degeneracy (see [Bor03]), a bound on the number of regions can be given by $R \leq \sum_{i=0}^{m_U} \binom{c}{i}$.

For quadratic control problems, the dimension m_U of U_N is equal to Nm (i.e. $m_U = Nm$), where N is the prediction horizon and m the input dimension for the LTI system (9.1). Hence, the x-U_N-polytope is in dimensions $n + Nm$ ($x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$). Since the number of controller regions is strongly correlated with the dimension of the constraint polytope, the prediction horizon N is the single most important factor determining the number of regions in the solution partition $\{\mathcal{P}_r\}_{r=1}^R$.

Runtime per Region: The runtime per region is almost entirely determined by the time required to obtain the minimal representation of a controller region. Once an active set $\mathcal{A}_N(x)$ has been determined, it is necessary to compute the associated region $\mathcal{P}_r$ (see (3.6)) in minimal representation (see Chapter 2). It is important to obtain the non-redundant representation of each $\mathcal{P}_r$ for two reasons. First, the necessary storage effort for the solution partition would be considerable if every $\mathcal{P}_r$ were stored in redundant form. Second and more importantly, the minimal representation is vital for the algorithms [TJB03a, Bao02, BMDP02, BBM00b] to guarantee that the entire feasible state space $\mathcal{X}_N$ will be covered, i.e. that there will be no 'gaps' in the partition.

When solving mp-QPs, in order to obtain a non-redundant representation of $\mathcal{P}_r$, it is necessary to solve c LPs, where c corresponds to the number of constraints in (9.6b), i.e. $W \in \mathbb{R}^c$ (see Remark 3.2.6). For instance, assume an optimization problem for a second order SISO system with prediction horizon $N = 10$ and min-max constraints on all states and inputs. This would yield $c = 60$ half-spaces[1] for the initial representation of each controller region. For mp-LPs, the number of LPs which need to be solved per region vary. Specifically, it is necessary to solve one LP for each constraint which is not contained in the active set associated with region $\mathcal{P}_r$ [Bor03, Gal95].

Overhead: The overhead of all multi-parametric solvers is almost exclusively determined by active set management. Active set management involves the identifi-

[1] Two states with $x_{\min} \leq x_k \leq x_{\max}$ and $k = 0, \ldots, 9$ yields 40 half-spaces. One input with $u_{\min} \leq u_k \leq u_{\max}$ and $k = 0, \ldots, 9$ yields 20 half-spaces.

cation of active sets $\mathcal{A}_N(x)$ and the guarantee that all optimal active sets will be found.

The first time consuming aspect is the identification of active sets. Depending on the specific multi-parametric solver, the identification of an optimal active set requires the solution of up to 2 LPs. Specifically, one LP is required to obtain a point x^* in the 'uncharted' part of the state-space and an additional LP is required to identify the active set $\mathcal{A}_N(x^*)$ for that point [Bao02]. The scheme in [TJB03a] is more efficient and identifies active sets without solving LPs in the non-degenerate case. In case of degeneracy however, the solver in [TJB03a] also requires the solution to 2 LPs, as in [Bao02].

The second time consuming aspect lies in ascertaining that the entire feasible state space is covered by the controller partition and that no duplicate regions are computed. Using the scheme in [Bao02], it is necessary to check whether a newly obtained state x^* is already contained in a previously computed controller region, before computing the associated active set $\mathcal{A}_N(x^*)$. Although this operation is not very expensive as such, it is performed once for every facet of every region $\mathcal{P}_r$ such that the total runtime can be considerable. Note that the algorithm in [TJB03a] requires only a simple string comparison for the general (non-degenerate) case.

The computational burden of active set management has been somewhat alleviated by the algorithm in [TJB03a] and the computation of simple controller partitions will be addressed in Chapter 11. We will therefore focus on the efficient polytope reduction in the following sections. Section 10.2 will introduce an efficient algorithm for obtaining the minimal representation of general polytopes. Although the algorithm can be applied to any polytope, it is specifically well suited for the type of polytopes which appear in the context of multi-parametric programming. In Section 10.3, an algorithm will be introduced which computes the infinite-time constrained LQR (see Section 4.4). As it turns out, the proposed scheme can be applied even for finite time optimal controller computation and has the intrinsic advantage of reducing the necessary effort of polytope reduction for certain classes of problems.

10.2 Efficient Computation of Minimal Representations of Polytopes

A detailed introduction to polytopes and the terminology used throughout this section is given in Chapter 2. The results of this section[2] have been published in [SLG+04].

As stated in the previous section, polytope reduction is important for the computational speed of multi-parametric solvers. However, when multi-parametric solvers are used in the context of controller computation for piecewise affine (PWA) systems, the importance of polytope reduction is even greater. In optimal control of PWA systems, it is necessary to intersect a large number of controller regions in order to obtain the feedback law which optimizes the cost objective [BCM03b, BBBM03]. Furthermore, it is often necessary to check whether two unions of polytopes are equal, which again requires extensive polytope computations [GKBM04a, BT03]. These issues are discussed in detail in Part IV. In this section, a polytope reduction method will be presented which can be used to efficiently obtain the minimal representations of polytopes which arise in the context of multi-parametric programming.

10.2.1 Efficient Polytope Reduction in Multi-Parametric Programming

Computing the minimal representation of polytopes has turned out to be a bottle neck in many multi-parametric programs solved by the Multi Parametric Toolbox [KGB04], and has been reported to be an issue also in other implementations [TJB01]. The standard approach to detect if the jth constraint in the set

$$\begin{aligned} Hx &\leq K, \\ H &= \begin{bmatrix} h_1 & h_2 & \dots & h_c \end{bmatrix}^T, \\ K &= \begin{bmatrix} k_1 & k_2 & \dots & k_c \end{bmatrix}^T, \end{aligned} \tag{10.1}$$

is redundant, is to define a new polyhedron with the jth constraint removed,

$$\tilde{H} = \begin{bmatrix} h_1 & h_{j-1} & h_{j+1} \dots & h_c \end{bmatrix}^T,$$

[2]The content of this section is the result of a collaboration with Johan Löfberg who was the primary contributor.

$$\tilde{K} = \begin{bmatrix} k_1 & k_{j-1} & k_{j+1} \dots & k_c \end{bmatrix}^T,$$

and maximize $h_j^T x$ in the reduced polytope $\tilde{H}x \leq \tilde{K}$

$$\max_x \; h_j^T x \tag{10.2a}$$

$$| \quad \tilde{H}x \leq \tilde{K}. \tag{10.2b}$$

If the optimal objective value of this problem is less than or equal to k_j, the constraint is redundant and can be removed [Fuk04c, OSS95].

To detect and remove all redundant constraints, the algorithm requires the solution of c LPs with, in the worst-case, $c-1$ constraints and n variables. To improve the performance of this algorithm, we need to reduce the number of LPs to be solved, and preferably also their size. Our approach to do this is to perform an initial, computationally cheap, pre-solve analysis to detect a sub-set of the redundant and non-redundant constraints.

10.2.2 Detecting Non-Redundant Half-Spaces

By detecting some of the non-redundant constraints, we can reduce the number of LPs that have to be solved to derive the minimal representation of a polytope. We first propose the application of a simple randomized ray-shooting approach [Bon83].

1. Initialize the set of non-redundant constraints $\mathcal{J}_N = \emptyset$.
2. Calculate an interior point x_{int}, $Hx_{\text{int}} < K$.
3. Generate a random direction $d \in \mathbb{R}^n$.
4. Calculate intersections between the line $x_{\text{int}} + t_i d$ and the hyper-plane $h_i^T x = k_i$, giving $t_i = \frac{k_i - h_i^T x_{\text{int}}}{h_i^T d}$.
5. Find the closest intersecting hyper-planes along positive and negative direction d, corresponding to smallest positive and largest negative t respectively. Let the corresponding indices to these hyper-planes be i_p and i_n. These constraints are non-redundant such that $\mathcal{J}_N := \mathcal{J}_N \cup i_p \cup i_n$.
6. Let the mid-point of the line between the two intersection points $x_{\text{int}} + t_p d$ and $x_{\text{int}} + t_n d$ serve as a new interior point, $x_{\text{int}} := x_{\text{int}} + \frac{t_{i_p} + t_{i_n}}{2} d$.

7. Repeat from 3).

An illustration of this algorithm is given in Figure 10.2. The algorithm requires an interior point to begin with in step 2. To find one, we calculate the Chebychev center of the polytope, requiring the solution of one LP (see (2.5)).

Remark 10.2.1 *Note that the active constraints (see Definition 3.1.4) which are obtained when solving the Chebychev-Ball problem can also be used to initialize the set of non-redundant constraints $\mathcal{J}_N$. Obviously, all half-spaces which are 'touched' by the ball are non-redundant, provided all duplicate half-spaces have been removed.*

Of-course the number of ray-shooting iterations is an important parameter. In the current implementation, $\lceil c/2 \rceil$ iterations are performed. This value was heuristically determined by numerous simulation runs.

Although there is no guarantee that we find all, or even a significant part of the non-redundant half-spaces, the algorithm is simple enough to motivate its use. Note that the algorithm is most efficient when the fraction of redundant constraints is low.

10.2.3 Detecting Redundant Half-Spaces

By detecting redundant half-spaces, we not only reduce the number of LPs that have to be solved in (10.2), but we also reduce the size of these LPs, since the corresponding constraints can be removed.

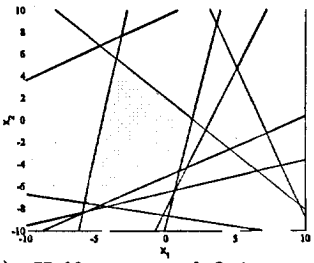
Detecting redundant constraints in LPs is a standard problem, and is done in most LP solvers during a pre-solve analysis of the problem. The key idea in pre-solve algorithms is to exploit variable bounds $L \leq x \leq U$ to detect obviously redundant constraints [Gon97].

To detect if $h_i^T x \leq k_i$, $h_i = \begin{bmatrix} h_{i1} & h_{i2} & \dots & h_{in} \end{bmatrix}$ is redundant, each term in $h_i^T x$ is individually maximized to obtain an upper bound on $h_i^T x$

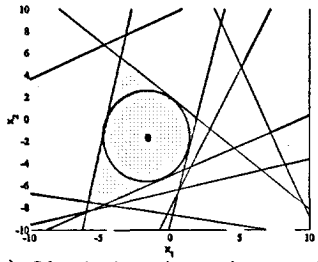
$$\sum_{j=1}^{n} h_i x_i \leq \sum_{j \in \{j:h_{ij}>0\}} h_{ij} U_j + \sum_{j \in \{j:h_{ij}<0\}} h_{ij} L_j. \tag{10.3}$$

If the right-hand side of (10.3) is less than k_i, the constraint is redundant and can be removed. Hence, the set of redundant constraints detected in the pre-solve analysis is defined by

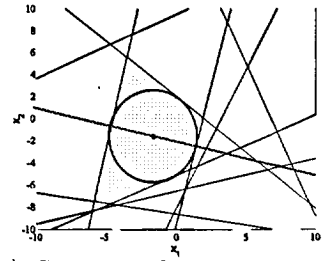
$$\mathcal{J}_R = \left\{ i \in \{1, \dots, c\} \mid \sum_{j \in \{j:h_{ij}>0\}} h_{ij} U_j + \sum_{j \in \{j:h_{ij}<0\}} h_{ij} L_j < k_i \right\}. \tag{10.4}$$



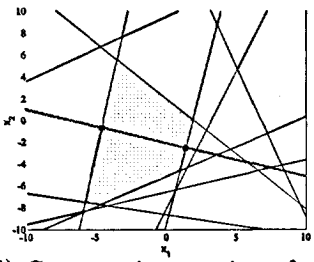
(a) Half-spaces defining the polytope.



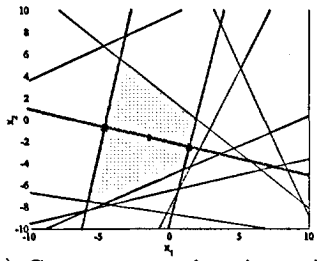
(b) Obtain interior point x_{int} by computing the Chebychev ball.



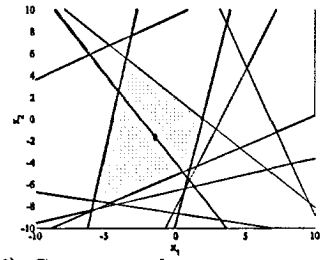
(c) Create random ray emanating from x_{int}.



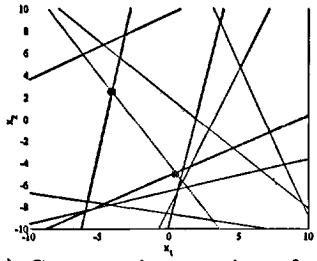
(d) Compute intersection of ray with closest half-spaces (bold circles). These half-spaces are non-redundant.



(e) Compute new interior point x_{int}.



(f) Create random ray emanating from x_{int}.



(g) Compute intersection of ray with closest-half spaces (bold circles).

Figure 10.2: Illustration of the scheme to detect non-redundant half-spaces. Here, all non-redundant constraints happen to be identified by computing the Chebychev ball (see Remark 10.2.1).

Tight variable bounds L and U are crucial for this pre-solve algorithm to be efficient. In a pre-solver used in an LP solver, crude bounds are typically given by a priori knowledge, and by applying a more advanced pre-solve algorithm iteratively, the bounds can in some cases be improved upon by inferring more information from the

constraints.

The standard pre-solve analysis that is applied before solving an LP is required to be cheap in order to actually yield runtime benefits, since the LP itself can be solved efficiently. In contrast, we are here solving a total of c LPs for polytope reduction. Hence, we can spend a lot more effort on a pre-solve analysis since it benefits each of the c LPs.

Since tight lower and upper bounds are crucial for the detection of redundant constraints using (10.3), we solve $2n$ LPs ($x \in \mathbb{R}^n$) to derive exact lower and upper bounds on x in the polytope $Hx \leq K$. Specifically we solve the following LP for all $i \in \{1, \ldots, n\}$:

$$\min_x \ \pm x_{(i)} \tag{10.5a}$$
$$\text{s.t.} \quad Hx \leq K, \tag{10.5b}$$

where $x_{(i)}$ denotes the i-th element of the vector $x \in \mathbb{R}^n$. Of-course, spending the effort of solving $2n$ LPs to find the bounding box of a polytope, to be used in the possibly inefficient algorithm (10.3), is only reasonable if the expected number of detected redundant constraints is large and n is sufficiently small compared to c. This is generally the case if multi-parametric programming is used in the context of controller computation. An illustration of the bounding box computation is given in Figure 10.3.

10.2.4 Complete Algorithm

Putting the two parts together, we obtain the reduction algorithm.

Algorithm 10.2.2 (Efficient Polytope Reduction)

1. *Calculate upper and lower bounds using (10.5).*
2. *Apply (10.4) to remove redundant constraints $\mathcal{J}_R$.*
3. *Compute the Chebychev ball to find interior points and a subset $\mathcal{J}_C$ of non-redundant constraints.*
4. *Find a subset $\mathcal{J}_N$ of the non-redundant constraints using ray-shooting on the constraints $\mathcal{J} \setminus \mathcal{J}_R$.*

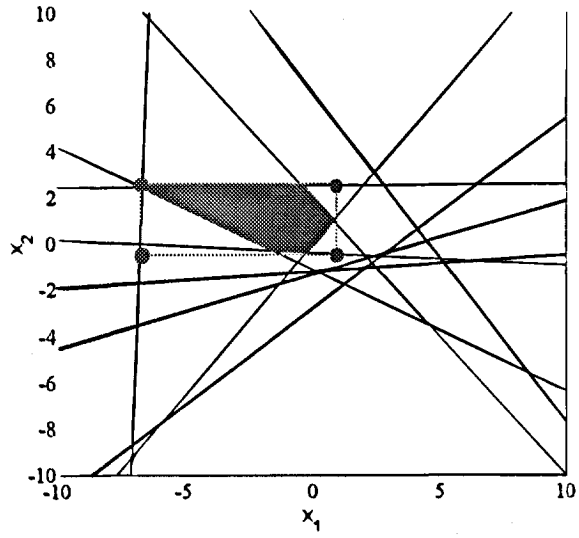


Figure 10.3: Illustration of a bounding box. If the bounding box does not intersect a hyper-plane, it is redundant.

5. *Check redundancy of remaining unresolved constraints* $h_i^T x \leq k_i$, $\forall i \in \mathcal{J}$ /($\mathcal{J}_R \cup \mathcal{J}_C \cup \mathcal{J}_N$) *by solving the LP (10.2).*

Observe that the ray-shooting algorithm is efficient for polytopes with few redundant constraints, while the bounding box method is most useful for polytopes with many easily detected redundant constraints. Hence, the two pre-analysis algorithms together cover many levels of redundancy.

The expected computational gains from the two pre-solve steps can be estimated if we take the computational complexity of solving an LP into account. A rough complexity analysis of a modern interior-point algorithm to solve an LP with n variables and s constraints would typically give $O((n^3 + n^2 s)\sqrt{s})$ operations[3] [dH94]. Hence, a polytope reduction algorithm, solving s LPs, will have super-quadratic complexity with respect to the number of constraints in the original polytope. Consequently, the effect of removing redundant constraints by using the bounding box approach will be super-quadratic, i.e. removing half of the constraints will reduce the computational

[3]The main computational burden in each interior-point iteration is the calculation of the Schur-matrix $H^T D H$ and factorization of this matrix (D is a diagonal matrix which depends on the particular algorithm). Creating the Schur-matrix requires $O(n^2 s)$ operations and the factorization $O(n^3)$. Additional computations also have linear complexity in s. The number of iterations can be bounded by $O((n^3 + n^2 s)\sqrt{s})$, but is typically between 5 and 50.

effort by more than a factor of four. The impact of the ray-shooting scheme on the total runtime will however only be linear, since the size of the remaining LPs are unaffected, only the number of LPs is reduced.

Remark 10.2.3 *It should be noted that the proposed method may not be suitable for all types of polytope reduction problems. Polytopes which arise in the context of multi-parametric programs in the field of control typically have a specific structure (limited number of facets) and are in low dimensions (e.g., below* 10*).*

10.2.5 Other Usage of Bounding Boxes

Outer box approximations defined by (10.5) can be efficiently used in many problems arising in fields of reachability analysis for hybrid systems, approximate projections and computation of explicit control laws for hybrid systems.

For instance in reach-set computation for hybrid systems [Tor03], bounding boxes can be used to decrease memory requirements by keeping only two extreme points of a bounding box instead of storing the complete half-space representation of a polytope $Hx \leq K$. This is mainly important because of the explosion of the number of polytopes at each step of the iterative exploration procedure.

As already indicated, bounding boxes can be effectively used in the area of multi-parametric programming for PWA systems. Optimal control problems for PWA systems are generally solved in a dynamic programming fashion [BCM03b, BCM03a, KM02]. At each step of the dynamic program, the cost expression associated with a polytope over which the control law is defined needs to be compared to the cost of each other region which intersects the first one. To avoid unnecessary computation, it is useful to detect any possible intersections before further processing. This feature is also relevant in the context of stability analysis of PWA systems, since answering the question if two boxes intersect reduces to a simple set of IF-THEN statements (see Remark 8.2.2). Despite the over-approximation nature of bounding boxes this method performs very well in practice.

Furthermore, search tree structures can be created more efficiently using the box approximations of polytopes [GTM04, TJB03b]. Search trees are important in the on-line implementation of the results of a multi-parametric program and will be briefly discussed in Chapter 12. Since the optimizer $U_N^*(x)$ is piecewise-affine over a polyhedral partition $\bigcup_{r\in\mathcal{R}} \mathcal{P}_r$, the procedure to obtain the control action for a given state x reduces to a simple membership test. Without a search tree, one would need

to check every region $\mathcal{P}_r$, $r \in \mathcal{R}$, which could be expensive when the number of regions becomes very large. In such search trees, each node of the tree consists of a hyperplane and a list of regions which satisfy this inequality and a list of regions which do not. Bounding boxes are a very effective tool in deciding to which list a region belongs to. Again, the speedup results from the fact that such an evaluation has to be performed only on two extreme points of the box, without the need to compute extreme points of the original polytope, which requires the solution to an LP. Hence, the construction of such search trees can be speeded-up significantly by the use of bounding boxes.

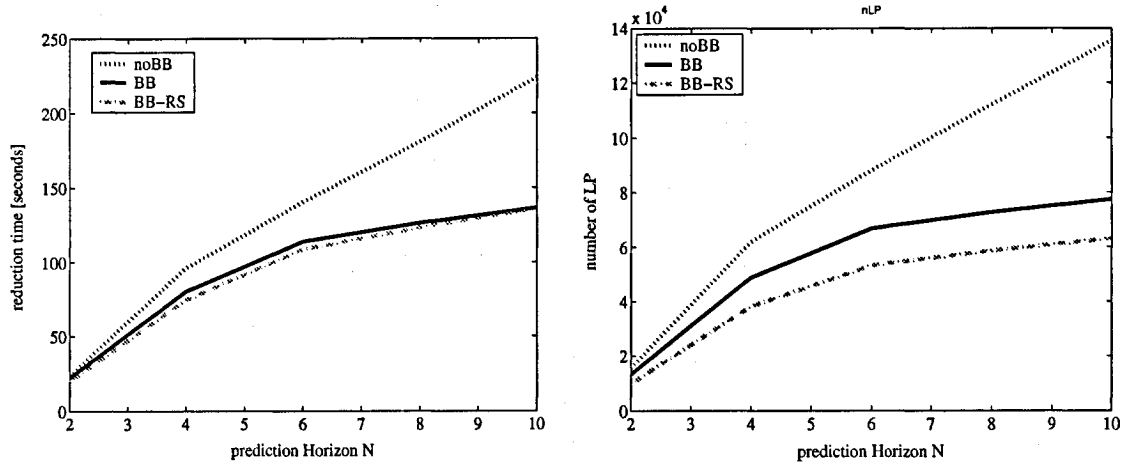
The computation of outer box approximations (10.5) in the algorithm described in the previous section is therefore not a one-purpose operation. The boxes can be stored along with the original polytope to significantly speed up subsequent operations, some of which were mentioned in this section.

10.2.6 Numerical Results

The computational improvements of the proposed pre-solve approach depend strongly on the multi-parametric problem being solved. To find a general trend for problems typically solved using multi-parametric techniques, 10 random stable systems with $n = 3$ states and $m = 2$ inputs were generated. The mp-QPs arising in optimal control problems with prediction horizons $N = 2, 4, 6, 8$ and 10 where solved. Averaged results for the proposed polytope reduction algorithms are depicted in Figure 10.4.

The experiments indicate that the impact of efficient polytope reduction is increasing with the prediction horizon N. This was to be expected from the construction of the controller regions in (3.6), i.e. as N increases, the number of initial half-spaces grows. On the other hand it has been observed that, in general, the number of half-spaces defining the controller regions grows sub-linearly. Therefore, the fraction of redundant half-spaces grows with increasing prediction horizon. As we described earlier, the computational efficiency of the bounding box approach grows quadratically with respect to the fraction of detected redundant constraints, so an improved performance for longer horizons is to be expected.

The impact of ray-shooting is less impressive. In the multi-parametric application, most half-spaces are redundant, hence few non-redundant half-spaces will be found. The number of solved LPs is decreased by the ray-shooting, but the cost to find the small number of non-redundant half-spaces is comparable to the cost of solving



(a) Total time spent on polytope reduction.

(b) Total number of LPs solved for polytope reduction.

Figure 10.4: Comparison of average time spent and average number of LPs solved for various polytope reduction schemes (noBB : standard polytope reduction, BB : polytope reduction using bounding boxes, BB-RS : polytope reduction using bounding boxes and ray-shooting)

the additional LPs, at least with the current implementation of the ray-shooting algorithm.

10.3 Computation of the Constrained Infinite-Time Linear Quadratic Regulator

In this section an algorithm to compute the explicit Infinite-Time Constrained LQR (CLQR) controller will be presented. This section is based on the publications in [GBTM03, GBTM04]. We refer the reader to Chapter 4 for an overview of general optimal control problems. The primary contribution of this section is an algorithm to compute the CLQR. However, for certain types of problems, the proposed scheme also has the intrinsic advantage of reducing the necessary effort for computing the minimal representation of controller regions. Therefore, the CLQR is covered in this chapter.

It is current practice to approximate the CLQR problem by receding horizon control

(RHC, see Chapter 5). For RHC, a finite horizon problem is solved at each time step, and then only the initial element of the optimal input sequence is applied to the plant. The main problem of RHC is that it does not, in general, guarantee stability. In order to make RHC stable, conditions have to be added to the original problem which may result in degraded performance [RM93, MRRS00].

A return to the infinite-horizon formulation is required to produce stabilizing control laws which guarantee global optimality [RBW90]. However, rather than address the CLQR problem, all Receding Horizon Control (RHC) variants [MRRS00], with few exceptions [SR98, CM96, SD87], rely on approximations. Sznaier and Damborg [SD87] showed that a finite horizon optimization over a horizon $\bar{N}$ can provide the solution to the infinite-horizon constrained optimal control problem. However, there is no technique to compute $\bar{N}$ for compact sets of points apart from the conservative upper bound in [CM96].

The contribution of this section is a novel approach to compute the piecewise affine (PWA) state feedback solution to the CLQR problem. The presented algorithm combines multi-parametric quadratic programming [BMDP02] with reachability analysis to obtain the infinite-time optimal PWA feedback law. The algorithm reduces the time necessary to compute the PWA solution for the CLQR when compared to other approaches [Bao02, BMDP02]. Furthermore, the algorithm does not rely on estimates of $\bar{N}$ but instead computes $\bar{N}$ for compact sets. Thus, the on-line computation of the control action can be reduced by either evaluating the PWA solution or by solving the finite horizon problem for a horizon of $N = \bar{N}$.

We will now briefly recap the results of Section 4.4 before introducing the proposed algorithm.

10.3.1 Problem Statement

If in (4.7) we set $N = +\infty$, we obtain the infinite-time constrained LQR (CLQR) problem:

$$J^*_\infty(x(0)) = \min_U \left\{ \sum_{k=0}^{+\infty} u_k^T Q_u u_k + x_k^T Q_x x_k \right\}, \tag{10.6a}$$

$$\text{subj. to} \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}, \quad k \in \{0, 1, \ldots\}, \tag{10.6b}$$

$$x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0), \tag{10.6c}$$

$$Q_x \succeq 0, \qquad Q_u \succ 0, \tag{10.6d}$$

where the infinite dimensional vector $U_\infty \triangleq [u_0^T, u_1^T, \ldots]^T$ is the optimization vector. We denote by U_∞^* the optimizer of (10.6). In order to show the equivalence of the finite time optimal control problem (4.7)($N < \infty$) and the infinite-time optimal control problem (10.6)($N = +\infty$) we also define the following vector:

$$U^*(x(0), N) = [U_N^*(x(0)), Kx_N, Kx_{N+1}, Kx_{N+2}, \ldots], \tag{10.7}$$

where K is the unconstrained optimal feedback law (4.3) obtained from the Algebraic Riccati Equation (4.4).

Next, some of the definitions and theorems required in this section will be restated. We refer the interested reader to Section 7.1 for detailed discussion of various invariant sets and simply state the key definitions:

Definition 10.3.1 (Maximal LQR Invariant Set $\mathcal{O}_\infty^{\mathrm{LQR}}$) *$\mathcal{O}_\infty^{LQR} \subseteq \mathbb{R}^n$ denotes the maximal invariant set of states for which the unconstrained LQR control law K obtained from the ARE satisfies the constraints in (9.2) for all time, i.e.,*

$$\begin{aligned}\mathcal{O}_\infty^{LQR} = \{x(0) \in \mathbb{R}^n | x(k) \in \mathbb{X}, Kx(k) \in \mathbb{U}, \forall k \geq 0, \\ x(k+1) = (A+BK)x(k)\},\end{aligned}$$

where $\mathcal{O}_\infty^{\mathrm{LQR}}$ is a positive invariant set containing an open neighborhood of the origin [SD87], provided the origin is contained in the interior of the set described by (9.2) [4].

Definition 10.3.2 (Maximal Stabilizable Set $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$) *For the control invariant set $\mathcal{O}_\infty^{LQR} \subseteq \mathbb{X}$, the maximal stabilizable set $\mathcal{K}_\infty(\mathcal{O}_\infty^{LQR})$ for the LTI system (9.1) subject to the constraints in (9.2) is the union of all N-step stabilizable sets contained in $\mathbb{X}$ ($N \in \mathbb{N}_+$).*

The following theorems are derived from [CM96, SR98] and establish the solutions properties of the CLQR:

Theorem 10.3.3 (Infinite-Horizon Optimization for Initial State, [GBTM03], [GBTM04]) *Assume an optimal control problem (4.7) is posed for an LTI system (9.1) with a terminal cost Q_{x_N} equal to the ARE solution (4.4). Then there*

[4] If the origin is not contained in (9.2), no solution to (10.6) exists, since $J_\infty^*(x(0))$ is infinite.

exists a finite horizon $\hat{N}(x(0))$ *such that* $U^*_\infty(x(0)) = U^*(x(0), \hat{N}(x(0)))$ *for any* $x(0) \in \mathcal{K}_\infty(\mathcal{O}^{LQR}_\infty)$. *The maximum stabilizable set* $\mathcal{K}_\infty(\mathcal{O}^{LQR}_\infty)$ *is described in Definition 10.3.2. The equality also holds for all horizons* $N \geq \hat{N}(x(0))$.

Definition 10.3.4 (Horizon $\bar{N}(x)$ of State x) *We define* $\bar{N}(x(0))$ *to be the minimal horizon satisfying Theorem 10.3.3, i.e.,* $U^*_\infty(x(0)) = U^*(x(0), \bar{N}(x(0)))$ *and* $U^*_\infty(x(0)) \neq U^*(x(0), \bar{N}(x(0)) - 1)$.

Lemma 10.3.5 (Horizon $\bar{N}_\mathcal{S}$ of Set $\mathcal{S}$) *Consider a compact set* $\mathcal{S} \subseteq \mathbb{R}^N$ *of initial conditions* $x(0)$. *If the feasible set* $\mathcal{S}_F = \mathcal{S} \cap \mathcal{K}_\infty(\mathcal{O}^{LQR}_\infty)$ *is closed, then there exists a finite horizon* $\bar{N}_\mathcal{S}$ *defined as*

$$\bar{N}_\mathcal{S} \triangleq \max_{x(0) \in \mathcal{S}_F} \bar{N}(x(0)),$$

such that $U^*_\infty(x(0)) = U^*(x(0), \bar{N}_\mathcal{S})$ *for any* $x(0) \in \mathcal{S}_F$. *The equality also holds for all horizons* $N \geq \bar{N}_\mathcal{S}$. *If* $\mathcal{S}_F$ *is not closed,* $\bar{N}(x(0))$ *may be unbounded.*

Theorem 10.3.6 (Properties of Infinite-Time Quadratic Optimal Control, [GBTM03, GBTM04]) *Consider a compact set* $\mathcal{S} \subseteq \mathbb{R}^n$ *of initial conditions* $x(0)$. *If the feasible set* $\mathcal{S}_F = \mathcal{S} \cap \mathcal{K}_\infty(\mathcal{O}^{LQR}_\infty)$ *is closed, then the horizon* $\bar{N}_\mathcal{S}$ *is finite and therefore the state feedback solution* $U^*_\infty = U^*(x(0), \bar{N}_\mathcal{S})$ *of problem (10.6) defined over* $\mathcal{S}_F$ *is PWA over a finite number of polytopic regions* R, *in particular*

$$U^*_{\bar{N}_\mathcal{S}}(x(0)) = F_r x(0) + G_r \quad \text{if} \quad x(0) \in \mathcal{P}_r, \tag{10.8a}$$

$$\mathcal{P}_r = \{x \in \mathbb{R}^n | H_r x \leq K_r\}, \; r = 0, \ldots, R. \tag{10.8b}$$

Proof Follows directly from Lemma 10.3.5 and Theorem 3.2.1 for $N = \bar{N}_\mathcal{S}$. Consequence of the results in [BMDP02]. □

Remark 10.3.7 *The cumbersome definition of the compact set* $\mathcal{S}_F$ *in Lemma 10.3.5 and Theorem 10.3.6 is necessary to avoid the case where the maximal stabilizable set* $\mathcal{K}_\infty$ *is bounded but has open boundaries. As the state* x *approaches an open boundary of* $\mathcal{K}_\infty$, $\bar{N}(x) \to \infty$.

Remark 10.3.8 *For any initial state $x(0) \in \mathcal{K}_\infty$ the following holds: If a PWA control law according to Theorem 10.3.6 is applied in a RHC manner, the resulting state trajectory is identical to that which is obtained if the infinite-time input sequence* (10.7) *is applied in open-loop [CZ99a].*

In view of the results of the previous section and Theorem 10.3.3, the implementation of CLQR can be performed either by solving the finite horizon optimization problem (4.7) for a given $x(0)$ with $N = \bar{N}(x(0))$, or in a given compact set $\mathcal{S}$ of the initial conditions by solving the mp-QP (4.7) for $N \geq \bar{N}_{\mathcal{S}}$.

Various methods have been proposed in the literature for the computation of $\bar{N}(x(0))$ [SR98] and the estimation of $\bar{N}_{\mathcal{S}}$ [CM96]. Note that $\bar{N}_{\mathcal{S}}$ is required for the computation of the infinite-time PWA solution presented in Theorem 3.2.1, using the techniques in [BMDP02].

Chmielewski and Manousiouthakis [CM96] presented an approach that provides a conservative estimate N_{est} of the finite horizon $\bar{N}_{\mathcal{S}}$ for a compact set $\mathcal{S}$ ($N_{\text{est}} \geq \bar{N}_{\mathcal{S}}$). They solve a single, finite dimensional, convex program of known size to obtain N_{est}. Their estimate can be used either to compute the PWA solution of (10.6) for an arbitrary set $\mathcal{S}$ or, alternatively, a quadratic program with horizon N_{est} for any initial state $x(0) \in \mathcal{S}$. Chisci and Zappa [CZ99a] presented a fast algorithm which is capable of speeding up the computation time for the CLQR problem by a factor of $\frac{N_{\text{est}}}{n}$, where n is the number of states. The procedure involves the solution of a QP as in (4.7) with horizon N_{est}. For a given initial state $x(0)$, Scokaert and Rawlings [SR98] presented an algorithm that attempts to identify $\bar{N}(x(0))$ iteratively. The key theorem is reformulated here for completeness.

Theorem 10.3.9 (Equality of Finite and Infinite Optimal Control, [SR98]) *For any given initial state $x(0)$, the solution to (4.7) is equal to the infinite-time solution (10.6), i.e., $J_N^*(x(0)) = J_\infty^*(x(0))$ and $U_\infty^*(x(0)) = U^*(x(0), N)$, if the terminal state x_N of (4.7) lies in the unconstrained positive invariant set $\mathcal{O}_\infty^{LQR}$ and no terminal set constraint is applied in* (4.7c)*, i.e. the state 'voluntarily' enters the set $\mathcal{O}_\infty^{LQR}$ after N steps.*

The method in [SR98] solves (4.7) for an initial horizon $N = N_0$. Then, until the final state lies in $\mathcal{O}_\infty^{\text{LQR}}$, N is increased according to a predefined iteration law. The iteration variable c is initialized to $c = 0$ and incremented by 1 at each iteration step. One iteration scheme is to increment N by c ($N = N_0 + c$) at each iteration step,

which yields the minimal horizon N such that $x_N \in \mathcal{O}_\infty^{\mathrm{LQR}}$, i.e., $N_{\mathrm{est}} = \bar{N}(x(0))$. An alternative is to increase N by a factor of 2 at each iteration, i.e., $N = 2^c N_0$, which results in fewer QPs to be solved at the cost of a larger N_{est} ($N_{\mathrm{est}} \geq \bar{N}(x(0))$). Note that this approach cannot be used to compute the PWA solution in Theorem 3.2.1 because it does not yield the horizon $\bar{N}_{\mathcal{S}}$ for a compact set $\mathcal{S}$.

10.3.2 Comparison of Available Techniques

In this section we will discuss and compare available methods for solving CLQR and some of the drawbacks of the approaches of Scokaert and Rawlings in [SR98] and Chmielewski and Manousiouthakis in [CM96], that were mentioned in the previous section, will be illustrated.

Example 10.3.10 *Consider the system [BMDP02]*

$$x(k+1) = \begin{pmatrix} 0.7326 & -0.0861 \\ 0.1722 & 0.9909 \end{pmatrix} x(k) + \begin{pmatrix} 0.0609 \\ 0.0064 \end{pmatrix} u(k).$$

The task is to regulate the system to the origin while fulfilling the input constraint

$$-2 \leq u(k) \leq 2, \qquad \forall k \geq 0,$$

We will solve this example in a set $\mathcal{S}$ of interest defined as

$$\mathcal{S} = \{x \in \mathbb{R}^2 | ||x||_\infty \leq 1000\}.$$

Note that this is not a constraint but merely an artificial bound on the state-space to be explored. The cost on the state is set to $Q_x = I$ and the input-cost is $Q_u = 0.01$.

Applying the approach in [CM96] to Example 10.3.10, we obtain $N_{\mathrm{est}} = 1.5 \cdot 10^7$ while the true minimal horizon is $\bar{N}_{\mathcal{S}} = 71$. An algorithm for computing $\bar{N}_{\mathcal{S}}$ will be provided in Section 10.3.3. The optimization approach in [SR98] is well suited for small $\bar{N}_{\mathcal{S}}(x(0))$. However, in general, if a large section of the state-space is to be covered, this implies a large $\bar{N}_{\mathcal{S}}(x(0))$. Therefore, the approach in [SR98] will require a large number of iterations and runtime. For random values of $x(0) \in \mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ in Example 10.3.10, the run-times of the algorithm in [SR98] are presented in Table 10.1.

Example 10.3.10	Average-Case (100 runs)	Worst-Case
$N = N_0 + c$	5.29 sec	10.13 sec
$N = 2^c N_0$	0.80 sec	2.48 sec
$N = \bar{N}_S$	0.47 sec	0.56 sec

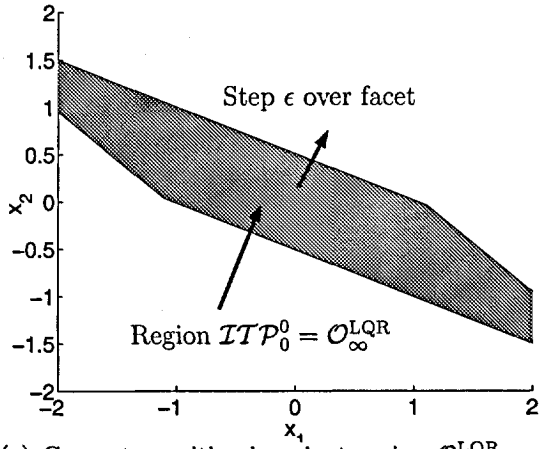
Table 10.1: Time to compute the optimal control input on a 1.2 GHz PC with the approach in [SR98]. N_0 is set to 1 and c is incremented by 1 at each iteration. The analysis is based on 100 random initial states.

10.3.3 CLQR Algorithm

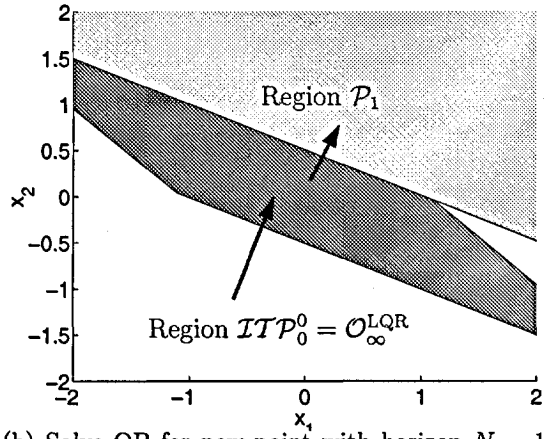
In this section we will provide an efficient algorithm to compute the PWA solution to the CLQR problem in (10.8) for a given set $\mathcal{S}$ of initial conditions. As a side product, the algorithm also computes $\bar{N}_S$ defined in Lemma 10.3.5.

The key idea of the algorithm is described next. For the optimization problem in (4.7), we choose the terminal set constraint $\mathcal{T}_{\text{set}} = \mathbb{R}^n$ (i.e. no terminal set constraint) and terminal cost $Q_{x_N} = P$, where P is the solution to the ARE 4.4 and solve an mp-QP with prediction horizon N. From Theorem 10.3.9 we can conclude that for all states which enter the invariant set $\mathcal{O}_\infty^{\text{LQR}}$ introduced in Definition 10.3.1 in N steps, the infinite-horizon problem has been solved. Therefore the associated feedback law is infinite-horizon optimal. For the sake of clarity, we will first introduce our algorithm by applying it to a generic example, before we conclude this section with a more general description. We denote the set of feasible initial conditions of problem (10.6) inside the compact set $\mathcal{S}$ as $\mathcal{S}_F = \mathcal{S} \cap \mathcal{K}_\infty(\mathcal{O}_\infty^{\text{LQR}})$. The user defined set $\mathcal{S}$ is introduced as an artificial bound on the state-space to make sure $\mathcal{S}_F$ is bounded. In practice, $\mathcal{S}$ should be chosen to be very large.

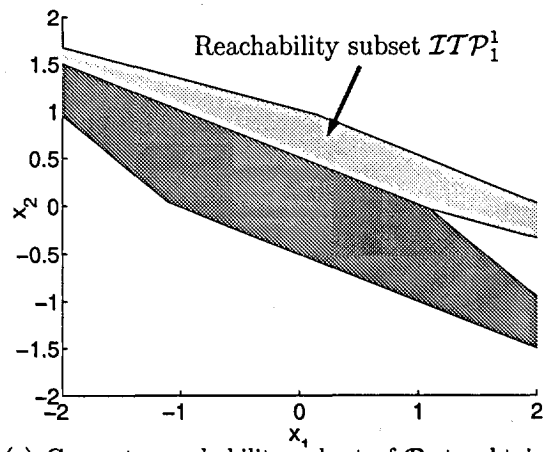
We start the procedure by computing the positive-invariant unconstrained set $\mathcal{O}_\infty^{\text{LQR}}$ introduced in Definition 10.3.1. The polyhedron $\mathcal{O}_\infty^{\text{LQR}} = \mathcal{P}_0 = \{x \in \mathbb{R}^n | H_0 x \leq K_0\}$ can be computed as in [GT91]. Figure 10.5(a) depicts $\mathcal{O}_\infty^{\text{LQR}}$. Then, the algorithm finds a point $\bar{x}$ by stepping over a facet f of $\mathcal{O}_\infty^{\text{LQR}}$ with a small step ϵ, as described in [Bao02]. If (4.7) is feasible for horizon $N = 1$ (terminal set constraint $\mathcal{T}_{\text{set}} = \mathbb{R}^n$, terminal cost $Q_{x_N} = P$ and $x(0) = \bar{x}$), the active constraints $\mathcal{A}_1^N(x(0))$ will define the neighboring polyhedron $\mathcal{P}_1 = \{x \in \mathbb{R}^n | H_1 x \leq K_1\}$ ($\bar{x} \in \mathcal{P}_1$, see Figure 10.5(b)) [BMDP02]. In order to avoid redundant exploration, one should keep track of the facets already explored. By Theorem 10.3.9, the finite time optimal so-



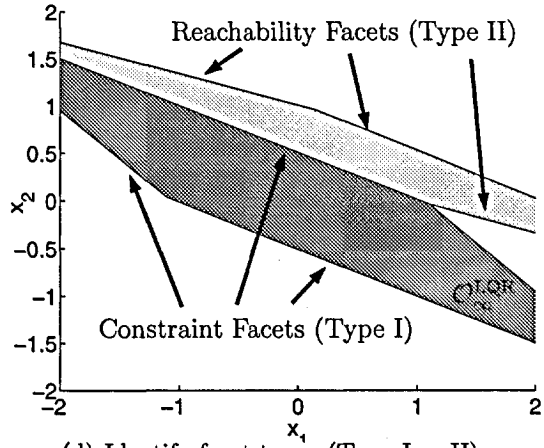
(a) Compute positive invariant region $\mathcal{O}_\infty^{\mathrm{LQR}}$ and step over facet with step-size ϵ.



(b) Solve QP for new point with horizon $N = 1$ to create the first constrained region $\mathcal{P}_1$.



(c) Compute reachability subset of $\mathcal{P}_1$ to obtain $\mathcal{ITP}_1^1$.



(d) Identify facet types (Type I or II).

Figure 10.5: Region Exploration.

lution computed above equals the infinite-time optimal solution if $x_N \in \mathcal{O}_\infty^{\mathrm{LQR}}$ (here $N = 1$). Therefore we extract from $\mathcal{P}_1$ the set of points that will enter $\mathcal{O}_\infty^{\mathrm{LQR}}$ in $N = 1$ time-steps, provided that the optimal control law associated with $\mathcal{P}_1$ (i.e., $U_N^* = F_r x(0) + G_r$; Here, $r = 1$ and $N = 1$) is applied. The Infinite-Time Polyhedron ($\mathcal{ITP}$) is therefore defined by the intersection of the following two polyhedra:

$$x_N \in \mathcal{O}_\infty^{\mathrm{LQR}}, \tag{10.9a}$$

$$x_0 \in \mathcal{P}_r. \tag{10.9b}$$

Equation (10.9a) is the reachability constraint and (10.9b) defines the set of states for which the computed feedback law is feasible and optimal over N steps (see [BMDP02] for details). Note that x_N can be described as a linear function of x_0 by substituting the feedback sequence[5] $U_N^* = F_r x_0 + G_r$ into the LTI system dynamics (9.1).

The identified region will be referred to as $\mathcal{ITP}_r^N$ ($\mathcal{ITP}_0^0 = \mathcal{O}_\infty^{\mathrm{LQR}}$, see Figure 10.5(c)) according to the following convention.

Definition 10.3.11 (Infinite-Time-Polyhedron $\mathcal{ITP}^N$) *We define the r-th Infinite-Time-Polyhedron $\mathcal{ITP}_r^N$ as follows:*

$$\forall x \in \mathcal{ITP}_r^N, \qquad \mathcal{A}_r^N(x) = \textit{constant},$$

and the reachability condition (10.9a) holds. The optimal feedback law for $\mathcal{ITP}_r^N$ is defined by $\mathcal{A}_r^N$ [BMDP02] and ensures that $x_N \in \mathcal{O}_\infty^{LQR}$. Once all redundant inequalities have been removed, this polyhedron has two types of facets:
Type I: *The facet originated from constraint restrictions in (10.9b).*
Type II: *The facet originated from reachability restrictions in (10.9a).*

The procedure for identifying the adjacent $\mathcal{ITP}$s is repeated for all facets f not previously explored, which originate from constraint restrictions (Type I). If a facet originates from the reachability restriction (Type II), we can conclude that the infinite-horizon optimal input sequence will not drive the states 'on the other side' of the facet into $\mathcal{O}_\infty^{\mathrm{LQR}}$ in N steps. This distinction is depicted in Figure 10.5(d). As depicted in Figure 10.5(d), Type I facets are shared by all $\mathcal{P}$ and their associated reachability

[5] The optimizer U_N^* can be represented by the time-varying feedback F_r, G_r whereby the dimension of U_N^* is a function of the input dimension m of the LTI system and the prediction horizon N which was used to compute F_r, G_r.

subsets $\mathcal{ITP}$, while all other facets are of Type II. Also note that we define all facets of $\mathcal{ITP}_0^0 = \mathcal{O}_\infty^{\mathrm{LQR}}$ to be of Type I, i.e., all facets of $\mathcal{O}_\infty^{\mathrm{LQR}}$ are explored at the first iteration (see Algorithm 10.3.12).

Once all facets have been explored for the finite horizon N, the horizon is increased to $N+1$ and the entire procedure is repeated. For all regions that have been computed for a smaller N, only facets of Type II are considered while for all newly identified regions, only facets of Type I are examined. This procedure is not applied to the facets which have already been explored. Note that the distinction between Type I and Type II facets merely serves to speed up the exploration procedure. It is not necessary for the algorithm to work correctly. The algorithm terminates as soon as all facets f of all $\mathcal{ITP}$s have been explored. A facet f is considered to be explored if the state $\bar{x}$ beyond f provides an $\mathcal{ITP}$ region or an infeasible problem (10.6) results for $x(0) = \bar{x}$. The presented procedure is summarized for a general problem in the following algorithm:

Algorithm 10.3.12 (CLQR Computation)

1. $\mathcal{ITP}_0^0 = \mathcal{O}_\infty^{LQR}$, $N = 0$, $r = 1$, $\mathcal{C} = \{\}$,$\mathcal{Z} = \{\}$;
2. $\mathcal{C} = \mathcal{C} \cup \{\mathcal{ITP}_0^0,\ F_0 = K_{LQR},\ G_0 = 0\}$;
3. ***repeat:***
4. $N = N + 1$
5. *forall* ($\mathcal{ITP}_i^q \in \mathcal{C}$ & $q < N$), *explore*($\mathcal{ITP}_i^q$,*Type II*);
6. *forall* ($\mathcal{ITP}_i^q \in \mathcal{C}$ & $q = N$), *explore*($\mathcal{ITP}_i^q$,*Type I*);
7. ***until:*** *all facets* f *of all* $\mathcal{ITP}_i^q \in \mathcal{C}$ *are contained in* $\mathcal{Z}$;
8. $\bar{N}_S = N$; *return* $\mathcal{C}$;

function *explore*($\mathcal{ITP}_i^q$,*facetType*);

1. *forall* ($f \in$ *facets*($\mathcal{ITP}_i^q$) & $f \notin \mathcal{Z}$) ,
2. *if facet type of* f *is not facetType, goto 1;*
3. $\mathcal{Z} = \mathcal{Z} \cup \{f\}$;

4. *step over f and get $\bar{x}$;*
5. *if $\exists\, \mathcal{ITP}_i^m \in \mathcal{C}$ $-\bar{x} \in \mathcal{ITP}_i^m$, return;*
6. *solve (4.7) for $x(0) = \bar{x}$ with horizon N;*
   if infeasible, return;
7. *compute $\mathcal{ITP}_r^N$ according to (10.9b) and (10.9a);*
8. *If $\mathcal{ITP}_r^N = \varnothing$, goto 1;*
9. *Use $\mathcal{A}_r^N$ to compute F_r and G_r according to [BMDP02];*
10. $\mathcal{C} = \mathcal{C} \cup \{\mathcal{ITP}_r^N, F_r,\ G_r\}$; $r = r + 1$;
11. *end forall.*

As before, N denotes the horizon for solving (4.7) and the integer r is a counter for the region number. The generated structure $\mathcal{C}$ is a list of all regions with their associated control law as well as a list containing all explored facets.

Theorem 10.3.13 (Finite-Time Convergence of Algorithm 10.3.12, [GBTM03], [GBTM04]) *Algorithm 10.3.12 always converges in finite time, provided $\mathcal{S}_F$ is bounded and closed.*

Proof If $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ is compact, a finite $\bar{N}_\mathcal{S}$ exists [CM96]. For a finite prediction horizon, the number of possible active constraint combinations is also finite. Since a region is uniquely identified by the active constraints, the associated region partition will consist of a finite number of regions. Since the algorithm increases the horizon N if no more regions are identified, the prediction horizon will eventually reach $\bar{N}_\mathcal{S}$. At this point, Algorithm 10.3.12 is identical to the one in [BMDP02] and will therefore converge in finite time. □

It should be noted that in theory, $\bar{N}_\mathcal{S}$ and the convergence time might not be finite if $\mathcal{S}_F$ is open or unbounded. However, in practice this is not an issue. First, there will always exist a compact set $\mathcal{S}$ to make $\mathcal{S}_F$ bounded. Second, if $\mathcal{S}_F$ has open boundaries, accumulation points will occur near those boundaries, i.e. an infinite number of regions will be located in a bounded subset of the state-space. Since the step-size ϵ which is taken in the algorithm is finite, the point $\bar{x}$ will not provide a feasible solution to (4.7) once the regions are close to the open boundaries. The resulting

partition will be an inner approximation of $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$, whereby the accuracy of the approximation can be adjusted by choice of the step-size ϵ. Therefore in practice the algorithm always converges in finite time, though not the entire set $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ may be covered by $\mathcal{ITP}$s if $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ has open boundaries, i.e. $\bigcup \mathcal{ITP}_r \subset \mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$.

Theorem 10.3.14 shows that each state $x \in \mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ is unambiguously associated with one $\mathcal{ITP}$.

Theorem 10.3.14 (Non-Overlapping $\mathcal{ITP}$s, [GBTM03, GBTM04]) *The intersection of the interior of $\mathcal{ITP}_i^N$ and $\mathcal{ITP}_j^M$ is non-empty, if and only if $i = j$ and $N = M$.*

Proof "$\Leftarrow$" trivial.
"$\Rightarrow$", from Theorem 10.3.9, (10.9b) and (10.9a), we can conclude that the $\mathcal{ITP}$ region partition is identical to the finite-time region partition computed for a horizon of $N \geq \bar{N}_{\mathcal{S}}$. Therefore each region has a distinct set of active constraints and from [BMDP02] we can conclude that if two $\mathcal{ITP}$s have a non-empty intersection then they are identical. □

The following Theorems state some properties of the solution provided by Algorithm 10.3.12.

Theorem 10.3.15 (Exact Computation of $\bar{N}_{\mathcal{S}}$, [GBTM03, GBTM04]) *If we explore any given compact set $\mathcal{S}$ with Algorithm 10.3.12, the largest resulting horizon is equal to $\bar{N}_{\mathcal{S}}$, i.e.,*

$$\bar{N}_{\mathcal{S}} = \max_{\mathcal{ITP}_r^N \; r=0,\ldots,R} N.$$

Proof Since $\bar{N}_{\mathcal{S}}$ is defined as $\bar{N}_{\mathcal{S}} \triangleq \max_{x(0)\in\mathcal{S}_F} \bar{N}(x(0))$ with $\mathcal{S}_F = \mathcal{S} \cap \mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$, we need to show that

$$\max_{x(0)\in\mathcal{S}_F} \bar{N}(x(0)) = \max_{\mathcal{ITP}_r^N \; r=0,\ldots,R} N.$$

We will denote the maximum horizon of all $\mathcal{ITP}$s as $N_{\max}$. Consider an initial feasible state $\tilde{x} \in \mathcal{S}$ which reaches $\mathcal{O}_\infty^{\mathrm{LQR}}$ in exactly $\bar{N}_{\mathcal{S}}$ steps if the optimal PWA control law is applied. This state would not be covered by any $\mathcal{ITP}$, if $N_{\max} < \bar{N}_{\mathcal{S}}$, since (10.9a) would be violated. Since Algorithm 10.3.12 always converges, the entire

feasible set $\mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ is covered by $\mathcal{ITP}$s and therefore $N_{\max} \geq \bar{N}_{\mathcal{S}}$. However, $N_{\max}$ can only be greater than $\bar{N}_{\mathcal{S}}$ if a region with horizon $N = \bar{N}_{\mathcal{S}}$ is bounded by a reachability facet (Type II). Only then would the algorithm increase N further; otherwise all facets would be covered for a horizon $N = \bar{N}_{\mathcal{S}}$ and the exploration would end. Since, by definition, all $x \in \mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}})$ can reach $\mathcal{O}_\infty^{\mathrm{LQR}}$ in at most $\bar{N}_{\mathcal{S}}$ steps, a region with horizon $N = \bar{N}_{\mathcal{S}}$ cannot be bounded by a Type II facet. These states have no impact on the result because $\bar{N}(x)$ is not defined for infeasible states (i.e., $x \in (\mathcal{S} \setminus \mathcal{K}_\infty(\mathcal{O}_\infty^{\mathrm{LQR}}))$). Therefore $N_{\max} = \bar{N}_{\mathcal{S}}$. □

Lemma 10.3.16 *In the infinite-horizon polyhedral state-space region partition a state can only remain within one region for at most one time step (except for $\mathcal{O}_\infty^{LQR}$).*

Proof Follows from Remark 10.3.8 and the implementation of Algorithm 10.3.12.□

Theorem 10.3.17 (Invariance of Feasible Set, [GBTM03, GBTM04]) *The union of all $\mathcal{ITP}$'s computed with Algorithm 10.3.12 is positive invariant if $\mathcal{S}_F$ is bounded and closed.*

Proof Follows from Lemma 10.3.16 and Theorem 3.2.1 and 10.3.13. The state will always move to a region with horizon $N - 1$ at the next time step until the unconstrained region $\mathcal{O}_\infty^{\mathrm{LQR}}$ is reached. □

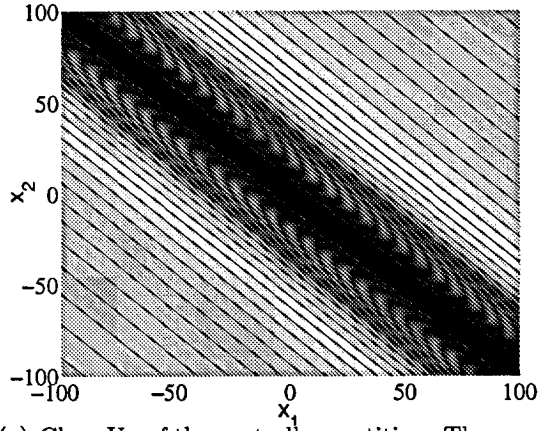
Remark 10.3.18 *As previously stated, $\bigcup \mathcal{ITP}_r \subset \mathcal{K}_\infty(\mathcal{O}_\infty^{LQR})$ if $\mathcal{K}_\infty(\mathcal{O}_\infty^{LQR})$ has open boundaries or $\mathcal{S} \subset \mathcal{K}_\infty(\mathcal{O}_\infty^{LQR})$. The union of all regions is generally not invariant in this case. However, the union of regions can easily be made invariant by modifying the on-line application of the feedback law. If the trajectory enters part of the state space where no region was found, the open loop solution of the previous region is applied. Since the open-loop is equal to the RHC closed-loop solution for the infinite-horizon controller (see Remark 10.3.8), optimality and invariance is preserved.*

For certain classes of problems Algorithm 10.3.12 is more efficient than standard multi-parametric solvers, even if finite horizon optimization problems are being solved. The initial polyhedral representation $\mathcal{P}_r$ contains redundant constraints which need to be removed in order to obtain a minimal representation of the controller region. As stated in Chapter 9, the number of initial half-spaces grows linearly with the prediction horizon. Hence, if a standard algorithm is used to compute the CLQR, it is necessary to compute the solution for a fixed horizon $N = \bar{N}_{\mathcal{S}}$.

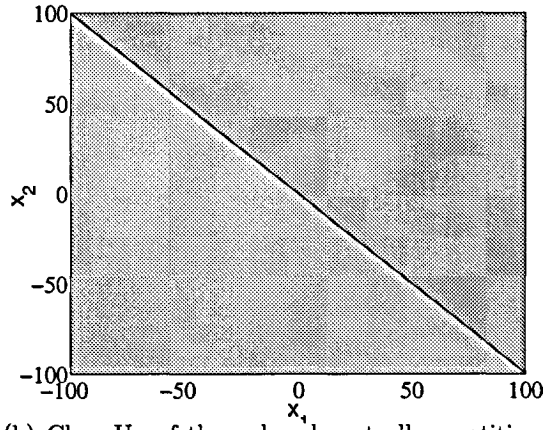
In the CLQR algorithm presented here, the number of initial half-spaces for each $\mathcal{ITP}^N$ (10.9) grows linearly from iteration to iteration. Specifically, the number of half-spaces in (10.9a) are constant and the number of half-spaces in (10.9b) grow linearly with N, i.e. (10.9b) is obtained for N, whereby N is increased at each iteration and takes on values between 1 and $\bar{N}_{\mathcal{S}}$. Therefore, the traditional multi-parametric solver needs to compute the minimal representation of (10.9b) for a fixed $N = \bar{N}_{\mathcal{S}}$, whereas the CLQR algorithm proposed here needs to compute the minimal representation of (10.9) for varying N. It is therefore not possible to draw general conclusions on the efficiency of the two schemes. It is easy to come up with examples where either one outperforms the other.

It is possible to extend Algorithm 10.3.12 to speed up the identification of the active PWA feedback law for a given $x(0)$. Since the closed-loop solution is equal to the open-loop solution (see Remark 10.3.8) and a state only remains in one region for one time-step (see Lemma 10.3.16), it is possible to merge regions according to [BFT01, GTM04, GTM03]. With this method, regions with the same PWA control law on the first input are joined. If this procedure is applied to the PWA controller partition obtained for Example 10.3.10, the number of regions is reduced from 185 to 45 (see Figure 10.6).

It should be noted that the PWA controller may consist of a very large number of regions. The resulting partition may therefore be computationally prohibitive for on-line implementation in the form of a look-up table. However, as the next section will show, the CLQR obtained with Algorithm 10.3.12 may also be of relatively low complexity. Furthermore, the procedure described in Algorithm 10.3.12 can easily be adjusted to compute the finite horizon controller which may yield an off-line speedup compared to other algorithms [BMDP02, Bao02].



(a) Close-Up of the controller partition. The number of regions is $R = 185$.



(b) Close-Up of the reduced controller partition. The number of regions is $R = 45$ (most regions are located on the thin diagonal slab).

Figure 10.6: Infinite-Time Region partitions obtained by applying Algorithm 10.3.12 on Example 10.3.10.

Algorithm 10.3.12	22.43 sec
mp-QP [Bao02], $N = \bar{N}_S = 71$	37.99 sec

Table 10.2: Comparisons of computation times to compute the PWA solution for Example 10.3.10 on a Pentium III, 1.2 GHz. The solution consists of 185 regions.

10.3.4 Numerical Results

In this section, we will compare the computation time needed to obtain the PWA solution using Algorithm 10.3.12 to other approaches. Subsequently, we will compare the necessary effort to identify the active feedback law with the time needed to solve a QP. The region partitions that were obtained for Example 10.3.10 can be seen in Figure 10.6(a). The times needed to compute the PWA solution for Examples 10.3.10 using various algorithms are given in Table 10.2. The abbreviation mp-QP in Table 10.2 signifies that (4.7) is solved explicitly for horizon $N = \bar{N}_S$; Note that there is currently no algorithm to compute $\bar{N}_S$. Also note that the run-time for both algorithms could be further reduced by almost 50% by taking into account that symmetric constraints produce symmetric region partitions [TJB01]. To the authors'

Example 10.3.10	Average-Case	Worst-Case
$N = N_0 + c$	5.29 sec	10.13 sec
$N = 2^c N_0$	0.80 sec	2.48 sec
$N = \bar{N}_S$	0.47 sec	0.56 sec
Algorithm 10.3.12 (185 regions)	0.0042 sec	0.0056 sec

Table 10.3: Comparisons of the computation time necessary to identify the optimal input on a Pentium III, 1.2 GHz. N_0 is set to 1 and c is incremented by 1 at each iteration. The analysis is based on 100 random initial states.

knowledge, all comparable algorithms which were published to date, have larger run-times, even under the assumption that the horizon $\bar{N}_S$ is known. This is shown in Table 10.2 and was verified on numerous other examples.

Assuming the PWA controller partition has been computed, we will now compare the on-line times necessary to extract the PWA feedback law with the iterative QP algorithm of Scokaert and Rawlings [SR98] which was presented in Section 10.3.2. Also note that the PWA solution can give hard bounds on the worst case run-time whereas a QP based solution cannot provide such a bound without knowledge of $\bar{N}_S$. A scheme to decrease the time necessary to find the PWA feedback law further, was published in [BBBM01]. The authors use a cost-function to identify the optimal feedback law, greatly decreasing the required storage-space and additionally reducing identification times for the controller by a factor of 2 for the examples given here. This and various other schemes to speed up the set membership test will be discussed in Chapter 12. Note that, according to Theorem 10.3.15, Algorithm 10.3.12 can be used to compute $\bar{N}_S$ exactly. As an alternative to the look-up table, this value could subsequently be used to speed up the algorithm in [SD87, SR98].

10.4 Conclusions

Section 10.2 presented computationally cheap algorithms to reduce the computation effort of polytope reduction in multi-parametric programs. Computational experiments show that the time spent in polytope reductions may be reduced, leading to overall time savings of the multi-parametric programs, in general. A side-effect of the polytope reduction is that a bounding box is obtained. This bounding box can

be used in other parts of the multi-parametric algorithms to speed up some polytopic manipulations, leading to dramatic gains for some problems.

Section 10.3 presented an efficient algorithm for solving the infinite-horizon constrained linear quadratic regulator (CLQR) problem. The algorithm is based on multi-parametric quadratic programming and reachability analysis. When compared to on-line computation procedures, the time necessary to obtain the optimal input was significantly decreased, making CLQR an attractive solution even for fast processes. In addition, a method to compute the horizon $\bar{N}_S$ for compact sets has been presented. Exact knowledge of $\bar{N}_S$ can serve to improve the performance of a wide array of algorithms presented in the literature.

Seite Leer /
Blank leaf

11

Robust Low Complexity Feedback Control of Constrained Systems

This chapter will address the second lever for complexity reduction. Namely, the computation of controller partitions consisting of few regions.

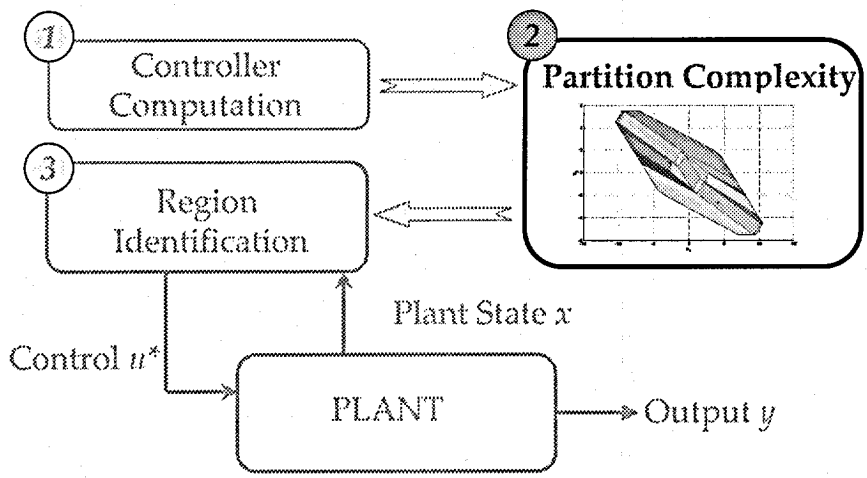


Figure 11.1: Illustration of the three levers for complexity reduction in receding horizon control. The second lever, which is the focus of this chapter, is highlighted.

As repeatedly stated in previous chapters, the complexity of multi-parametric programming *solutions* is *the* major bottle-neck in practical applications. The issue of complexity has been addressed in the literature mainly by reducing the storage space requirements for the solution and speeding up the on-line set-membership test necessary to find the active feedback law [BBBM01, TJB03b, RG04b, BBBM01]. However, because the initial computation grows exponentially with the problem size, these

methods do not make large problems tractable, since the initial computation is already prohibitive. Other procedures such as [BF01, GLPM03] may directly provide solutions of lower complexity (no post-processing), but a reduction is by no means guaranteed and, even when present, is not sufficient to make practical problems tractable.

There are numerous other (efficient) ways to obtain low complexity feedback controllers, but most of these schemes (e.g., [TJ02, RKC01, CKD03, JG03, KM04b][1]) do not meet the objectives considered in this section, namely, providing robust feedback control for all controllable states with guaranteed robust stability and constraint satisfaction.

In this section it will be shown how invariant set computation schemes [Bla92, Bla94, KG98, Bla99] can be combined with multi-parametric programming to simultaneously compute robust control invariant sets *and* the associated state-feedback controller. Furthermore, a computation scheme is presented which yields the linear feedback controller which produces the largest ellipsoidal robust invariant set for the closed-loop system. The results are derived from the publications in [GM03, GPM03].

Subsequently, it is shown how the controller computations may be combined with the stability analysis schemes presented in Section 8 to obtain feedback controllers of very low complexity. Specifically, the computation of a robust minimum-time controller and a so called 'N'-step controller is presented. Both controllers guarantee robust constraint satisfaction and robust convergence. In an extensive case study we demonstrate that, in general, the controller complexity for these controllers is orders of magnitude smaller than what is obtained by computing stabilizing optimal controllers which rely on terminal set constraints. The numerical results furthermore suggest a negligible performance decrease relative to linear and quadratic optimal control.

Definition of Systems with Uncertainties

In this chapter, we will consider systems subject to time-varying polytopic uncertainty and additive disturbances of the form

$$x(k+1) = A(\eta(k))x(k) + B(\eta(k))u(k) + w(k), \tag{11.1}$$

[1] Specifically, these schemes do not guarantee that the resulting controller will cover the maximum controllable set of states.

with

$$w(k) \in \mathbb{W}, \quad x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U}, \quad \forall k \geq 0, \tag{11.2}$$

where the sets $\mathbb{W}$, $\mathbb{X}$ and $\mathbb{U}$ are polytopic and $0 \in \text{int}(\mathbb{W})$, $0 \in \text{int}(\mathbb{X})$, $0 \in \text{int}(\mathbb{U})$. In addition,

$$[A(\eta)|B(\eta)] = \sum_{l=1}^{L} \eta^{(l)}[A^{(l)}|B^{(l)}], \quad \eta \in \Lambda, \ \eta = [\eta^{(1)}, \ldots, \eta^{(L)}], \tag{11.3a}$$

$$\Lambda \triangleq \{\eta \in \mathbb{R}^L \mid \sum_{l=1}^{L} \eta^{(l)} = 1 \ , \ \eta^{(l)} \geq 0\}, \tag{11.3b}$$

$$\Pi \triangleq \{[A(\eta))|B(\eta)] \in \mathbb{R}^{n \times (n+m)} \mid \eta \in \Lambda\}. \tag{11.3c}$$

The set Π is assumed to be polytopic. Furthermore, (11.1) can be written as

$$x(k+1) \in f(x(k), u(k), \Lambda, \mathbb{W}) = \{A(\eta(k))x(k) + B(\eta(k))u(k) + w(k), \\ \eta(k) \in \Lambda, \ w(k) \in \mathbb{W}\},$$

where $f(\cdot)$ is a set-valued function. If (11.1) is subject to an optimal (PWA) feedback law as given in Theorem 3.2.1, the following autonomous PWA system is obtained

$$x(k+1) = A_r(\eta(k))x(k) + g_r(\eta(k)) + w(k), \quad \eta(k) \in \Lambda, \ w(k) \in \mathbb{W}, \ x(k) \in \mathcal{P}_r, \tag{11.4}$$

with $A_r = A(\eta(k)) + B(\eta(k))F_r$ and $g_r(\eta(k)) = B(\eta(k))G_r$ whereby F_r and G_r are defined in Theorem 3.2.1. The set-valued equivalent to (11.4) can now be stated for $x(k) \in \mathcal{P}_r$:

$$x(k+1) \in f_{\text{PWA}}(x(k), \Lambda, \mathbb{W}) = \{A_r(\eta(k))x(k) + g_r(\eta(k)) + w(k), \\ \eta(k) \in \Lambda, \ w(k) \in \mathbb{W}\}.$$

Here, $x(k) \in \mathcal{P}_r$ defines the active dynamics r.

11.1 Invariant Set Computation

Invariant sets were covered in-depth in Chapter 7. We will briefly restate the main definitions and algorithms here:

Definition 11.1.1 (Maximal Robust Control Invariant Set $\mathcal{C}_\infty$) *The set $\mathcal{C}_\infty$ is said to be the maximal robustly control invariant set for the linear system in (11.1) subject to the constraints in (11.2) if it is robust control invariant and contains all the robustly control invariant sets contained in $\mathbb{X}$.*

Definition 11.1.2 (Robust 1-step Set $\mathrm{Pre}(\Phi)$) *Given the set $\Phi \subseteq \mathbb{X}$ the robust one step set $\mathrm{Pre}(\Phi)$ for system (11.1) subject to the constraints in (11.2) is*

$$\mathrm{Pre}(\Phi) \triangleq \{x \in \mathbb{X} \mid \exists u \in \mathbb{U} \text{ s.t. } f(x, u, \Lambda, \mathbb{W}) \subseteq \Phi\}.$$

Definition 11.1.3 (Maximal Robust Stabilizable Set $\mathcal{K}_\infty(\Phi)$) *For a robust invariant target set $\Phi \subseteq \mathbb{X}$, the N-step robustly stabilizable set $\mathcal{K}_N(\Phi)$ for system (11.1) subject to the constraints in (11.2) is defined by the set sequence $\mathcal{K}_c$*

$$\mathcal{K}_c \triangleq \mathrm{Pre}(\mathcal{K}_{c-1}),\ \mathcal{K}_0 = \Phi,\ c \in \{1, \ldots, N\},$$

where $\mathrm{Pre}(\cdot)$ is defined in Definition 11.1.2. The maximal robust stabilizable set $\mathcal{K}_\infty(\Psi)$ is the union of all N-step stabilizable sets.

The set $\mathcal{K}_\infty(\Phi) \subseteq \mathcal{C}_\infty$ contains all states which can be robustly steered into the robust invariant set Φ and can be computed as follows (cf. [Bla99, Ber72]):

Algorithm 11.1.4 (The Maximal Robust Stabilizable Set $\mathcal{K}_\infty(\Phi)$)

1. $\mathcal{K}_0 = \Phi$, *where Φ is robust invariant.*
2. $\mathcal{K}_{c+1} = \mathrm{Pre}(\mathcal{K}_c)$.
3. *If $\mathcal{K}_{c+1} = \mathcal{K}_c$, then $\mathcal{K}_\infty(\Phi) = \mathcal{K}_c$, return; Else, set $c = c+1$ and goto 2.*

Since Φ is robust invariant, it holds $\forall c \in \mathbb{N}$ that $\mathcal{K}_c(\Phi)$ is robust control invariant and $\mathcal{K}_c \subseteq \mathcal{K}_{c+1}$. Note that Algorithm 11.1.4 is not guaranteed to terminate in finite time.

The maximal control invariant set $\mathcal{C}_\infty$ can be computed as follows (cf. [Bla94]):

Definition 11.1.5 (λ Scaled Polytope) *We define the λ* Scaled Polytope *$\lambda\mathcal{P}$ ($\lambda \in \mathbb{R},\ 0 \in \mathcal{P}$) as*

$$\lambda P \triangleq \{\lambda x \in \mathbb{R}^n \mid x \in \mathcal{P}\}.$$

Algorithm 11.1.6 (The Maximal Robust Control Invariant Set $\mathcal{C}_\infty^\lambda$)

1. $\mathcal{C}_0 = \mathbb{X}$.
2. $\mathcal{C}_{c+1} = \mathrm{Pre}(\lambda\mathcal{C}_c)$.

3. If $C_c \subseteq C_{c+1}$, *then* $C_\infty^\lambda = C_{c+1}$, *return; Else, set* $c = c+1$ *and goto 2.*

All intermediate sets C_{c+1} (i.e. $C_c \nsubseteq C_{c+1}$) in Algorithm 11.1.6 are *not* robust control invariant. If λ is chosen as $0 \leq \lambda < 1$, then Algorithm 11.1.6 will converge to a robust invariant subset $C_\infty^\lambda \subseteq C_\infty$ in finite time[2] [Bla94].

The invariance properties of the intermediate sets in Algorithm 11.1.4 as well as the finite time termination of Algorithm 11.1.6 are relevant for the controller computation methods described in Section 11.3.

Remark 11.1.7 *Although* $\mathcal{K}_c(\Phi)$ *and* C_∞^λ *can be computed in finite time, the sets may become arbitrarily complex even for problems of low dimensions, placing a practical limit on the number of iterations which are tractable in Algorithms 11.1.4 and 11.1.6.*

11.2 Computing PWA Controllers that Enforce Set Invariance

This section will illustrate how polytopic robust control invariant sets along with an associated PWA control law can be constructed. The traditional method to achieve this task is based on projection and triangulation and will be discussed in Section 11.2.1. In Section 11.2.2, we will demonstrate how multi-parametric programming can be used to simultaneously obtain robust control invariant sets *and* the associated PWA feedback controllers. Finally, a case study to compare the two approaches is presented in Section 11.2.3.

11.2.1 PWA Control via Triangulation

In most control schemes based on set invariance, the computation of the invariant set and the associated control law are dealt with separately (e.g. [MS97, Bla94, GWKM04]).

Specifically, the sets $\mathcal{K}_\infty$ or C_∞ are computed by applying projection algorithms (e.g., [JKM04]) in Step 2 of Algorithms 11.1.4 and 11.1.6. Subsequently, the robust invariant set is divided into simplices using triangulation methods

[2] In order to obtain C_∞ exactly, it is necessary to set $\lambda = 1$. For $\lambda = 1$ there is no finite time termination guarantee.

[MS97, Bla94, GWKM04]. Finally, in order to obtain a feedback law, control sequences are computed for each vertex of each simplex. The linear interpolation of these input sequences yields an affine feedback law over each simplex. The input resulting from this controller is therefore a PWA function defined over a simplex partition of the robust invariant sets $\mathcal{K}_\infty$ or $\mathcal{C}_\infty$.

In our experience [GWKM04], triangulation is ill suited for control problems due to the inherent computational complexity of the off-line computations. For instance, the most established triangulation method, the *Delaunay Triangulation*, requires vertex enumeration and subsequent convex hull computation in a higher dimensional space [Zie94, Fuk04c], which is very expensive.

Furthermore, in on-line application, triangulation based controllers generally need to rely on open loop control in order to provide stability guarantees [GWKM04, WK03b] (this is not true for the minimum-time controller proposed in [MS97]). In addition, the triangulation based controller does not guarantee optimal performance.

11.2.2 PWA Control via Multi-Parametric Programming

The contribution of this section is to establish that multi-parametric programming can be used to simultaneously obtain the sets $\mathcal{K}_\infty$ or $\mathcal{C}_\infty$ *and* a PWA feedback law which makes these sets robust invariant. Unlike the methods in [Bla92, Bla94, MS97], there is no separation between set and controller computation in the scheme proposed here.

We will now show how to compute Pre$(\cdot)$ in Step 2 of Algorithms 11.1.4 and 11.1.6 via multi-parametric programming. This computation will yield both the set Pre$(\mathcal{T}_{\text{set}})$ along with a PWA feedback controller which will robustly drive all states into the target set, i.e. $x_k \in \text{Pre}(\mathcal{T}_{\text{set}}) \Rightarrow x_{k+1} \in \mathcal{T}_{\text{set}}$. Problem (11.5a) can be reformulated as

$$
\begin{aligned}
J_1^*(x(0)) \;=\; & \min_{u_0} (u_0^T Q_u u_0 + x_0^T Q_x x_0 + x_1^T Q_{x_N} x_1) && \text{(11.5a)}\\
\text{s.t.} \quad & x_1^l \in \mathbb{X} \ominus \mathbb{W},\; x_1^l \in \mathcal{T}_{\text{set}} \ominus \mathbb{W},\; u_0 \in \mathbb{U}, && \text{(11.5b)}\\
& x_1^l \;=\; A^{(l)} x_0 + B^{(l)} u_0,\; x_0 = x(0), \forall l \in \{1, \ldots, L\}, && \text{(11.5c)}\\
& x_1 \;=\; \bar{A} x_0 + \bar{B} u_0,\; x_0 = x(0), && \text{(11.5d)}\\
& Q_x \succeq 0,\;\; Q_{x_N} \succeq 0,\;\; Q_u \succ 0. && \text{(11.5e)}
\end{aligned}
$$

Here, $\bar{A}$ and $\bar{B}$ denote the nominal dynamics of the system. The feasible set $\mathcal{X}_1$ (here $N = 1$) is obtained by solving (11.5) as an mp-QP and Pre$(\mathcal{T}_{\text{set}}) = \mathcal{X}_1$ (see (11.5b)).

It follows from Theorem 3.2.1 that the mp-QP will also yield the PWA feedback law which will robustly drive all states in $\mathcal{X}_1$ into $\mathcal{T}_{\text{set}}$ in one time step. Hence, by applying multi-parametric programming techniques in Step 2 of Algorithm 11.1.4 ($\mathcal{T}_{\text{set}} = \mathcal{K}_c$) or Algorithm 11.1.6 ($\mathcal{T}_{\text{set}} = \mathcal{C}_c$) both the robust control invariant set *and* the associated control law are obtained. Note that this result is independent of the objective (11.5a), i.e. it would also be possible to use linear objectives.

11.2.3 Triangulation versus Multi-Parametric Programming

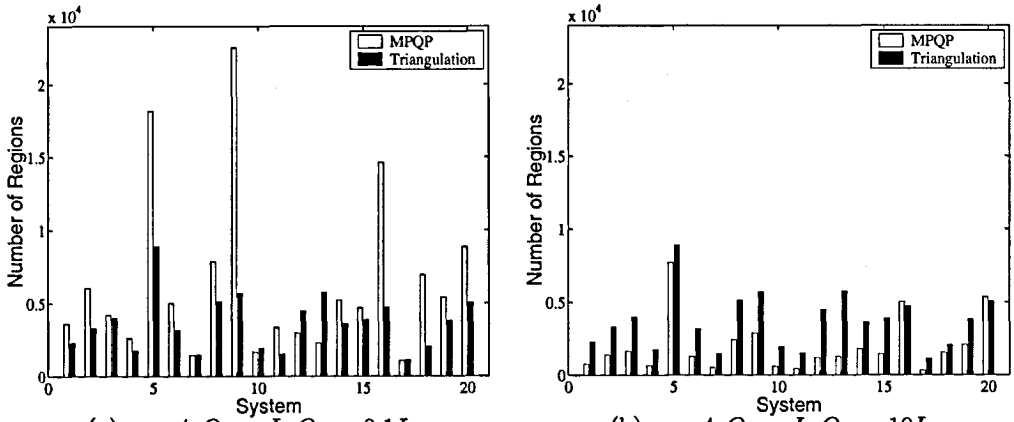
In this section, multi-parametric controllers will be compared with the only other established method, triangulation, to obtain explicit state feedback controllers which cover the maximal control invariant (or stabilizable) set $\mathcal{C}_\infty$ (or $\mathcal{K}_\infty$).

The focus of the comparison is on controller complexity, i.e. how many affine feedback laws define the PWA controller. For triangulation-controllers, the number of simplex regions depends only on the complexity of $\mathcal{C}_\infty$ (or $\mathcal{K}_\infty$) and not on the control objective, as is the case for multi-parametric programming.

The number of regions obtained with multi-parametric programming is compared to the number of regions of a simplex-controller in Figure 11.2. Specifically, 20 random stable systems with $n = 4$ states and $m = 2$ inputs were generated. Subsequently, both the infinite horizon optimal solution with the multi-parametric algorithm in [GBTM04] and a triangulation controller [GWKM04] were computed for different cost objectives in (11.5a). The systems considered here are nominal, i.e. $\mathbb{W} = \{0\}$. The number of simplices given in Figure 11.2 was obtained by applying the Delaunay triangulation [Zie94] to the maximal robust stabilizable set $\mathcal{K}_\infty(0)$.

Although the suboptimal triangulation based controller may consist of fewer regions than the infinite horizon optimal multi-parametric counterpart, we will show in this chapter that controller computation based on parametric programming can be far superior if the only objective is to enforce robust set invariance. It trivially holds that a simplex partition of a robust invariant set will always consist of more (or an equal number of) control laws than the *simplest* polytopic partition of the same set.

Section 11.3 will illustrate how polytopic controller partitions of low complexity can be obtained and in Section 11.4 these methods are applied to the systems considered in this section (i.e. in Figure 11.2).



(a) $n = 4, Q_x = I, Q_u = 0.1I$ (b) $n = 4, Q_x = I, Q_u = 10I$

Figure 11.2: Number of controller partitions mp-QP [GBTM04] vs. Triangulation [GWKM04] for different cost objectives in (11.5a). The same systems were used for the two cost objectives.

11.3 Computing Robust Low Complexity Controllers

This section will illustrate how the set computation algorithms in Section 11.1 and the stability analysis schemes in Chapter 8 can be combined to obtain feedback controllers of very low complexity. Before giving an outline of this section, we will describe the control objectives that are pursued:

1. **Constraints Satisfaction:** The controller covers all states $\mathcal{K}_\infty(\mathcal{O}_\infty)$ or $\mathcal{C}_\infty$ and constraint satisfaction is guaranteed for all time for those states.

2. **Convergence:** The closed-loop system is robust convergent (see Definition 8.5.1).

3. **Low Complexity:** The PWA controller partition $\{\mathcal{P}_r\}_{r=1}^R$ consists of few regions (R is small).

Remark 11.3.1 *Since the set $\mathcal{K}_\infty$ or $\mathcal{C}_\infty$ may not be computable in finite time, we will only consider the finite time computable sets $\mathcal{K}_N$ and $\mathcal{C}_\infty^\lambda$ $(0 \leq \lambda < 1)$ in this section.*

In Section 11.3.1, we will first present a method to compute the linear feedback controller that is associated with the largest volume ellipsoidal robust invariant set of an LTI system. This computation is crucial for the computation of the minimum-time feedback controller which is subsequently presented in Section 11.3.2. Finally, the computation of a so-called 'N-step' controller is presented in Section 11.3.3.

11.3.1 Computing Linear Controllers for Enforcing Set Invariance

The contribution of this section is the introduction of an algorithm for computing the linear feedback controller which produces the largest ellipsoidal robust invariant set for uncertain linear systems of type (11.1).

We will first motivate the need for this section, before describing the proposed computation scheme in detail. The algorithms presented in the subsequent sections are closely correlated with Algorithm 11.1.4, which computes the maximal stabilizable set $\mathcal{K}_\infty$. If Algorithm 11.1.4 is initialized with a robust invariant set Φ, all sets obtained at subsequent iterations are robust invariant, which is of great importance, since finite time termination of Algorithm 11.1.4 cannot be guaranteed and the iteration may have to be aborted after a finite number of steps. The initialization set Φ is also crucial for minimum-time control, as will be illustrated in the next section. Furthermore, if the set Φ is large in volume, the subsequent iteration scheme converges more quickly. Hence, the problem of finding a robust invariant set or even the largest volume robust invariant set is of great interest, in practice. Although we are dealing with polytopic sets, it is safe to assume that the feedback law associated with the largest volume invariant ellipsoids will also yield large (though not necessarily the largest) polytopic invariant sets. This assumption is sound since the polytopic invariant set will always be larger in volume than the associated ellipsoidal invariant set for the problem setup described in this section.

In [Löf03, Appendix 5], it was shown how to apply LMI techniques to find the largest robust invariant ellipsoid for systems of type (11.1) subject to additive disturbances bounded by an ellipsoid. Here we will extend that concept by considering additive disturbances bounded by a polytope. Furthermore, we require the ellipsoid to be contractive for the nominal system, i.e. if no uncertainty is present the closed-

loop system is asymptotically stable in the ellipsoid. This is crucial for the robust convergence proof in Theorem 11.3.5.

The search for the maximal volume robust invariant ellipsoid $\mathcal{E} = \{x \in \mathbb{R}^n \mid x^T P x \leq 1\}$ and the linear feedback law F ($u = Fx$) can be posed as:

$$\min_{P,F} \det(P), \quad \text{such that,} \tag{11.6a}$$

$$x_{k+1} = (A(\eta) + B(\eta)F)x_k + w_k, \quad Fx_k \in \mathbb{U}, \tag{11.6b}$$

$$x_k \in \mathcal{E} \quad \Rightarrow \quad x_{k+1} \in \mathcal{E}, \qquad \forall w_k \in \mathbb{W},\ \forall [A(\eta) \mid B(\eta)] \in \Pi,\ \mathcal{E} \subset \mathbb{X}. \tag{11.6c}$$

We assume $\mathbb{W}$ to be a polytope with $0 \in \text{int}(\mathbb{W})$. The invariance condition in (11.6c) can be posed as an LMI by applying the S-procedure [BGFB94]:

$$\begin{pmatrix} (A^{(l)} + B^{(l)}F)^T P (A^{(l)} + B^{(l)}F) & (A^{(l)} + B^{(l)}F)^T P w^{(p)} \\ (w^{(p)})^T P (A^{(l)} + B^{(l)}F) & (w^{(p)})^T P w^{(p)} - 1 \end{pmatrix} + \tau \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0, \tag{11.7}$$

$$\forall [A^{(l)} \mid B^{(l)}] \in \text{vert}(\Pi),\ \forall w^{(p)} \in \text{vert}(\mathbb{W}), \tau \in \mathbb{R}_+.$$

A congruence transformation with $\begin{pmatrix} P^{-1} & 0 \\ 0 & I \end{pmatrix}$ and subsequent Schur complement turns (11.7) into:

$$\begin{pmatrix} \tau Z & 0 & (A^{(l)}Z + B^{(l)}K)^T \\ 0 & 1-\tau & (w^{(p)})^T \\ A^{(l)}Z + B^{(l)}K & w^{(p)} & Z \end{pmatrix} \succeq 0, \tag{11.8a}$$

$$\forall [A^{(l)} \mid B^{(l)}] \in \text{vert}(\Pi),\ \forall w^{(p)} \in \text{vert}(\mathbb{W}), \tag{11.8b}$$

where $Z = P^{-1}$ and $K = FP^{-1}$. Furthermore, we will require the feedback law to stabilize the system if no additive uncertainty is present. Again, this can be written as an LMI [BGFB94]:

$$\begin{pmatrix} Z & (A^{(l)}Z + B^{(l)}K)^T & Z & K^T \\ A^{(l)}Z + B^{(l)}K & Z & 0 & \\ Z & 0 & Q_x^{-1} & 0 \\ K & 0 & 0 & Q_u^{-1} \end{pmatrix} \preceq 0, \quad \forall l \in \{1, \ldots, L\}, \tag{11.9}$$

where Q_x and Q_u are defined in (11.5a). Condition (11.9) is crucial for the robust convergence proof in Theorem 11.3.5 to hold. Furthermore, if the obtained feedback

law F satisfies (11.9), we can apply the methods in [KG98] to obtain $\mathcal{O}_\infty$ in finite time, since $A^{(l)} + B^{(l)}F$ is Hurwitz $\forall l \in \{1, \ldots, L\}$ and there exists a common Lyapunov function (see (11.9)).

Although problem (11.8) is bilinear (τZ), it can be efficiently solved by doing a bisection on the scalar $0 \leq \tau \leq 1$ (follows from element [2,2] in (11.8) and $\tau \in \mathbb{R}_+$), so that we can now solve our original problem in polynomial time by using standard Semi-Definite Programming (SDP) software [Löf04, Stu99], i.e.,

$$\max_{Z,K} \det(Z), \quad \text{such that (11.2), (11.8), and (11.9) hold.} \tag{11.10}$$

The state and input constraints (11.2) can easily be posed as an LMI [KBM96].

Remark 11.3.2 *If the control input u is constrained by symmetric lower and upper bounds, i.e. $\|u\|_\infty \leq u_{max}$ and $\|x\|_\infty \leq x_{max}$, then the constraints can be imposed in a non-conservative manner. If this is not the case, conservative approximations of the constraints sets $\mathbb{U}$ and $\mathbb{X}$ are needed [KBM96], such that no solution to the SDP* (11.10) *may exist. In such cases, it is advisable to remove the system constraints from* (11.10) *and solve the relaxed problem instead*[3]*. The subsequent computation of $\mathcal{O}_\infty$ with the method in [KG98] will yield a robust invariant set satisfying the original problem constraints $u \in \mathbb{U}$ and $x \in \mathbb{X}$, if it exists. However, the volume of that set may be very small.*

11.3.2 Minimum-Time Controller

The contribution of this section is the introduction of an algorithm for the construction of minimum-time state feedback controllers of low complexity.

A constructive algorithm for computing a minimum-time controller was first presented in [KG87] and the concepts therein were later extended to uncertain systems in [Bla92]. A robust minimum-time state feedback controller not relying on on-line optimization was first presented in [MS97].

The difference between the approach proposed here and the scheme in [MS97] is in the construction of the state feedback controller and in the the structure of the feedback law. In [MS97], the authors proposed an implementation of Algorithm

[3] In order to obtain a bounded result, it is advisable to add 'artificial' constraints $\|u\|_\infty \leq u_{\max}$ and $\|x\|_\infty \leq x_{\max}$, with $u_{\max}$ and $x_{\max}$ such that the artificial constraints encompass the sets $\mathbb{U}$ and $\mathbb{X}$.

11.1.4 which required all polytopes $\mathcal{K}_c$ to be available in both half-space and vertex representation. In addition, the authors relied on triangulation to obtain the feedback controllers which is computationally very expensive as pointed out in Section 11.2.1. Furthermore, it is necessary to triangulate each set $\mathcal{K}_c$ obtained at Step 2 of Algorithm 11.1.4, such that the final controller complexity is much higher than indicated by Figure 11.2.

Off-Line Computation

The controller computation proposed here is based on multi-parametric programming and works as follows:

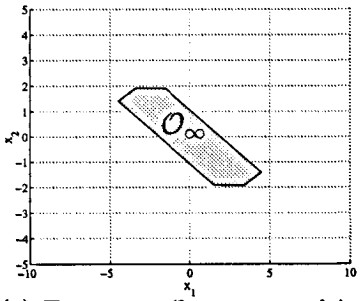
Algorithm 11.3.3 (Minimum-Time Controller: Off-Line Computation)

1. *Compute a linear feedback controller F such that the nominal system is asymptotically stabilized and the maximal robust invariant set $\mathcal{O}_\infty$ is non-empty (see Section 11.3.1 for details).*
2. *Compute $\mathcal{O}_\infty$ for the closed-loop system ($x_{k+1} = (A + BF)x_k$) according to [KG98].*
3. *Compute $\mathcal{K}_N(\mathcal{O}_\infty)$ according to Algorithm 11.1.4 using multi-parametric programming and store all controller partitions computed at intermediate iteration steps.*

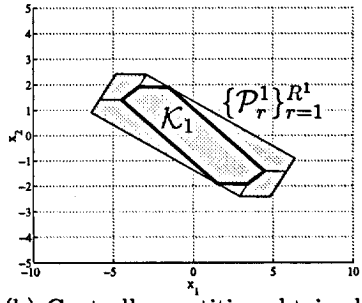
In Algorithm 11.3.3, the first two steps are needed to obtain the target set $\mathcal{O}_\infty$, while the iterative computation occurs in Step 3 (see Figure 11.3). Instead of solving one multi-parametric program for prediction horizon N, Step 3 in Algorithm 11.3.3 solves N multi-parametric programs for prediction horizon 1. Since, the overall complexity of a multi-parametric program is exponential in N [BMDP02], this scheme can be expected to yield controllers of lower complexity than standard optimal control schemes (e.g. [BMDP02, Bao02]), in general.

On-Line Computation

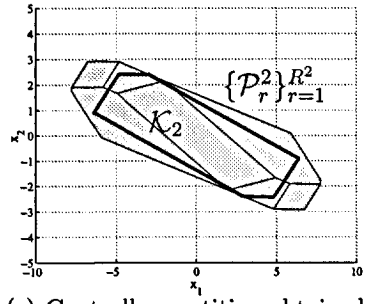
Since numerous multi-parametric programs are solved in Algorithm 11.3.3, several controller regions may overlap. In order to guarantee robust convergence and feasibility, the feedback law associated with the region computed at the smallest iteration number c, is selected for any given state x,



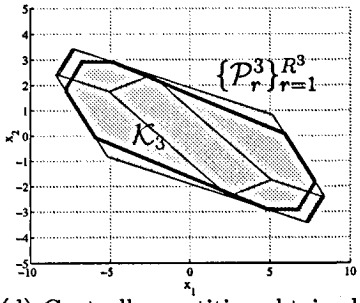
(a) Target set $\mathcal{O}_\infty$ computed in Step 2.



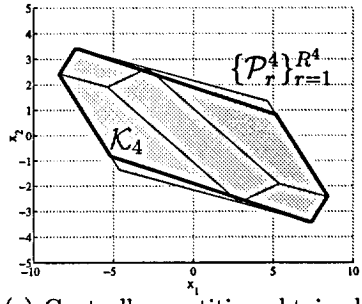
(b) Controller partition obtained after fist iteration ($\mathcal{T}_{\text{set}} = \mathcal{O}_\infty$) in Step 3.



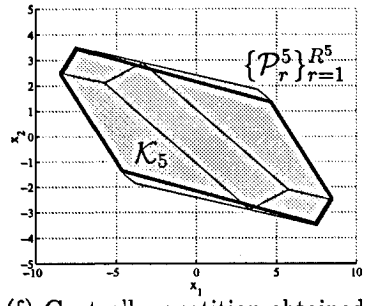
(c) Controller partition obtained after second iteration ($\mathcal{T}_{\text{set}} = \mathcal{K}_1 = \bigcup_{r=1,\ldots,R^1} \mathcal{P}_r^1$) in Step 3.



(d) Controller partition obtained after third iteration ($\mathcal{T}_{\text{set}} = \mathcal{K}_2 = \bigcup_{r=1,\ldots,R^2} \mathcal{P}_r^2$) in Step 3.



(e) Controller partition obtained after fourth iteration ($\mathcal{T}_{\text{set}} = \mathcal{K}_3 = \bigcup_{r=1,\ldots,R^3} \mathcal{P}_r^3$) in Step 3.



(f) Controller partition obtained after fifth iteration ($\mathcal{T}_{\text{set}} = \mathcal{K}_4 = \bigcup_{r=1,\ldots,R^4} \mathcal{P}_r^4$) in Step 3.

Figure 11.3: Illustration of Algorithm 11.3.3 applied to Example 5.2.7. The target sets $\mathcal{T}_{\text{set}}$ are depicted by bold outlines and the controller partitions are shaded.

Algorithm 11.3.4 (Minimum-Time Controller: On-Line Application)

1. *Obtain state measurement x.*
2. *Find controller partition* $c_{min} = \min_{c\in\{0,\ldots,N\}} c, \quad s.t.\ x \in \mathcal{K}_c$.
3. *Find controller region r, such that $x \in \mathcal{P}_r^{c_{min}}$ and compute* $u_0 = F_r^{c_{min}} x + G_r^{c_{min}}$.
4. *Apply input u_0 to system and go to Step 1.*

Here the sets $\mathcal{K}_c$ are defined as in Algorithm 11.1.4 and $\{\mathcal{P}_r^c\}_{r=1}^{R^c}$ is the controller partition computed at iteration c, consisting of R^c regions. Note that the region identification for this type of controller partition is much more efficient than simply checking all the regions. Steps 2 and 3 in Algorithm 11.3.4 correspond to two levels of

a search tree, where the search is first performed over the feasible sets $\mathcal{K}_c$ and then over the controller partition $\{\mathcal{P}_r^c\}_{r=1}^{R^c}$. Furthermore, one may discard all regions $\mathcal{P}_r^i$ which are completely covered by previously computed controllers (i.e. $\mathcal{P}_r^i \subseteq \bigcup_{j\in\{1,\ldots,i-1\}} \mathcal{K}_j$) since they are not time optimal.

Theorem 11.3.5 (Properties of Minimum-Time Control, [GM04, GM03]) *If the feedback law obtained with Algorithm 11.3.3 is applied as in Algorithm 11.3.4 to system (11.1), then the closed-loop system is robust convergent on $\mathcal{K}_N(\mathcal{O}_\infty)$. In addition, the constraints (11.2) will be satisfied for all time.*

Proof If the partitions obtained with Algorithm 11.3.3 are applied as in Algorithm 11.3.4, any state $x \in \mathcal{K}_c \setminus \mathcal{K}_{c-1}$ will be robustly driven into the set $\mathcal{K}_{c-1} \setminus \mathcal{K}_{c-2}$ in one time step. Since Algorithm 11.3.3 terminates in finite time, the state will enter the robust invariant set $\mathcal{O}_\infty$ in finite time. Once the state enters the robust invariant set $\mathcal{O}_\infty$, the trajectory will converge to the minimal robust invariant set $\mathcal{F}_\infty$ because the nominal closed-loop system in $\mathcal{O}_\infty$ is asymptotically stable (see Section 11.3.1, (11.9)) [KG98]. Constraint satisfaction is trivially guaranteed by the mp-QP (11.5) and the initialization $\mathcal{K}_0 = \mathcal{O}_\infty$ of Algorithm 11.3.3. □

11.3.3 N-Step Controller

The contribution of this section is the introduction of a control scheme which separately deals with the issues of robust constraint satisfaction and robust convergence. For small prediction horizon N, the resulting controllers are of very low complexity, as will be illustrated in Section 11.4.2.

In most receding horizon control schemes closed-loop stability is guaranteed by imposing an 'artificial' terminal set constraint with an associated cost on the final state [MRRS00]. This terminal constraint generally requires the use of large prediction horizons N which in turn results in significant computational complexity. The scheme proposed in this section does not rely on any artificial terminal set constraints, such that the use of large prediction horizons is not necessary. Specifically, we will present a controller computation scheme where robust constraint satisfaction is enforced by construction and robust stability is analyzed a posteriori. Since stability is analyzed a posteriori, there is no a priori guarantee that a stabilizing controller will be obtained.

The cornerstone of the proposed algorithm is the following mp-QP:

$$
\begin{aligned}
J_N^*(x(0)) \;=\; & \min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_N^T Q_{x_N} x_N && \text{(11.11a)}\\
\text{s.t.} \quad & x_1^{(l)} \in \mathcal{C}_\infty^\lambda \ominus \mathbb{W}, \quad \forall l \in \{1,\ldots,L\}, && \text{(11.11b)}\\
& x_k \in \mathbb{X},\ u_{k-1} \in \mathbb{U}, \qquad \forall k \in \{1,\ldots,N\}, && \text{(11.11c)}\\
& x_1^{(l)} \;=\; A^{(l)} x_0 + B^{(l)} u_0, \quad x_0^{(l)} = x(0), && \text{(11.11d)}\\
& x_{k+1} = \bar{A} x_k + \bar{B} u_k, \quad x_0 = x(0), && \text{(11.11e)}\\
& Q_x \succeq 0, \quad Q_{x_N} \succeq 0, \quad Q_u \succ 0. && \text{(11.11f)}
\end{aligned}
$$

Here, $\bar{A}$ and $\bar{B}$ denote the nominal system dynamics. In (11.11) robust constraints are enforced on the first time step (11.11b), but only nominal constraints are enforced for future time steps (11.11c), as in [CZ99b,Ker00]. This implementation is necessary to avoid problems of infeasibility which may occur when solving robust optimization problems in open-loop. The PWA state-feedback controller which is obtained when solving the mp-QP (11.11) will cover the maximal robust control invariant set (i.e. $\mathcal{X}_N = \mathcal{C}_\infty^\lambda$) and keep the receding horizon control (RHC) state trajectories within $\mathcal{C}_\infty^\lambda$ for all time.

The proposed algorithm can now be stated:

Algorithm 11.3.6 (N-Step Controller Design)

1. *Compute $\mathcal{C}_\infty^\lambda$ according to Algorithm 11.1.6 using projection methods (e.g., [JKM04]).*
2. *Solve* (11.11) *as an mp-QP to obtain a PWA feedback control law.*
3. *Compute a robust invariant set $\mathcal{O}$ with $0 \in \text{int}(\mathcal{O})$.*
4. *Analyze asymptotic stability (see Theorem 8.1.5) of the nominal system* (9.1) *on the set $\mathcal{O}$ and analyze robust convergence (see Theorem 8.5.4) of system* (11.1) *on the set $\mathcal{X}_N \setminus \mathcal{O}$ (see Chapter 8 for details).*

The advantage of computing $\mathcal{C}_\infty^\lambda$ instead of $\mathcal{K}_\infty(\Phi)$ in Step 1 of Algorithm 11.3.6 is that we do not require the computation (or even existence) of a robustly stabilizing linear feedback controller F as was the case in Algorithm 11.3.3. Furthermore, finite time convergence of Algorithm 11.3.6 is guaranteed by the scaling with $0 \leq \lambda < 1$,

as was illustrated in Section 11.1. Although the closed-loop system is guaranteed to remain within the bounded control invariant set $\mathcal{C}_\infty^\lambda$, asymptotic stability still needs to be verified in Step 3. of Algorithm 11.3.6.

Theorem 11.3.7 (Properties of N-step Control, [GM04, GM03]) *If the stability analysis in Algorithm 11.3.6 is successful and the feedback law obtained in Step 2 of Algorithm 11.3.6 is applied to system (11.1) in a RHC fashion, then the closed-loop system is robust convergent on $\mathcal{C}_\infty^\lambda$. In addition, the constraints (11.2) will be satisfied for all time.*

Proof The PWA controller obtained in Step 2 of Algorithm 11.3.6 ensures that $\mathcal{C}_\infty^\lambda$ is control invariant, since the constraint $x_1^{(l)} \in \mathcal{C}_\infty^\lambda$ in (11.5b) is imposed on the mp-QP (11.11). This guarantees constraint satisfaction for all time if $x(0) \in \mathcal{C}_\infty^\lambda$ and RHC is applied. Since the stability analysis in Step 4 was successful the rest of the proof follows directly from Theorem 8.5.5. □

Remark 11.3.8 *Note that the controller computed in Algorithm 11.3.6 is not subject to any artificial constraints (e.g., terminal set constraint or contraction constraints) since the constraints in* (11.11) *are non-restrictive, i.e. they are met by all controllers which satisfy Objective 1 (controller covers $\mathcal{C}_\infty^\lambda$) in Section 11.3.*

Remark 11.3.9 *If the stability analysis in Step 4 of Algorithm 11.3.6 fails, it is advisable to solve the mp-QP in Step 2 of Algorithm 11.3.6 using different weights Q_x, Q_u, Q_{x_N} and/or a different prediction horizon N. Alternatively, a different robust invariant set $\mathcal{O}$ in Step 3 can be computed. Slight modifications in Step 2 or 3 may make the subsequent stability analysis feasible.*

Remark 11.3.10 *If a system is not subject to uncertainty, it is advisable to select $\mathcal{O} = \emptyset$ which will make the implementation of Algorithm 11.3.6 significantly easier.*

11.4 Numerical Results

In this section we will first compare the controller complexity obtained with Algorithms 11.3.3 and 11.3.6 with other comparable controllers published in the literature. In Section 11.4.2 we will present a large number of random systems on which

we have tested the algorithms presented in this chapter and we will illustrate how controller complexity is drastically reduced compared to standard constrained finite time optimal control via mp-QP [BMDP02, TJB03a, GBTM04].

11.4.1 Construction of Robust Control Laws

In this section we will demonstrate that the controller complexity obtained with Algorithm 11.3.6 is significantly lower than the controller complexity obtained with other robust multi-parametric computation procedures published in the literature [KM04a, BBM03].

Example 11.4.1 *Consider the second order system in [KM04a]*

$$x(k+1) = \begin{pmatrix} 1 & 0.8 \\ 0 & 0.7 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k).$$

The task is to regulate the system to the origin while fulfilling the constraints $\|u(k)\|_\infty \leq 3$, $\|x(k)\|_\infty \leq 10$, $\forall k \geq 0$. *The weight on the state is set to* $Q_x = I$ *and the input-weight is* $Q_u = 1$. *The terminal weight* Q_{x_N} *is the solution to the algebraic Riccati equation.*

Parametric Uncertainty

We will assume a simple type of parametric uncertainty given by $L = 2$ and $\delta = 0.4$:

$$[A^{(1)}|B^{(1)}] = [(1-\delta)A \mid B] \qquad [A^{(2)}|B^{(2)}] = [(1+\delta)A \mid B]. \tag{11.12}$$

The algorithm in [BBM03], computed a robust controller for Example 11.4.1 in 35 seconds and 154 regions covering $\mathcal{C}_\infty$ were obtained. Using $N = 1$ in Algorithm 11.3.6, the computation took under 3 seconds and 3 regions were obtained using a quadratic objective[4]. Using an infinity-norm objective and $N = 3$, 64 regions were computed using the proposed 'N-step' controller. Note that although we have focused on quadratic objectives in this section, all methods are directly extendable to linear objectives in (11.11a).

The controllers obtained with the two algorithms both cover $\mathcal{C}_\infty$ and guarantee robust convergence and feasibility. The approach in [BBM03] enforces robust performance (i.e., min-max optimal for linear objective) whereas the algorithm presented

[4]Both simulations were run on a 2.2GHz Pentium III, with the NAG Library LP solver [Num02] and the SeDuMi [Stu99] LMI solver.

here only enforces nominal performance for prediction horizon $N = 1$. For a performance analysis, we refer the reader to Section 11.4.2 where a detailed comparison of Algorithm 11.3.6 with other methods in the literature is given.

Additive Uncertainty

We will assume additive uncertainty according to (11.1) as $\mathbb{W} = \{ w \mid \|w\|_\infty \leq \gamma \}$. With the LMI in (8.13), robust convergence on the maximal robust control invariant set $\mathcal{C}_\infty$ can be shown for $|\gamma| \leq 0.7$ in Example 11.4.1, using an N-Step controller with N=1. In [KM04a], Example 11.4.1 was solved for $\gamma = 0.1$, horizon $N = 2$ and 71 regions were obtained. The maximal robust invariant set $\mathcal{C}_\infty$ was not covered by the 71 regions. Using $N = 1$ in Algorithm 11.3.6, it was possible to compute 3 regions covering the entire set $\mathcal{C}_\infty$ with robust convergence and feasibility guarantees. For a performance analysis, we refer the reader to Section 11.4.2 where a detailed comparison of Algorithms 11.3.6 with other methods in the literature is given.

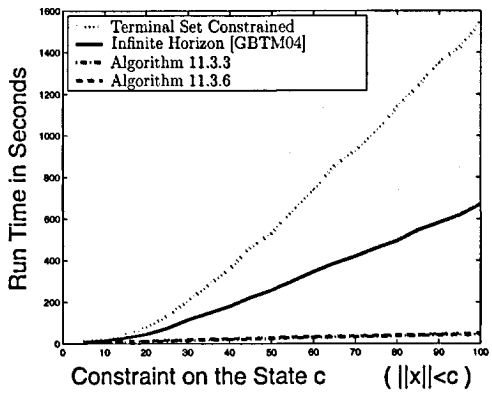
11.4.2 Case Study on Nominal Systems

In this subsection we give a detailed comparison of Algorithms 11.3.3 and 11.3.6 versus the finite- and infinite-horizon optimal controllers in [Bao02] and [GBTM04]. All controllers considered here cover the largest stabilizable set $\mathcal{K}_\infty(\mathcal{O}_\infty)$ and provide stability and feasibility guarantees. The set $\mathcal{O}_\infty$ was computed for the optimal LQR feedback law using the invariant set computation algorithm in [GT91]. Hence it is only necessary to compare complexity and performance. We will first study the influence of constraints (i.e., the volume of $\mathcal{K}_\infty(\mathcal{O}_\infty)$) on complexity before showing more general results for randomly selected systems.

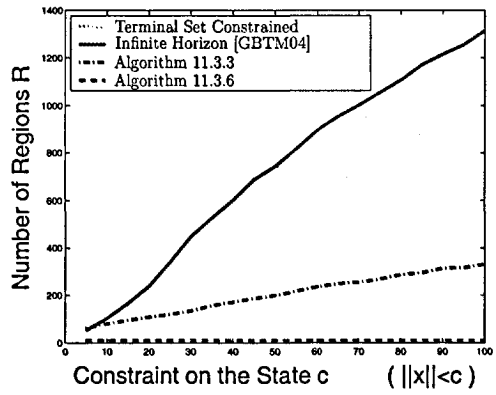
Example 11.4.2 *Consider the second order system with two inputs and outputs:*

$$x(k+1) = \begin{pmatrix} 0.7326 & -0.0861 \\ 0.1722 & 0.9909 \end{pmatrix} x(k) + \begin{pmatrix} 0.0609 & 0 \\ 0.064 & 1 \end{pmatrix} u(k).$$

The task is to regulate the system to the origin while fulfilling the constraints $\|u(k)\|_\infty \leq 5$, $\|x(k)\|_\infty \leq c$, $\forall k \geq 0$, *where the parameter c is used to change the volume of* $\mathcal{K}_\infty(\mathcal{O}_\infty)$. *The weight on the state is set to* $Q_x = I$ *and the input-weight is* $Q_u = 0.01I$. *The terminal weight* Q_{x_N} *is the solution to the algebraic Riccati equation.*



(a) Off-line computation time to obtain the feedback controller. Here, the computation times for Algorithms 11.3.3 and 11.3.6 are almost identical.



(b) Number of controller regions. Note that the results for the infinite horizon and terminal set algorithms are almost equal.

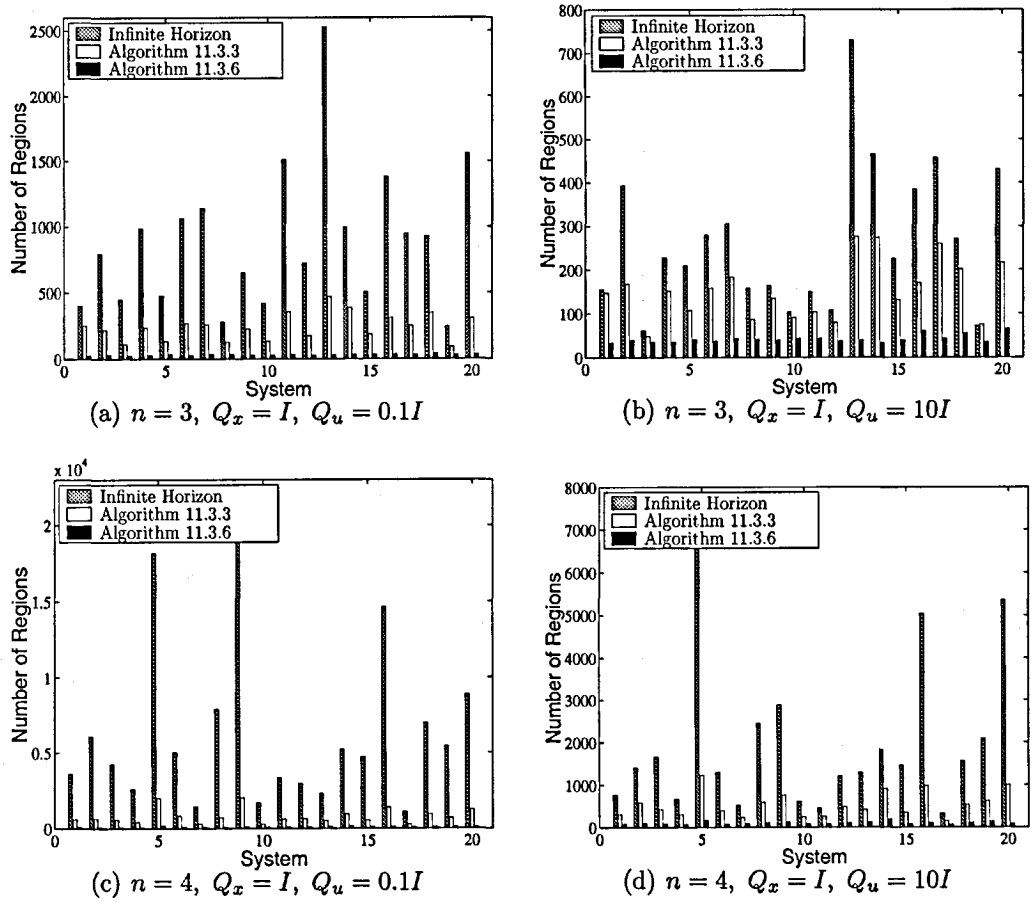
Figure 11.4: For Example 11.4.2 a controller which covers $\mathcal{K}_\infty(\mathcal{O}_\infty)$ is computed with the methods in [Bao02], [GBTM04] and Algorithms 11.3.3 and 11.3.6 (for $N = 1$). The abscissa represents the varying constraints on the system state ($\|x(k)\|_\infty \leq c$), i.e. the volume of the maximal robust invariant set is continually increased.

We will now compare Algorithm 11.3.3 and 11.3.6 with the infinite-horizon algorithm presented in [GBTM04] and a terminal set constrained algorithm [Bao02, MRRS00], where the terminal set was chosen to be $\mathcal{O}_\infty$. Figure 11.4 depicts the results for various values of the constraint parameter c in Example 11.4.2.

As can be seen from Figure 11.4, Algorithms 11.3.3 and 11.3.6 outperform the other algorithms in solution complexity (number of regions) and necessary computation time. It might be surprising that the infinite horizon algorithm in [GBTM04] is faster than the standard finite horizon solutions with terminal set constraints [Bao02]. We refer the reader to Section 10.3 for a detailed discussion of this property.

Having established the infinite-horizon algorithm in [GBTM04] as a valid basis for comparison, we will now examine the complexity decrease and degradation in performance incurred by Algorithms 11.3.3 and 11.3.6 based on 40 random stable systems with $n = 3$ to 4 states and $m = 2$ inputs. The inputs for all systems were constrained to $\|u(k)\|_\infty \leq 1$ and the states were limited to $\|x(k)\|_\infty \leq 10$,

$\forall k \geq 0$. Two different performance objectives in (11.5a) were considered: small and large weights on the input, i.e. $Q_u = 0.1I$ and $Q_u = 10I$. $Q_x = I$ was used throughout. As can be gathered from Figures 11.5 and 11.6, the decrease in controller

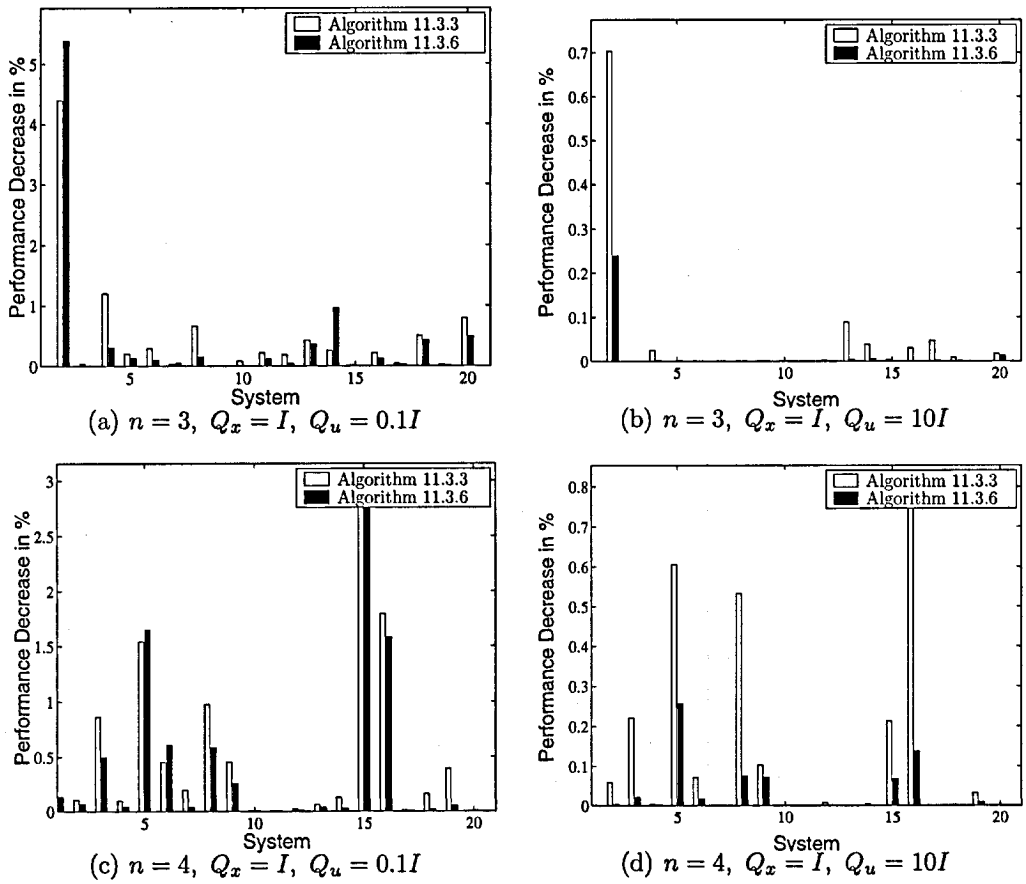


(a) $n = 3,\ Q_x = I,\ Q_u = 0.1I$ (b) $n = 3,\ Q_x = I,\ Q_u = 10I$

(c) $n = 4,\ Q_x = I,\ Q_u = 0.1I$ (d) $n = 4,\ Q_x = I,\ Q_u = 10I$

Figure 11.5: Complexity reduction versus the infinite horizon optimal controller in [GBTM04]. The results for 40 random systems with $n = 3$ to 4 states and $m = 2$ inputs for varying cost objectives in (11.5a) are given. Prediction horizon $N = 1$ was used in Algorithm 11.3.6.

complexity is substantial (e.g. Figure 11.5(c), system 9: $R = 22529$ for [GBTM04] vs. $R = 2021$ for Algorithm 11.3.3 vs. $R = 85$ for Algorithm 11.3.6). The LMI analysis in Algorithm 11.3.6 always succeeded in finding a PWQ Lyapunov function. Figure 11.5 indicates that the relative complexity decrease incurred by both Algorithms 11.3.3 and 11.3.6 grows with problem size. On average, Algorithm 11.3.3 decreases



(a) $n = 3,\ Q_x = I,\ Q_u = 0.1I$ (b) $n = 3,\ Q_x = I,\ Q_u = 10I$

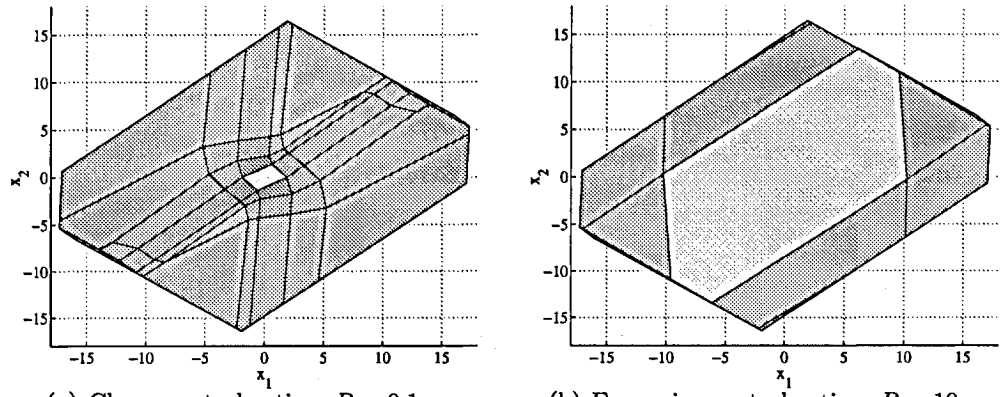
(c) $n = 4,\ Q_x = I,\ Q_u = 0.1I$ (d) $n = 4,\ Q_x = I,\ Q_u = 10I$

Figure 11.6: Performance decrease vs. [GBTM04]: Results for 40 random systems with $n = 3$ to 4 states and $m = 2$ inputs for varying cost objectives in (11.5a). Prediction horizon $N = 1$ was used in Algorithm 11.3.6.

complexity versus [GBTM04] by a factor of 10 and Algorithm 11.3.6 for $N = 1$ by a factor of 100 for the fourth order systems. At the same time, the average closed-loop performance is only 1% below the infinite-time optimal controller [GBTM04]. Performance was measured by gridding the state space and computing the closed-loop trajectory cost to the origin.

Since the methods presented in this chapter can easily be combined with the improved set-membership tests in [TJB03b, BBBM01, RG04b], on-line complexity may be decreased even further.

Remark 11.4.3 *It may seem surprising that the performance degradation is greater if the weight on the input is small (see Figure 11.6). Small weights on inputs would*



(a) Cheap control action, $R = 0.1$.

(b) Expensive control action, $R = 10$.

Figure 11.7: Two partitions obtained for a randomly generated LTI system with $R = 0.1$ and $R = 10$, respectively. The maximum admissible set for the LQR controller is depicted in yellow. See Remark 11.4.3 for a detailed discussion.

seem to indicate that the minimum-time controller in Algorithm 11.3.3 is optimal. However, small weights on the input inherently result in smaller invariant LQR sets $\mathcal{O}_\infty$, thus the degradation in performance. This is illustrated in Figure 11.7. For large weights on the input, almost no degradation is incurred with Algorithm 11.3.6 for $N = 1$.

Remark 11.4.4 *The random systems selected here are all stable such that the polytopic constraints* (11.2) *imply that the sets $\mathcal{K}_\infty(\mathcal{O}_\infty)$ and $\mathcal{C}_\infty$ are bounded and closed. This will guarantee finite time termination of Algorithms 11.3.4 and 11.3.6. For unstable systems, the sets $\mathcal{K}_\infty(\mathcal{O}_\infty)$ may have open boundaries such that the the associated numerical issues make unstable systems ill-suited for large-scale case studies.*

In our experience, larger prediction horizons ($N > 1$) are generally needed to stabilize an unstable system with the N-step control scheme. However, the incurred complexity reduction versus quadratic optimal control is still significant.

11.5 Conclusion

In this chapter, methods for computing polytopic robust invariant sets along with the associated feedback controllers are presented and novel schemes for combining

these methods with stability analysis of piecewise affine systems (see Section 8) are introduced.

Based on these tools, algorithms for the computation of two different robust feedback controllers for systems subject to polytopic and additive uncertainty are given (minimum-time and N-step controller). The extensive numerical examples clearly indicate that the complexity of the resulting controllers is generally orders of magnitude lower than that obtained with comparable algorithms. The results also indicate that the relative complexity decrease grows with the problem size, thus making large problems tractable. The difference in the solution complexity is mainly due to the choice of different control objectives. However, the proposed algorithms incur only a negligible penalty in terms of performance with respect to traditional control methods for the presented examples. The infinite-time stabilizable set $\mathcal{K}_\infty(\mathcal{O}_\infty)$ is covered by all controllers presented in this chapter and robust convergence and robust feasibility guarantees are given.

The presented algorithms are part of the MPT toolbox [KGB04] and can be downloaded from http://control.ee.ethz.ch/~mpt.

Seite Leer /
Blank leaf

12

Efficient Set Membership Tests

12.1 Introduction

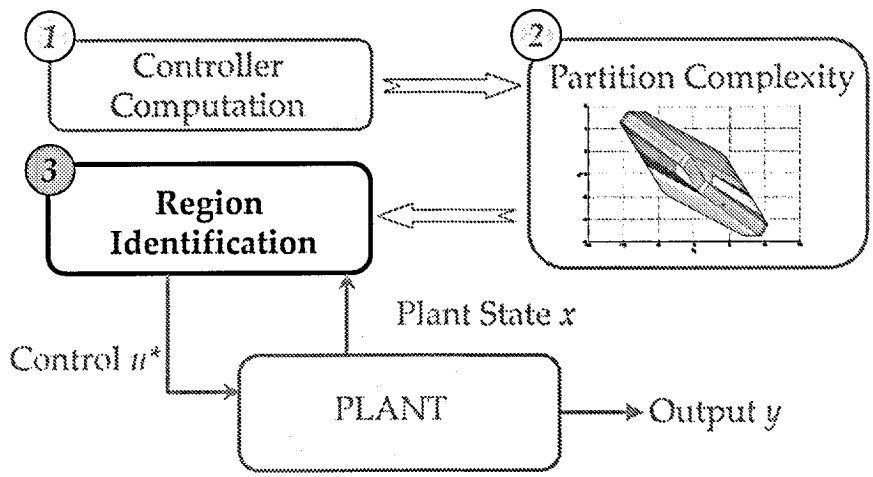


Figure 12.1: Illustration of the three levers for complexity reduction in receding horizon control. The third lever, which is the focus of this chapter, is highlighted.

This chapter will address the third lever for complexity reduction. Namely, efficient identification of the currently active feedback law.

As stated in Chapter 9, identifying which region a given state is contained in, is one of the key aspects determining the efficiency of explicit feedback controllers. This section will illustrate how controller partitions obtained via multi-parametric programming can be post-processed, such that the set-membership test can be performed efficiently. The aim is to reduce the data storage requirements as well as the on-line implementation time for the optimal control algorithms. Several authors have

investigated these issues although little has yet appeared in the published literature which gives significant reductions in complexity *and* is applicable to large controller partitions (i.e. several thousand regions).

In [BBBM01] the authors exploit the convexity properties of the piecewise affine (PWA) value function associated with linear MPC problems to solve the point location problem efficiently. Instead of checking whether the point is contained in the interior of a polyhedral region, each affine element of the value function is evaluated for a given state x. Since the value function is PWA and convex, the region containing x is associated with the affine function which yields the largest value for the state x (see Section 12.3 for details). Although this scheme is efficient and reduces storage space and region identification time, it is still linear in the number of regions R.

A different direction is taken in [TJB03b], where the authors propose to construct a bilinear search tree over the polyhedral state-space partition. Therein, auxiliary hyper-planes are used to subdivide the partition at each tree level. Note that these auxiliary hyper-planes may subdivide existing regions. The necessary on–line identification time is logarithmic in the number of subdivided regions, which may be significantly larger than the original number of regions. Although the scheme works very well for smaller partitions, it is not applicable to large controller structures due to the prohibitive pre-processing time. If R is the number of regions and $\bar{F}$ the average number of facets defining a region, then the approach requires the solution to $R^2 \cdot \bar{F}$ LPs[1]. However, the scheme in [TJB03b] is applicable to *any* type of *closed-form* MPC controller, whereas the algorithms proposed in this chapter are only applicable to controllers which have been obtained for LTI systems.

A similar approach was taken in [GTM04, GTM03]. Instead of computing a search tree with minimal depth as in [TJB03b], the authors compute the search tree with the minimal number of nodes, leading to significantly reduced storage complexity. Again, the necessary computations quickly grow prohibitive such that this approach cannot be applied to partitions consisting of several hundred regions[2].

Two alternative schemes for complexity reduction are proposed in this chapter. The first approach will henceforth be referred to as Interpolation mp-QP (IMPQP)

[1] It is possible to improve the pre-processing time at the cost of lest efficient (non-logarithmic) on-line computation times.

[2] It is possible to significantly improve on this limit at the cost of larger search-trees.

control [RG04b, RG05]. IMPQP is based on interpolation (e.g. [MKR00, RKC01]) of input sequences which provides feasibility and stability for the closed-loop system. The applied interpolation scheme allows for the removal of a certain number of regions, such that both storage space and region identification time is reduced.

The second approach is based on power-diagrams (extended Voronoi Diagrams) and combines the advantages of value function identification in [BBBM01] with the search tree in [TJB03b] to yield a region identification scheme which runs in $O(log(R))$ time, where R is the number of controller regions [JGR04b]. The two approaches are discussed in Sections 12.2 and 12.3, respectively, before concluding this Chapter in Section 12.4.

12.2 The IMPQP Algorithm

In this section[3], a novel interpolation scheme, *Interpolation mp-QP* (IMPQP), will be presented. The results in this section are derived from [RG04b, RG05].

The IMPQP scheme takes a controller partition (e.g., a partition computed with the methods in Chapter 11) as an input, and processes that partition in such a way that the number of controller regions is reduced, i.e. several regions are simply removed from the partition. In order to have a control law for the now 'missing' regions, an interpolation scheme is applied. IMPQP makes use of two interpolations: the first interpolation enforces infinite time feasibility while the second guarantees stability. We address each of the two schemes separately in the following subsections before concluding with an overview of the properties of IMPQP control.

Consider the quadratic optimal control problem

$$J_N^*(x(0)) = \min_{u_0,\ldots,u_{N-1}} \left\{ \sum_{k=0}^{N-1} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_N^T Q_{x_N} x_N \right\} \quad (12.1a)$$

$$\text{subj. to} \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}, \qquad k \in \{0,\ldots,N-1\}, \quad (12.1b)$$

$$x_N \in \mathcal{T}_{\text{set}}, \quad (12.1c)$$

$$x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0), \quad (12.1d)$$

$$Q_x \succeq 0,\ Q_{x_N} \succeq 0,\ Q_u \succ 0. \quad (12.1e)$$

[3]Note that the content of this section is the result of a collaboration with Anthony J. Rossiter who was the primary contributor.

Assumption 12.2.1 *The standing assumption throughout this section will be that the optimization problem (12.1) is posed such that stability and infinite-time feasibility are guaranteed for RHC (see Chapter 5), i.e. a terminal set constraint $\mathcal{T}_{set} = \mathcal{O}_{\infty}^{LQR}$ and appropriate terminal cost Q_{x_N} are imposed.*

For convenience [RRK91], the free inputs u_k will be reformulated in terms of a new variable c_k

$$u_k = -Kx_k + c_k, \qquad k = 0, ..., N-1, \tag{12.2}$$

where K denotes the Riccati LQR feedback (4.3). Hence the MPC optimization problem (12.1) can be written as:

$$\begin{aligned} J_N^*(x) &= \min_{C_N} C_N^T S C_N \\ \text{s.t. } & \tilde{G}C_N \leq \tilde{W} + \tilde{E}x(0), \quad C_N = [c_0^T, \ldots, c_{N-1}^T]^T. \end{aligned} \tag{12.3}$$

Details of how to derive the positive definite matrix S and matrices $\tilde{G}$, $\tilde{W}$ and $\tilde{E}$ from (12.1) are omitted (see e.g. [Mac02, RRK91] for details).

12.2.1 IMPQP: Interpolation Guaranteeing Feasibility

The first level of IMPQP computes an input sequence $C_N^{\text{feas}} = [c_0^T, c_1^T, \ldots, c_{N-1}^T]^T$ by optimizing the scalar interpolation parameter α between two input sequences $C_N^{(1)}$ and $C_N^{(2)}$ according to,

$$J_N^*(x(0)) = \min_{\alpha} (C_N^{\text{feas}})^T S C_N^{\text{feas}} \tag{12.4a}$$

$$\text{s.t. } \tilde{G}C_N^{\text{feas}} \leq \tilde{W} + \tilde{E}x, \quad x = x(0), \tag{12.4b}$$

$$C_N^{\text{feas}} = (1-\alpha)C_N^{(1)} + \alpha C_N^{(2)}. \tag{12.4c}$$

The input sequences $C_N^{(1)}$ and $C_N^{(2)}$ are selected, such that we can always guarantee the existence of a feasible interpolation parameter α. We furthermore want to ascertain an acceptable performance (i.e. small $J_N^*(x)$) for the resulting control scheme.

Specifically, the first input sequence, $C_N^{(1)}$, is obtained by assuming that the optimal Riccati LQR feedback law is applied for N steps to the system, i.e. it follows from (12.2) that $C_N^{(1)} = 0$ and therefore $C_N^{\text{feas}} = \alpha C_N^{(2)}$. The second sequence, $C_N^{(2)}$, is obtained by first projecting the current state x onto the nearest facet of the feasible

set $\mathcal{X}_N$, such that $\hat{x} \in \partial\mathcal{X}_N$ is obtained. The state $\hat{x}$ can be computed by intersecting the line touching the origin and x with the boundary of the feasible set $\partial\mathcal{X}_N$. The optimal constrained input sequence in (12.3) for $\hat{x}$ corresponds to $C_N^{(2)}$. Note that the sequence $C_N^{(1)}$ is optimal for the origin and $C_N^{(2)}$ is optimal for $\hat{x}$ and the current state x lies on the line intersecting the origin and $\hat{x}$. The purpose of this section is to find the optimal interpolation parameter α such that C_N^{feas} is obtained for x.

The following lemma can be used to establish the nearest facet of $\mathcal{X}_N$ for a given state $x \in \mathcal{X}_N$.

Lemma 12.2.2 *(i) Assume $\mathcal{X}_N$ is compact and contains the origin in it's interior. (ii) Normalize the inequalities defining $\mathcal{X}_N$ according to*

$$\mathcal{X}_N \triangleq \{x \in \mathbb{R}^n \mid Hx \leq K\}; \quad K = \begin{bmatrix} 1 \\ 1 \\ \vdots \end{bmatrix}, \quad H = \begin{bmatrix} h_1^T \\ h_2^T \\ \vdots \end{bmatrix}. \tag{12.5}$$

(ii) Compute the value $\gamma = \max_j h_j^T x$ and corresponding integer j. Then if $x \in \mathcal{X}_N$, it follows that $\frac{x}{\gamma}$ is located on the j-th facet of $\mathcal{X}_N$.

Proof The state $\frac{x}{\gamma}$ corresponds to the intersection of a ray emanating from the origin and moving through x, with the boundary of the set $\mathcal{X}_N$. Hence, we are looking for the biggest scaling parameter λ and associated facet j, such that $h_j \frac{x}{\lambda} = 1$. Therefore, $\gamma = \max_j h_j^T x$ is the maximum scaling factor. □

Theorem 12.2.3 (Existence of Feasible Parameter α, [RG04b, RG05]) *Given $x \in \mathcal{X}_N$ and $\gamma = \max_j h_j^T x$, find the optimal control sequence in* (12.3) *for the scaled state $\hat{x} = \frac{x}{\gamma}$, i.e. $C_N^{(2)} = C_N^*(\frac{x}{\gamma})$. Then the interpolated control move $C_N^{feas} = \gamma C_N^{(2)}$ satisfies the constraints* (12.4b) *for x and therefore $\alpha = \gamma$ is a feasible solution to the interpolation problem* (12.4).

Proof It holds by construction of (12.3) that

$$\tilde{G}C_N^{(2)} - \tilde{E}\hat{x} \leq \tilde{W}.$$

Because $0 \leq \gamma \leq 1$, this directly implies

$$\gamma\left(\tilde{G}C_N^{(2)} - \tilde{E}\hat{x}\right) = \tilde{G}\gamma C_N^{(2)} - \tilde{E}\gamma\hat{x} = \tilde{G}C_N^{\text{feas}} - \tilde{E}x \leq \tilde{W}.$$

Hence, $\alpha = \gamma$ is a feasible solution to the interpolation problem (12.4). □

While theorem 12.2.3 shows that a feasible solution to (12.4) exists (i.e. $\alpha = \gamma$), we will show in the following how the optimal interpolation can be obtained. We will now introduce the first interpolation of the proposed IMPQP control procedure.

Algorithm 12.2.4 (Feasibility Interpolation)

1. *Off-Line: Solve* (9.6) *as an mp-QP (e.g. Figure 12.2(a)) and remove all regions not sharing a facet with* $\mathcal{X}_N$ *(e.g. Figure 12.2(b,c)).*

2. *On-Line: For an initial state* $x(k)$*, compute* $\gamma = \max_j [h_j^T x(k)]$ *and the corresponding* j*, as in* (12.5).

3. *On-Line: The state* $x(k)/\gamma$ *is located on the* j*-th facet of* $\mathcal{X}_N$*. From all controller regions* $\mathcal{P}_i$ *sharing that facet, find* i *such that* $x(k)/\gamma \in \mathcal{P}_i$ *and evaluate the associated PWA feedback law to obtain* $C_N^{(2)}$.

4. *On-Line: Interpolate between the LQR sequence* $C_N^{(1)} = 0$ *and the constrained sequence* $C_N^{(2)}$ *The following minimization optimizes*[4] *the predicted performance over the interpolation* $C_N^{feas} = \alpha C_N^{(2)}$*, i.e. we can restate* (12.4) *as*

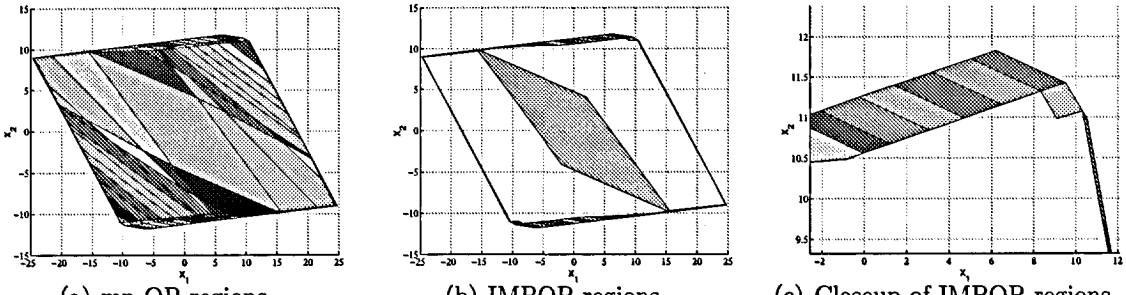
$$\min_{\alpha} \quad \alpha \quad s.t. \quad \begin{cases} \tilde{G}(\alpha C_N^{(2)}) \leq \tilde{W} + \tilde{E}x(k) \\ 0 \leq \alpha \leq 1 \end{cases} \tag{12.6}$$

   It follows from Theorem 12.2.3 that a feasible solution to (12.6) *will be found.*

5. *We obtain the sequence* $C_N^{feas} = \alpha C_N(x(k))$ *which will be used in the next section.*

Remark 12.2.5 *The resulting closed-loop sequence has the property of infinite-time feasibility because satisfaction of (12.6) directly implies that the state will remain within* $\mathcal{X}_N$*, where a feasible interpolated sequence can be found according to Theorem 12.2.3.*

[4]The objective is to 'push' the interpolated input sequence as far towards the LQR sequence $C_N^{(1)} = 0$ as possible. Since the objective in (12.4) is convex in C_N, minimizing α will thus always yield the optimal interpolation, e.g. in the case $\alpha = 0$ and we obtain the optimal LQR control law.



(a) mp-QP regions. (b) IMPQP regions. (c) Closeup of IMPQP regions.

Figure 12.2: Controller partition for a standard controller compared to the IMPQP controller.

In the next section we add a second interpolation which will guarantee the closed loop system is asymptotically stable.

12.2.2 IMPQP: Stability Interpolation

The classic RHC approach of 'shifting' the input sequence of the previous time step in order to prove stability [MRRS00] does not apply here [RBC03] (see Chapter 5). Without the 'tail' in the class of possible predictions, one cannot easily argue that the cost J_N^* is monotonically decreasing in time and in fact one can easily find simulations corresponding to the IMPQP feasibility interpolation (Algorithm 12.2.4) where J_N^* does not decrease monotonically.

Lemma 12.2.6 *Let $C_N(x(k)) = [c_0^T, c_1^T, \ldots, c_{N-1}^T]^T$ be the input sequence that is obtained at time k and that satisfies Assumption 12.2.1. If the input sequence $[c_1^T, \ldots, c_{N-1}^T, 0]^T$ is feasible at time $k+1$ for (12.6), this is sufficient to guarantee stability in the nominal case.*

The result of this Lemma is well known in the literature (see Section 5.2) and applies to the optimization problem (12.3) satisfying Assumption 12.2.1. Obviously, Algorithm 12.2.4 does not necessarily satisfy Lemma 12.2.6, since the input sequence $C_N^{(2)}$ and therefore the optimization problem (12.6) change at each time step, i.e. at time $k+1$ there may not exist an interpolation parameter α which yields the shifted sequence (i.e. $[c_1^T, \ldots, c_{N-1}^T, 0]^T$).

The proposal here is to add a second degree of freedom (e.g. [MKR00]) which corresponds to an interpolation of the input sequence C_N^{feas} of Algorithm 12.2.4 and

the 'tail' of the input sequence which was obtained at the previous time step. This second interpolation guarantees that the shifted sequence will be feasible at time $k+1$ and that Lemma 12.2.6 applies to the IMPQP scheme. Next, the IMPQP Algorithm is introduced.

Algorithm 12.2.7 (IMPQP Algorithm)

1. *At time k, store the input sequence $C_N^{feas} = [c_0^T, c_1^T, \ldots, c_{N-1}^T]^T$ obtained with Algorithm 12.2.4 as*
$$C_N^{tail} = [c_1^T, c_2^T, \ldots, c_{N-1}^T, 0]^T. \tag{12.7}$$

2. *At time $k+1$, define a linear interpolation between C_N^{feas} obtained by Algorithm 12.2.4 and the tail C_N^{tail}*
$$C_N^{mix} = (1-\beta)C_N^{feas} + \beta C_N^{tail}, \qquad 0 \leq \beta \leq 1. \tag{12.8}$$

3. *Minimize the predicted cost over the prediction class in (12.8).*
$$\min_{\beta} \quad J = (C_N^{mix})^T S C_N^{mix} = \beta^2 f + 2\beta g \tag{12.9}$$
$$s.t. \quad \begin{cases} 0 \leq \beta \leq 1 \\ f = [C_N^{tail} - C_N^{feas}]^T S [C_N^{tail} - C_N^{feas}] \\ g = [C_N^{tail} - C_N^{feas}]^T S C_N^{feas} \end{cases}$$

4. *Implement the control law $\mathbf{u} = -Kx_k + \mathbf{e}_1^T C_N^{mix}$, where K is the Riccati LQR feedback law and $\mathbf{e}_1 = [I\ 0\ \cdots\ 0]$ is used to extract only the first element of C_N^{mix}.*

Note that β will be zero unless the solution obtained with Algorithm 12.2.4 can be improved upon by moving towards C_N^{tail}.

12.2.3 Complexity and Properties of IMPQP Control

Some of the properties of IMPQP control are stated next.

Theorem 12.2.8 (Stability and Feasibility of IMPQP, [RG04b, RG05]) *Algorithm IMPQP guarantees both infinite-time feasibility and stability in the nominal case.*

Proof By definition both C_N^{feas} and C_N^{tail} are feasible and therefore, from convexity arguments, C_N^{mix} in (12.8) must also be feasible. It follows from feasibility of C_N^{mix}, that the predicted state x_N will be contained in the target set $\mathcal{T}_{\text{set}} = \mathcal{O}_\infty^{\text{LQR}}$. Therefore, Assumption 12.2.1 is satisfied and Lemma 12.2.6 applies to optimization (12.9), and hence comes the guarantee of stability. □

The on-line computational burden of IMPQP control is significantly smaller than that of comparable methods.

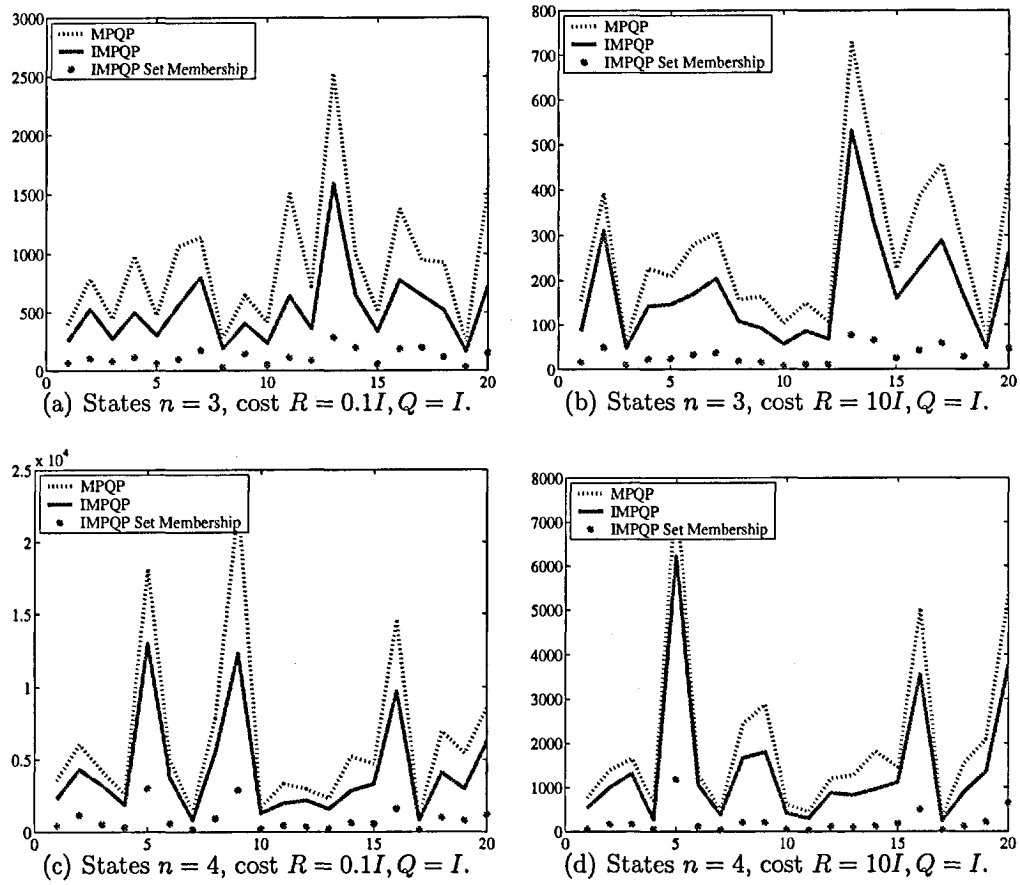
- It is only necessary to store controller regions which lie on a facet of $\mathcal{X}_N$, thus reducing the storage effort.
- A simple lookup table which associates facets of $\mathcal{X}_N$ to the controller regions sharing that facet can be created, such that the number of set-membership tests is reduced significantly.
- The additional on-line computations in (12.6) and (12.9) are also negligible since the implied minimizations are over scalars and thus are easy to implement.

12.2.4 Numerical Results

This section presents an extensive comparison of the IMPQP algorithm with traditional mp-QP controllers. The infinite horizon controller in Section 10.3 (i.e. [GBTM03, GBTM04]) is used as a basis for comparison. Both controllers cover the feasible set $\mathcal{X}_N$ and provide stability and feasibility properties. Hence it is necessary only to compare complexity and performance.

The comparison is based on 20 random systems with 3 and 4 states and 2 inputs (total of 40 systems). The inputs for all systems were constrained to $\|u\|_\infty \leq 1$ and the states were limited to $\|x\|_\infty \leq 10$.

Two different variations of the performance objective of (12.1a) were considered: that is the cases of small and large weights on the input, i.e. in (12.1a) the weights $Q_u = 0.1I$ and $Q_u = 10I$ were applied. $Q_x = I$ was used throughout. For consistency with other work, the cases considered are identical to those presented in [GLPM03, GM03].

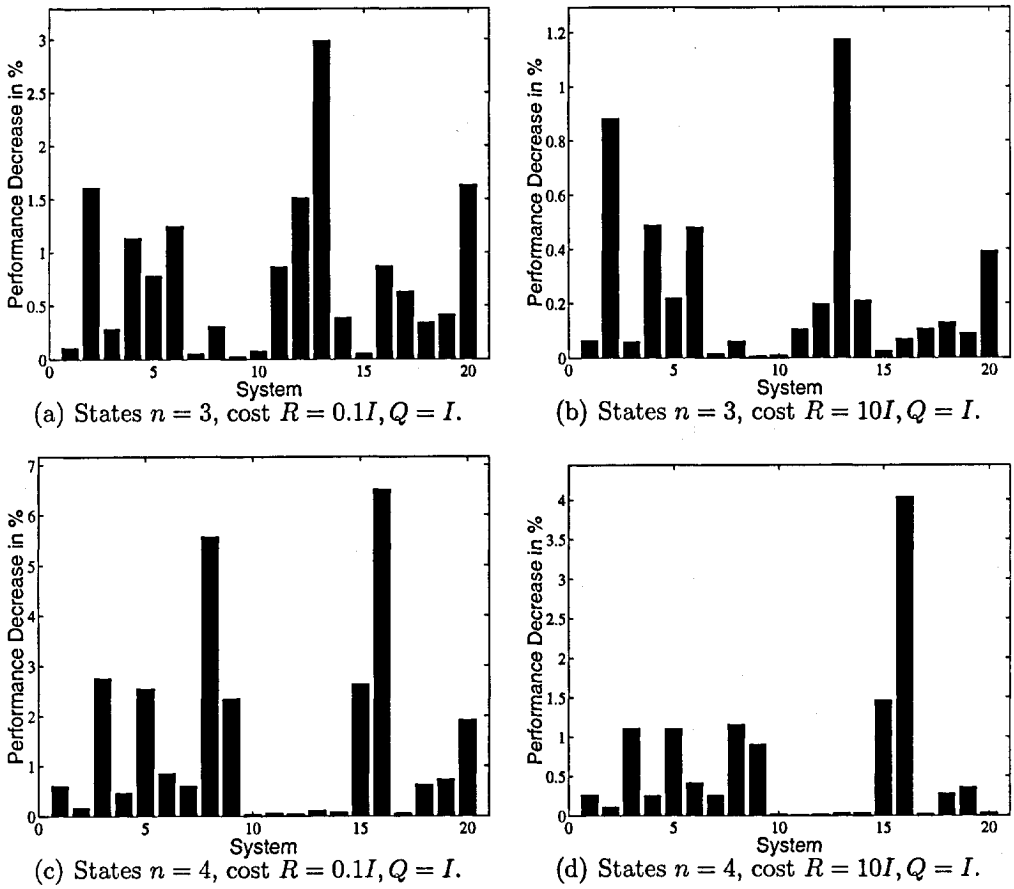


(a) States $n = 3$, cost $R = 0.1I, Q = I$.
(b) States $n = 3$, cost $R = 10I, Q = I$.
(c) States $n = 4$, cost $R = 0.1I, Q = I$.
(d) States $n = 4$, cost $R = 10I, Q = I$.

Figure 12.3: Comparison of IMPQP complexity versus infinite-horizon optimal control [GBTM04] for 40 random systems. The x-axis serves as index for the dynamic systems (20 for 3 and 4 states, respectively) and the y-axis indicates the number of regions obtained for each system.

Complexity comparisons

Figure 12.3 gives a comparison of the controller complexity of IMPQP versus the controller in [GBTM04]. The figure displays the number of regions which need to be stored for [GBTM04] and IMPQP respectively. The on-line effort for the set-membership test in [GBTM04] is proportional to the total number of regions whereas the IMPQP only needs to check the regions associated with the facet identified in step 3 of Algorithm 12.2.4. Hence for IMPQP, the dashed line is an indication of the storage space and the dotted line denotes the worst case on-line computational



(a) States $n = 3$, cost $R = 0.1I, Q = I$.

(b) States $n = 3$, cost $R = 10I, Q = I$.

(c) States $n = 4$, cost $R = 0.1I, Q = I$.

(d) States $n = 4$, cost $R = 10I, Q = I$.

Figure 12.4: Comparison of IMPQP performance versus infinite-horizon optimal control [GBTM04] for 40 random systems. The x-axis serves as index for the dynamic systems (20 for 3 and 4 states, respectively) and the y-axis indicates the performance decrease with respect to [GBTM04].

effort.

- The average storage requirements for the IMPQP algorithm is 50% of the mp-QP.

- The average on-line computational effort associated with set membership identification for the IMPQP algorithm is 10% of the requirement for the mp-QP solution (reduction by a factor of 10).

Performance comparisons

Figure 12.4 gives a comparison of the performance (calculating J_N^* of (12.1a) for the closed-loop trajectories) of IMPQP versus the controller in [GBTM04]. Performance was evaluated by gridding the feasible state space and summing up all the closed loop trajectory costs. The average performance decrease over all runs is 2%.

Summary of comparison

As can be gathered from Figures 12.3 and 12.4, IMPQP control exhibits a significant decrease in complexity in storage and set-membership test at very little cost in terms of performance. Furthermore, other post-processing schemes (e.g. [TJB03b, BBBM01]) may be used in combination with IMPQP to obtain an even greater reduction in complexity. However, similar to other simplification techniques (e.g. [TJB03b, BBBM01]), the IMPQP procedure relies on the a priori computation of the explicit control law, which may be prohibitive for large problems.

12.3 A Logarithmic Solution to the Point Location Problem for mp-LPs

The complexity of solving the set membership test (point location problem) as it occurs in explicit MPC formulations is clearly dependent on the number of regions in the solution. The number of regions is known to grow very quickly and possibly exponentially, with horizon length and state/input dimension [BMDP02]. The complexity of the solution therefore implies that an efficient method for solving the point location problem is needed.

As stated in the introduction to this chapter, the key contributions to this end have been made in [TJB03b] and [BBBM01]. In this section[5], we combine the concept of region identification via the value-function [BBBM01] with the construction of search trees [TJB03b], by using the link between *parametric* linear programming, Voronoi Diagrams and Delaunay triangulations, recently established in [RGJ04]. We demonstrate that the PWA cost function can be interpreted as a weighted power diagram, which is a type of Voronoi diagram, and exploit the results in [AMN+98]

[5]Note that the content of this section is the result of a collaboration with Colin Jones and Sasa Raković [JGR04b].

to solve the point location problem for Voronoi diagrams in logarithmic time at the cost of very simple pre-processing operations on the controller partition.

We focus on MPC problems with 1- or ∞-norm objectives and show that evaluating the optimal PWA function for a given state can be posed as a nearest neighbor search over a finite set of points. In [AMN+98] an algorithm is introduced that solves the nearest neighbor problem in n dimensions with R regions in time $O(c_{n,\epsilon} n \log R)$ and space $O(dR)$ after a pre-processing step taking $O(dR \log R)$, where $c_{n,\epsilon}$ is a factor depending on the state dimension and an error tolerance ϵ. Hence, the optimal control input can be found on–line in time logarithmic in the number of regions R.

The remainder of this section is organized as follows. In Section 12.3.1 the basic MPC problem is formulated, the structure of the closed-form solution is discussed and the problem addressed in this section is formulated. Section 12.3.2 demonstrates that the point location problem can be posed as a nearest neighbor search over R points. Section 12.3.3 provides a brief overview of the logarithmic nearest neighbor algorithm from [AMN+98]. Section 12.3.4 provides numerical examples and compares the approach to the current state of the art. Finally, conclusions are given in Section 12.4.

12.3.1 Problem Formulation

The problem formulation for linear performance objectives has been described in Chapter 9 such that we merely restate some of the notation here. If the linear p–norm used is the 1– or the ∞–norm, then (9.3) can be re-written as a linear program (LP):

$$J_N^*(x) = \min_y c^T y \tag{12.10a}$$

$$\text{s.t. } (x, y) \in \mathcal{Z}, \tag{12.10b}$$

by introducing an appropriate set of l slack variables ($\dim(y) > \dim(U_N)$). The polytope $\mathcal{Z}$ is closed and bounded and incorporates all system constraints (i.e., see (9.3)). The interested reader is referred to [Bor03, BBM00a] for details on how to compute an appropriate polytope $\mathcal{Z}$ and cost c such that (12.10) is equivalent to (9.3).

The first Nm elements of the optimizer $y^*(x)$ of LP (12.10) define the optimal control sequence $U_N^*(x) \triangleq [u_0^*(x)^T, \ldots, u_{N-1}^*(x)^T]^T$ for the optimal control problem (9.3). In MPC, the problem (9.3) is solved at each sampling instant, and the control law

$\kappa(\cdot)$ is defined as the first element in the optimal input sequence:

$$\kappa(x) \triangleq u_0^*(x).$$

Solution Structure

Since the problem (12.10) is an LP, it can be solved off-line as a multi-parametric linear program (mp-LP). See, for instance, [Bor03] for an algorithm for computing the solution to an mp-LP. First, we need to restate the notion of a *complex* of polytopes from Section 2:

Definition 12.3.1 (Complex, [Grü00]) *A finite family $\mathscr{C}$ of polytopes in $\mathbb{R}^n$ is a complex if*

- *Every face of a member of $\mathscr{C}$ is itself a member of $\mathscr{C}$*
- *The intersection of any two members of $\mathscr{C}$ is a face of each of them*

If a polytope $\mathcal{Q}$ is a member of a complex $\mathscr{C}$ we call $\mathcal{Q}$ a face of $\mathscr{C}$ and write $\mathcal{Q} \in \mathscr{C}$. Faces of dimension n are called *cells* of the complex. A controller partition obtained via multi-parametric programming is a complex. We introduce the concept of a 'complex' here in order to draw parallels to the computational geometry literature.

A basic result on the nature of the solution to a parametric linear program is given next:

Theorem 12.3.2 (Solution to an mp-LP, [Bor03])
Let $\mathcal{Z} \subset \mathbb{R}^{n+n_y}$ be a polyhedron and

$$\pi(\mathcal{Z}) \triangleq \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^{n_y} \text{ such that } (x,y) \in \mathcal{Z}\}.$$

For each x in $\pi(\mathcal{Z})$, let

$$J_N^*(x) \triangleq \inf_y \{c^T y \mid (x,y) \in \mathcal{Z}\}, \tag{12.11}$$

where $c \in \mathbb{R}^{n_y}$.

Then $J_N^*(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a convex, piecewise affine function defined over a complex $\mathscr{C}$ whose cells partition $\pi(\mathcal{Z})$. Furthermore, there exists a continuous, piecewise affine function[6] $v(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_y}$ such that $c^T v(x) = J_N^*(x)$ for every $x \in \pi(\mathcal{Z})$.

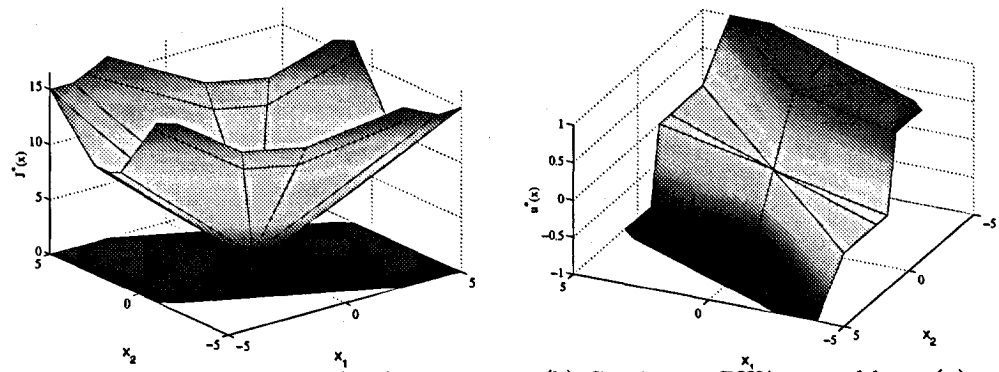
[6]Note that in general, the optimizer of (12.11) is set-valued.

Thus by Theorem 12.3.2, the optimal cost of (12.10) is a convex, piecewise affine function of the state x, taking $\mathbb{R}^n$ to $\mathbb{R}$ and is defined over a complex $\mathscr{C} \triangleq \{\mathcal{P}_1, \ldots, \mathcal{P}_R\}$:

$$J_N^*(x) = \alpha_r^T x + \beta_r, \quad \text{if } x \in \mathcal{P}_r, \quad r \in \mathcal{R}, \tag{12.12}$$

where each cell $\mathcal{P}_r$ is a polytope and $\mathcal{R} \triangleq \{1, \ldots, R\}$ is the index set for the regions. Furthermore, the optimizer of LP (12.10) is a piecewise affine function of x taking $\mathbb{R}^n$ to $\mathbb{R}^{N(m+l)}$ as is the control law $\kappa(\cdot)$, which takes $\mathbb{R}^n$ to $\mathbb{R}^m$ and is defined over the same complex:

$$\kappa(x) = u_0^*(x) = F_r x + G_r, \text{ if } x \in \mathcal{P}_r, \ r \in \mathcal{R}.$$



(a) Continuous and convex PWA value function $J_N^*(x)$

(b) Continuous PWA control law $\kappa(x)$

Figure 12.5: Illustration of the value function $J_N^*(x)$ and control law $\kappa(x)$ for a randomly generated mp-LP.

Point location problem

In on-line application of PWA controllers the following problem statement defines the active feedback law:
Given a measured state x and complex $\mathscr{C} \triangleq \{\mathcal{P}_1, \ldots, \mathcal{P}_R\}$, determine any integer[7]

[7]The state may be on the boundary of several regions.

$\mathsf{i}(x) \in \mathcal{R}$ such that polytope $\mathcal{P}_{\mathsf{i}(x)}$ contains x.

The function $\mathsf{i}(x)$ defines the control law $\kappa(x)$ as

$$\kappa(x) = u_0^*(x) = F_{\mathsf{i}(x)}x + G_{\mathsf{i}(x)}.$$

As $J_N^*(x)$ is convex, the calculation of $\mathsf{i}(x)$ can be written as [BBBM01]:

$$\mathsf{i}(x) = \arg\max_{r \in \mathcal{R}} \left\{ \alpha_r^T x + \beta_r \right\}. \tag{12.13}$$

As was proposed in [BBBM01], $\mathsf{i}(x)$ can be computed from (12.13) by simply evaluating the cost $\alpha_r^T x + \beta_r$ for each $r \in \mathcal{R}$ and then taking the largest. This procedure requires $2nR$ floating point operations and has a storage requirement of $(n+1)R$ real numbers.

In the following sections we will show that with a negligible pre-processing step, (12.13) can be computed in *logarithmic time.*

12.3.2 Point Location and Nearest Neighbors

In this section we show that for mp-LPs, the point location problem can be written as an *additively weighted nearest neighbor search*, or a search over R points in $\mathbb{R}^n$ to determine which is closest to the state x.

Consider the finite set of points called *sites* $\mathcal{S} \triangleq \{s_1, \ldots, s_R\}$ and the weights $\mathcal{W} \triangleq \{w_1, \ldots, w_R\}$, where $(s_r, w_r) \in \mathbb{R}^n \times \mathbb{R}$, $\forall r \in \mathcal{R}$. Given a point x in $\mathbb{R}^n$, the weighted nearest neighbor problem is the determination of the pair (s_r, w_r) that is closest to x. Associated with each site is a set of points $\mathcal{L}_r \subset \mathbb{R}^n$ such that for each $x \in \mathcal{L}_r$, x is closer to (s_r, w_r) than to any other site:

$$\mathcal{L}_r \triangleq \{x \mid \left\|s_r - x\right\|_2^2 + w_r \leq \left\|s_j - x\right\|_2^2 + w_j,\ \forall j \in \mathcal{R}\}. \tag{12.14}$$

Note that the sets $\mathcal{L}_r$ form a complex $\mathscr{C}_V \triangleq \{\mathcal{L}_1, \ldots, \mathcal{L}_R\}$ [Aur91]. If the weights w_r are all zero, then the sets $\mathcal{L}_r$ form a *Voronoi diagram*, otherwise they are called a *power diagram* [Aur91]. An example Voronoi diagram is shown in Figure 12.6 for a random set of sites. We now state the following result:

Theorem 12.3.3 (Existence of Power Diagram, [JGR04b]) *If $\mathscr{C}$ is a solution complex of an arbitrary parametric linear program, then there exists a power diagram with the solution complex $\mathscr{C}$.*

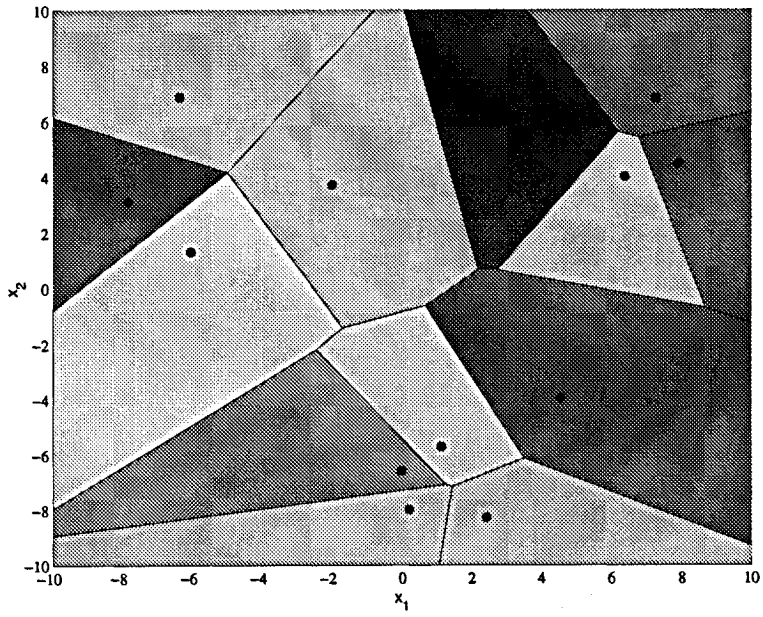


Figure 12.6: Example of a Random Voronoi Diagram

Proof It suffices to show that it is possible to define a set of sites and weights such that their power diagram is the solution complex of mp-LP (12.10), $\mathscr{C} \triangleq \{\mathcal{P}_1, \ldots, \mathcal{P}_R\}$.

It follows from Theorem 12.3.2 and (12.12)–(12.13) that x is contained in cell $\mathcal{P}_r$ if and only if

$$\alpha_r^T x + \beta_r \geq \alpha_j^T x + \beta_j, \qquad \forall j \in \mathcal{R},$$

or equivalently, if and only if:

$$-\alpha_r^T x - \beta_r \leq -\alpha_j^T x - \beta_j, \qquad \forall j \in \mathcal{R}.$$

Define the R sites and weights as:

$$\begin{aligned} s_r &\triangleq \frac{\alpha_r}{2}, \\ w_r &\triangleq -\beta_r - \left\|\frac{\alpha_r}{2}\right\|_2^2 = -\beta_r - \left\|s_r\right\|_2^2. \end{aligned} \tag{12.15}$$

For all $r \in \mathcal{R}$ and a given x it follows that:

$$\left\|s_r - x\right\|_2^2 + w_r = -\alpha_r^T x - \beta_r + \left\|x\right\|_2^2.$$

Recalling the definition of $\mathcal{L}_r$ in (12.14) we obtain to following:

$$\begin{aligned}
\mathcal{L}_r &\triangleq \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \left\| s_r - x \right\|_2^2 + w_r \\ \qquad \leq \left\| s_j - x \right\|_2^2 + w_j, \end{array} \quad \forall j \in \mathcal{R} \right\} \\
&= \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} -\alpha_r^T x - \beta_r + \left\| x \right\|_2^2 \\ \qquad \leq -\alpha_j^T x - \beta_j + \left\| x \right\|_2^2, \end{array} \quad \forall j \in \mathcal{R} \right\} \\
&= \left\{ x \in \mathbb{R}^n \;\middle|\; -\alpha_r^T x - \beta_r \leq -\alpha_j^T x - \beta_j, \;\; \forall j \in \mathcal{R} \right\} \\
&= \left\{ x \in \mathbb{R}^n \;\middle|\; \alpha_r^T x + \beta_r \geq \alpha_j^T x + \beta_j, \;\; \forall j \in \mathcal{R} \right\} \\
&= \mathcal{P}_r.
\end{aligned}$$

Thus the equivalence of the power diagram of the set of sites and weights (12.15) and the solution complex $\mathscr{C}$ of a corresponding mp-LP is established. □

A very important consequence of Theorem 12.3.3 is the point location problem (12.13) can be solved by determining which site s_r is closest to the current state x:

$$\begin{aligned}
\mathrm{i}(x) &= \left\{ r \in \mathcal{R} \;\middle|\; \begin{array}{l} \left\| s_r - x \right\|_2^2 + w_r \leq \\ \qquad \left\| s_j - x \right\|_2^2 + w_j, \end{array} \quad \forall j \in \mathcal{R} \right\} \\
&= \operatorname*{argmin}_{r \in \mathcal{R}} \left\| \begin{pmatrix} s_r \\ \sqrt{w_r} \end{pmatrix} - \begin{pmatrix} x \\ 0 \end{pmatrix} \right\|.
\end{aligned}$$

Since this problem has been well studied in the computational geometry literature we propose to adapt an efficient algorithm introduced in [AMN+98] that solves the nearest neighbor problem in *logarithmic time* and thereby solves the point location problem in *logarithmic time.*

Note that a necessary prerequisite for this approach to succeed is, that the state x is contained in the complex, i.e. $x \in \mathscr{C}$. If $x \notin \mathscr{C}$, the nearest neighbor search will yield a solution even though none exists (there does not exist an integer i such that $x \in \mathcal{P}_i$). For practical control problems this is not an issue, since the correct solution will be found if a solution exists.

The next section will give a brief introduction to the algorithm introduced in [AMN+98].

Remark 12.3.4 *In [Aur87] it was shown that a complex is a power diagram if and only if there exists a piecewise affine, continuous and convex function in* $\mathbb{R}^{n+1}$ *such that the projection of each affine piece of the function from* $\mathbb{R}^{n+1}$ *to* $\mathbb{R}^n$ *is a cell in the complex. This piecewise affine function is called a* lifting *of the complex. From the proof of Theorem 12.3.3, it is clear that the solution complex of every mp-LP has a lifting.*

Remark 12.3.5 *If a quadratic norm is used in the formulation of the MPC problem (e.g., see* (12.1)*) then the resulting solution complex may or may not have a lifting. Although it is not difficult to find problems for which a lifting does not exist, general conditions for the existence of a lifting for quadratic costs are not known. See [Aur91, Ryb99] for details on testing when a complex has an appropriate lifting.*

12.3.3 Approximate Nearest Neighbor: Logarithmic Solution

In this section, the key aspects of the approximate nearest neighbor search algorithm presented in [AMN+98] will be restated. Given a point $q \in \mathbb{R}^n$, a positive real ϵ and a set of R points in $\mathbb{R}^n$, the point p is a $(1+\epsilon)$-approximate nearest neighbor of q, if its distance from q is within a factor of $(1+\epsilon)$ of the distance from the true nearest neighbor.

Remark 12.3.6 *The* ϵ *error is required in order to prove the logarithmic search time [AMN+98]. As the optimal feedback* $\kappa^*(x)$ *is continuous (see Theorem 12.3.2) this error in determining the region translates into a maximum error in the input that is proportional to* ϵ*. Therefore, the error in the control input can be made arbitrarily small with an appropriate selection of* ϵ*.*

As shown in [AMN+98], it is possible to pre-process the R data points in $O(nR\log R)$ time and $O(nR)$ space, such that the approximate nearest neighbor can be identified in $O(c_{n,\epsilon}logR)$ time, where $c_{n,\epsilon}$ is a factor depending only on state-space dimension n and accuracy ϵ.

The authors in [AMN+98] propose to construct a so called *balanced box-decomposition tree* or BBD-tree. The BBD-tree is a hierarchical decomposition of the state-space into hyper-rectangles (cells) whose sides are orthogonal to the coordinate axes. The BBD tree has two key properties which are vital in obtaining

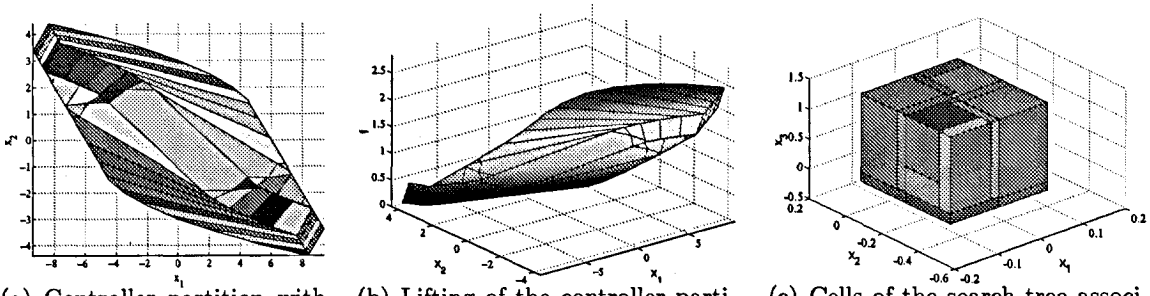
the logarithmic runtime bounds. Namely, as one descends the BBD-tree, the number of points associated with each cell decreases exponentially *and* the aspect ratio (ratio of longest to shortest side of each cell) is bounded by a constant.

The BBD-tree is constructed through the repeated application of two operations, *splits* and *shrinks*. A *split* subdivides a cell into two equally sized *children* by adding an axis-orthogonal hyperplane. This operation guarantees the exponential decrease in the number of points associated with each cell but it cannot give bounds on the aspect ratio. The *shrink*, partitions a cell into two subcells by using a hyper-rectangle which is located in the interior of the parent cell. The shrink operation corresponds to 'zooming in' to regions where points are highly clustered. A simple strategy to construct the BBD-tree is to apply splits and shrinks alternately. This procedure is repeated until the number of points associated with each cell is at most one.

In order to describe the on-line search, we will introduce the following definition: the *distance* between a point q and a cell is the closest distance between q and any part of the cell (Hausdorff distance, see Definition 8.5.2). Given a query point q, the algorithm first identifies the associated leaf cell by a simple descent through the tree in $O(\log R)$ time. It is then possible to enumerate the c cells closest to q in increasing order in $O(cn \log R)$ time [AMN+98]. The necessary number of cells c is bounded by a constant which can be determined without constructing the BBD-tree [AMN+98]. Each cell is then visited (closest cell first) and the closest point seen so far is stored as p. As soon as the distance from a cell to q exceeds $dist(p,q)/(1+\epsilon)$, it follows that the search can be terminated and p can be reported as the approximate nearest neighbor [AMN+98].

12.3.4 Numerical Results

In this section we consider various systems and compare the on–line calculation times of the method proposed in this section to the scheme in [BBBM01]. Although the scheme in [TJB03b] may lead to more significant runtime improvements than [BBBM01], the necessary pre-processing time is prohibitive for large partitions and we therefore refrain from a comparison to that scheme.



(a) Controller partition with $R = 65$ regions.

(b) Lifting of the controller partition (see Remark 12.3.5).

(c) Cells of the search tree associated with the lifting.

Figure 12.7: Search tree construction for Example 12.3.7.

Double Integrator

Example 12.3.7 *Consider the double integrator*

$$x_{k+1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} u_k$$

The task is to regulate the system to the origin while fulfilling the input constraint $||u_k||_\infty \leq 1$ *and state constraint* $||x_k||_\infty \leq 5$.

For Example 12.3.7, we consider the optimization problem (12.10) with a 2-norm objective for a prediction horizon $N = 10$. The objective weight matrices are set to $Q_x = Q_{x_N} = I$ and $Q_u = I$. For this example, there exists a lifting according to Remark 12.3.5 such that it is possible to construct the associated search tree. The construction process for Example 12.3.7 is depicted in Figure 12.7.

Large Random System

Example 12.3.8 *Consider the following 4-dimensional LTI system:*

$$x_{k+1} = \begin{bmatrix} 0.7 & -0.1 & 0 & 0 \\ 0.2 & -0.5 & 0.1 & 0 \\ 0 & 0.1 & 0.1 & 0 \\ 0.5 & 0 & 0.5 & 0.5 \end{bmatrix} x_k + \begin{bmatrix} 0 & 0.1 \\ 0.1 & 1 \\ 0.1 & 0 \\ 0 & 0 \end{bmatrix} u_k.$$

Subject to constraints $||u_k||_\infty \leq 5$ *and* $||x_k||_\infty \leq 5$.

Example 12.3.8 was solved for the infinity norm $p = \infty$, prediction horizon $N = 5$ and for weighting matrices $Q_x = I$ and $Q_u = I$. The resulting controller partition consists of $R = 12290$ regions. The construction of the search tree required 0.03 seconds. In comparison, the approach in [TJB03b] would require the solution to approximately 151'000'000 LPs, which is clearly prohibitive. For $\epsilon = 0.01$, the average and worst-case input computation times for ANN [MA98] are 29'450 and 36'910 floating point operations respectively[8]. In comparison, the approach in [BBBM01] always takes exactly 160'000 operations.

Randomly Generated Regions

In this section we compare the computational complexity of the approach presented in this section with that discussed in [BBBM01] for very large systems. The currently available multi-parametric solvers [KGB04] produce reliable results for partitions of up to approximately 30'000 regions [GM03]. However, methods are currently being developed that will provide solutions for much larger problems. Therefore, in order to give a speed comparison we have randomly generated vectors α_r and β_r in the form of (12.12). The code developed in [AMN+98], which is available at [MA98], was then used to execute 1000 random queries and the worst-case is plotted in Figure 12.8. For all of the queries the error parameter ϵ was set to zero and therefore the solution returned is the exact solution. It should be noted that the preprocessing time for one million regions in dimension 20 is merely 22.2 seconds.

Figure 12.8 shows the number of floating point operations (flops) as a function of the number of regions for the two approaches and the dimension of the state-space. Note that both axes are logarithmic.

A 3.0GHz Pentium 4 computer can execute approximately 800×10^6 flops/second. It follows that for a 10 dimensional system whose solution has one million regions, the control action can be computed at a rate of 20kHz using the proposed method, whereas that given in [BBBM01] could run at only 35Hz.

It is clear from Figure 12.8 that the calculation speed of the proposed method is very good for systems with a large number of regions. Furthermore note that controller partitions where ANN does worse than [BBBM01] are virtually impossible to generate, i.e. a partition in dimensions $n = 10$ with less than $R = 100$ regions is

[8] It us possible to derive hard upper bounds for the number of floating point operations. However, due to limited insights into the ANN code, we have refrained from doing so here.

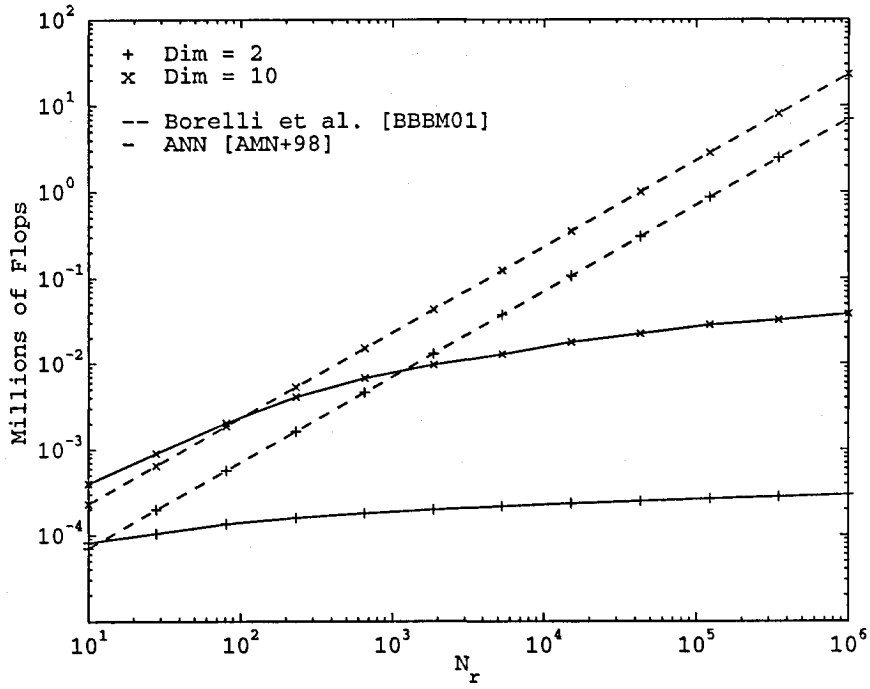


Figure 12.8: Comparison of the nearest neighbor search [AMN+98] with the set membership-test proposed in [BBBM01].

very difficult to contrive. Hence, it can be expected that in practice the proposed scheme will always result in a significant increase in speed. Since explicit feedback MPC is generally being applied to systems with very fast dynamics, any speedup in the set-membership test is useful in practice.

12.4 Conclusion

Two different post-processing schemes to simplify the set-membership test for a given partition have been introduced in this chapter. The first scheme (IMPQP, Section 12.2) can be applied to controller partitions which result from control problems with quadratic performance objectives. The second approach (nearest neighbor search, Section 12.3) is primarily applicable to controller partitions which result from control problems with linear performance objectives.

A novel interpolation based control scheme (IMPQP) was presented in Section

12.2, which allows for significant simplification of the on-line set-membership test necessary for mp-QP control for the nominal case. The main algorithm is based on two interpolations, the first of which aims at feasibility while the second guarantees stability. In extensive simulations it was shown that the procedure can reduce the necessary on-line effort by a factor of 10 with very little performance degradation, making it an attractive option for fast processes.

In Section 12.3 a method of solving the point location problem for linear-cost RHC problems was presented. If the controller partition exhibits a specific structure, the proposed scheme can also be applied to quadratic-cost RHC problems. It has been shown that the method is linear in the dimension of the state-space and logarithmic in the number of regions. Numerical examples have demonstrated that this approach is superior to the current state of the art.

The schemes proposed in this chapter are expected to significantly increase the sampling rates by which explicit feedback MPC can be applied.

Part IV

EFFICIENT CONTROL OF PWA SYSTEMS

Seite Leer /
Blank leaf

13

Problem Description

13.1 Introduction

This part of the thesis will address the topic of efficient feedback control of discrete-time, time-invariant, piecewise affine (PWA), systems subject to constraints. Here, the term *efficient* refers to any scheme which is able to simplify or speed-up the process of controller design and/or application, compared to current techniques.

Optimal control of PWA systems has garnered increasing interest in the research community since this system type represents a powerful tool for approximating non-linear systems and because of its equivalence to many classes of hybrid systems [Tor03, HSB01, Son96, Son81, Bem04]. The optimal control inputs for PWA systems may be obtained by solving mixed-integer optimization problems on-line [BM99a, MR03], or as was shown in [BCM03a, BBBM03, KM02, Bor03, DP00], by solving a number of multi-parametric programs off-line. Additional methods for controlling hybrid systems are reported in [LR03, MR03, KA02, MR02, BZ00, LTS99, TLS00].

In their pioneering work [BMDP02] the authors show how to formulate an optimal control problem for constrained linear discrete-time systems as a multi-parametric program (by treating the state vector as a parameter) and how to solve such a program (see Chapter 3). Basic ideas from [BMDP02] for linear systems were extended to PWA systems in [BCM03a, BBBM03, KM02, Bor03]. The associated solution (optimal control inputs) takes the form of a PWA state feedback law. If the control objective is linear, the state-space is partitioned into polyhedral sets and for each of these sets the optimal control law is given as an affine function of the state. For quadratic objectives the state space partition is not polyhedral, in general [BBBM03].

In the on-line implementation of these explicit controllers, input computation reduces to a simple set-membership test. Even though the approaches in [BCM03a,

BBBM03, KM02, Bor03] rely on off-line computation of a feedback law, the computation quickly becomes prohibitive for larger problems. This is not only due to the high complexity of the multi-parametric programs involved [GM03, BMDP02], but mainly because of the large number of multi-parametric programs which need to be solved when a controller is computed in a dynamic programming fashion [BBBM03, KM02].

In addition, there are few results in the literature which explicitly address the issue of computing feedback controllers which provide stability guarantees. The few publications which address this issue (e.g., [MR03]) assume the the origin is contained in the interior of one unique dynamic or rely on end-point constraints (e.g., [BBM00c]). The only exception is the infinite horizon solution proposed in [BCM03b], which is computationally intractable for large problems.

The subsequent chapters will deal with the following issues

Chapter 14: Posing an optimization problem such that receding horizon control of a PWA system guarantees stability and constraint satisfaction.

Chapter 15: Efficient computation of explicit feedback controllers for PWA systems.

Chapter 16: Problem formulations which yield low complexity feedback controllers for PWA systems.

13.2 Background and Definitions

A a detailed overview of multi-parametric programming principles is given in Chapter 3 and Receding Horizon Control (RHC) of linear time-invariant systems is addressed in Chapter 9. We will give a basic introduction to RHC of PWA systems

It was shown in [BBBM03, KM02, BBM00b] how to compute the optimal explicit feedback controller for PWA systems of the form

$$x(k+1) = f_{\text{PWA}}(x(k), u(k)) = A_i x(k) + B_i u(k) + f_i, \quad (13.1a)$$

$$\text{if} \quad [x(k)^T \; u(k)^T]^T \in \mathcal{D}_i, \quad i \in \mathcal{I}, \quad (13.1b)$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control vector and $\{\mathcal{D}_i\}_{i=1}^D$ is a bounded polyhedral partition of $(x, u) \subset \mathbb{R}^{n+m}$ space. For simplicity, the sets $\mathcal{D}_i$ here

define both regions in which a particular state update equation is valid as well as the constraints on the state and input variables. The set $\mathcal{I}$ is defined as $\mathcal{I} \triangleq \{1, 2, \ldots, D\}$ where D denotes the number of different dynamics. We will henceforth assume that the sets $\mathcal{D}_i$ are non-intersecting.
Henceforth, we will abbreviate (13.1a) and (13.1b) with $x(k+1) = f_{PWA}(x(k), u(k))$. The optimization problem considered here is thus given by

$$J_N^*(x) = \min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(||Q_u u_k||_p + ||Q_x x_k||_p\right) + ||Q_{x_N} x_N||_p \tag{13.2a}$$

$$\text{subj. to} \quad x_N \in \mathcal{T}_{\text{set}}, \tag{13.2b}$$

$$x_{k+1} = f_{PWA}(x_k, u_k), \; x_0 = x, \tag{13.2c}$$

using either the standard squared Euclidean norm ($p = 2$) or linear norms ($p = 1$ and $p = \infty$).

Definition 13.2.1 (Feasible Set $\mathcal{X}_N$) *We define the N-step feasible set $\mathcal{X}_N \subseteq \mathbb{R}^n$ as the set of initial states x_0 for which the optimal control problem (13.2) is feasible, i.e.*

$$\mathcal{X}_N = \{x_0 \in \mathbb{R}^n | \exists U_N = [u_0^T, \ldots, u_{N-1}^T]^T, \; s.t. \; x_N \in \mathcal{T}_{set}, \; x_{k+1} = f_{\text{PWA}}(x_k, u_k)\}.$$

In [BCM03a, KM02], multi-parametric Linear Programs (mp-LP) were solved in a dynamic programming fashion to obtain the feedback solution to (13.2). It was shown that the resulting feedback law is piecewise affine over polyhedra. In [BBBM03], the feedback solution to (13.2) with a quadratic objective in (13.2a) was computed by solving a sequence of multi-parametric Quadratic Programs (mp-QP) in a dynamic programming fashion. It was shown that the resulting feedback law is piecewise affine over (possibly) non-convex sets bounded by quadratic surfaces. Various additional methods to obtain explicit feedback solutions to linear or quadratic optimization problems for PWA systems are given in [Bor03, MR03, BCM03b, KM02].

Seite Leer /
Blank leaf

14

Construction of Stabilizing Controllers for Piecewise Affine Systems

A large part of the literature has focussed on end-point constraints to guarantee asymptotic stability of the closed-loop system (e.g., [Bor03, BBM00c, BM99a]). This type of constraint generally requires the use of large prediction horizons for the controller to cover the maximal attractive set (see Definition 7.1.9), such that the computational complexity quickly becomes prohibitive. Other methods (e.g., [MR03]) only provide stability guarantees if the origin is contained in the interior of one of the dynamics $\mathcal{D}_i$. In this section, a method is presented for obtaining stabilizing controllers for generic PWA systems. The results in this section are derived from [GKBM04b][1].

For any dynamical system, stability is guaranteed if an invariant set is imposed as a terminal state constraint in (13.2b) and the terminal cost in (13.2a) corresponds to a Lyapunov function for that set. In addition, the decay rate of the 'terminal Lyapunov function' must be greater than the stage cost (see Section 5.2). Here we show how to compute a control invariant set $\mathcal{O}_\infty^{\text{PWA}}$ with the associated Lyapunov function such that stability and constraint satisfaction of RHC is guaranteed. The scheme is based on the results in [MFTM00, RGK+04a] and was first published in [GKBM04a].

Remark 14.1.2 *We here cover the case where the origin is located on the boundary of multiple dynamics $\mathcal{D}_i$. The case where the equilibrium point is located on the boundary of multiple dynamics is by no means a pathologically rare case. Many physical systems exhibit a change in their dynamic behavior when certain states change their sign.*

[1] Note that identical results were simultaneously obtained by others in [LHWB04].

Before introducing the computation scheme, some of the assumptions must be stated. First, we require that the system dynamics $\mathcal{D}_i$ in (13.1b) are defined over the x-space only. Considering x-u-partitions would make the proposed scheme conservative to such an extent that the practicality of the approach proposed here becomes questionable. Specifically, it would be necessary to require that the feedback law F_i associated with each dynamic $\mathcal{D}_i$ satisfy the input constraints over the entire state space, i.e. $K_i x \in \mathbb{U}$, $\forall x \in \bigcup_{i \in \mathcal{I}} \mathcal{D}_i$.

We are furthermore assuming that the origin is an equilibrium state of the PWA system and hence the closed loop dynamics $f_i = 0$, $\forall i \in \mathcal{I}_0$ (see (13.1)). If this assumption is not satisfied, the approach proposed here will fail.

We will now show how terminal set $\mathcal{T}_{\text{set}}$ and cost Q_{x_N} can be computed such that stabilizing RHC controllers can be constructed for generic PWA systems. In a first step, we select all dynamics $i \in \mathcal{I}_0$ which contain the origin, i.e.

$$\mathcal{I}_0 \triangleq \{i \in \mathcal{I} \mid 0 \in \mathcal{D}_i\}.$$

The search for stabilizing piecewise linear feedback controllers F_i and an associated common quadratic Lyapunov function $V(x) = x^T P x$ can now be posed as

$$x^T P x \geq 0, \qquad \forall x \in \mathbb{X},$$
$$x^T (A_i + B_i F_i)^T P (A_i + B_i F_i) x - x^T P x \leq -x^T Q_x x - x^T F_i^T Q_u F_i x, \quad \forall x \in \mathcal{D}_i, \forall i \in \mathcal{I}_0.$$

If we relax this condition by setting $\mathcal{D}_i = \mathbb{R}^n$, $\forall i \in \mathcal{I}_0$, the problem can be rewritten as an SDP by using Schur complements and introducing the new variables $Y_i = F_i Z$ and $Z = \frac{1}{\gamma} P^{-1}$ (see [BGFB94, KBM96, MFTM00] for details),

$$\min_{Y_i, Z, \gamma} \gamma, \quad \text{s.t. ,} \tag{14.1a}$$

$$Z \succ 0, \tag{14.1b}$$

$$\begin{bmatrix} Z & (A_i Z + B_i Y_i) & (Q_x^{0.5} Z)^T & (Q_u^{0.5} Y_i)^T \\ (A_i Z + B_i Y_i)^T & Z & 0 & 0 \\ (Q_x^{0.5} Z) & 0 & \gamma I & 0 \\ (Q_u^{0.5} Y_i) & 0 & 0 & \gamma I \end{bmatrix} \succeq 0, \quad \forall i \in \mathcal{I}_0. \tag{14.1c}$$

where the scalar γ is introduced to optimize for the worst case performance, whereby the 'worst case' corresponds to an arbitrary switching sequence. Note that it may

not be possible for the worst case switching sequence considered in (14.1) to occur in practice, since not all dynamics i are defined over the entire state space.

Remark 14.1.3 *If* (14.1) *is posed for an LTI system (i.e.* $\mathcal{I}_0 = 1$*), the optimal LQR state feedback solution* K *and the solution to the Algebraic Riccati Equation* P *is recovered.*

Alternatively, one can solve a MAXDET problem (see Chapter 1) to obtain the largest invariant ellipsoidal target set [BGFB94] . Large target sets generally make the subsequent controller computations simpler, since fewer iterations are required to cover the state space of interest. Note however, that the feedback laws associated with the maximal volume invariant ellipsoidal set may not yield the maximal volume invariant polytopic set.

In a second step, the maximal admissible set $\mathcal{O}_\infty^{\text{PWA}}$ of the PWA system subject to the feedback controllers F_i can be computed with the algorithm in [RGK+04a] (see Chapter 7), which is guaranteed to terminate in finite time for the problem at hand, since the closed loop system is asymptotically stable.

The proposed computation scheme is summarized in the following algorithm:

Algorithm 14.1.4 (Computation of Maximal Admissible Set $\mathcal{O}_\infty^{\text{PWA}}$)

1. *Identify all dynamics* i *which contain the origin, i.e.* $i \in \mathcal{I}_0 \triangleq \{i \in \mathbb{N}^+ \mid 0 \in \mathcal{D}_i\}$.
2. *Solve* (14.1) *for all* $i \in \mathcal{I}_0$*, to obtain* F_i *and* P*. If* (14.1) *is infeasible, abort the algorithm.*
3. *Compute the maximal output admissible set* $\mathcal{O}_\infty^{PWA}$ *corresponding to the closed loop system* $x^+ = (A_i + B_i F_i)x$*, if* $x \in \mathcal{D}_i$ *with the method in [RGK+04a] (see Chapter 7).*
4. *Return the target set* $\mathcal{O}_\infty^{PWA}$*, the feedback laws* F_i *and the associated matrix* P.

Theorem 14.1.5 (Exponential Stability of RHC for PWA Systems, [GKBM04a, GKBM04b]) *Assume the optimization problem* (13.2) *is given with a quadratic objective, i.e.* (13.2a) *corresponds to*

$$J_N^*(x(0)) = \min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \left(u_k^T Q_u u_k + x_k^T Q_x x_k\right) + x_N^T Q_{x_N} x_N$$

$$s.t. \qquad Q_x \succeq 0, \quad Q_{x_N} \succeq 0, \quad Q_u \succ 0,$$

the terminal set is $\mathcal{T}_{set} = \mathcal{O}_\infty^{PWA}$ *and the terminal cost is* $\mathcal{Q}_{x_N} = P$ *(obtained with Algorithm 14.1.4). If this problem is solved at each time step for the PWA system* (13.1) *and only the first input is applied (Receding Horizon Control as described in Chapter 5), then the closed loop system is exponentially stable.*

Proof Algorithm 14.1.4 trivially satisfies the conditions for exponential stability in [MRRS00, Section 3.3] (also given in Section 5.2). □

Note that we only need to consider a single convex terminal set for linear systems [GT91, MRRS00] whereas for PWA systems, the terminal set $\mathcal{O}_\infty^{\text{PWA}}$ is given as a union of several convex sets $\mathcal{O}_\infty^{\text{PWA}} = \bigcup \mathcal{X}_i$. If the union $\bigcup \mathcal{X}_i$ is convex, the regions can be merged with the method in [BFT01]. Convexity of the target set is a desirable property since simpler target sets $\mathcal{T}_{\text{set}}$ generally lead to reduced algorithm run-time and solution complexity for the type of optimization problem given in (13.2).

Remark 14.1.6 *The procedure described in this section is merely sufficient for asymptotic stability. We cannot guarantee that the Lyapunov function and the associated state feedback laws will be found in the suggested manner. However, we have observed in an extensive case study that the approach works very well in practice. Short of the computationally very demanding construction of the infinite horizon solution proposed in [BCM03b], there is currently no alternative method for guaranteeing closed-loop stability for control of generic PWA systems. Furthermore, the method we propose here can easily be combined with most other controller computation schemes (e.g., [BBBM03, MR03, KM02, BCM03a]).*

15

Optimal Controller Computation for Piecewise Affine Systems

This chapter will address the first lever[1] for complexity reduction. Namely, the efficient computation of explicit control laws.

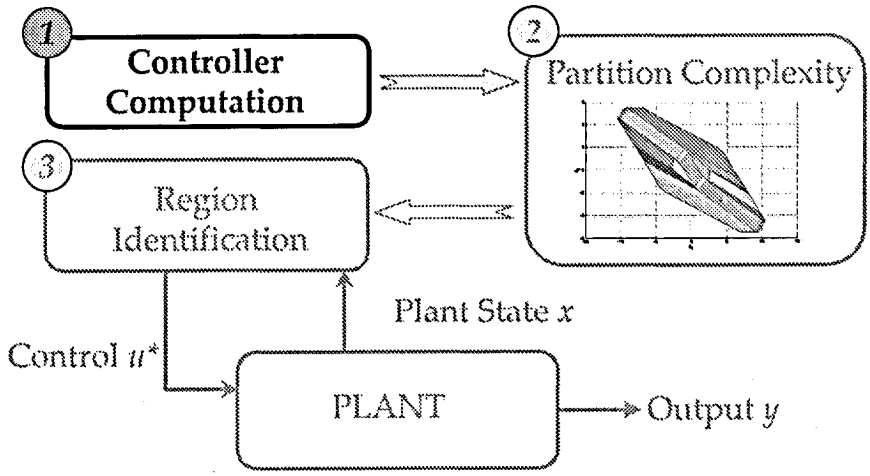


Figure 15.1: Illustration of the three levers for complexity reduction in receding horizon control. The first lever, which is the focus of this chapter, is highlighted.

In the following, a novel algorithm for the computation of explicit optimal control laws for piecewise affine (PWA) systems with linear performance indices is presented. The algorithm is based on dynamic programming (DP) and represents an extension of ideas initially proposed in [KM02, BCM03a]. Specifically, it will be shown here how to exploit the underlying geometric structure of the control problem in order to

[1]See Chapter 9 for a discussion of the three levers for complexity reduction.

significantly improve the efficiency of the off-line computations. An extensive case study is provided, which clearly indicates that the algorithm proposed in this chapter is preferable to other schemes published in the literature. The results given here are derived from [BGBM05].

15.1 Introduction

Constrained finite time optimal control (CFTOC) of PWA systems has been widely addressed in the literature, e.g. [Bor03, KM02, BCM03a, MR02]. The explicit off-line solution to the CFTOC problem can be obtained by solving a *multi-parametric mixed-integer linear program* (mp-MILP) for linear performance indices [DP00], or a *multi-parametric mixed-integer quadratic program* (mp-MIQP) for a quadratic cost [BM99a]. In [Bor03], the author proposed a computational scheme for obtaining quadratic optimal controllers for PWA systems which was based on dynamic programming (DP). For PWA systems and linear performance indices, a DP-based approach was introduced in [KM02], where the authors consider the optimal control of PWA systems affected by bounded disturbances. The scheme proposed in [KM02] uses a series of *multi-parametric linear programs* (mp-LPs) instead of one mp-MILP. An implementation of the DP-based algorithm in [KM02] for performance indices based on 1 and ∞ norms is described in detail in [BCM03a]. Practical experience with these computation schemes shows that for PWA systems of higher dimensions and a large number of affine dynamics, the off-line computation of the explicit optimal control law may become too demanding to be applicable in practice.

Therefore, in this chapter, we address the efficiency of the off-line computation of the CFTOC law. We show how to exploit problem convexity, such that fewer mp-LPs need to be solved. For this new formulation, we show how to reduce the number of constraints defining polyhedral critical regions in the explicit control law by exploiting region adjacency information. Finally, we present an extensive case study in which we compare the runtime of the algorithm in [KM02, BCM03a] with the runtime of the algorithm proposed here.

15.2 Problem Statement and Preliminary Results

In this section we will define the CFTOC problem and give preliminary results which characterize its explicit solution. Also, we will shortly describe the DP-based algorithm in [BCM03a, KM02].

15.2.1 Problem Statement and Properties of the Solution

Consider the PWA system (13.1) and the following cost function:

$$J(U_0, x(0)) := \|Q_{x_N} x(N)\|_\ell + \sum_{k=0}^{N-1} \|Q_x x(k)\|_\ell + \|Q_u u(k)\|_\ell, \tag{15.1}$$

where N is the time horizon, Q_{x_N} is a matrix defining the weight on the terminal state $x(N)$, $\|\cdot\|_\ell$ denotes the vector norm with $\ell \in \{1, \infty\}$ and $U_0 = \left[u^T(0), \ldots, u^T(N-1)\right]^T \in \mathbb{R}^{mN}$ is the vector of control inputs. The goal of CFTOC is to minimize the cost function (15.1), i.e.:

$$(J^*)^{\{N\}}(x(0)) := \min_{U_0} J^{\{N\}}(U_0, x(0)) \tag{15.2a}$$

$$\text{subj. to} \begin{cases} x(k+1) = f_{\text{PWA}}(x(k), u(k)), \\ x(N) \in \mathcal{T}_{\text{set}}, \end{cases} \tag{15.2b}$$

where $\mathcal{T}_{\text{set}}$ is a *terminal set*, i.e. the set of admissible states at the final time instance N. The following theorem characterizes the solution of the CFTOC problem (13.1)-(15.2).

Theorem 15.2.1 (CFTOC Solution Properties, [Bor03]) *The solution to the optimal control problem (13.1)-(15.2) with $\ell \in \{1, \infty\}$ is a* polyhedral piecewise affine (PPWA) *(affine in every polyhedron) state feedback control law of the form*

$$u^*(k) = F_r^{\{k\}} x(k) + G_r^{\{k\}} \quad \text{if } x(k) \in \mathcal{R}_r^{\{k\}}, \tag{15.3}$$

where $\mathcal{R}_r^{\{k\}}$, $r = 1, \ldots, R^{\{k\}}$ are polyhedra defining a polyhedral partition of the set $\mathcal{X}^{\{k\}}$ of feasible states $x(k)$ at time step $k = 0, \ldots, N-1$.

The PPWA solution to the CFTOC problem can be obtained by formulating the problem as a DP and solving a number of mp-LPs [KM02, BCM03a]. In each mp-LP, the state vector x is considered to be a vector of parameters and the control input u is the optimization variable. For further discussion, we will need the following result related to the character of the solution of an mp-LP:

Theorem 15.2.2 (mp-LP Solution Properties, [Bor03]) *Consider the mp-LP:*

$$J^*(x) = \min_z J(z,x) = c^T z, \qquad \text{subj. to } Gz \leq Sx + W, \tag{15.4}$$

where $z \in \mathbb{R}^s$ is a vector of optimization variables, $x \in \mathbb{R}^n$ is a vector of parameters, $J(z,x) : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}$ is the objective function and $G \in \mathbb{R}^{q \times s}$, $S \in \mathbb{R}^{q \times n}$, $W \in \mathbb{R}^q$. Let $\mathcal{P}^*$ be the set of parameters x for which the linear program (15.4) has a finite optimal solution. Then:

1. *$\mathcal{P}^*$ is a closed polyhedral set in $\mathbb{R}^n$,*
2. *The value function $J^*(x)$ is convex and PPWA over $\mathcal{P}^*$, i.e.*

$$J^*(x) = \alpha_i x + \beta_i, \quad \text{if} \quad x \in \mathcal{CR}_i, \tag{15.5}$$

where $\{\mathcal{CR}_i\}_{i=1}^R$, are polyhedra and $\mathcal{P}^* = \bigcup_{i=1}^R \mathcal{CR}_i$.

Polyhedra $\mathcal{CR}_i$, defining the partition of the feasible set of parameters $\mathcal{P}^*$, are known in the theory of parametric programming as *critical regions*. To make a clear distinction between the solution of a single mp-LP and the general solution to a CFTOC problem (13.1)-(15.2), i.e. between critical regions $\mathcal{CR}_i$, and regions $\mathcal{R}_i^{\{k\}}$ in Theorem 15.2.1, we will refer to $\mathcal{R}_i^{\{k\}}$ as *controller regions*. Note also that the convexity results given in Theorem 15.2.2 are valid only for the solution of a single mp-LP. The set of feasible parameters and the value function of the CFTOC problem (13.1)-(15.2) are in general non-convex. For more details about the theory of mp-LP, the reader is referred to Chapter 3.

15.2.2 DP-Based Algorithm with Affine "Cost-To-Go"

In this subsection we briefly describe the DP-based algorithm previously published in [KM02,BCM03a]. Problem (15.2) can be stated as an equivalent dynamic program:

$$(J^*)^{\{k\}}(x(k)) := \min_{u(k)} \|Q_x x(k)\|_\ell + \|Q_u u(k)\|_\ell + (J^*)^{\{k+1\}}(f_{\text{PWA}}(x(k),u(k))) \tag{15.6a}$$

$$\text{s.t.} \quad f_{\text{PWA}}(x(k),u(k)) \in \mathcal{X}^{\{k+1\}}, \tag{15.6b}$$

for $k = N-1, \ldots, 0$, with boundary conditions

$$\mathcal{X}^{\{N\}} = \mathcal{T}_{\text{set}}, \quad \text{and} \quad (J^*)^{\{N\}}(x) = \|Q_{x_N} x\|_\ell,$$

where

$$\mathcal{X}^{\{k\}} = \left\{ x \in \mathbb{R}^n \mid \exists u,\ f_{\text{PWA}}(x,u) \in \mathcal{X}^{\{k+1\}} \right\}, \tag{15.7}$$

is the set of all initial states for which the problem (15.6) is feasible. The term $(J^*)^{\{k+1\}}(x(k+1))$, representing the cost of all future control actions, is commonly referred to as "Cost-To-Go". Since the set $\mathcal{X}^{\{k+1\}}$ and $(J^*)^{\{k+1\}}(x(k+1))$ are in general non-convex for PWA systems, mp-LP solvers cannot be applied directly to solve (15.6). Instead, the non-convex problem (15.6) is split into a number of convex sub-problems by formulating one mp-LP for each controller region obtained at the previous iteration of the DP:

$$(J^*)^{\{k\}}(x(k)) := \min_{u(k)} \|Q_x x(k)\|_\ell + \|Q_u u(k)\|_\ell + \alpha_r^{\{k+1\}} x(k+1) + \beta_r^{\{k+1\}} \tag{15.8a}$$

$$\text{s.t. } x(k+1) = f_{\text{PWA}}(x(k),u(k)) \in \mathcal{R}_r^{\{k+1\}}, \tag{15.8b}$$

In (15.8) the non-convex PWA "Cost-To-Go" has been replaced by the affine term $(J^*)^{\{k+1\}} = \alpha_r^{\{k+1\}} x(k+1) + \beta_r^{\{k+1\}}$. This way, problem (15.8) needs to be solved for all controller regions $\left\{\mathcal{R}_r^{\{k+1\}}\right\}_{r=1}^{R^{\{k+1\}}}$ and all dynamics $\{\mathcal{D}_i\}_{i=1}^{D}$. The result of each of $R^{\{k+1\}} \cdot D$ mp-LPs (15.8) is a closed polyhedral partition $\mathcal{P}_i^{\{k\}}$, the union of which is the set of feasible parameters in step k, i.e. $\mathcal{X}^{\{k\}} = \bigcup_{i=1}^{R^{\{k+1\}} \cdot D} \mathcal{P}_i^{\{k\}}$. Note that the partitions $\mathcal{P}_i^{\{k\}}$ will overlap, in general. In order to obtain a suitable target partition for the next step of the DP, it is therefore necessary to compare the cost $(J^*)^{\{k\}}(x)$ wherever controller regions overlap and to remove the controller regions which are not cost optimal. Detection of overlapping critical regions within partitions $\mathcal{P}_i^{\{k\}}$ and comparison of the cost are done by solving a (possibly large) number of LPs ([GKBM03] or [Bor03], pg. 158-160). The computational complexity of removing overlaps grows exponentially with the number of regions covering any given state x. As the number of mp-LPs solved in one step of the DP grows with each iteration of the DP, the number of overlaps grows as well. Thus, a significant amount of time is spent on the removal of overlaps.

15.3 Dynamic Programming with Convex PWA "Cost-To-Go"

In this section, the main contribution of this chapter is presented. It is shown how to reformulate the DP problem presented in the previous section such that the number

of mp-LPs which need to be solved is reduced. Before stating the algorithm formally, a simple example to illustrate the key ideas is provided.

Consider a one-dimensional PWA system and a single DP iteration of the CFTOC problem for the $i-$th dynamic of the system:

$$(J^*)^{\{k\}}(x(k)) := \min_{u(k)} (J^*)^{\{k+1\}}(x(k+1)) \tag{15.9a}$$

$$\text{s.t.} \quad x(k+1) = f_{\text{PWA}}(x(k), u(k)) \in \mathcal{X}^{\{k+1\}}. \tag{15.9b}$$

For the sake of simplicity, we consider only the minimization of the "cost-to-go" $(J^*)^{\{k+1\}}(x(k+1))$. The target set $\mathcal{X}^{\{k+1\}}$ is assumed to be convex and consisting of two subsets (see Fig. 15.2):

$$\begin{aligned} \mathcal{X}_1^{\{k+1\}} &:= \left\{x \mid x \in \mathcal{X}^{\{k+1\}} \wedge x \leq x_c\right\}, \\ \mathcal{X}_2^{\{k+1\}} &:= \left\{x \mid x \in \mathcal{X}^{\{k+1\}} \wedge x \geq x_c\right\}. \end{aligned}$$

We further assume that $(J^*)^{\{k+1\}}$ is non-convex on $\mathcal{X}^{\{k+1\}}$ and affine in each of the subsets. The constraints (15.9b) define a polyhedron Π in (x, u) space (Fig. 15.2), the projection of which to the x space defines the set of parameters (system states) $\mathcal{P}$ for which the optimization problem (15.9) is feasible. We split this non-convex problem into two subproblems by considering each segment of the target set separately, as explained in Subsection 15.2.2. A cut $x(k+1) = x_c$ in $(x, u)-$space separates polyhedron Π into two polyhedra Π_1 and Π_2 (see Fig. 15.2). Projections of these polyhedra to $x-$space define sets $\mathcal{P}_1$ and $\mathcal{P}_2$, which represent sets of feasible states for each mp-LP subproblem. In general, the cut introduced by the additional constraint $x(k+1) = x_c$ separates the polyhedron Π in such a way that the sets $\mathcal{P}_1$ and $\mathcal{P}_2$ overlap. On the other hand, if $(J^*)^{\{k+1\}}$ is convex PWA in $\mathcal{X}^{\{k+1\}}$, one can formulate the problem as a *single mp-LP* and obtain the solution as a set of *non-overlapping* critical regions whose union is $\mathcal{P}$. This will be shown in the following section.

15.3.1 Dynamic Programming with Convex PWA "Cost-To-Go"

Consider the DP formulation of the CFTOC problem (15.6) and assume that a terminal set is given by:

$$\mathcal{X}^{\{N\}} = \mathcal{T}_{\text{set}} = \bigcup_{c=1}^{C^{\{N\}}} \mathcal{P}_c^{\{N\}}, \tag{15.10}$$

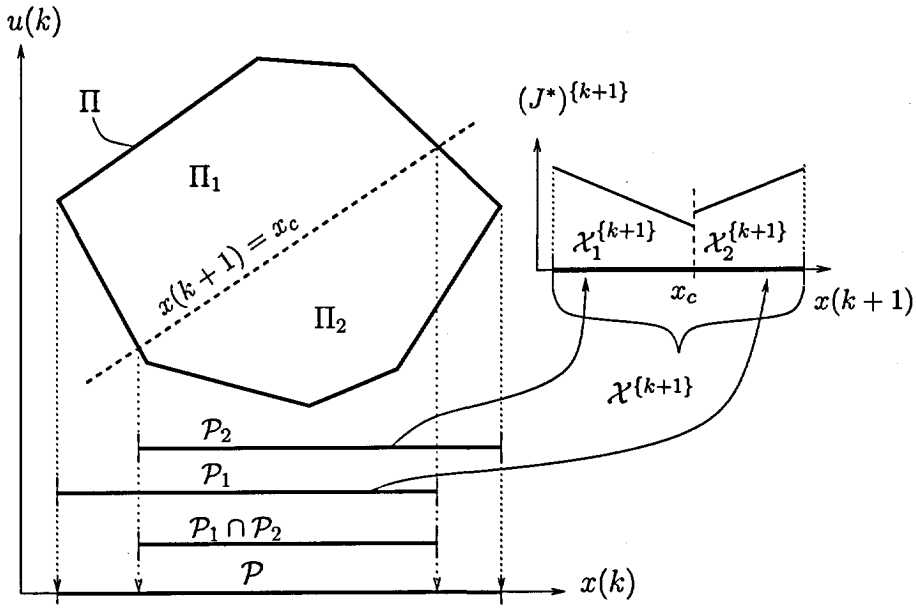


Figure 15.2: DP-based algorithm with affine "cost-to-go": for two adjacent target sets, two overlapping partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ are generated.

where $\mathcal{P}_c^{\{N\}}$ are *closed polyhedral sets*. Furthermore, assume that the value functions $(J_c^*)^{\{N\}}$ defined over each subset $\mathcal{P}_c^{\{N\}}$ are *convex PPWA*, i.e.:

$$(J_c^*)^{\{N\}}(x) = \alpha_{r,c}x + \beta_{r,c}, \quad \text{if } x \in \mathcal{R}_{r,c}^{\{N\}}, \tag{15.11}$$

where $r = 1, \ldots, R_c^{\{N\}}$ is the number of polyhedral regions in $\mathcal{P}_c^{\{N\}} = \bigcup_{r=1}^{R_c^{\{N\}}} \mathcal{R}_{r,c}^{\{N\}}$.

Being convex and PWA affine, each $(J_c^*)^{\{N\}}$ can be written as [Sch87]:

$$(J_c^*)^{\{N\}}(x) = \max_{r \in \{1,\ldots,R_c^{\{N\}}\}} \alpha_{r,c}x + \beta_{r,c},$$

and the problem:

$$\min_x (J_c^*)^{\{N\}}(x), \quad \text{s.t. } x \in \mathcal{P}_c^{\{N\}}, \tag{15.12}$$

can be equivalently formulated as:

$$\min_{r,\gamma} \gamma, \qquad \text{s.t. } \alpha_{r,c}x + \beta_{r,c} \leq \gamma, \tag{15.13}$$

where $r = 1, \ldots, R_c^{\{N\}}$. This well-known principle is also applied in Section 12.3.

By taking into account Theorem 15.2.2, the non-convex problem (15.6) can be split

into a number of $c = 1, \ldots, C^{\{k+1\}}$ mp-LP subproblems of the following form

$$(J_c^*)^{\{N\}}(x) := \min_{u,\gamma} \ \|Q_x x\|_\ell + \|Q_u u\|_\ell + \gamma \tag{15.14a}$$

$$\text{s.t. } f_{PWA}(x(k), u(k))) \in \mathcal{P}_c^{\{k+1\}}, \tag{15.14b}$$

$$\alpha_{r,c} x(k+1) + \beta_{r,c} \leq \gamma, \ \ \forall r \in \{1, \ldots, R_c^{\{k+1\}}\}, \tag{15.14c}$$

where $C^{\{k+1\}}$ denotes the number of polyhedral target partitions $\mathcal{P}_c^{\{k+1\}}$ from iteration $k+1$ and $R_c^{\{k+1\}}$ corresponds to the number of polyhedral regions in $\mathcal{P}_c^{\{k+1\}}$. Hence, it is necessary to solve $C^{\{k\}} = C^{\{k+1\}} \cdot D$ mp-LPs in order to obtain the solution to (15.14). The $C^{\{k\}}$ resulting partitions will overlap, in general, and a comparison of the mp-LP solutions needs to be performed in order to get the solution of the original CFTOC problem, i.e. it is still necessary to remove overlaps in the partitions. So far, the only difference to the scheme described in Section 15.2.2 is that we consider only large convex controller partitions $\mathcal{P}_c^{\{k\}}$ as target sets, instead of the smaller convex controller regions. In order to avoid exponential growth of the number of partitions $\mathcal{P}_c^{\{k\}}$ with each iteration (mp-LPs which need to be solved in each step of the DP), it is advisable to remove partitions which do not contain any optimal critical region, i.e. if for all $x \in \mathcal{P}_k^{\{n\}}$ there exists a partition $\mathcal{P}_k^{\{m\}}$ and a corresponding value function $(J_m^*)^{\{k\}}$ such that $x \in \mathcal{P}_m^{\{k\}}$ and $(J_n^*(x))^{\{k\}} > (J_m^*(x))^{\{k\}}$, then the partition $\mathcal{P}_n^{\{k\}}$ can be removed. Detection of suboptimal partitions can be done by using the value functions $(J_c^*)^{\{k\}}$ as a selection criterion and solving a (possibly large) number of LPs. Note that in the worst-case, it may not be possible to discard any partitions.

Comparing the new algorithm (15.14) to the approach in Section 15.2.2, it is obvious that the mp-LPs in (15.14) are more complex, since the number of constraints is (considerably) higher due to the introduction of *value function constraints* (15.14c). As a remedy, we propose the following scheme based on region adjacency information.

15.3.2 Constraint Reduction Using Adjacency List

Using algebraic manipulations, the mp-LP (15.14) can be put into the form (15.4). For further discussion we will need the notion of *active constraints.*

Definition 15.3.1 (Active Constraints) *The set of active constraints $\mathcal{A}(x)$ for a*

given state x of problem (15.4) is defined as[2]*:*

$$\mathcal{A}(x) := \left\{ i \in \mathcal{I} \,|\, \forall z \,:\, J^{\{k\}}(x,z) = (J^*)^{\{k\}}(x),\ G_{(i)}z - S_{(i)}x - W_{(i)} = 0 \right\}, \quad (15.15)$$

where $G_{(i)}$, $S_{(i)}$ and $W_{(i)}$ denote the $i-th$ row of matrices G,S and W respectively, and $\mathcal{I} = \{1, \ldots, q\}$.

Critical regions, defining the solution of the mp-LP, are constructed as follows. For a given state (parameter) x^* an LP is solved and a set of active constraints $\mathcal{A}(x^*)$ is identified. For the construction of the critical region, the set of *inactive constraints* $\mathcal{N}(x^*) = \mathcal{I} \backslash \mathcal{A}(x^*)$ is used, defining $q_{\mathcal{N}}$ half-spaces whose intersection describes the polyhedral critical region, i.e. the cardinality of $\mathcal{N}$ is $q_{\mathcal{N}}$. In order to obtain the minimal representation of the critical region, all redundant half-spaces need to be removed, so that the final representation of the critical region is defined as an intersection of $q^*_{\mathcal{N}} \leq q_{\mathcal{N}}$ half-spaces. This procedure requires the solution to $q_{\mathcal{N}}$ LPs per critical region [Bor03][3]. In our case, $q_{\mathcal{N}}$ increases with the number of value function constraints, i.e. the number of critical regions in the target partitions from the previous step of the DP, which may grow exponentially with each DP iteration. In the following we show how to reduce the number of initial half-spaces $q_{\mathcal{N}}$ by exploiting region adjacency information, and thus significantly decrease the number of LPs which need to be solved per mp-LP subproblem.

Before proceeding further, it is necessary to point out that, in the strict sense, critical regions are *open sets*. However, for any practical computation and analysis, a critical region is usually replaced by its *closure*. In the rest of the text, when speaking of a critical region, we will consider its closure.

Definition 15.3.2 (Adjacent Regions) *Polyhedral critical regions $\mathcal{CR}_i$ and $\mathcal{CR}_j$ are called adjacent if they share a common facet.*

In the following, the term *adjacent constraints* will be used for value function constraints (15.14c) which correspond to adjacent critical regions. Let $\mathcal{A}(x^*)$ be a set of active constraints for a given $x^* \in \mathcal{CR}_l^{\{k\}}$, where $\mathcal{CR}_l^{\{k\}}$ is the critical region whose polyhedral representation we want to compute. The critical region $\mathcal{CR}_l^{\{k\}}$ can be

[2]We here use the variable z instead of the input U because mp-LP problem formulations additionally introduce slack variables (see Chapter 9).

[3]Assuming the improved scheme proposed in Section 10.2 is not applied.

obtained as a projection from (x, z)-space to x-space of the polyhedron defined by:

$$G_{(i)}z - S_{(i)}x = W_{(i)}, \quad i \in \mathcal{A}(x^*), \tag{15.16a}$$

$$G_{(j)}z - S_{(j)}x \leq W_{(j)}, \quad j \in \mathcal{N}(x^*). \tag{15.16b}$$

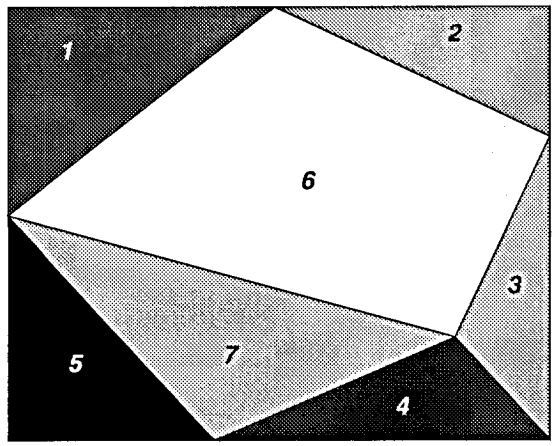
Theorem 15.3.3 (Using Adjacency in Region Computation, [BGBM05]) *The representation of the critical region $\mathcal{CR}_l^{\{k\}}$ (i.e. the projection of (15.16)) remains the same if the set $\mathcal{N}$ is reduced to those constraints which become active on the facets of the region $\mathcal{CR}_l^{\{k\}}$.*

Proof Consider only the value function constraints (15.14c), and, for the moment, assume that only one value function constraint is in the set of active constraints. When computing a critical region $\mathcal{CR}_l^{\{k\}}$, the active value function constraint is known and enforced in (15.16a). This active value function constraint directly identifies the region $\mathcal{CR}_{r^*,c}^{\{k+1\}}$ containing the state at time $k+1$. All inactive value function constraints are forced to be inactive by (15.16b). Geometrically, the value function constraints in (15.14c) represent a polyhedron in (x,u,γ)-space, whereby the active value function constraint in (15.16a) defines one of it's facets. It now follows directly from convexity of the value function $(J^*)^{\{k+1\}}$, that all value function constraints which do not originate from regions adjacent to this facet (which corresponds to region $\mathcal{CR}_{r^*,c}^{\{k+1\}}$), are redundant (e.g., see also (15.13)). Hence, it is sufficient to consider only the value function constraints in (15.16b) which originate from regions adjacent to $\mathcal{CR}_{r^*,c}^{\{k+1\}}$. □

A list of adjacent regions for every critical region can be obtained when solving an mp-LP at no additional computational cost [Bor03]. An example of the solution of an mp-LP and the constructed adjacency list is shown on Fig. 15.3. The approach described above is easily extended to cases where more than one value function constraint is active. It is then sufficient to consider only those value function constraints in $\mathcal{N}$, which are adjacent to at least one of the active value function constraints.

15.3.3 A Note on Complexity

No tight bounds on the computational complexity of solving multi-parametric programs exist. Hence, it is not possible to perform a detailed complexity comparison



region	adjacent regions
1	6
2	6
3	4, 6
4	3, 7
5	7
6	1, 2, 3, 7
7	4, 5, 6

Figure 15.3: Critical regions and the corresponding adjacency list

of the two algorithms described here. Instead we will discuss the two most crucial aspects of complexity from an intuitive point of view.

Using an affine cost-to-go as in Section 15.2.2, the CFTOC computation requires the solution to $R^{\{k+1\}} \cdot D$ mp-LPs, while the proposed algorithm with a piecewise affine cost-to-go solves $C^{\{k+1\}} \cdot D$ mp-LPs, with an additional variable γ and considerably more constraints. If the adjacency scheme in Section 15.3.2 is applied, the run times for solving the mp-LPs do not differ significantly for the two approaches. It always holds that $C^{\{k+1\}} \leq R^{\{k+1\}}$ and in practice it generally holds that $C^{\{k+1\}} \ll R^{\{k+1\}}$. Hence, fewer mp-LPs need to be solved for our algorithm, i.e. the PWA cost-to-go approach.

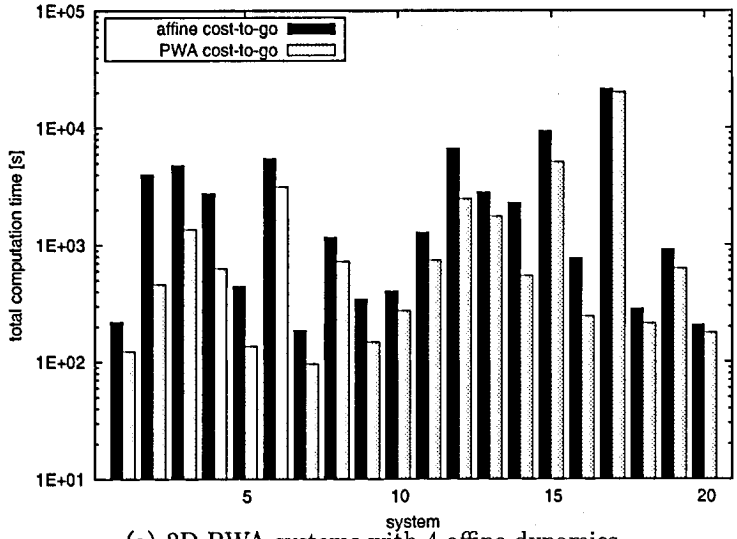
The other critical component of the CFTOC algorithms in terms of overall runtime is the removal of overlapping partitions, i.e. associating the unique optimal feedback law to each state. On one hand, the affine cost-to-go algorithm solves more mp-LPs and is hence likely to produce more controller regions and thus more overlaps. On the other hand, our algorithm solves mp-LPs for larger volume target sets, leading to larger volume partitions. Thus, it is possible that more controller regions will cover any given state. Therefore, we cannot draw theoretical conclusions on the complexity of overlap removal, although extensive simulations clearly suggest the PWA cost-to-go approach to be superior.

15.4 Numerical Results

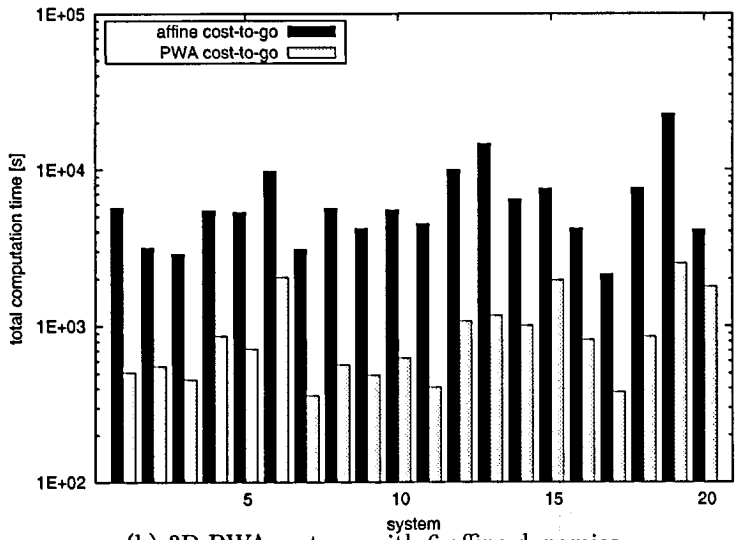
In order to demonstrate the efficiency of the proposed algorithm, we show the results of an extensive case study. An ∞-norm performance objective with $Q_x = Q_{x_N} = I$ and $Q_u = I$ was used in formulating the CFTOC problem. The total computation run times were measured for the proposed algorithm and the algorithm presented in Section 15.2.2. Both algorithms are implemented using the *Multi Parametric Toolbox* [KGB04] for MATLAB. The explicit control laws were computed for 20 randomly generated 2D PWA systems with 4 affine dynamics and for 20 randomly generated 3D PWA systems with 6 affine dynamics. The horizons $N = 7$ and $N = 3$ were used for 2D and 3D systems respectively. The results are shown on Figure 15.4 for computations run on a Pentium 4 PC, 3GHz, using MATLAB 6.5 [The03] and the NAG LP solver [Num02].

15.5 Conclusion

In this chapter, a novel algorithm was proposed to solve CFTOC problems for discrete-time PWA systems. The algorithm exploits problem structure (i.e. region adjacency information and convexity) to yield faster run times than previously published algorithms. For the analyzed 3$^{\text{rd}}$ order PWA systems, the speedup with the new algorithm is typically of one order of magnitude. We cannot claim that the proposed algorithm will outperform alternative schemes in every case, though it was true for all examples studied.



(a) 2D PWA systems with 4 affine dynamics.



(b) 3D PWA systems with 6 affine dynamics.

Figure 15.4: Total computation time for various random PWA systems. Note that the plots are in log-scale.

Seite Leer /
Blank leaf

16

Low Complexity Feedback Control of Piecewise Affine Systems

This chapter will address the second lever[1] for complexity reduction. Namely, the computation of controller partitions consisting of few regions.

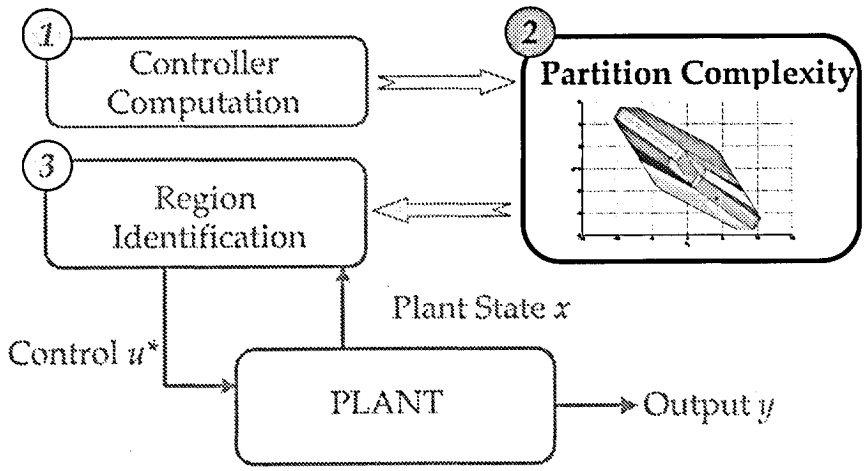


Figure 16.1: Illustration of the three levers for complexity reduction in receding horizon control. The second lever, which is the focus of this chapter, is highlighted.

It will be shown in the following how to compute low complexity controllers for piecewise affine (PWA) systems which provide stability guarantees, even if the origin is located on the boundary of multiple dynamic regions. The schemes presented in this chapter are basically identical to those presented in Chapter 11, except that they are applied to PWA systems.

[1]See Chapter 9 for a discussion of the three levers for complexity reduction.

Without loss of generality, we restrict ourselves to the regulation problem, i.e. how the state $x(k)$ can be steered to the origin without violating any of the system constraints along the closed loop trajectory. General tracking problems can easily be formulated as regulation problems by augmenting the state space appropriately (e.g. [PK03]).

One of the key problems in control of PWA systems is the lack of convexity in the controlled sets, which produces a significant computational overhead. Furthermore, the complexity of the cost-to-go function in the dynamic programming approach in [BBBM03, KM02] makes it necessary to explore an exponentially growing number of possible target sets during the iterations. The algorithms presented here avoid these issues to some extent by considering 'simpler' control objectives (e.g. minimum-time control). However, all controllers presented here guarantee constraint satisfaction for all time as well as asymptotic stability.

The rest of this chapter is structured as follows: In Section 16.1, the computation of a minimum-time feedback controller is presented which drives the system state into a pre-specified target set in minimum time. Section 16.2 will introduce a control scheme which aims at obtaining a low (but not necessarily minimal) number of switches in the system dynamics. In Section 16.3, we show how controllers of even lower complexity can be obtained by separately dealing with the issue of constraint satisfaction and asymptotic stability. As the final Section 16.4 will show, the resulting controllers are of such low complexity compared to the traditional methods [BCM03a, BBBM03, KM02] that a whole new class of problems becomes tractable.

16.1 Minimum-Time Controller

A minimum-time controller aims at driving the system state $x(k)$ into a pre-specified target set (here $\mathcal{O}_\infty^{\mathrm{LQR}}$) in minimum time. Unlike the approaches in [BBBM03], the cost-to-go for the minimum-time controller assumes only integer values. Because of the 'simple' cost-to-go, the target sets which need to be considered at each iteration step are larger and fewer in number than those which would be obtained if an optimal controller with a different cost objective were to be computed [BBBM03, BCM03b]. Thus, both the complexity of the feedback law as well as the computation time are greatly reduced, in general.

A minimum-time controller computation scheme for PWA systems was first in-

troduced in [KM02], using projection methods. Though giving general ideas about the computation concept and the character of the minimum-time solution, computational issues are not addressed in detail. A detailed algorithmic implementation of the minimum-time algorithm will be described in the following, using multi-parametric programming[2]. For a comparison of projection based controller computation with multi-parametric programming, we refer the reader to Section 11.2.

When the minimum-time algorithm terminates, the associated feedback controller will cover the N-step stabilizable set $\mathcal{K}_N^{\mathrm{PWA}}(\mathcal{O}_\infty^{\mathrm{LQR}})$.

Definition 16.1.1 (N-step stabilizable set $\mathcal{K}_N^{\mathrm{PWA}}(\mathcal{O}_\infty^{\mathrm{LQR}})$) *The set $\mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR})$ denotes the N-step stabilizable set for a PWA system* (13.1)*, i.e., it contains all states which can be steered into $\mathcal{O}_\infty^{LQR}$ in N steps. Specifically,*

$$\mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR}) = \{x(0) \in \mathbb{R}^n \mid \exists u(k) \in \mathbb{R}^m,\ s.t.\ \ x(N) \in \mathcal{O}_\infty^{LQR},\\ x(k+1) = f_{PWA}(x(k), u(k)),\ \forall k \in \{0, \dots, N-1\}\}.$$

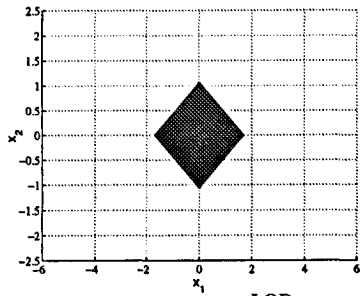
Accordingly, the set $\mathcal{K}_\infty^{PWA}(\mathcal{O}_\infty^{LQR})$ denotes the maximal stabilizable set for $N \to \infty$.

16.1.1 Minimum-Time Controller: Off-Line Computation

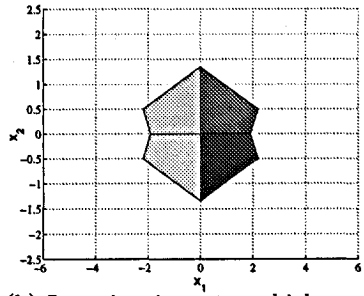
An algorithm for computing the minimum-time controller will be presented in this section. The computation scheme is based on solving a sequence of multi-parametric programs at each iteration step. The number of iterations corresponds to the number of time steps which are needed to reach the target set. At each iteration, a controller partition is computed which drives the state into the partition that was obtained in the previous iteration. The scheme is illustrated in Figure 16.2.

Before presenting the algorithm, some preliminaries will be introduced. Assume a P-collection $\mathcal{S}^0$ of L^0 polytopes $\mathcal{S}_l^0$, i.e. $\mathcal{S}^0 = \bigcup_{l \in \mathcal{L}^0} \mathcal{S}_l^0$, where $\mathcal{L}^0 \triangleq \{1, 2, \dots, L^0\}$. In the following, the set $\mathcal{S}$ without subscript will be used to denote P-collections while the subscript is used to denote polytopes. All states which can be driven into the set

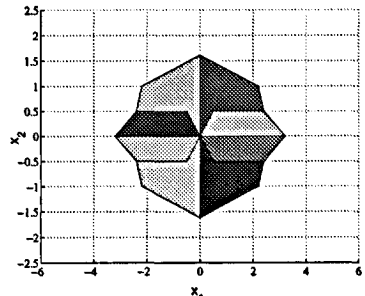
[2]Multi-parametric programming can be seen as a form of projection and thus the content of this section can be viewed as a special case of [KM02].



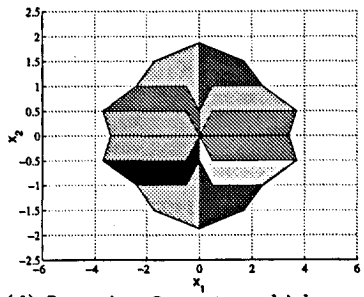
(a) Target Set $\mathcal{O}_\infty^{\mathrm{LQR}}$.



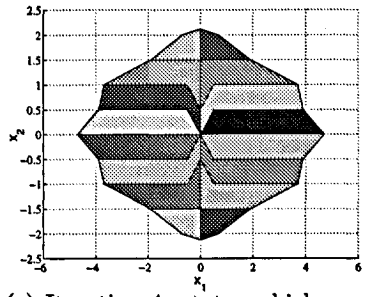
(b) Iteration 1: states which can be driven into $\mathcal{O}_\infty^{\mathrm{LQR}}$ in one step.



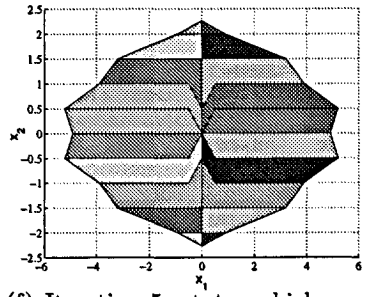
(c) Iteration 2: states which can be driven into $\mathcal{O}_\infty^{\mathrm{LQR}}$ in two steps.



(d) Iteration 3: states which can be driven into $\mathcal{O}_\infty^{\mathrm{LQR}}$ in three steps.



(e) Iteration 4: states which can be driven into $\mathcal{O}_\infty^{\mathrm{LQR}}$ in four steps.



(f) Iteration 5: states which can be driven into $\mathcal{O}_\infty^{\mathrm{LQR}}$ in five steps.

Figure 16.2: Minimum-time controller computation for Example 16.4.3. The controller partitions for the first five iterations are shown. Different colors are used to depict different control laws.

$\mathcal{S}^0$ for the PWA system (13.1) are defined by:

$$\begin{aligned}\mathrm{Pre}(\mathcal{S}^0) &= \{x \in \mathbb{R}^n \mid \exists u \in \mathbb{R}^m,\ f_{\mathrm{PWA}}(x,u) \in \mathcal{S}^0\} \\ &= \bigcup_{i\in\mathcal{I}} \bigcup_{l\in\mathcal{L}^0} \left\{x \in \mathbb{R}^n \mid \exists u \in \mathbb{R}^m,\ [x^T\ \ u^T]^T \in \mathcal{D}_i,\ A_i x + B_i u + f_i \in \mathcal{S}_l^0\right\} \\ &= \bigcup_{j\in\mathcal{J}^0} \mathcal{X}_{1,j}.\end{aligned}$$

For the feasible set $\mathcal{X}_{1,j}$, the subindex 1 denotes that the set was obtained for a prediction horizon of 1 (see Definition 13.2.1). The second subindex, j, is used to access the different feasible sets which are obtained for various combinations of active dynamics and target sets. The index set $\mathcal{J}^0$ contains all valid values for j in $\mathcal{X}_{1,j}$.

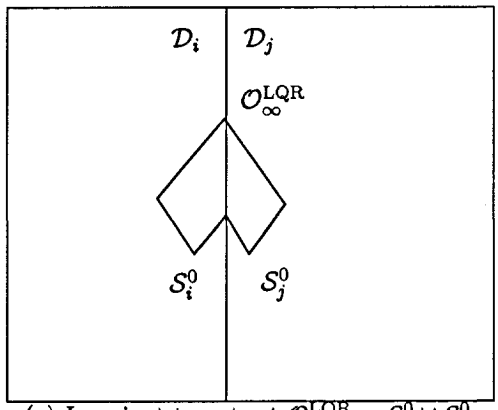
For a fixed i and l, the target set $\mathcal{S}_l^0$ is convex and the dynamics affine, such that it is possible to apply standard multi-parametric programming techniques to

compute the set of states which can be driven into $\mathcal{S}_f^0$ [BMDP02]. Therefore the set $\text{Pre}(\mathcal{S}^0)$ is a union of polytopes and can be computed by solving $J^0 = D \cdot L^0$ multi-parametric programs, where D denotes the number of dynamics and L^0 is the number of polytopes which define $\mathcal{S}^0$. Each of these multi-parametric programs will yield a controller partition $\{\mathcal{P}_{j,r}^0\}_{r=1}^R$ consisting of R controller regions whose union covers the feasible set $\mathcal{X}_{1,j} = \bigcup_{r=1,\ldots,R} \mathcal{P}_{j,r}^0$ (see Definition 13.2.1). Since the set $\text{Pre}(\mathcal{S}^0)$ is computed via multi-parametric programming, we also obtain an associated feedback law $u(x)$ which provides feasible inputs as a function of the state (see Theorem 3.2.3). Note that the various controller partitions may overlap, but that each controller will drive the state into $\mathcal{S}^0$ in one time step, i.e. $f_{\text{PWA}}(x, u(x)) \in \mathcal{S}^0$. Henceforth, we will use the notation $\mathcal{S}^{iter+1} = \text{Pre}(\mathcal{S}^{iter}) = \bigcup_{j \in \mathcal{J}^{iter+1}} \mathcal{S}_j^{iter+1}$.

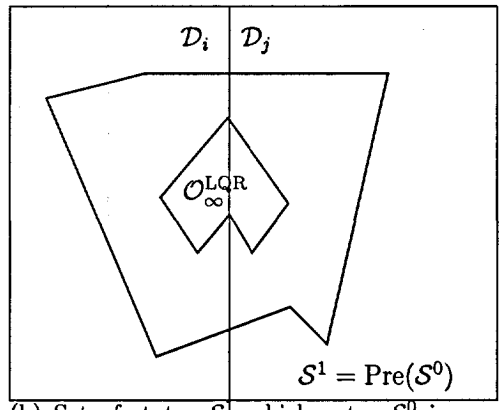
In the following, the algorithm for computing the minimum-time controller for PWA systems will be introduced.

Algorithm 16.1.2 (Minimum-Time Controller Computation)

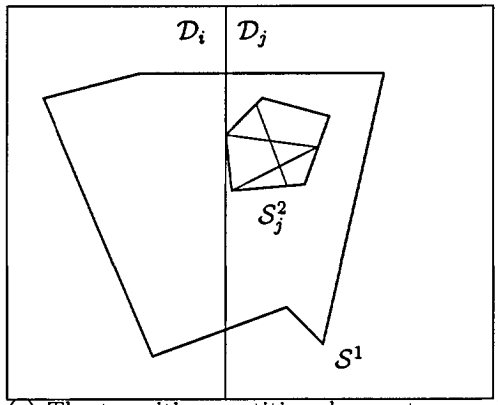
1. *Compute the invariant set $\mathcal{O}_\infty^{LQR}$ around the origin (see Figure 16.3(a)) as well as the associated Lyapunov function $V(x) = x^T P x$ and feedback laws F_i as described by Algorithm 14.1.4.*
2. *Initialize the set list $\mathcal{S}^0 = \mathcal{O}_\infty^{LQR}$ and initialize the iteration counter $iter = 0$.*
3. *Compute $\mathcal{S}^{iter+1} = \text{Pre}(\mathcal{S}^{iter}) = \bigcup_{j \in \mathcal{J}^{iter+1}} \mathcal{S}_j^{iter+1}$, by solving a sequence of multi-parametric programs (see Figure 16.3(b)). Thus, a feedback controller partition $\{\mathcal{P}_{j,r}^{iter+1}\}_{r=1}^R$ is associated with each obtained set $\mathcal{S}_j^{iter+1}$. Obviously, the number of regions R of each partition is a function of iter and j.*
4. *For all $j^* \in \mathcal{J}^{iter+1}$: If $\mathcal{S}_{j^*}^{iter+1} \subseteq \left\{\bigcup_{j \in \mathcal{J}^{iter+1} \setminus \{j^*\}} \mathcal{S}_j^{iter+1}\right\} \cup \left\{\bigcup_{i \in \{1,\ldots,iter\}} \mathcal{S}^i\right\}$, then discard $\mathcal{S}_{j^*}^{iter+1}$ from $\mathcal{S}^{iter+1}$ and set $\mathcal{J}^{iter+1} = \mathcal{J}^{iter+1} \setminus \{j^*\}$ (see Figures 16.3(c) and 16.3(d)).*
5. *If $\mathcal{S}^{iter+1} \neq \varnothing$, set $iter = iter + 1$ and goto step 3.*
6. *For all $k \in \{1, \ldots, iter - 1\}$ and $r \in \mathbb{N}^+$ discard all controller regions $\mathcal{P}_{j,r}^{k+1}$ for which $\mathcal{P}_{j,r}^{k+1} \subseteq \bigcup_{i \in \{1,\ldots,k\}} \mathcal{S}^i$ since the associated control laws are not time-optimal and will never be applied.*



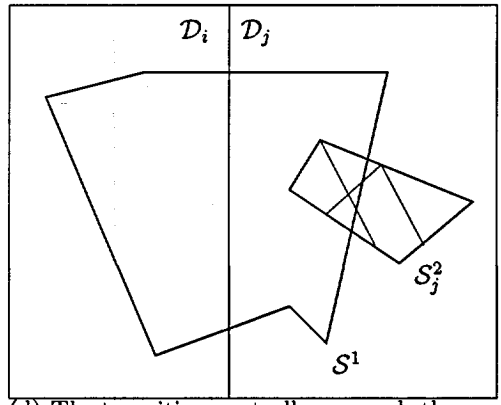
(a) Invariant target set $\mathcal{O}_\infty^{\mathrm{LQR}} = \mathcal{S}_i^0 \cup \mathcal{S}_j^0$.



(b) Set of states $\mathcal{S}^1$ which enter $\mathcal{S}^0$ in one time step. The individual controller partitions defining $\mathcal{S}^1$ are not depicted.



(c) The transition partition does not expand the controlled set of states. The individual controller partitions defining $\mathcal{S}^1$ are not depicted.



(d) The transition controller expands the controllable set of states. The individual controller partitions defining $\mathcal{S}^1$ are not depicted.

Figure 16.3: Description of Algorithm 16.1.2.

The index *iter* corresponds to the number of steps in which a state trajectory will enter the terminal set $\mathcal{O}_\infty^{\mathrm{LQR}}$ if a receding horizon control policy is applied. If the algorithm terminates in finite time, then the set $\mathcal{S}^{iter}$ is the maximum controllable set $\mathcal{K}_\infty^{\mathrm{PWA}}(\mathcal{O}_\infty^{\mathrm{LQR}})$ as given in Definition 16.1.1.

Remark 16.1.3 *Note that Algorithm 16.1.2 may not terminate in finite time (e.g. if states are unbounded). This is a problem inherent property and not a result of the computation scheme. It is therefore advisable to specify a maximum step distance N which can be used as a termination criterion in step 5 of Algorithm 16.1.2. The*

resulting controller computation will then terminate in finite time and the feedback controller will cover $\mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR})$.

Remark 16.1.4 *The implementation of Algorithm 16.1.2 requires a function that can detect if a convex polyhedron* $\mathcal{P}_0$ *is covered by a finite set of non-empty convex polyhedra* $\{\mathcal{P}_r\}_{r=1}^R$, *i.e. if* $\mathcal{P}_0 \subseteq \bigcup_{r\in\{1,\dots,R\}} \mathcal{P}_r$. *For instance, this operation is needed to check if two unions of polyhedra cover the same non-convex set [RGK+04a] (e.g., Step 5 of Algorithm 16.1.2). We refer the reader to [BT03], where an efficient algorithm is given to perform this task.*

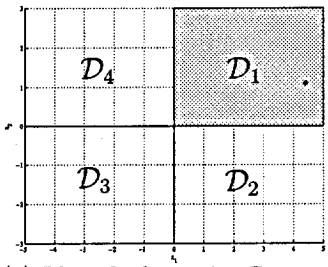
16.1.2 Minimum-Time Controller: On-Line Application

In the minimum-time algorithm presented in this paper, we can take advantage of some of the algorithm features to speed up the on-line region identification procedure. We propose a three-tiered search tree structure which serves to significantly speed up the region identification. Unlike the search tree proposed in [TJB03b], the tree structure proposed here is computed automatically by Algorithm 16.1.2, i.e. no post-processing is necessary. The following algorithm illustrates how the controller obtained with Algorithm 16.1.2 can be applied, such that the resulting closed-loop trajectories are minimum-time optimal.

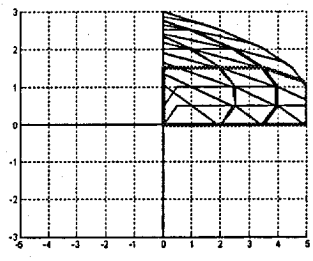
Algorithm 16.1.5 (On-Line Application of Minimum-Time Controller)

1. *Identify the active dynamics* i, *such that* $x \in \mathcal{D}_i$, $i \in \mathcal{I}$ *(see Figure 16.4(a))*[3].
2. *Identify controller set* $\mathcal{S}_j^{iter}$ *associated with dynamic* i *which is 'closest' to the target set* $\mathcal{S}^0$, *i.e.* $\min_{iter,j}$ $iter$, *s.t.* $x \in \mathcal{S}_j^{iter}$, $j \in \mathcal{J}^{iter}$ *(see Figure 16.4(b)).*
3. *Extract the controller partition* $\{\mathcal{P}_{j,r}^{iter}\}_{r=1}^R$ *with the corresponding feedback laws* F_r, G_r *and identify the region* r *which contains the state* $x \in \mathcal{P}_{j,r}^{iter}$ *(see Figure 16.4(c)).*
4. *Apply the control input* $u = F_r x + G_r$. *Goto 1.*

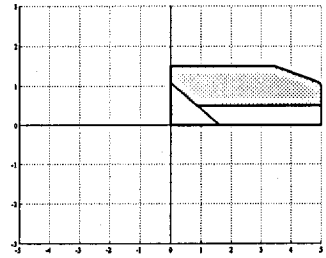
[3] Note that once the control law has been computed, a unique dynamic i can be associated with each state, even if the original PWA system was defined in x-u-space.



(a) Identify dynamics $\mathcal{D}_i$ containing the state[3].



(b) Identify feasible controller set $\mathcal{X}_j^{iter}$ containing the state that has the smallest value for *iter*.



(c) Extract controller partition $\{\mathcal{P}_{j,r}^{iter}\}_{r=1}^{R}$ associated with feasible set $\mathcal{S}_j^{iter}$ and identify region $\mathcal{P}_{j,r}^{iter}$ containing the state.

Figure 16.4: Illustration of Algorithm 16.1.5.

Note that the association of controller partitions $\mathcal{S}_j^{iter}$ to active dynamics in step 2 is trivially implemented by building an appropriate lookup-table during the off-line computation in Algorithm 16.1.2.

A minimum-time control scheme for PWA systems based on projection was proposed in [KM02]. The on-line application of that scheme is similar to Algorithm 16.1.5, except that in Step 3, the authors need to find an interior point to a polytope in x-U_N-space. Depending on the number of controller regions R, the interior point approach may or may not be faster than the algorithm proposed here.

Theorem 16.1.6 (Properties of Minimum-Time Control, [GKBM04a] [GKBM04b]) *The controller obtained with Algorithm 16.1.2 and applied to a PWA system* (13.1) *in a receding horizon control fashion according to Algorithm 16.1.5, guarantees asymptotic stability and feasibility of the closed loop system, provided* $x(0) \in \mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR})$.

Proof Assume the initial state $x(0)$ is contained in the set $\mathcal{S}^{iter}$ with a step distance to $\mathcal{O}_\infty^{\mathrm{LQR}}$ of *iter*. The control law at step 4 of Algorithm 16.1.5 will drive the state into a set $\mathcal{S}^{iter-1}$ in one time step (see step 3 of Algorithm 16.1.2). Therefore, the state will enter $\mathcal{O}_\infty^{\mathrm{LQR}}$ in *iter* steps. Once the state enters $\mathcal{O}_\infty^{\mathrm{LQR}}$ the feedback controllers associated with the common quadratic Lyapunov ensure stability. □

16.2 Reduced Switching Controller

In general, it is possible to obtain even simpler controllers and faster computation times by modifying Algorithm 16.1.2. Instead of computing a minimum-time controller, an alternative scheme which aims at reducing the number of switches can be applied. A change in the active system dynamic $\mathcal{D}_i \rightarrow \mathcal{D}_j$, $(i \neq j)$ is referred to as a switch. The proposed procedure does not guarantee the minimum number of switches, though straightforward modifications to the algorithm would yield such a solution. The "minimum number of switches" solution is not pursued here since computation time is the primary objective.

The proposed reduced switch controller will avoid switching the active dynamics for as long as possible. We will introduce the following operator for $i \in \mathcal{I}$:

$$\text{Pre}_i(\mathcal{S}_j^{iter}) = \{x \in \mathbb{R}^n \mid \exists u \in \mathbb{R}^m,\ [x^T\ \ u^T]^T \in \mathcal{D}_i,\ A_i x + B_i u + f_i \in \mathcal{S}_j^{iter}\}.$$

Once the $j-th$ controller set $\mathcal{S}_j^{iter}$ obtained at iteration *iter* is computed (see Algorithm 16.1.2, step 3) for dynamics i, the set is subsequently used as a target set for as long as the controllable set of states can be enlarged without switching the active dynamics i. With this scheme, the total number of convex sets needed to describe the controlled set $\mathcal{S}^{iter}$ remains constant while the size of $\mathcal{S}^{iter}$ increases. Therefore, this scheme generally results in significantly fewer sets during the dynamic programming iterations compared to Algorithm 16.1.2. Specifically, the algorithm works as follows:

Algorithm 16.2.1 (Computation of Controller with Reduced Number of Switches)

1. *Compute the invariant set $\mathcal{O}_\infty^{LQR}$ around the origin (see Figure 16.5(a)) as well as the associated Lyapunov function $V(x) = x^T P x$ and linear feedback laws F_r as described by Algorithm 14.1.4.*

2. *Initialize the set list $\mathcal{S}^0 = \mathcal{O}_\infty^{LQR} = \bigcup_{j \in \mathcal{J}^0} \mathcal{S}_j^0$ and initialize the iteration counter $iter = 0$.*

3. *Initialize $\mathcal{S}^{iter+1} = \emptyset$ and execute the following for all dynamics $i \in \mathcal{I}$ and set-indices $j \in \mathcal{J}^{iter}$:*

   a) *Initialize counter $c = 0$ and set $C^0 = \mathcal{S}_j^{iter}$.*

b) Compute $\mathcal{C}^{c+1} = \mathrm{Pre}_i(\mathcal{C}^c)$ *(see Figure 16.5(b)) by using multi-parametric programming and store the associated controller partition. Thus, a feedback controller partition* $\{\mathcal{P}_{j,r}^{c+1,iter}\}_{r=1}^{R}$ *is obtained.*

c) If $\mathcal{C}^c \subset \mathcal{C}^{c+1}$ *(see Figure 16.5(c)), set* $c = c+1$ *and goto step 3b.*

d) Set $\mathcal{S}^{iter+1} = \mathcal{S}^{iter+1} \cup \mathcal{C}^c$ *(see Figure 16.5(d)).*

4. If $\mathcal{S}^{iter+1} \neq \mathcal{S}^{iter}$, *set* $iter = iter + 1$ *and goto 3.*

5. For all $k \in \{1, \ldots, iter-1\}$, $c \in \mathbb{N}$ *and* $r \in \mathbb{N}^+$ *discard all controller regions* $\mathcal{P}_{j,r}^{c,k+1}$ *for which* $\mathcal{P}_{j,r}^{c,k+1} \subset \bigcup_{i\in\{1,\ldots,k\}} \mathcal{S}^i$ *since the associated control law has a non-minimum number of switches and will never be applied.*

The on-line computation is identical to the scheme described in Section 16.1.2 and the same finite time termination conditions as in Remark 16.1.3 apply.

Remark 16.2.2 *In Algorithm 16.2.1 the counter 'iter' associated with the control sets* $\mathcal{S}^{iter}$ *corresponds to the number of dynamic switches which will occur before the target set* $\mathcal{O}_\infty^{LQR}$ *is reached.*

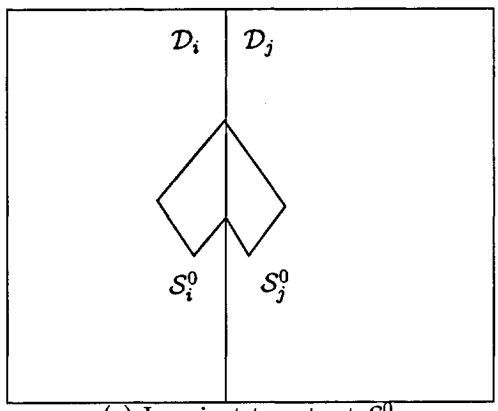
Remark 16.2.3 *If we always have* $\mathcal{C}^c \not\subset \mathcal{C}^{c+1}$ *in step 3c of Algorithm 16.2.1, then Algorithm 16.2.1 is identical to Algorithm 16.1.2. However if* $\mathcal{C}^c \subset \mathcal{C}^{c+1}$, *it is possible to perform a large part of the computations on convex sets, which makes Algorithm 16.2.1 more efficient than Algorithm 16.1.2, in general.*

Theorem 16.2.4 (Properties of Minimum-Switch Control, [GKBM04a] [GKBM04b]) *A controller computed according to Algorithm 16.2.1 and applied to a PWA system (13.1) according to Algorithm 16.1.5, guarantees stability and feasibility of the closed loop system, provided* $x(0) \in \mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR})$.

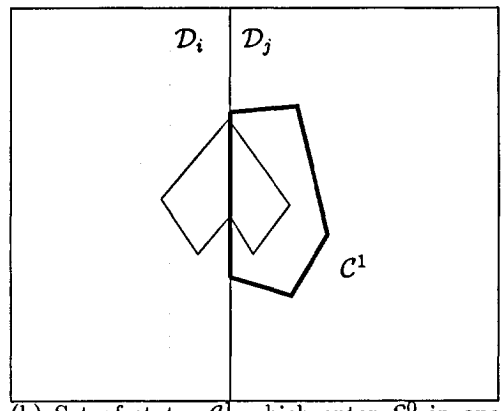
Proof Follows from Theorem 16.1.6. □

16.3 N-step Controller

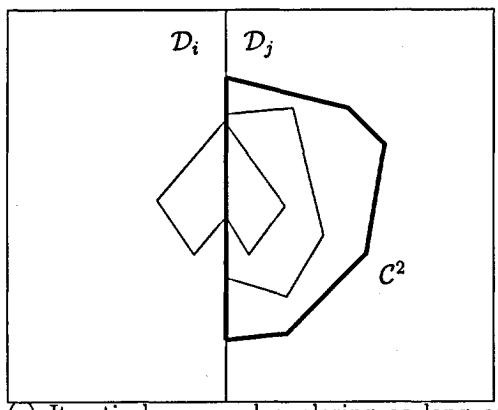
In the previous sections, stability was guaranteed by imposing an appropriate terminal set constraint. In order to cover large parts of the state space, this type of constraint



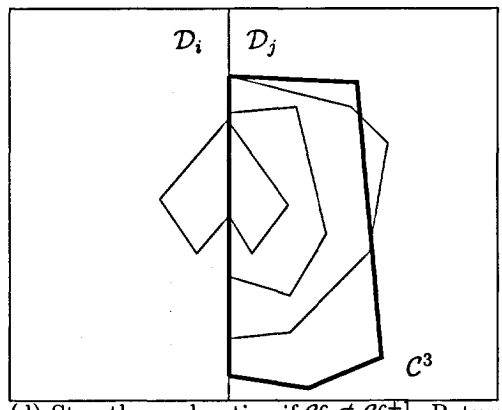
(a) Invariant target set $\mathcal{S}_j^0$.



(b) Set of states $\mathcal{C}^1$ which enter $\mathcal{S}_j^0$ in one time step.



(c) Iteratively proceed exploring as long as $\mathcal{C}^c \subset \mathcal{C}^{c+1}$.



(d) Stop the exploration if $\mathcal{C}^c \not\subset \mathcal{C}^{c+1}$. Return $\mathcal{C}^c$ in such case.

Figure 16.5: Description of Algorithm 16.2.1.

generally entails the use of large prediction horizons which results in controllers with a large number of regions.

In this section, instead of enforcing asymptotic stability with an appropriate terminal set constraint (and the associated cost), we propose to enforce constraint satisfaction only. This can be easily achieved by imposing a set-constraint on the first predicted state in the MPC formulation. Hence, the terminal-set constraint (13.2b) becomes superfluous and we do not need to rely on large prediction horizons. Asymptotic stability is analyzed in a second step. This scheme is inspired by promising complexity reduction results for LTI systems in [GPM03, GM03].

16.3.1 Constraint Satisfaction

If the constrained finite time optimal control problem ((13.2)) is solved via multi-parametric programming for any prediction horizon $N' < N$ with $x_{N'} \in \mathcal{T}_{\text{set}} = \mathbb{R}^n$ in (13.2b) and the additional constraint $x_1 \in \mathcal{K}_N^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}})$, the resulting MPC controller guarantees that the state remains within $\mathcal{K}_N^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}})$ for all time. The set constraint on the first step guarantees that the resulting controller partition will be positive invariant, which directly implies feasibility for all time [Bla99, Ker00]. The set $\mathcal{O}_\infty^{\text{LQR}}$ can be computed as described by Algorithm 14.1.4 and $\mathcal{K}_N^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}})$ can be obtained by applying Algorithm 16.1.2.

Note that this allows us to control large volume sets $\mathcal{K}_N^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}})$ with short prediction horizons N', i.e. $N' \ll N$. Although we have set $N \rightarrow \infty$ for the examples provided in Section 16.4, the set $\mathcal{K}_\infty^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}})$ was always finitely determined. This is not always the case such that in practice it is advisable to limit N to be a large but finite value.

Since the target set $\mathcal{K}_\infty^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}}) = \bigcup_{c\in\{1,\ldots,C\}} \mathcal{K}_\infty^c$ is non-convex in general (i.e. a union of C polytopes $\mathcal{K}_\infty^c$) a controller partition can be obtained by solving a sequence of $C \cdot D$ multi-parametric programs, e.g. (9.4) or (9.6), where D corresponds to the total number of different dynamics. Specifically, the N-step controller can be obtained by solving $C \cdot D$ multi-parametric programs (e.g., (9.4) or (9.6)) for an arbitrary N' with $\mathcal{T}_{\text{set}} = \mathcal{K}_\infty^c$ in (13.2b) (C different sets) and for D different dynamics in (13.1). The smaller N' the lower the controller complexity. However, N' has no impact on the size of the controlled set.

16.3.2 Stability Analysis

The controller partition obtained in the previous subsection will generally contain overlaps such that the closed-loop dynamics associated with a given state $x(0)$ may not be unique. It is therefore not possible to perform a non-conservative stability analysis of the closed-loop system. However, by using the PWA value function $J_N^*(x)$ in (13.2a) as a selection criterion it is possible to obtain a non-overlapping partition ([GKBM03] or [Bor03], pg. 158-160) by solving a number of LPs, i.e. only the cost optimal controller is stored.

The resulting controller partition is invariant and a unique controller region r ($x \in \mathcal{P}_r$) and unique dynamics l ($x \in \mathcal{D}_l$) is associated with each state x, i.e. the

closed loop system corresponds to an autonomous PWA system

$$x_{k+1} = (A_l + B_l F_r)x_k + B_l G_r + f_l, \qquad x_k \in \mathcal{P}_r \cap \mathcal{D}_l \tag{16.1a}$$

$$= \tilde{A}_r x_k + \tilde{f}_r, \qquad x_k \in \mathcal{P}_r. \tag{16.1b}$$

Since every controller region $\mathcal{P}_r$ is only contained in one unique dynamic $\mathcal{D}_l$, the update matrix $\tilde{A}_r$ and vector $\tilde{f}_r$ are uniquely defined. The search for a Lyapunov function which guarantees asymptotic stability of the closed-loop PWA system can now be performed as described in Chapter 8.

16.3.3 N-step Controller Computation

The N-step control scheme utilizes tools from invariant set computation and stability analysis in order to compute controllers with small prediction horizons which guarantee constraint satisfaction as well as asymptotic stability. The basic procedure consists of two main stages. In the first stage, a N-step optimal controller is computed which guarantees constraint satisfaction for all time. Since constraint satisfaction does not imply asymptotic stability, it is necessary to analyze the stability properties of the closed-loop system in a second stage. Specifically, the algorithm works as follows.

Algorithm 16.3.1 (N-step Controller Computation)

1. *Compute the invariant set $\mathcal{O}_\infty^{LQR}$ around the origin and an associated Lyapunov function as described by Algorithm 14.1.4.*
2. *Compute the set $\mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR}) = \bigcup_{c\in\{1,\dots,C\}} \mathcal{K}_N^c$ $(N \to \infty)$ by applying Algorithm 16.1.2.*
3. *Solve a sequence of $C \cdot D$ mp-LPs (9.4) for prediction horizon N' with $\mathcal{T}_{set} = \mathcal{K}_N^c$, $\forall c \in \{1,\dots,C\}$ in (13.2b), affine dynamics $i \in \mathcal{I} = \{1,\dots,D\}$ in (13.1) and $N' \leq N$.*
4. *Remove the region overlaps by using the PWA value function $J_N^*(x)$ as a selection criterion (see [GKBM03] or [Bor03] for details).*
5. *Attempt to find a PWA or PWQ Lyapunov function as described in Chapter 8.*

There is no guarantee that step 2 of Algorithm 16.3.1 will terminate in finite time or that a Lyapunov function can be found in step 5. The finite time termination conditions are discussed in Remark 16.1.3. If no Lyapunov function is found, the resulting controller is guaranteed to satisfy the system constraints for all time, but no proof of asymptotic stability can be given. However, it always holds that the state of the closed-loop system cannot become arbitrarily large, since it is guaranteed to remain within a bounded invariant set.

Theorem 16.3.2 (Properties of N-step Control, [GKBM04a, GKBM04b]) *If the stability analysis in Step 5 of Algorithm 16.3.1 is successful and the feedback law obtained in Step 4 is applied to system* (13.1) *in a RHC fashion, then the closed-loop system is exponentially stable on $\mathcal{K}_N^{PWA}(\mathcal{O}_\infty^{LQR})$ and the system constraints are satisfied for all time.*

Proof The partition computed in Step 4 is invariant by construction, hence constraint satisfaction is guaranteed. Exponential stability follows trivially from the successful stability analysis in Step 5. □

Remark 16.3.3 *If the stability analysis in Step 5 of Algorithm 16.3.1 fails, it is advisable to recompute the controller in Step 3 using different weights Q_u, Q_x, Q_{x_N} and/or a different prediction horizon N' in* (13.2). *Slight modifications in these parameters may make the subsequent stability analysis in Step 5 feasible.*

16.4 Numerical Results

As was shown in [GM03,GPM03] and will also be illustrated in this section, algorithms of type 16.1.2–16.3.1 generally yield controllers of significantly lower complexity than those which are obtained if a linear norm-objective is minimized as in (13.2) [BCM03b, BCM03a, KM02].

Example 16.4.1 *Consider the 2-dimensional problem adopted from [MR03],*

$$x(k+1) = \begin{cases} \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{if } x_{(1)}(k) \leq 1 \\ \begin{bmatrix} 0.5 & 0.2 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} & \text{if } x_{(1)}(k) \geq 1 \end{cases}, \quad (16.2)$$

subject to constraints $-x_{(1)}(k)+x_{(2)}(k)\leq 15$, $-3x_{(1)}(k)-x_{(2)}(k)\leq 25$, $0.2x_{(1)}(k)+x_{(2)}(k)\leq 9$, $x_{(1)}(k)\geq -6$, $x_{(1)}(k)\leq 8$, *and* $-1\leq u(k)\leq 1$. *Weight matrices in the cost function were chosen as* $Q_x=I$ *and* $Q_u=0.1$ *in (13.2).*

Example 16.4.2 *Consider the 3-dimensional PWA system introduced in [MR03],*

$$x(k+1)=\begin{cases}\begin{bmatrix}1 & 0.5 & 0.3\\ 0 & 1 & 1\\ 0 & 0 & 1\end{bmatrix}x(k)+\begin{bmatrix}0\\0\\1\end{bmatrix}u(k)+\begin{bmatrix}0\\0\\0\end{bmatrix} & \text{if } x_{(2)}(k)\leq 1\\ \begin{bmatrix}1 & 0.2 & 0.3\\ 0 & 0.5 & 1\\ 0 & 0 & 1\end{bmatrix}x(k)+\begin{bmatrix}0\\0\\1\end{bmatrix}u(k)+\begin{bmatrix}0.3\\0.5\\0\end{bmatrix} & \text{if } x_{(2)}(k)\geq 1\end{cases}, \tag{16.3}$$

subject to constraints $-10\leq x_{(1)}(k)\leq 10$, $-5\leq x_{(2)}(k)\leq 5$, $-10\leq x_{(3)}(k)\leq 10$, *and* $-1\leq u(k)\leq 1$. *Again, weights in the cost function are* $Q_x=I$, $Q_u=0.1$.

Example 16.4.3 *Consider the 4-dimensional PWA system introduced in [MR03],*

$$x(k+1)=\begin{cases}\begin{bmatrix}1 & 0.5 & 0.3 & 0.5\\ 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}x(k)+\begin{bmatrix}0\\0\\0\\1\end{bmatrix}u(k)+\begin{bmatrix}0\\0\\0\\0\end{bmatrix} & \text{if } x_{(2)}(k)\leq 1\\ \begin{bmatrix}1 & 0.2 & 0.3 & 0.5\\ 0 & 0.5 & 1 & 1\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}x(k)+\begin{bmatrix}0\\0\\0\\1\end{bmatrix}u(k)+\begin{bmatrix}0.3\\0.5\\0\\0\end{bmatrix} & \text{if } x_{(2)}(k)\geq 1\end{cases}, \tag{16.4}$$

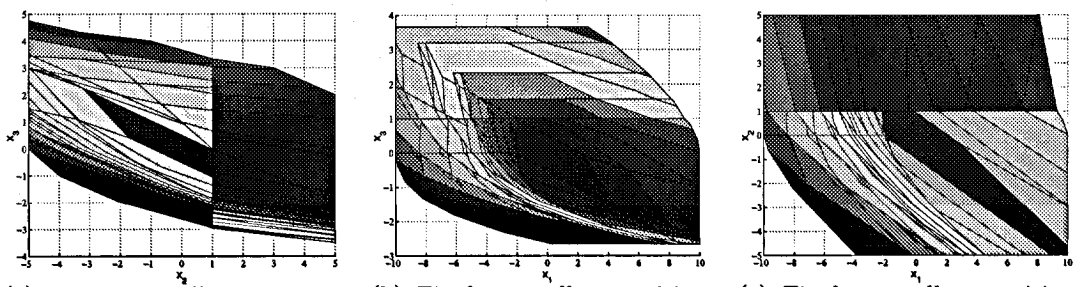
subject to constraints $-10\leq x_{(1)}(k)\leq 10$, $-5\leq x_{(2)}(k)\leq 5$, $-10\leq x_{(3)}(k)\leq 10$, $-10\leq x_{(4)}(k)\leq 10$, *and* $-1\leq u(k)\leq 1$. *Weighting matrices in the cost function are* $Q_x=I$, $Q_u=0.1$.

Once the set $\mathcal{O}_\infty^{\mathrm{LQR}}$ is computed, Algorithms 16.1.2–16.3.1 are applied to Examples 16.4.1 – 16.4.3. A runtime comparison of the computation procedures as well as complexity of the resulting solutions are reported in Table 16.1.
Controller regions for Example 16.4.2 are depicted in Figures 16.6(a)–16.6(c).

In order to compare low complexity control strategies discussed in this paper with the cost optimal approach of [BCM03b], we generated 10 random PWA systems

	Alg. 16.1.2		Alg. 16.2.1		Alg. 16.3.1		Alg. [BCM03b]	
	t	$\#R$	t	$\#R$	t	$\#R$	t	$\#R$
Ex. 16.4.1	61 sec.	279	40 sec.	186	53 sec.	61	5.5 h	1413
Ex. 16.4.2	1153 sec.	1519	755 sec.	1044	286 sec.	522	$\star$	$\star$
Ex. 16.4.3	92 h	7894	2.2 h	2434	†	†	$\star$	$\star$

Table 16.1: Off-line CPU-time t and number of controller regions $\#R$ for different algorithms. The CPU-time for Algorithm 16.3.1 includes the stability analysis. The $\star$ denotes that the computations were not completed after 7 days. The † symbol denotes that the stability analysis procedure failed. The computation was run on a 2.8GHz Pentium IV CPU running the Windows version of MATLAB 6.5 along with the NAG foundation LP solver.



(a) Final controller partition (cut on $x_{(1)} = 0$).

(b) Final controller partition (cut on $x_{(2)} = 0$).

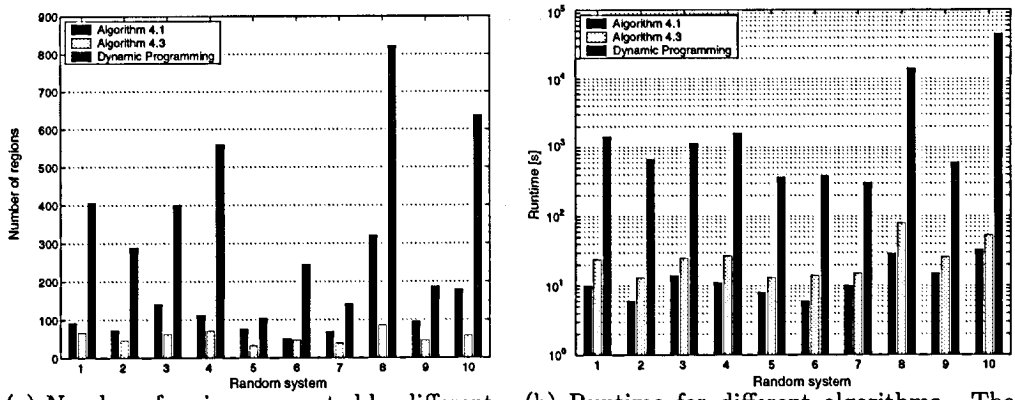
(c) Final controller partition (cut on $x_{(3)} = 0$).

Figure 16.6: The controller partition obtained by applying Algorithm 16.1.2 on Example 16.4.2. The actual partition is three dimensional (see (16.3)), but only the axis intersections are shown.

with 2 states, 1 input and 4 piecewise-affine dynamics. All elements in the state space matrices were assigned random values between $[-2, 2]$ (i.e., stable and unstable systems were considered). Each of the random PWA systems consists of 4 different affine dynamics which are defined over non-overlapping random sets whose union covers the square $\mathbb{X} = [-5, 5] \times [-5, 5]$. The origin was chosen to be on the boundary of multiple dynamics. All simulation runs as well as the random system generation were performed with the MPT toolbox [KGB04][4].

[4]For random PWA systems `mpt_randPWAsys` of the MPT toolbox [KGB04] was called.

Algorithms 16.1.2, 16.2.1 and 16.3.1, as well as the cost-optimal strategy of [BCM03b] were applied to these systems. Complexity of the resulting solution and run time of each algorithm are depicted graphically in Figures 16.7(a) and 16.7(b).



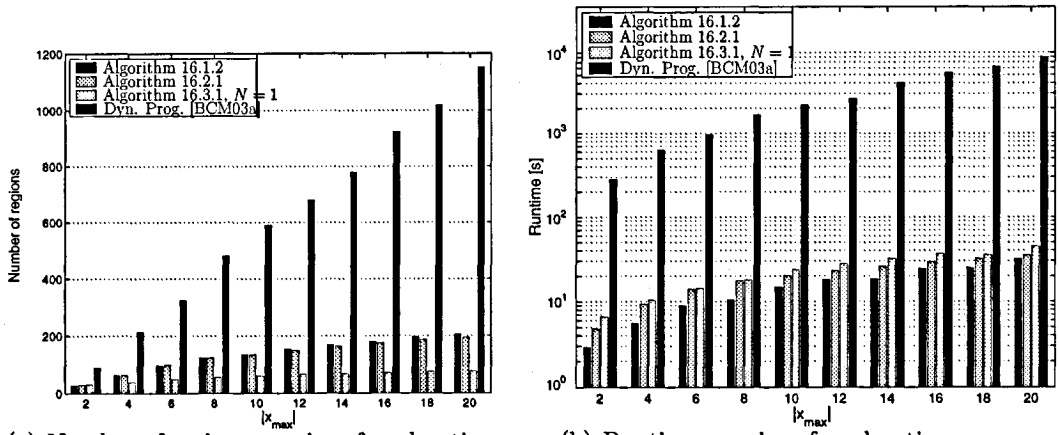
(a) Number of regions generated by different algorithms.

(b) Runtime for different algorithms. The runtime for Algorithm 16.3.1 includes the stability analysis.

Figure 16.7: Complexity and runtime for 10 random PWA systems.

To further investigate the behavior of different control strategies, another test on a set of 10 random PWA systems was performed to show how the complexity of Algorithms 16.1.2, 16.2.1 and 16.3.1 scales with increasing volume of the exploration space. A comparison with the approach in [BCM03b] is depicted in Figures 16.8(a) and 16.8(b). For the random systems considered here, the necessary runtime is reduced by two orders of magnitude and the solution complexity is reduced by one order of magnitude, on average. In addition, these differences become larger with increasing size of the state constraints. Although we have not come across any examples where the proposed schemes are inferior to the approaches in [BBBM03, KM02], we are not able to proof that no such cases exist.

However, none of the algorithms presented in this paper guarantee optimal closed-loop performance in the sense of the cost-objective (13.2). In order to assess the degradation in performance, equidistantly spaced data points inside the set $\mathcal{K}_\infty^{\text{PWA}}(\mathcal{O}_\infty^{\text{LQR}})$ were generated as feasible initial states. Subsequently, the closed-loop trajectory cost for these initial states was computed according to the performance index (13.2a). The average decrease in performance with respect to the cost-optimal solution of [BCM03b] is summarized in Figures 16.9(a) and 16.9(b). It can be seen that



(a) Number of regions vs. size of exploration space.

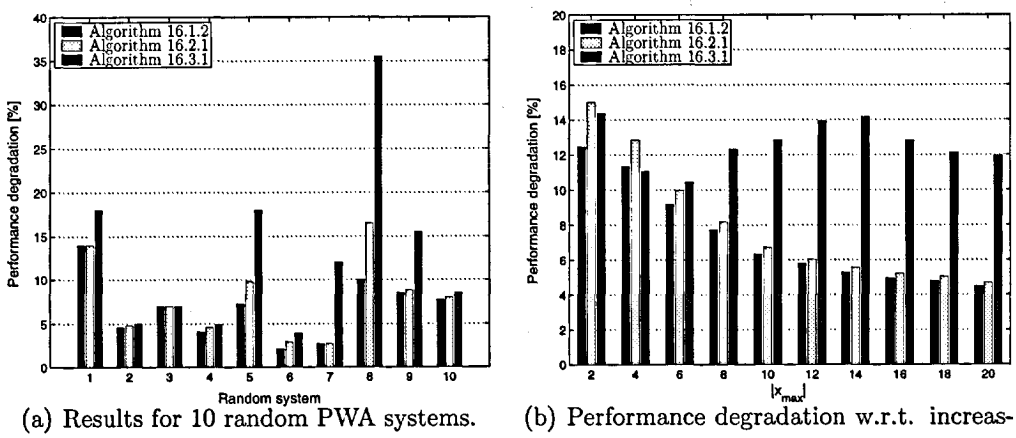
(b) Runtime vs. size of exploration space.

Figure 16.8: Complexity and runtime versus size of exploration space (average over 10 random PWA systems).

closed-loop performance gets better with increasing size of the exploration space. The intuitive explanation of this observation is as follows: if the state is far away from the origin, going at "full throttle" will be the optimal strategy, since the contribution of the state penalty term in (13.2a) is much higher than the term which penalizes the control action. Therefore almost the same performance is achieved with low complexity strategies as with cost-optimal algorithms for a majority of the controllable state-space, resulting in good average performance.

16.5 Conclusion

In this chapter, three novel algorithms to compute low complexity feedback controllers for constrained PWA systems are presented. All controllers guarantee constraint satisfaction for all time as well as asymptotic stability. The proposed computation scheme iteratively solves a series of multi-parametric programs such that a feedback controller is obtained which drives the state into a target set in minimum time. An alternative controller which aims at reducing the number of switches between different dynamics is also presented and the provided examples suggest that this approach may further reduce complexity. Furthermore, a search tree for efficient on-line identification of the optimal feedback law is automatically constructed by



(a) Results for 10 random PWA systems.

(b) Performance degradation w.r.t. increasing size of exploration space (average over 10 random systems).

Figure 16.9: Performance degradation with respect to cost-optimal solution of [BCM03b]. The performance of Algorithm 16.3.1 can be improved by increasing N.

both algorithms. A third computation scheme (referred to as N-step control) is also presented, which separately deals with the requirement of constraint satisfaction and asymptotic stability. In the N-step scheme, stability is not enforced but merely verified a posteriori. While the resulting controller is of even lower complexity than the previous two (for $N = 1$), there is no a priori guarantee that the closed-loop system will be asymptotically stable. In addition, the closed-loop performance may not be satisfactory (see Figure 16.9).

An extensive case study is provided which clearly indicates that all three algorithms reduce complexity versus optimal controllers [BBBM03, KM02] by several orders of magnitude, in general. The proposed procedures make problems tractable that were previously too complex to be tackled by standard methods.

Seite Leer /
Blank leaf

Part V

THE MULTI PARAMETRIC TOOLBOX

Seite Leer /
Blank leaf

17

Overview of The Toolbox

Optimal control of constrained linear and piecewise affine (PWA) systems has garnered great interest in the research community due to the relative ease with which complex problems can be stated and solved. The aim of the *Multi-Parametric Toolbox* (MPT)[1] is to provide an efficient computational means to obtain the explicit solutions to these types of constrained optimal control problems. The MPT toolbox is implemented in a MATLAB [The03] programming environment and consists of three software-blocks:

- Polytope Library
- Multi-Parametric Programming Solvers
- Computation of Feedback Controllers for Constrained Systems

Specifically, the toolbox contains efficient implementations of all polytope- and P-collectionoperations described in Chapter 2. In addition, efficient mp-LP and mp-QP solvers are provided and various feedback control schemes which rely on the previous two software-blocks are contained in MPT. In short, MPT contains a large part of all algorithms which were developed at the Automatic Control Laboratory (ETH Zürich) during the last two years as well as a plethora of standard functions which are often needed in the context of controller computation for constrained systems.

Furthermore, the MPT software package includes several state of the art solvers (CDD [Fuk04a], ESP [Jon04], SeDuMi [Stu99], Yalmip [Löf04]) such that the toolbox is truly 'unpack and use'. In addition to these freeware solvers which are distributed as part of MPT, several additional solvers are also compatible. Namely MATLABs

[1]The MPT toolbox is the result of a close collaboration with Michal Kvasnica and Mato Baotić. For a full list of contributors, see the acknowledgement at the end of this chapter.

`linprog` and `quadprog`, the LP and QP solvers of the Numerical Algorithms Group (NAG) [Num02] as well as the CPLEX [ILO03] and GLPK [Mak01] solvers. Aside from the functionality, a lot of work has gone into making the toolbox easily accessible, so that people with little background in control will (hopefully) be able to apply it without too many difficulties.

The MPT toolbox is available from

http://control.ee.ethz.ch/~mpt

and is updated on a regular basis. For an in-depth introduction to MPT, we refer the reader to the MPT web-page, where a detailed software manual is available for download.

17.1 Classes and Basic Polytope Manipulations

The toolbox defines a new class `polytope` inside the MATLAB programming environment along with overloaded operators which are presented in Table 17.1. The functions for polytope manipulations are given in Table 17.2.

Note that MPT does not handle polyhedral sets and is designed for use with bounded sets only. All functions take either polytopes or P-collections as an input argument which is illustrated in the following example:

Example 17.1.1

```
>> P=polytope([eye(2);-eye(2)],[1 1 1 1]');          %Create Polytope P
>> [r,c]=chebyball(P)                              %Chebychev ball inside P
   r=[0 0]'
   c=1
>> W=polytope([eye(2);-eye(2)],0.1*[1 1 1 1]');      %Create Polytope W
>> DIF=P-W;                                       %Pontryagin difference P-W
>> ADD=P+W;                                         %Minkowski addition P+W
>> plot(ADD, P, DIF, W);                                %Plot P-collection
```

`P=polytope(Px,Pc)`	Constructor for creating the polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$;
`double(P)`	Access internal data of the polytope, e.g. `[Px,Pc]=double(P);`
`display(P)`	Displays details about the polytope $\mathcal{P}$;
`nx=dimension(P)`	Returns dimension of a given polytope $\mathcal{P}$;
`nc=nconstr(P)`	For a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$ returns number of constraints of the P^x matrix (i.e. number of rows);
`[ , ]`	Horizontal concatenation of polytopes into an array, e.g. `PA=[P1,P2,P3];`
`( )`	Subscripting operator for P-collections, e.g. `PA(i)` returns the i-th polytope in `PA`;
`length(PA)`	Returns number of elements in a P-collection $\mathcal{P}_A$;
`end`	Indexing function which returns the final element of a P-collection;
`P == Q`	Check if two polytopes are equal ($\mathcal{P} = \mathcal{Q}$);
`P ~= Q`	Check if two polytopes are not-equal ($\mathcal{P} \neq \mathcal{Q}$);
`P >= Q`	Check if $\mathcal{P} \supseteq \mathcal{Q}$;
`P <= Q`	Check if $\mathcal{P} \subseteq \mathcal{Q}$;
`P > Q`	Check if $\mathcal{P} \supset \mathcal{Q}$;
`P < Q`	Check if $\mathcal{P} \subset \mathcal{Q}$;
`P & Q`	Intersection of two polytopes, $\mathcal{P} \cap \mathcal{Q}$;
`P | Q`	Union of two polytopes, $\mathcal{P} \cup \mathcal{Q}$. If the union is convex, the polytope $\mathcal{P} \cup \mathcal{Q}$ is returned, otherwise the P-collection (polytope array) $[\mathcal{P}\ \mathcal{Q}]$ is returned;
`P + Q`	Minkowski sum, $\mathcal{P} \oplus \mathcal{Q}$;
`P - Q`	Pontryagin difference, $\mathcal{P} \ominus \mathcal{Q}$;
`P \ Q`	Set difference operator. Works with polytopes and P-collections;

Table 17.1: Short overview of overloaded operators for the class `polytope`.

The resulting plot is depicted in Figure 17.1. When a `polytope` object is created, the constructor automatically normalizes its representation and removes all redundant

`B=bounding_box(P)`	Computes minimal hyper-rectangle containing a polytope $\mathcal{P}$;
`[c,r]=chebyball(P)`	Returns center c and radius r of the Chebychev ball inside $\mathcal{P}$;
`V=extreme(P)`	Computes extreme points (vertices) of a polytope $\mathcal{P}$;
`E=envelope(PA)`	Computes envelope $\mathcal{E}$ of a P-collection $\mathcal{P}_A$;
`[c,r]=facetcircle(P,i)`	Computes the center c and radius r of the largest lower dimensional ball inside facet i of polytope $\mathcal{P}$;
`P=hull(PA)`	Computes hull of a P-collection $\mathcal{P}_\mathcal{A}$;
`P=hull(V)`	or hull of an array of vertices V;
`bool=isfulldim(P)`	Checks if polytope $\mathcal{P}$ is full dimensional;
`bool=isinside(P,x)`	Checks if $x \in \mathcal{P}$. Also works for P-collections.
`plot(P)`	Plots a given polytope or P-collection in 2D or 3D;
`P=range(Q,A,f)`	Affine transformation of a polytope $\mathcal{P} = \{Ax + f \in \mathbb{R}^n \mid x \in \mathcal{Q}\}$;
`P=domain(Q,A,f)`	Compute polytope that is mapped to $\mathcal{Q}$ $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax + f \in \mathcal{Q}\}$;

Table 17.2: Functions defined for class `polytope`.

constraints. Note that all elements of the polytope class are private and can only be accessed as described in the tables. Furthermore, all information on a polytope is stored in the internal polytope structure. More functions on polytopes are given in Table 17.2 and are illustrated in the following example.

Example 17.1.2

```
>> P=polytope([eye(2);-eye(2)],[1 1 1 1]');        %Create Polytope P
>> Q=polytope([eye(2);-eye(2)],0.1*[1 1 1 1]');    %Create Polytope Q
>> D=P\Q;                    %Compute set difference between P and Q
>> length(D)                   %D is a P-collection with 4 elements
    ans=4
>> U=D|Q;                                    %Compute union of D and Q
>> length(U)                                %Union is again a polytope
    ans=1
```

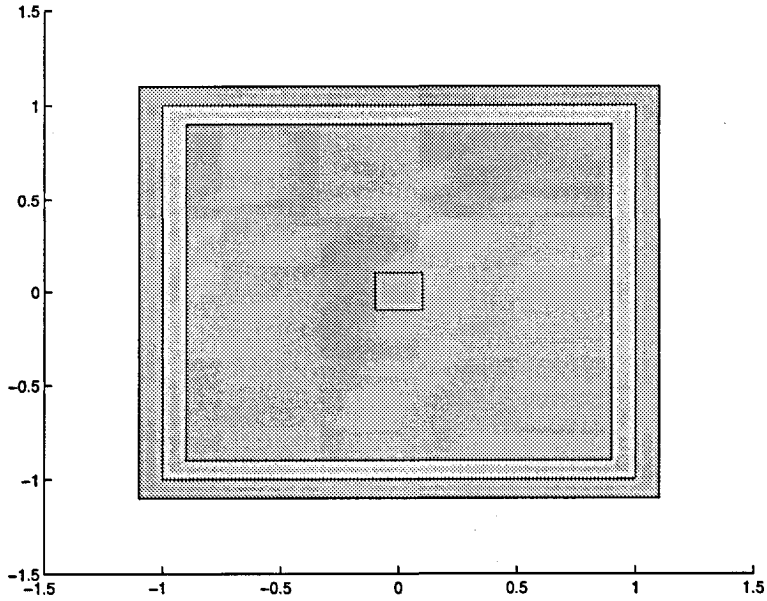
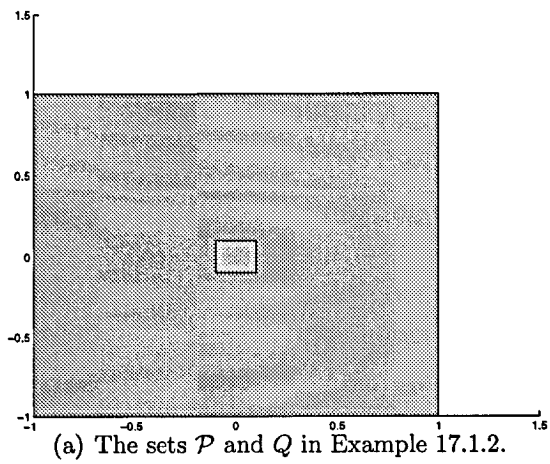
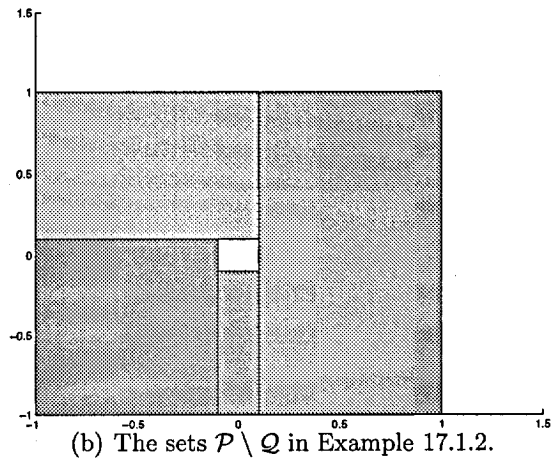


Figure 17.1: The result of the `plot` call in Example 17.1.1

```
>> U==P                                    %Check if two polytopes are equal
    ans=1
```



(a) The sets $\mathcal{P}$ and $\mathcal{Q}$ in Example 17.1.2.



(b) The sets $\mathcal{P} \setminus \mathcal{Q}$ in Example 17.1.2.

The polytopes $\mathcal{P}$ and $\mathcal{Q}$ are depicted in Figure 17.1. The following will illustrate the `hull` and `extreme` functions.

Example 17.1.3

```
>> P=polytope([eye(2);-eye(2)],[0 1 1 1]');      %Create Polytope P
>> Q=polytope([eye(2);-eye(2)],[1 1 0 1]');      %Create Polytope Q
>> VP=extreme(P);                    %Compute extreme vertices of P
>> VQ=extreme(Q);                    %Compute extreme vertices of P
>> D1=hull([P Q]);                   %Create convex Hull of P and Q
>> D2=hull([VP;VQ]);       %Create convex Hull of vertices VP and VQ
>> D1==D2                                %Check if hulls are equal
    ans=1
```

The `hull` function is overloaded such that it takes both elements of the `polytope` class as well as matrices of points as input arguments.

17.2 Control Functions

This section will give a brief overview of the main control functions which are provided with the MPT toolbox. All controller computation algorithms may be called by using the accessor function

```
[ctrlStruct]=mpt_Control(sysStruct,probStruct,Options)
```

which takes the structures defined in Chapter 20 as parameters and automatically calls one of the functions described below depending on the parameters which were passed. Every function returns a P-collection which contains the regions over which the feedback law is unique, i.e. $u = F_r x + G_r$ if $x \in \mathcal{P}_r$. The different functions for obtaining these solutions are:

`[ctrlStruct]=mpt_optControl(sysStruct,probStruct,Options)`:
: This function solves a constrained finite-time optimal control problem as defined in (4.7) for linear and quadratic cost objectives and for linear systems. See [Bao02, BBM00b] for additional details.

`[ctrlStruct]=mpt_optControlPWA(sysStruct,probStruct,Options)`:
: Calculates a solution to the constrained finite-time optimal control problem for linear cost objective and PWA system. See Section 15 or [BCM03a] for additional details.

`[ctrlStruct]=mpt_optInfControl(sysStruct,probStruct,Options)`:
This function computes a solution to the constrained infinite-time optimal control problem for quadratic cost objective and linear systems. See Section 10.3 or [GBTM03, GBTM04] for additional details.

`[ctrlStruct]=mpt_optInfControlPWA(sysStruct,probStruct,Options)`:
Solution to the constrained infinite-time optimal control problem for linear cost objective and PWA system. See [BCM03b] for details.

`[ctrlStruct]=mpt_iterative(sysStruct,probStruct,Options)`:
This function applies the minimum-time computation scheme described in [GM03] to linear systems. See Chapter 11.3.2 or [GPM03] for additional details.

`[ctrlStruct]=mpt_iterativePWA(sysStruct,probStruct,Options)`:
This function implements the minimum-time or N-step controller computation schemes described in [GKBM04a], depending on the `Options` setting. A stabilizing 'N-step' or minimal-time control is returned (see Chapter 16). This scheme can also be used to obtain robust solutions for PWA systems affected by additive disturbance by setting the appropriate flags [KM02].

As mentioned before, the solution to an optimal control problem is obtained by a call to the `mpt_control` function. This function takes the system and problem description as input arguments and calls one of the functions above to calculate the state feedback controller. There is no need to call the individual functions directly. The function `mpt_control` returns the control structure `ctrlStruct` which encompasses the control law $u = F_r x + G_r$ as well as the polyhedral partition $\{\mathcal{P}_r\}_{r=1}^{R}$ over which this PWA control law is active. Consult Chapter 20 for a detailed description of the mentioned structures.

17.3 Analysis Functions

Various scripts which serve to plot the obtained results as well as analysis functions are included in the toolbox. Some of these function are vital for obtaining low complexity controllers (see Chapters 11 and 16). The most important functions are given in the following Table:

`mpt_getPWALyapFct`	Computes a PWA Lyapunov function for a given closed-loop system;
`mpt_getPWQLyapFct`	Computes a PWQ Lyapunov function for a given closed-loop system;
`mpt_getCommonLyapFct`	Computes a common quadratic Lyapunov function for a set of linear systems;
`mpt_infset`	Calculates the maximal (robust) positively invariant set for an LTI system;
`mpt_infsetPWA`	Computes the maximal (robust) positive invariant subset for PWA systems;
`mpt_maxCtrlSet`	Computes the maximal (robust) stabilizable set $\mathcal{K}_\infty$ or the maximal (robust) controllable set $\mathcal{C}_\infty$ for LTI and PWA systems;
`mpt_plotPartition`	Plots controller partitions of type `ctrlStruct`.
`mpt_plotTrajectory`	Graphical interface for plotting closed-loop trajectories in state-space;
`mpt_plotTimeTrajectory`	Plots closed-loop trajectories of states, inputs and outputs as a function of time;
`mpt_plotU`	Plots the value of the control input over the controller partition;
`mpt_plotPWA`	Plots a PWA function in 3D;
`mpt_plotPWQ`	Plots a PWQ function in 3D;
`mpt_plotArrangement`	Plots a hyperplane arrangement of a polytope in half-space representation;

18

MPT in 15 minutes

This short introduction is not meant to (and does not) replace the MPT manual. It serves to clarify some key points of Model Predictive Control and application thereof within the framework of the MPT toolbox. Specifically, the main problems which arise in practice are illustrated in a concise manner without going into the technical details.

18.1 First Steps

Before reading the rest of this chapter, have a close look at the provided demonstrations and go through them slowly. At the MATLAB command prompt, type `mpt_demo1`, `mpt_demo2`, ..., `mpt_demo6`. After completing the demos, run some examples by typing `runExample` at the command prompt. Finally, for a good overview, type `help mpt` and `help mpt/polytope` to get the list and short descriptions of (almost) all available functions.

Guidelines for Modelling a Dynamical System

Before actually computing a controller, the first step is to obtain a suitable system representation. The most important aspects in system modelling for MPT are given below:

1. Always make sure your dynamic matrices and states/inputs are well scaled. Ideally all variables exploit the full range between ± 10. See [SP96] for details.

2. Try to have as few different dynamics as possible when designing your PWA system model. If possible, use an LTI model.

3. The fewer states and inputs your system model has, the easier all subsequent computations will be.

4. Use the largest possible sampling time when discretizing your system.

18.2 State Regulation Problems

In this section the regulation problem is discussed, i.e. the objective is to drive the state to the origin. See the subsequent section for the special case of tracking. In order to compute a controller, only one function call is needed:

```
ctrlStruct = mpt_control(sysStruct,probStruct)
```

For a detailed description of how to define your system `sysStruct` and problem `probStruct`, see Chapter 20, the MPT manual or type `help mpt_sysStruct` and `help mpt_probStruct`. We also suggest you examine the m-files in the 'Examples' directory of the MPT toolbox and take a close look at the `RunExample.m` file. Additional examples for controller computations are provided in Section 20.4.

Computing explicit state feedback controllers via multi-parametric programming may easily lead to controllers with prohibitive complexity and the following is intended to give a brief overview of the existing possibilities to obtain tractable controllers for the problems MPT users may face. Specifically, there are three controller properties which are important in this respect: performance, stability and constraint satisfaction.

Infinite-Time Optimal Control: [GBTM03, BCM03b] [see Section 10.3]
To use this method, set `probStruct.N=Inf`. This will yield the infinite time optimal controller, i.e., the best possible performance for the problem at hand. Asymptotic stability and constraint satisfaction are guaranteed and all states which are stabilizable will be covered by the resulting controller. However, the complexity of the associated controller may be prohibitive.

Finite-Time Optimal Control [BMDP02, BCM03a, Bor03, MRRS00] [see Chapter 5]
To use this method, set `probStruct.N` $\in \mathbb{N}_+$ and `probStruct.subopt_lev=0`. This will yield the finite time optimal controller, i.e. performance will be

N-step optimal but may not be infinite horizon optimal. The complexity of the resulting controller depends strongly on the prediction horizon (large $N \Rightarrow$ complex controller). It is furthermore necessary to differentiate the following cases:

`probStruct.Tconstraint=0`: No terminal set constraint. The controller will be defined over a superset of the maximum controllable set, but no guarantees on stability or closed-loop constraint satisfaction can be given. As the prediction horizon N is increased the feasible set of states will converge to the maximum controllable set $\mathcal{C}_\infty$ from 'the outside-in', i.e. the controlled set will shrink as N increases (see Remark 5.2.4 and Theorem 7.4.2). To extract the set of states which satisfy the constraints for all time, call `mpt_infsetPWA`. To analyze these states for stability, call `mpt_getPWALyapFct` or `mpt_getPWQLyapFct`. Note that the analysis functions may have prohibitive run times for large partitions.

`probStruct.Tconstraint=1`: A stabilizing terminal set is automatically computed. The resulting controller will guarantee stability and constraint satisfaction for all time, but will only cover a subset of the maximum stabilizable set of states $\mathcal{K}_\infty$. By increasing the prediction horizon, the controllable set of states will converge to the maximum controllable set from 'the inside-out', i.e. the controlled set will become larger as N increases (see Remark 5.2.4 and Theorem 7.4.3).

`probStruct.Tset=T`: User defined terminal set. Depending on the properties (e.g., invariance, size) of the target set `T`, any combination of the two cases previously described may occur.

Minimum-Time Control [GM03, GKBM04a] [see Sections 11.3.2 and 16.1]

To use this method, set `probStruct.subopt_lev=1`. This will yield the minimal time controller with respect to a target set around the origin, i.e. the controller will drive the state into this set in minimal time. In general, the complexity of minimum time controllers is significantly lower than that of their $1/2/\infty$-norm cost optimal counterparts. The controller is guaranteed to cover all controllable states and asymptotic stability and constraint satisfaction are

guaranteed. Note that if you choose to manually define your target set by setting `probStruct.Tset=T`, these properties may not hold.

N-step Control [GM03, GPM03] [see Sections 11.3.3 and 16.3]

To use this method, set `probStruct.subopt_lev=2`. This will yield a controller for prediction horizon N, with additional constraints which guarantee asymptotic stability and constraint satisfaction in closed-loop. The controller covers all controllable states. The complexity of this N-step controller is generally significantly lower than all other control schemes in MPT which cover the maximal controllable set. However, the computation of the controller may take a long time.

Conclusion

The key influence on controller complexity are as follows

1. Prediction horizon N
2. Number of different dynamics of the PWA system
3. Dimension of state and input.
4. Type of control scheme.

Furthermore, 2-norm objectives generally yield controllers of lower complexity than their $1/\infty$-norm counterparts. Therefore, we suggest you try the control schemes in the following order to trade-off performance for complexity

1. Finite Horizon Optimal Control for small N (i.e., $N = 1, 2$);

   `probStruct.Tconstraint=0`
2. N-step Control
3. Minimum-Time Control
4. Finite Horizon Optimal Control for large N

   `probStruct.Tconstraint=1`
5. Infinite Horizon Optimal Control

Note that for a specific system, the order of preference may be different, so it may yet be best to investigate all methods.

18.3 State Tracking Problems

When solving tracking problems, computation schemes become more complex compared to the standard regulation problems covered in the previous section. It is necessary to differentiate between the case of constant reference tracking (reference state is fixed a priori) and time varying reference tracking (user defined reference is arbitrarily time varying).

For constant reference tracking (`probStruct.xref` $\in \mathbb{R}^n$), the problem setup reduces to a normal regulation problem where all of the observations from the previous section hold.

Time varying reference tracking (`probStruct.tracking=1`) is implemented for both LTI and PWA systems. For time varying reference states, it is necessary to augment the state space matrices. The process of augmenting the state update equations is performed automatically by MPT, the following exposition is intended to give you some flavor of what is going on.

First the state vector x is extended with the reference state vector x_{ref}, i.e. the reference states are added to the dynamical model. The input which is necessary such that the state remains at the reference is not generally known. Therefore the state update equations are reformulated in Δu-form. In this framework the system input at time k is $\Delta u(k)$ whereby $u(k-1)$ is an additional state in the dynamical model, i.e. the system input can be obtained as $u(k) = u(k-1) + \Delta u(k)$. The state update equation is thus given by

$$\begin{pmatrix} x(k+1) \\ u(k) \\ x_{ref}(k+1) \end{pmatrix} = \begin{pmatrix} A & B & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} x(k) \\ u(k-1) \\ x_{ref}(k) \end{pmatrix} + \begin{pmatrix} B \\ I \\ 0 \end{pmatrix} \Delta u(k).$$

Assume a 3rd order system with 2 inputs. In Δu-tracking formulation, the resulting dynamical model will have 8 states (3 system states $x \in \mathbb{R}^3$ + 3 reference states $x_{ref} \in \mathbb{R}^3$ + 2 input states $u(k-1) \in \mathbb{R}^2$) and 2 inputs ($\Delta u(k) \in \mathbb{R}^2$). If we solve the regulation problem for the augmented system (see Section 18.2) we obtain a controller which allows for time varying references. For control purposes, the reference state x_{ref} is imposed by the user, i.e. x_{ref} is set to a specific value. By choosing appropriate

objective weights, e.g. $Q_u \succ 0$ and

$$Q_x = \begin{pmatrix} I & 0 & -I \\ 0 & 0 & 0 \\ -I & 0 & I \end{pmatrix},$$

the regulation controller automatically steers the state x to the reference state x_{ref}.

Note that time varying tracking problems are generally of high dimension, such that controller computation is expensive. If the control objective can be reduced to a regulation problem for a set of predefined reference points, we suggest to solve a sequence of fixed state tracking problems instead of the time varying tracking problem. None of the control schemes in MPT enforce offset free control in case of persistent disturbances. However, this type of control can be achieved by appropriate modification of the system model (e.g., [PK03]).

19

Polytope Library

As already mentioned in Chapter 2, a polytope is a convex bounded set which can be represented either as an intersection of a finite number of half-spaces or as the convex hull of a set of points. Both representations are supported in MPT and it is possible to switch between the two. Note however, that internally MPT performs almost all computations on half-space representations of polytopes.

19.1 Creating a polytope

A polytope in MPT is created by a call to the polytope constructor as follows

```
P = polytope(H,K)
```

where the matrices `H` and `K` describe the polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Hx \leq K\}$. The constructor automatically computes the polytope $\mathcal{P}$ in non-redundant form. In addition, center and diameter of the Chebychev ball (see Chapter 2) are computed and the half-space representation is normalized. The constructor then returns a polytope object. A polytope can also be defined by its vertices :

```
P = polytope(V)
```

where `V` is a matrix which contains vertices of the polytope in the following format:

$$\mathtt{V} = \begin{bmatrix} v_{1,1} & \dots & v_{1,n} \\ \vdots & \vdots & \vdots \\ v_{k,1} & \dots & v_{k,n} \end{bmatrix} \tag{19.1}$$

where k is the total number of vertices and n is the state dimension. Hence, vertices are stored row-wise. Before the polytope object is created, the vertex representation is first converted to half-space description by computing the convex hull. The extreme vertices are stored in the polytope object and can be returned upon request without additional computational effort.

19.2 Accessing data stored in a polytope object

Each `polytope` object is internally represented as a structure, but because of the object-oriented approach, this information cannot be directly obtained by using structure referencing through the . (dot) operator. Special functions have to be called in order to retrieve individual fields.

In order to access the half-space representation of the polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Hx \leq K\}$, one has to use the command `double` as described below.

```
[H,K] = double(P)
```

The center and radius of the Chebyshev ball can be obtained by:

```
[xCheb, RCheb] = chebyball(P)
```

The polytope is bounded if

```
flag = isbounded(P)
```

returns 1 as the output. Dimension of a polytope can be obtained by

```
d = dimension(P)
```

and

```
nc = nconstr(P)
```

will return number of constraints (i.e. number of half-spaces) defining the given polytope `P`. The vertex representation of a polytope can be obtained by:

```
V = extreme(P)
```

which returns vertices stored row-vise in the matrix `V`. As enumeration of extreme vertices is an expensive operation, the computed vertices can be stored in the polytope object. To do this, the function must be called as

```
[V,P] = extreme(P)
```

which returns the vertices `V` and the updated polytope object `P` with the stored vertices. To check if a given point `x` lies in a polytope `P`, use the following call:

```
flag = isinside(P,x)
```

The function returns 1 if $x \in \mathcal{P}$, 0 otherwise. If P is a P-collection (see Chapter 2), the function call can be extended to provide additional information:

```
[flag, inwhich, closest] = isinside(P,x)
```

which returns a 1/0 flag which denotes if the given point x belongs to any polytope in a P-collection P. If the given point is contained in more than one polytope, inwhich contains the indices of the regions which contain x. If there is no such region, the index of the region which is closest to the given point x is returned in closest. A more detailed overview of the polytope library is given in Table 19.1.

`P=polytope(H,K)`	Constructor for creating the polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Hx \leq K\}$;
`P=polytope(V)`	Constructor for creating the polytope out of extreme points;
`double(P)`	Access internal data of the polytope, e.g. `[H,K]=double(P)`;
`display(P)`	Displays details about the polytope $\mathcal{P}$;
`nx=dimension(P)`	Returns dimension of a given polytope $\mathcal{P}$;
`nc=nconstr(P)`	For a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid Hx \leq K\}$ returns number of rows of the H matrix;
`[ , ]`	Horizontal concatenation of polytopes into an array, e.g. `PA=[P1,P2,P3]`;
`( )`	Subscripting operator for polytope arrays, e.g. `PA(i)` returns the i-th polytope in PA;
`length(PA)`	Returns number of elements in a polytope array P_A;
`end`	In indexing functions returns the final element of an array;
`[c,r]=chebyball(P)`	Returns center c and radius r of the Chebychev ball of $\mathcal{P}$;
`V=extreme(P)`	Computes extreme points (vertices) of a polytope $\mathcal{P}$;
`bool=isfulldim(P)`	Checks if polytope $\mathcal{P}$ is full dimensional;
`bool=isinside(P,x)`	Checks if $x \in \mathcal{P}$. Works also for polytope arrays;

Table 19.1: Functions defined for class `polytope`.

19.3 P-collections

Polytope objects can be concatenated into arrays and it does not matter if the elements are stored row-wise or column-wise. A P-collection is created using standard MATLAB concatenation operators `[,]`, e.g. `A = [B C D]`.

It does not matter whether the concatenated elements are single polytopes or P-collections. To illustrate this, consider the polytopes `P1, P2, P3, P4, P5` and P-collections `A = [P1 P2]` and `B = [P3 P4 P5]`. Then the following P-collections `M` and `N` are equivalent:

```
M = [A B]
N = [P1 P2 P3 P4 P5]
```

Individual elements of a P-collection can be obtained using the standard referencing `(i)` operator, i.e.

```
P = M(2)
```

will return the second element of the P-collection `M` which is equal to `P2`, in this case. More complicated expressions can be used for referencing:

```
Q = M([1,3:5])
```

Here, `Q` is a P-collection which contains `P1, P3, P4, P5`.

If you want to remove some element from a P-collection, use the referencing command as follows:

```
M([1 3]) = []
```

which will remove the first and third element from the P-collection `M`. If some element of a P-collection is deleted, the remaining elements are shifted towards the start of the P-collection. This means that for `N = [P1 P2 P3 P4 P5]`, the command

```
N([1 3]) = []
```

will yield the P-collection `N = [P2 P4 P5]` and the length of the array is 3. No empty positions in a P-collection are allowed. Analogously, empty polytopes are not being added to a P-collection.

A P-collection is still a polytope object, such that all functions which work on polytopes also support P-collections. This is a key feature of MPT. All functions

automatically adapt to the type of input (i.e. polytope or P-collection) provided by the user.

The length of a given P-collection is obtained by

```
l = length(N)
```

For additional information on polytopes and P-collections in MPT, we refer the reader to the manual.

Seite Leer /
Blank leaf

20

Control Functions, Structures and Objects

As indicated in Section 17.2, the solution to an optimal control problem can be obtained by a simple call to `mpt_control`. The general syntax is given below:

```
ctrlStruct = mpt_control(sysStruct, probStruct, Options)
```

Based on the system definition `sysStruct` and problem description `probStruct`, the main control routine `mpt_control` automatically calls one of the functions reported in Section 17.2 to calculate the explicit solution to a given problem. Once the control law is calculated, the solution is returned in form of the controller structure `ctrlStruct`. The system-, problem- and control- objects will be discussed in this chapter.

MPT provides a variety of control routines which are being called from `mpt_control`. Solutions to the following problems can be obtained

A. Constrained Finite-Time Optimal Control (CFTOC) Problem for LTI and PWA systems,

B. Constrained Infinite-Time Optimal Control Problem (CITOC) for LTI and PWA systems,

C. Constrained Minimum-Time Optimal Control (CMTOC) Problem for LTI and PWA systems,

D. N-step controller scheme for LTI and PWA systems

The problem which will be solved depends on parameters of the system and problem structure, namely on type of the system (LTI or PWA), prediction horizon (finite or infinite) and the level of sub-optimality (optimal solution, minimum-time solution,

System	N	Problem	Function	Reference
LTI	N	CFTOC	`mpt_optControl`	[Bao02, Bor03]
LTI	Inf	CITOC	`mpt_optInfControl`	[GBTM03]
LTI	N / Inf	CMTOC	`mpt_iterative`	[GM03, GPM03]
LTI	N / Inf	N-step	`mpt_oneStepCtrl`	[GM03, GPM03]
PWA	N	CFTOC	`mpt_optControlPWA`	[BBBM03]
PWA	Inf	CITOC	`mpt_optInfControlPWA`	[BCM03b]
PWA	N / Inf	CMTOC	`mpt_iterativePWA`	[GKBM04a]
PWA	N / Inf	N-step	`mpt_iterativePWA`	[GKBM04b]

Table 20.1: List of control strategies applied to different system and problem definitions.

N-step controller). Different combinations of these three parameters lead to different optimization procedures, as reported in Table 20.1. See the documentation of the individual functions for implementation details.

20.1 System Structure sysStruct

The system object `sysStruct` is a structure which describes the system to be controlled. MPT can deal with two types of systems:

1. Discrete-time linear time-invariant (LTI) systems
2. Discrete-time piecewise affine (PWA) systems

Both system types can be subject to constraints on control inputs, system states and/or outputs. In addition, constraints on the slew rate of the control inputs can also be given.

LTI systems

In general, a constrained linear time-invariant system is defined by the following relations:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

$$\begin{aligned} &\text{subj. to} \\ y_{min} &\leq y(k) \leq y_{max} \\ u_{min} &\leq u(k) \leq u_{max} \end{aligned}$$

Such an LTI system is defined by the following mandatory fields:

```
sysStruct.A = A;
sysStruct.B = B;
sysStruct.C = C;
sysStruct.D = D;
sysStruct.ymax = ymax;
sysStruct.ymin = ymin;
sysStruct.umax = umax;
sysStruct.umin = umin;
```

Constraints on the slew rate of the control input $u(k)$ can also be imposed by:

```
sysStruct.dumax = dumax;
sysStruct.dumin = dumin;
```

which enforces $\Delta u_{min} \leq u(k) - u(k-1) \leq \Delta u_{max}$. In order to deactivate a certain constraint, simply set the associated limiter to `Inf`. An LTI system subject to parametric uncertainty and/or additive disturbances is described by the following set of relations:

$$\begin{aligned} x(k+1) &= A_{unc}x(k) + B_{unc}u(k) + w(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

where $w(k)$ is an unknown, but bounded additive disturbance, i.e.

$$w(k) \in \mathbb{W} \qquad \forall k \in \mathbb{N}$$

To specify an additive disturbance, set `sysStruct.noise = W` where `W` is a polytope object bounding the disturbance. Parametric uncertainty can be specified by a cell array of matrices `Aunc` and `Bunc` as follows:

```
sysStruct.Aunc = {A1, ..., An};
sysStruct.Bunc = {B1, ..., Bn};
```

where `Aunc` and `Bunc` denote the vertices of the polytopic uncertainty as described in (11.3).

PWA Systems

PWA systems are widely used to model hybrid and non-linear systems. The dynamical behavior of such systems is captured by relations of the following form:

$$\begin{aligned} x(k+1) &= A_i x(k) + B_i u(k) + f_i \\ y(k) &= C_i x(k) + D_i u(k) + g_i \\ \text{subj. to} & \\ y_{min} &\leq y(k) \leq y_{max} \\ u_{min} &\leq u(k) \leq u_{max} \\ \Delta u_{min} &\leq u(k) - u(k-1) \leq \Delta u_{max} \end{aligned}$$

Each dynamic i is active in a polyhedral partition $\mathcal{D}_i$ bounded by the so-called guardlines:

$$\mathcal{D}_i \triangleq \{[x^T \;\; u^T]^T \in \mathbb{R}^{n+m} \mid guardX_i x(k) + guardU_i u(k) \leq guardC_i\},$$

which means dynamic i will be applied if the state/input is contained in $\mathcal{D}_i$. Fields of `sysStruct` describing a PWA system are listed below:

```
sysStruct.A = {A1, ..., AR}
sysStruct.B = {B1, ..., BR}
sysStruct.C = {C1, ..., CR}
sysStruct.D = {D1, ..., DR}
sysStruct.f = {f1, ..., fR}
sysStruct.g = {g1, ..., gR}
sysStruct.guardX = {guardX1, ..., guardXR}
sysStruct.guardU = {guardU1, ..., guardUR}
sysStruct.guardC = {guardC1, ..., guardCR}
```

Note that all fields have to be cell arrays of matrices of compatible dimensions, R denotes the total number of different dynamics. If `sysStruct.guardU` is not provided, it is assumed to be zero. The system constraints are defined by:

```
sysStruct.ymax  = ymax;
sysStruct.ymin  = ymax;
```

```
sysStruct.umax  = umax;
sysStruct.umin  = umin;
sysStruct.dumax = dumax;
sysStruct.dumin = dumin;
```

Constraints on slew rate are optional and can be omitted. MPT is able to deal also with PWA systems which are affected by bounded additive disturbances:

$$x(k+1) = A_i x(k) + B_i u(k) + f_i + w(k)$$

where the disturbance $w(k)$ is assumed to be bounded for all time instances by some polytope $\mathbb{W}$. To indicate that your system is subject to such a disturbance, set

```
sysStruct.noise = W;
```

where `W` is a polytope object of appropriate dimension. Polytopic uncertainty in the dynamics cannot be treated by the control schemes for PWA systems. We leave it up to the user to implement the scheme in [RKM03], if this functionality is required. Mandatory and optional fields of the system structure are summarized in Tables 20.1 and 20.1, respectively.

`A, B, C, D, f, g`	State-space dynamic matrices for LTI (4.1) and PWA (6.3b) systems. Set elements to empty if they do not apply;
`umin, umax`	Bounds on inputs `umin` $\leq$ u(k) $\leq$ `umax`;
`ymin, ymax`	Constraints on the outputs `ymin` $\leq$ y(k) $\leq$ `ymax`;
`guardX, guardU, guardC`	Polytope cell array defining where the dynamics are active (for PWA systems). $\mathcal{D}_i = \{(x,u) \mid guardX\{i\}\ x + guardU\{i\}\ u \leq guardC\{i\}\}$;

Table 20.2: Mandatory fields of the system structure `sysStruct`.

20.2 Problem Structure probStruct

The problem object `probStruct` is a structure which defines the optimization problem to be solved by MPT. The `probStruct` object contains all information which does

`dumin, dumax`	Bounds on `dumin` $\leq$ u(k)-u(k-1) $\leq$ `dumax`;
`noise`	A `polytope` bounding the additive disturbance;
`Aunc, Bunc`	Cell arrays containing the vertices of the polytopic uncertainty;
`Pbnd`	Polytope limiting the state-space of interest, i.e., the defining the exploration space;

Table 20.3: Optional fields of the system structure `sysStruct`.

not directly originate from the dynamical system (e.g. control objective, etc.). Let us recall a standard finite time optimization problems as described in Chapter 9:

$$J_N^*(x) = \min_{u_0,\ldots,u_{N-1}} \quad ||P_N x(N)||_{\text{norm}} + \sum_{k=0}^{N-1} ||Ru(k)||_{\text{norm}} + ||Qx(k)||_{\text{norm}} \tag{20.1a}$$

$$\text{subj. to} \tag{20.1b}$$

$$x(k+1) = f_{dyn}(x(k), u(k), w(k)), x(0) = x, \tag{20.1c}$$

$$u_{min} \leq u(k) \leq u_{max} \tag{20.1d}$$

$$du_{min} \leq u(k) - u(k-1) \leq du_{max} \tag{20.1e}$$

$$y_{min} \leq g_{dyn}(x(k), u(k)) \leq y_{max} \tag{20.1f}$$

$$x(N) \in T_{set} \tag{20.1g}$$

The function $f_{dyn}(x(k), u(k), w(k))$ is the state-update function as defined in Section 20.1 and $g_{dyn}(x(k), u(k))$ yields the system output as a function of state and input. Here,

N	prediction horizon
norm	objective norm, can be `1`, `2` or `Inf`
Q	weighting matrix on the states
R	weighting matrix on the manipulated variables
P_N	weight imposed on the terminal state
T_{set}	terminal set constraint

are parameters which do not originate from the system dynamics and are defined in the `probStruct` object. Note that the entries N, norm, Q and R are mandatory. Optional fields are summarized in Table 20.4.

probStruct.y0bounds	Boolean variable. If false, no constraints are imposed on the initial output $y(0)$ (default is 0);
probStruct.tracking	Boolean variable, if set to 1, the problem will be formulated as a state-tracking problem (default is 0);
probStruct.P_N	Weight on the terminal state. If not specified, it is assumed to equal to the ARE solution (4.4) for quadratic cost, or $P_N = Q$ for linear cost;
probStruct.Tset	Polytope object describing the terminal set. If not provided and probStruct.norm is 2, the LQR set around the origin will be calculated automatically to guarantee stability properties (see Chapter 5.2);
probStruct.Tconstraint	An integer (0, 1, 2) denoting which auxiliary stability constraint to apply. 0 - no terminal constraint, 1 - LQR terminal set 2 - user-provided terminal set constraint. Note that if probStruct.Tset is given, Tconstraint will be set to 2 automatically;
probStruct.feedback	Boolean variable, if set to 1, the problem is augmented such that $U = Kx + c$ where K is a state-feedback gain (typically an LQR controller) and the optimization aims to identify the proper offset c (default is 0);
probStruct.FBgain	If the former option is activated, a specific state-feedback gain matric K can be provided (otherwise a LQR controller will be computed automatically);
probStruct.xref	By default, the toolbox designs a controller which forces the state vector to converge to the origin. If you want to track some a priori given reference point, provide the reference state in this variable. probStruct.tracking has to be 0 (zero) to use this option;
probStruct.uref	A reference point for the manipulated variable (i.e. the equilibrium u for state probStruct.xref can be specified here. If it is not given, it is assumed to be zero;

Table 20.4: Optional field of the probStruct structure.

MPT provides different control strategies with different levels of optimality. Specifically, it is possible to trade off controller performance for controller complexity by manipulation of the `.subopt_lev` field, as described in the following:

1. The cost-optimal solution leads to a control law which minimizes a given performance index. This strategy is enforced by

   ```
   probStruct.subopt_lev = 0
   ```

2. Another possibility is to use the minimum-time solution, i.e. the control law will drive a given state into an invariant set around the origin as quickly as possible (see Sections 11.3.2 and 16.1). This strategy usually leads to simpler control laws (i.e. less controller regions are generated). This approach is enforced by

   ```
   probStruct.subopt_lev = 1
   ```

3. The last option is to use a N-step control scheme (see Sections 11.3.3 and 16.3). This approach constructs an N-step solution (default $N = 1$) and subsequently attempts to verify stability by constructing a PWA or PWQ Lyapunov function. The approach generally results in a small number of regions and asymptotic stability as well as closed-loop constraint satisfaction is guaranteed. In order to compute this type of controller, use:

   ```
   probStruct.subopt_lev = 2
   ```

An overview of the implications of the `subopt_lev` field is given in Table 20.1.

20.3 Controller Structure ctrlStruct

The controller structure is an object which includes all information obtained while solving a given optimal control problem. In general, it describes the obtained control law and can be used both for analysis of the solution, as well as for implementation of the control law. The fields of the structure are summarized in Table 20.3.

`Pn`	The polyhedral partition over which the control law is defined is returned in this field. It is, in general, a polytope array;
`Fi, Gi`	The PWA control law for a given state $x(k) \in \mathcal{P}_r$ is given by `u = Fi{r} x(k) + Gi{r}`. `Fi` and `Gi` are cell arrays;
`Ai, Bi, Ci`	The value function $J_N^*(x)$ is returned in these three cell arrays and for a given state $x(k)$ can be evaluated as `J(x) = x(k)' Ai{r} x(k) + Bi{r} x(k) + Ci{r}` where the prime denotes a transpose and r is the index of the active region (i.e. the region of `Pn` containing the given state $x(k)$);
`Pfinal`	In this field, the feasible set $\mathcal{X}_N$ is returned (see Definition 3.1.1). For LTI systems, $\mathcal{X}_N$ corresponds to the convex union of all polytopes in `Pn`. For PWA systems, $\mathcal{X}_N$ is a P-collection;
`dynamics`	A vector which denotes which dynamics is active in which region of `Pn` (only relevant for PWA systems);
`overlaps`	Boolean variable denoting whether regions of the controller partition overlap;
`sysStruct`	System description in the `sysStruct` format;
`probStruct`	Problem description in the `probStruct` format;
`details`	More details about the solution (e.g. total run time);

Table 20.5: Fields of the controller structure `ctrlStruct`.

20.4 Examples

In order to obtain a feedback controller, it is necessary to specify both the system as well as the control problem. We now illustrate the computation procedure in MPT

with a simple second-order double integrator, with bounded scalar input $|u| \leq 1$ and output $\|y\|_\infty \leq 5$:

Example 20.4.1

```
>> sysStruct.A=[1 1; 0 1];                    %x(k+1)=Ax(k)+Bu(k)
>> sysStruct.B=[0 1];                         %x(k+1)=Ax(k)+Bu(k)
>> sysStruct.C=[1 0; 0 1];                      %y(k)=Cx(k)+Du(k)
>> sysStruct.D=[0;0];                           %y(k)=Cx(k)+Du(k)

>> sysStruct.umin=-1;                  %Input constraints umin<=u(k)
>> sysStruct.umax=1;                   %Input constraints u(k)<=umax
>> sysStruct.ymin=[-5 -5]';           %Output constraints ymin<=y(k)
>> sysStruct.ymax=[5 5]';             %Output constraints y(k)<=ymax
```

For this system we will now formulate the problem with quadratic cost objective in (20.1) and a prediction horizon of $N = 5$:

```
>> probStruct.norm=2;                                %Quadratic Objective
>> probStruct.Q=eye(2);          %Objective: min_U J=sum x'Qx + u'Ru...
>> probStruct.R=1;               %Objective: min_U J=sum x'Qx + u'Ru...
>> probStruct.N=5;                    %...over the prediction horizon 5
>> probStruct.subopt_lev=0;                  %Compute optimal solution
```

If we now call

```
>> ctrlStruct=mpt_Control(sysStruct,probStruct);   %Compute controller
>> mpt_plotPartition(ctrlStruct);             %Plot controller partition
```

the controller for the given problem is returned and plotted (see Figure 20.1(a)), i.e., if the state $x \in \mathrm{PA}(r)$, then the optimal input for prediction horizon $N = 5$ is given by $u = F_i\{r\}x + G_i\{r\}$. If we wish to compute a N-step controller with $N = 1$ (see Section 11.3.3), we can run the following:

```
>> probStruct.subopt_lev=2;                  %Compute N-step controller
>> [ctrlStruct]=mpt_Control(sysStruct,probStruct); %Compute controller
>> mpt_plotPartition(ctrlStruct)             %Plot controller partition
>> Q = ctrlStruct.details.lyapQ;             %Extract Lyapunov Function
>> L = ctrlStruct.details.lyapL;             %Extract Lyapunov Function
>> C = ctrlStruct.details.lyapC;             %Extract Lyapunov Function
>> mpt_plotPWQ(ctrlStruct.finalPn,Q,L,C);      %Plot Lyapunov Function
```

The resulting partition and Lyapunov function is depicted in Figures 20.1(b) and 20.1(c) respectively. In the following we will solve the PWA problem introduced in [MR03] by defining two different dynamics which are defined in the left- and right half-plane of the state space respectively.

Example 20.4.2

```
>> H=[-1 1; -3 -1; 0.2 1; -1 0; 1 0; 0 -1];          %Polytopic state
>> K=[  15;    25;     9;    6;   8;   10];       %constraints Hx(k)<=K

>> sysStruct.C{1} = [1 0];                          %System Dynamics 1:
>> sysStruct.D{1} = 0;                              %y(k)=Cx(k)+Du(k)+g
>> sysStruct.g{1} = [0];
>> sysStruct.A{1} = [0.5 0.2; 0 1];                 %System Dynamics 1:
>> sysStruct.B{1} = [0; 1];                        %x(k+1)=Ax(k)+Bu(k)+f
>> sysStruct.f{1} = [0.5; 0];
>> sysStruct.guardA{1} = [1 0; H];                  %Dynamics 1 defined
>> sysStruct.guardC{1} = [  1; K];                %in guardA x <= guardC

>> sysStruct.C{2} = [1 0];                          %System Dynamics 2:
>> sysStruct.D{2} = 0;                              %y(k)=Cx(k)+Du(k)+g
>> sysStruct.g{2} = [0];
>> sysStruct.A{2} = [0.5 0.2; 0 1];                 %System Dynamics 2:
>> sysStruct.B{2} = [0; 1];                        %x(k+1)=Ax(k)+Bu(k)+f
>> sysStruct.f{2} = [0.5; 0];
>> sysStruct.guardA{2} = [-1 0; H];                 %Dynamics 2 defined
>> sysStruct.guardC{2} = [  -1; K];               %in guardA x <= guardC

>> sysStruct.ymin = -10;                            %Output constraints
>> sysStruct.ymax = 10;                            %for dynamic 1 and 2
>> sysStruct.umin = -1;                             %Input constraints
>> sysStruct.umax = 1;                             %for dynamic 1 and 2
```

we can now compute the N-step feedback controller (see Section 16.3) by defining the problem

```
>> probStruct.norm=2;                              %Quadratic Objective
```

```
>> probStruct.Q=eye(2);          %Objective: min_U J=sum x'Qx + u'Ru...
>> probStruct.R=0.1;             %Objective: min_U J=sum x'Qx + u'Ru...
>> probStruct.subopt_lev=1;                  %Compute N-step controller
```

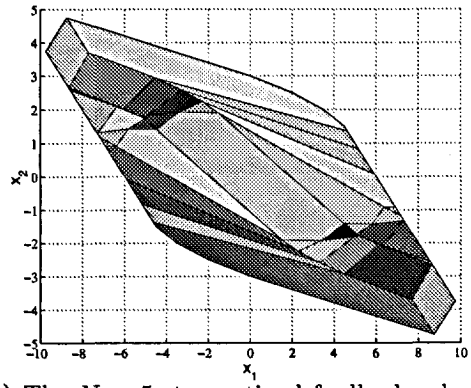
and calling the control function,

```
>> [ctrlStruct]=mpt_Control(sysStruct,probStruct);
>> mpt_plotPartition(ctrlStruct)
```

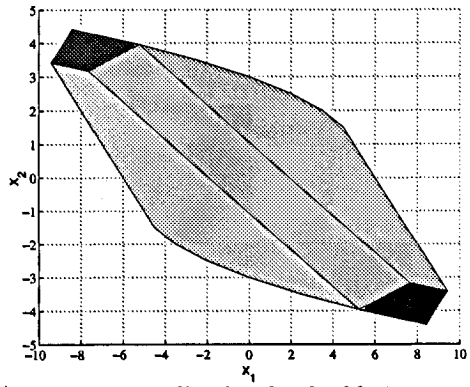
The result is depicted in Figure 20.2.

Acknowledgment

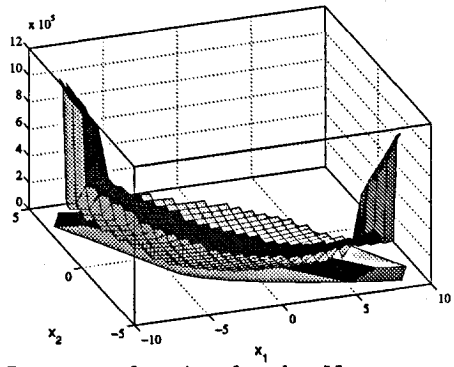
We would like to thank all contributors to the toolbox which are not on the author list. Specifically (in alphabetical order): Miroslav Barić, Alberto Bemporad, Pratik Biswas, Francesco Borrelli, Frank J. Christophersen, Eric Kerrigan, Marco Lüthi, Saša V. Raković, Sasikanth Manipatruni, Raphael Suard, Fabio Torrisi and Kari Unneland. A special thanks goes to Komei Fukuda (cdd), Colin Jones (ESP) and Johan Löfberg (Yalmip) for allowing us to include their respective packages in the distribution. Thanks to their help we are able to say that MPT truly is an 'unpack-and-use' toolbox.



(a) The $N = 5$ step optimal feedback solution.



(b) N-step controller for the double integrator.



(c) Lyapunov function for the N-step controller.

Figure 20.1: Results obtained for Example 20.4.1.

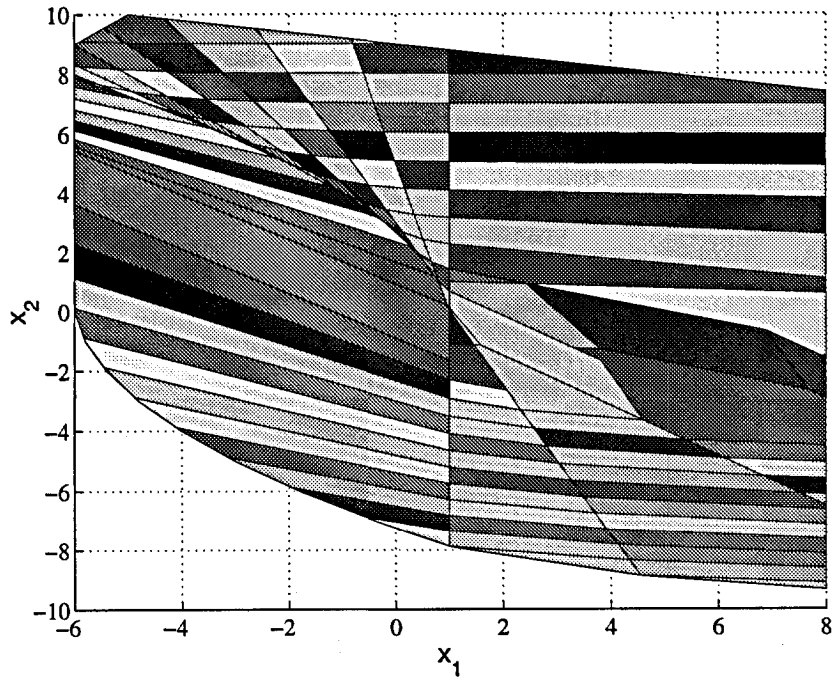


Figure 20.2: Controller partition obtained for Example 20.4.2.

Part VI

APPENDIX

Seite Leer /
Blank leaf

A

Publication List

Here, all my publications and technical reports are listed in chronological order:

- *Robust Adaptive Control of Hypnosis During Anesthesia*; P. Grieder, A. Gentilini, M. Morari and T. W. Schnider, International Conference of the IEEE Engineering in Medicine and Biology Society 2001, Istanbul, Turkey.
- *A Control Theoretic Approach to the U.S. National Airspace System*; A.M. Bayen, P. Grieder and C.J. Tomlin, AIAA Guidance, Navigation & Control Conference 2002, Monterey, USA.
- *Delay Predictive Models of the National Airspace System Using Hybrid Control Theory*; A.M. Bayen, P. Grieder, H. Sipma, G. Meyer and C.J. Tomlin; American Control Conference 2002, Alaska, USA.
- *Lagrangian delay predictive model for sector based air traffic flow*; A.M. Bayen, P. Grieder and C.J. Tomlin; To appear in the AIAA Journal on Guidance, Control and Dynamics.
- *Computation of the Constrained Infinite Time Linear Quadratic Regulator*; P. Grieder, F. Borrelli, F.D. Torrisi and M. Morari, American Control Conference 2003, Denver, USA. [GBTM03].
- *Analysis of Multi Parametric Quadratic Programming Algorithms*; K. Unneland, P. Grieder and F.D. Torrisi; Tech-Report AUT03-12, 2003; Automatic Control Lab, ETH, Switzerland. [UGT03].
- *Stability & Feasibility of Receding Horizon Control*; P. Grieder, M. Lüthi, P. Parillo and M. Morari; European Control Conference 2003, Cambridge, UK. [GLPM03].

- *Robust Receding Horizon Control - Analysis & Synthesis*; P. Grieder, P. Parillo and M. Morari; Conference on Decision and Control 2003, Hawaii, USA. [GPM03].

- *Complexity Reduction of Receding Horizon Control*; P. Grieder and M. Morari; Conference on Decision and Control 2003, Hawaii, USA. [GM03].

- *Robust Low Complexity Feedback Control of Constrained Linear Systems*, P. Grieder and M. Morari, submitted to journal 2004. [GM04].

- *Correction Note - The Explicit Linear Quadratic Regulator for Constrained Systems*, P. Grieder, M. Morari and A. Bemporad, Automatica, Volume 39, Issue 10, October 2003, Pages 1845–1846.

- *Computation of the Constrained Infinite Time Linear Quadratic Regulator*; P. Grieder, F. Borrelli, F.D. Torrisi and M. Morari, Automatica, Volume 40, Issue 4, April 2004, Pages 701–708. [GBTM04].

- *Multi Parametric Toolbox (MPT)*, M. Kvasnica, P. Grieder, M. Baotić and M. Morari, Hybrid Systems: Computation and Control Conference 2004, Pennsylvania, USA. [KGB04].

- *Two Level Model Predictive Control for the Maximum Control Invariant Set*; P. Grieder, Z. Wan, M. Kothare and M. Morari, American Control Conference 2004, Boston, USA. [GWKM04].

- *Using Interpolation to Simplify Explicit Model Predictive Control*; J.A. Rossiter and P. Grieder, American Control Conference 2004, Boston, USA. [RG04b].

- *Low Complexity Control of Piecewise Affine Systems with Stability Guarantee*, P. Grieder, M. Kvasnica, M. Baotić and M. Morari, American Control Conference 2004, Boston, USA. [GKBM04a].

- *Stabilizing Low Complexity Feedback Control of Constrained Piecewise Affine Systems*, P. Grieder, M. Kvasnica, M. Baotić and M. Morari, accepted for publication by Automatica. [GKBM04b].

- *Computation of Invariant Sets for Piecewise Affine Discrete Time Systems subject to Bounded Disturbances*, S. Raković, P. Grieder, M. Kvasnica, D.

Q. Mayne and M. Morari, Conference on Decision and Control 2004, Bahamas. [RGK+04a].

- *Efficient Computation of Controller Partitions in Multi-Parametric Programming*, R. Suard, J. Löfberg, P. Grieder, M. Kvasnica and M. Morari, Conference on Decision and Control 2004, Bahamas. [SLG+04].

- *Move Blocking Strategies in Receding Horizon Control*, R. Cagienard, P. Grieder, E.C. Kerrigan and M. Morari, Conference on Decision and Control 2004, Bahamas. [CGKM04].

- *Invariant Sets for Switched Discrete Time Systems subject to Bounded Disturbances*, P. Grieder, S. Raković, M. Morari, D. Q. Mayne, IFAC World Congress 2005, Prague, Czech Republic. [GRMM05].

- *Efficient implementation of constrained finite time optimal control of piecewise affine systems with linear performance indices*, M. Baric, M. Baotic, P. Grieder and M. Morari, IFAC World Congress 2005, Prague, Czech Republic. [BGBM05].

- *Approximations and Properties of the Disturbance Response Set of PWA Systems*, S. Raković and P. Grieder; Tech-Report AUT04-02, 2004; Automatic Control Lab, ETH, Switzerland. [RG04a].

- *Computation of Voronoi Diagrams and Delaunay Triangulation via Parametric Linear Programming*, S. Raković, P. Grieder and C. Jones; Tech-Report AUT04-03, 2004; Automatic Control Lab, ETH, Switzerland. [RGJ04].

- *A Logarithmic Solution to the Point Location Problem for Closed-Form Linear MPC*, C. Jones, P. Grieder and S. Raković, IFAC World Congress 2005, Prague, Czech Republic. [JGR04b].

- *A Survey on Stability Analysis of Discrete-Time Piecewise Affine Systems*, P. Biswas, P. Grieder, J. Löfberg, and M. Morari, IFAC World Congress 2005, Prague, Czech Republic. [BGLM05].

- *Using Interpolation to Simplify Explicit Model Predictive Control*; J.A. Rossiter and P. Grieder, Automatica, Volume 41, Issue 4, April 2005, Pages 637–643. [RG05].

Seite Leer /
Blank leaf

B

Curriculum Vitae

Pascal Grieder
Born on February 28th, 1977 in Basel, Switzerland

2002	–	2004	Doctorate Studies at the Automatic Control Laboratory, ETH Zürich. (Dr. sc.)
2002	–	2004	Graduate Studies in Business, ETH Zürich (Dipl. NDS ETHZ in Betriebswissenschaften)
2001	–	2002	Diploma project at Stanford University, California, USA
1997	–	2002	Undergraduate studies in Electrical Engineering, ETH Zürich (Dipl. El.-Ing. ETH)
1993	–	1996	Kantonsschule Alpenquai Luzern (Matura - Focus on Natural Sciences)

Seite Leer /
Blank leaf

Bibliography

[ABQ+99] ALLGÖWER, G., T.A. BADGWELL, S.J. QIN, J.B. RAWLINGS and S.J. WRIGHT: *Nonlinear Predictive Control and Moving Horizon Estimation - An Introduction Overview.* Advance in Control: Highlights of ECC'99, pages 391–449, 1999.

[AF96] AVIS, D. and K. FUKUDA: *Reverse search for enumeration.* Discrete Applied Mathematics, 65(1–3):21–46, March 1996.

[ALQ+02] AUBIN, J.-P., J. LYGEROS, M. QUINCAMPOIX, S. SASTRY, and N. SEUBE: *Impulse differential inclusions: A viability approach to hybrid systems.* IEEE Trans. on Automatic Control, 47(1):2–20, 2002.

[AM71] ANDERSON, B.D.O. and J.B. MOORE: *Linear Optimal Control.* Prentice–Hall Networks Series. Prentice–Hall, 1971.

[AM85] ADLER, I. and N. MEGIDDO: *A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension.* Journal of the ACM (Association for Computing Machinery), 32(4):871–895, 1985.

[AMN+98] ARYA, S., D.M. MOUNT, N.S. NETANYAHU, R. SILVERMAN and A.Y. WU: *An optimal algorithm for approximate nearest neighbor searching fixed dimensions.* Journal of the ACM, 45(6):891–923, 1998.

[Aub91] AUBIN, J. P.: *Viability theory.* Systems & Control: Foundations & Applications. Birkhäuser, 1991.

[Aur87] AURENHAMMER, F.: *A Criterion for the Affine Equivalence of Cell Complexes in $\mathbb{R}^d$ and Convex Polyhedra in $\mathbb{R}^{d+1}$.* Discrete and Computational Geometry, 2:49–64, 1987.

[Aur91] AURENHAMMER, F.: *Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure.* ACM Computing Surveys, 23(3), September 1991.

[Avi00] AVIS, D.: *lrs: A revised implementation of the reverse search vertex enumeration algorithm.* Polytopes, Combinatorics and Computation, pages 177–198, 2000.

[Bal98] BALAS, E.: *Projection with a minimum system of inequalities.* Computational Optimization and Applications, 10:189–193, 1998.

[Bao02] BAOTIĆ, M.: *An Efficient Algorithm for Multi-Parametric Quadratic Programming.* Technical Report AUT02–04, Automatic Control Laboratory, ETH Zurich, Switzerland, February 2002. http://control.ee.ethz.ch.

[BBBM01] BORRELLI, F., M. BAOTIĆ, A. BEMPORAD and M. MORARI: *Efficient On-Line Computation of Constrained Optimal Control.* In *Proc. 40th IEEE Conf. on Decision and Control*, Orlando, Florida, December 2001.

[BBBM03] BORRELLI, F., M. BAOTIĆ, A. BEMPORAD and M. MORARI: *An Efficient Algorithm for Computing the State Feedback Optimal Control Law for Discrete Time Hybrid Systems.* In *Proc. of the American Control Conference*, Denver, Colorado, USA, June 2003.

[BBM00a] BEMPORAD, A., F. BORRELLI and M. MORARI: *Explicit Solution of Constrained $1/\infty$-Norm Model Predictive Control.* In *Proc. 39th IEEE Conf. on Decision and Control*, December 2000.

[BBM00b] BEMPORAD, A., F. BORRELLI and M. MORARI: *Explicit Solution of LP-Based Model Predictive Control.* In *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.

[BBM00c] BEMPORAD, A., F. BORRELLI and M. MORARI: *Optimal Controllers for Hybrid Systems: Stability and Piecewise Linear Explicit Form.* In *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.

[BBM03] BEMPORAD, A., F. BORRELLI and M. MORARI: *Min-max control of constrained uncertain discrete-time linear systems.* IEEE Trans. Automatic Control, 48(9):1600 – 1606, 2003.

[BCLK03] BACIC, M., M. CANNON, Y. I. LEE and B. KOUVARITAKIS: *General Interpolation MPC and Its Advantages.* IEEE Trans. Automatic Control, 48(6):1092–1096, 2003.

[BCM03a] BAOTIĆ, M., F. J. CHRISTOPHERSEN and M. MORARI: *A new Algorithm for Constrained Finite Time Optimal Control of Hybrid Systems with a Linear Performance Index.* In *European Control Conference*, Cambridge, UK, September 2003.

[BCM03b] BAOTIĆ, M., F. J. CHRISTOPHERSEN and M. MORARI: *Infinite Time Optimal Control of Hybrid Systems with a Linear Performance Index.* In *Proc. of the Conf. on Decision and Control, Maui, Hawaii, USA*, December 2003.

[BDH96] BARBER, C. B., D. P. DOBKIN and H. T. HUHDANPAA: *The Quickhull algorithm for convex hulls.* ACM Trans. on Mathematical Software, 22(4):469–483, December 1996. http://www.qhull.org.

[Bem04] BEMPORAD, A.: *Efficient Conversion of Mixed Logical Dynamical Systems into an Equivalent Piecewise Affine Form.* IEEE Trans. Automatic Control, 49(5):832–838, May 2004.

[Ber71] BERTSEKAS, D. P.: *Control of Uncertain Systems with a set-membership description of the uncertainty.* PhD thesis, MIT, 1971.

[Ber72] BERTSEKAS, D.P.: *Infinite-time reachability of state-space regions by using feedback control.* IEEE Trans. Automatic Control, 17:604–613, October 1972.

[Ber95] BERTSEKAS, D.P.: *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, 1995.

[BF01] BEMPORAD, A. and C. FILIPPI: *Suboptimal explicit MPC via approximate multiparametric quadratic programming.* In *Proc. 40th IEEE Conf. on Decision and Control*, Orlando, Florida, USA, 2001.

[BFT01] BEMPORAD, A., K. FUKUDA and F.D. TORRISI: *Convexity Recognition of the Union of Polyhedra.* Computational Geometry, 18:141–154, April 2001.

[BFTM00] BEMPORAD, A., G. FERRARI-TRECATE and M. MORARI: *Observability and Controllability of Piecewise Affine and Hybrid Systems.* IEEE Trans. Automatic Control, 45(10):1864–1876, 2000.

[BGBM05] BARIĆ, M., P. GRIEDER, M. BAOTIĆ and M. MORARI: *Optimal Control of PWA Systems by Exploiting Problem Structure.* In *Proceedings of 16th IFAC World Congress*, July 2005.

[BGFB94] BOYD, S., L. EL GHAOUI, E. FERON and V. BALAKRISHNAN: *Linear Matrix Inequalities in System and Control Theory.* Studies in Applied Mathematics. SIAM, 1994.

[BGLM05] BISWAS, P., P. GRIEDER, J. LÖFBERG and M. MORARI: *A Survey on Stability Analysis of Discrete-Time Piecewise Affine Systems.* In *Proceedings of 16th IFAC World Congress*, July 2005.

[BGT00] BLONDEL, V.D., S. GAUBERT and J.N. TSITSIKLIS: *Approximating the spectral radius of sets of matrices in the max-algebra is NP-hard.* IEEE Trans. Automatic Control, 45(9):1762–1765, September 2000.

[Bit88] BITSORIS, G.: *On the positive invariance of polyhedral sets for discrete–time systems.* Systems & Control Letters, 11:243–248, 1988.

[Bla92] BLANCHINI, F.: *Minimum-Time Control for Uncertain Discrete-Time Linear Systems.* In *Proc. 31st IEEE Conf. on Decision and Control*, pages 2629–2634, Tucson, Arizona, USA, December 1992.

[Bla93] BLANCHINI, F.: *Robust control for uncertain linear systems via polyhedral Lyapunov functions.* Proc. 32rd IEEE Conf. on Decision and Control, 3:2592–2593, 1993.

[Bla94] BLANCHINI, F.: *Ultimate boundedness control for uncertain discrete–time systems via set–induced Lyapunov functions.* IEEE Trans. Automatic Control, 39(2):428–433, February 1994.

[Bla95] BLANCHINI, F.: *Nonquadratic Lyapunov functions for robust control.* Automatica, 31(3):451–461, March 1995.

[Bla99] BLANCHINI, F.: *Set invariance in control — A survey.* Automatica, 35(11):1747–1767, November 1999.

[BM99a] BEMPORAD, A. and M. MORARI: *Control of Systems Integrating Logic, Dynamics, and Constraints.* Automatica, 35(3):407–427, March 1999.

[BM99b] BEMPORAD, A. and M. MORARI: *Robust Model Predictive Control: A Survey.* In GARULLI, A., A. TESI and A. VICINO (editors): *Robustness in Identification and Control*, number 245 in *Lecture Notes in Control and Information Sciences*, pages 207–226. Springer–Verlag, 1999.

[BMDP02] BEMPORAD, A., M. MORARI, V. DUA and E.N. PISTIKOPOULOS: *The Explicit Linear Quadratic Regulator for Constrained Systems.* Automatica, 38(1):3–20, January 2002.

[Bon83] BONEH, A.: *Redundancy in Mathematical Programming*, volume 206 of *Lecture Notes in Economics and Mathematical Systems*, chapter PREDUCE - A Probabilistic Algorithm Identifying Redundancy by a Random Feasible Point Generator (RFPG). Springer-Verlag, 1983.

[Bor03] BORRELLI, F.: *Constrained Optimal Control Of Linear And Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences.* Springer, 2003.

[BR71] BERTSEKAS, D. P. and I. B. RHODES: *On the Minimax Reachability of Target Sets and Target Tubes.* Automatica, 7:233–247, 1971.

[BT03] BAOTIĆ, M. and F.D. TORRISI: *Polycover.* Technical Report AUT03-11, Automatic Control Lab, ETHZ, Switzerland, 2003. http://control.ee.ethz.ch.

[BV04] BOYD, S. and L. VANDENBERGHE: *Convex Optimization.* Cambridge University Press, 2004. http://www.stanford.edu/class/ee364/.

[BZ00] BRANICKY, M.S. and G. ZHANG: *Solving Hybrid Control Problems: Level Sets and Behavioral Programming.* In *Proc. American Contr. Conf.*, Chicago, Illinois USA, June 2000.

[CB99] CAMACHO, E.F. and C. BORDÒNS: *Model Predictive Control.* Springer-Verlag, London, 1999.

[Cer63] CERNIKOV, S.N.: *Contraction of finite systems of linear inequalities (in russian).* Doklady Akademiia Nauk SSSR, 152(5):1075–1078, 1963. (English translation in Societ Mathematics Doklady, Vol. 4, No. 5 (1963), pp.1520–1524).

[CGKM04] CAGIENARD, R., P. GRIEDER, E.C. KERRIGAN and M. MORARI: *Move Blocking Strategies in Receding Horizon Control.* In *Proc. 43th IEEE Conf. on Decision and Control*, Bahamas, December 2004.

[CKD03] CANNON, M., B. KOUVARITAKIS and V. DESHMUKH: *Enlargement of Polytopic Terminal Region in NMPC by Interpolation and Partial Invariance.* In *Proc. of the American Control Conference*, Denver, Colorado, USA, June 2003.

[CM86] CAMPO, P. J. and M. MORARI: *∞-norm Formulation of Model Predictive Control Problems.* In *Proc. of the American Control Conference*, pages 339–343, Seattle, Washington, 1986.

[CM96] CHMIELEWSKI, D. and V. MANOUSIOUTHAKIS: *On constrained infinite-time linear quadratic optimal control.* Systems & Control Letters, 29(3):121–130, November 1996.

[CZ99a] CHISCI, L. and G. ZAPPA: *Fast algorithm for a constrained infinite horizon LQ problem.* Int. J. Control, 72(11):1020–1026, August 1999.

[CZ99b] CHISCI, L. and G. ZAPPA: *Robustifying a predictive controller against persistent disturbances.* Proc. of the European Control Conference, 1999.

[dH94] HERTOG, D. DEN: *Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity.* Mathematics and Its Applications. Kluwer Academic Publishers, 1994.

[dM00] DE OLIVEIRA KOTHARE, S.L. and M. MORARI: *Contractive Model Predictive Control for Constrained Nonlinear Systems.* IEEE Trans. Automatic Control, 45(6):1052–1071, June 2000.

[DP00] DUA, V. and E.N. PISTIKOPOULOS: *An algorithm for the solution of multiparametric mixed integer linear programming problems.* Annals of Operations Research, 99:123–139, 2000.

[Fen02] FENG, G.: *Stability Analysis of Piecewise Discrete-Time Linear Systems.* IEEE Trans. on Automatic Control, 47(7):1108–1112, 2002.

[FLL00] FUKUDA, K., T. M. LIEBLING and CHRISTINE LÜTOLF: *Extended convex hull.* In *Proc. 12th Canadian Conference on Computational Geometry*, page 5764, July 2000.

[FP96] FUKUDA, K. and A. PRODON: *Double description method revisited.* Combinatorics and Computer Science, 1120:91111, 1996. ftp://ftp.ifor.math.ethz.ch/pub/fukuda/reports/ddrev960315.ps.gz.

[FTCMM02] FERRARI-TRECATE, G., F. A. CUZZOLA, D. MIGNONE and M. MORARI: *Analysis of discrete-time piecewise affine and hybrid systems.* Automatica, 38(12):2139–2146, 2002.

[Fuk04a] FUKUDA, K.: *Cdd/cdd+ Reference Manual*, June 2004. http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html.

[Fuk04b] FUKUDA, K.: *From the Zonotope Construction to the Minkowski Addition of Convex Polytopes.* Journal of Symbolic Computation, 38:1261–1272, May 2004.

[Fuk04c] FUKUDA, K.: *Polyhedral computation FAQ*, June 2004. http://www.ifor.math.ethz.ch/staff/fukuda/polyfaq/polyfaq.html.

[Gal95] GAL, T.: *Postoptimal Analyses, Parametric Programming, and Related Topics.* de Gruyter, Berlin, 2nd ed. edition, 1995.

[GBTM03] GRIEDER, P., F. BORRELLI, F.D. TORRISI and M. MORARI: *Computation of the Constrained Infinite Time Linear Quadratic Regulator.* In *Proc. of the American Control Conference*, Denver, Colorado, USA, June 2003.

[GBTM04] GRIEDER, P., F. BORRELLI, F.D. TORRISI and M. MORARI: *Computation of the Constrained Infinite Time Linear Quadratic Regulator.* Automatica, 40:701–708, April 2004.

[GKBM03] GRIEDER, P., M. KVASNICA, M. BAOTIĆ and M. MORARI: *Low Complexity Control of Piecewise Affine Systems with Stability Guarantee.* Technical Report AUT03-13, Automatic Control Lab, ETHZ, Switzerland, 2003. http://control.ee.ethz.ch.

[GKBM04a] GRIEDER, P., M. KVASNICA, M. BAOTIĆ and M. MORARI: *Low Complexity Control of Piecewise Affine Systems with Stability Guarantee.* In *Proc. of the American Control Conference*, Boston, USA, June 2004.

[GKBM04b] GRIEDER, P., M. KVASNICA, M. BAOTIĆ and M. MORARI: *Stabilizing Low Complexity Feedback Control of Constrained Piecewise Affine Systems.* Accepted for publication by Automatica, 2004.

[GLPM03] GRIEDER, P., M. LÜTHI, P. PARILLO and M. MORARI: *Stability & Feasibility of Receding Horizon Control.* In *European Control Conference*, Cambridge, UK, September 2003.

[GM03] GRIEDER, P. and M. MORARI: *Complexity Reduction of Receding Horizon Control.* In *Proc. 42th IEEE Conf. on Decision and Control*, Maui, Hawaii, USA, December 2003.

[GM04] GRIEDER, P. and M. MORARI: *Robust Low Complexity Feedback Control of Constrained Linear Systems.* submitted to journal, 2004.

[GN93] GENCELI, H. and M. NIKOLAOU: *Robust stability analysis of constrained ℓ_1-norm model predictive control.* AIChE J., 39(12):1954–1965, 1993.

[Gon97] GONDZIO, J.: *Presolve Analysis of Linear Programs Prior to Applying an Interior Point Method.* OSRA Journal on Computing, 9(1):73–91, 1997.

[GPM89] GARCIA, C.E., D.M. PRETT and M. MORARI: *Model Predictive Control: Theory and Practive - A Survey.* Automatica, 25(3):335–348, 1989.

[GPM03] GRIEDER, P., P. PARILLO and M. MORARI: *Robust Receding Horizon Control - Analysis & Synthesis.* In *Proc. 42th IEEE Conf. on Decision and Control*, Maui, Hawaii, USA, December 2003.

[GRMM05] GRIEDER, P., S. V. RAKOVIĆ, M. MORARI and D. Q. MAYNE: *Invariant Sets for Switched Discrete Time Systems subject to Bounded Disturbances.* In *Proceedings of 16th IFAC World Congress*, July 2005.

[Grü00] GRÜNBAUM, B.: *Convex Polytopes.* Springer-Verlag, Second edition, 2000.

[GS93] GRITZMANN, P. and B. STURMFELS: *Minkowski Addition of Polytopes: Computational Complexity and Applications to Gröbner Bases.* SIAM J. Discrete Math., 6:246–269, 1993.

[GT91] GILBERT, E. G. and K. T. TAN: *Linear systems with state and control constraints: the theory and applications of maximal output admissible sets.* IEEE Trans. Automatic Control, 36(9):1008–1020, 1991.

[GTM03] GEYER, T., F.D. TORRISI and M. MORARI: *Efficient Mode Enumeration of Compositional Hybrid Models.* In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, Volume 2623, pages 216–232. Springer Verlag, March 2003.

[GTM04] GEYER, T., F.D. TORRISI and M. MORARI: *Optimal Complexity Reduction of Piecewise Affine Models Based on Hyperplane Arrangements.* In *Proc. of the American Control Conference*, Boston, Massachusetts, 2004.

[GWKM04] GRIEDER, P., Z. WAN, M. KOTHARE and M. MORARI: *Two Level Model Predictive Control for the Maximum Control Invariant Set.* In *American Control Conference*, Boston, Massachusetts, June 2004.

[HSB01] HEEMELS, W.P.M.H., B. DE SCHUTTER and A. BEMPORAD: *Equivalence of Hybrid Dynamical Models.* Automatica, 37(7):1085–1091, July 2001.

[ILO03] ILOG, INC.: *CPLEX 8.0 User Manual.* Gentilly Cedex, France, 2003. http://www.ilog.fr/products/cplex/.

[JG03] JOHANSEN, T.A. and A. GRANCHAROVA: *Approximate explicit constrained linear model predictive control via orthogonal search tree.* IEEE Trans. Automatic Control, 48:810–815, May 2003.

[JGR04a] JONES, C., P. GRIEDER and S. V. RAKOVIĆ: *A Logarithmic Solution to the Point Location Problem for Closed-Form Linear MPC.* Technical Report AUT04-03, Automatic Control Lab, ETHZ, Switzerland, May 2004. http://control.ee.ethz.ch/.

[JGR04b] JONES, C., P. GRIEDER and S. V. RAKOVIĆ: *A Logarithmic Solution to the Point Location Problem for Closed-Form Linear MPC.* In *Proceedings of 16th IFAC World Congress*, July 2004.

[JKM04] JONES, C.N., E.C. KERRIGAN and J.M. MACIEJOWSKI: *Equality Set Projection: A new algorithm for the projection of polytopes in halfspace representation.* Technical Report CUED Technical Report CUED/F-INFENG/TR.463, Department of Engineering, Cambridge University, UK, 2004. http://www-control.eng.cam.ac.uk/~cnj22/.

[Joh02] JOHANSSON, M.: *Piecewise Linear Control Systems - A Computational Approach*, volume 284 of *Lecture Notes in Control and Information Sciences.* Springer, 2002.

[Jon04] JONES, C.N.: *Equality Set Projection Algorithm: Download.* Cambridge University, UK, 2004. http://www-control.eng.cam.ac.uk/~cnj22/.

[JR98] JOHANSSON, M. and A. RANTZER: *Computation of piece-wise quadratic Lyapunov functions for hybrid systems.* IEEE Trans. Automatic Control, 43(4):555–559, 1998.

[JvdS02] JULIUS, A.A. and A. J. VAN DER SCHAFT: *The maximal controlled invariant set of switched linear systems.* In *Proc. of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002.

[KA02] KOUTSOUKOS, X. D. and P. J. ANTSAKLIS: *Design of stabilizing switching control laws for discrete and continuous–time linear systems*

using piecewise linear Lyapunov functions. Int. J. Control, 75(12):932–945, 2002.

[KBM96] KOTHARE, M. V., V. BALAKRISHNAN and M. MORARI: *Robust constrained model predictive control using linear matrix inequalities.* Automatica, 32(10):1361–1379, 1996.

[KC01] KOUVARITAKIS, B. and M. CANNON: *Nonlinear Predictive Control: Theory and Practice.* The Institution of Electrical Engineers, London, UK, 2001.

[Ker00] KERRIGAN, E. C.: *Robust Constraints Satisfaction: Invariant Sets and Predictive Control.* PhD thesis, Department of Engineering, The University of Cambridge, Cambridge, England, 2000.

[KG87] KEERTHI, S.S. and E.G. GILBERT: *Computation of Minimum-Time Feedback Control Laws for Discrete-Time Systems with State-Control Constraints.* IEEE Trans. Automatic Control, AC-32:432–435, May 1987.

[KG88] KEERTHI, S. S. and E. G. GILBERT: *Optimal infinite-horizon feedback control laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations.* J. Opt. Theory and Applications, 57:265–293, 1988.

[KG98] KOLMANOVSKY, I. and E. G. GILBERT: *Theory and Computation of Disturbance Invariant Sets for Discrete-Time Linear Systems.* Mathematical Problems in Egineering, 4:317–367, 1998.

[KGB04] KVASNICA, M., P. GRIEDER and M. BAOTIĆ: *Multi Parametric Toolbox (MPT)*, 2004. http://control.ee.ethz.ch/~mpt/.

[Kha96] KHALIL, HASSAN K.: *Nonlinear Systems.* Prentice Hall, 2nd edition, 1996.

[KM02] KERRIGAN, E. C. and D. Q. MAYNE: *Optimal control of constrained, piecewise affine systems with bounded disturbances.* In *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002.

[KM03] KERRIGAN, E. C. and J. M. MACIEJOWSKI: *On robust optimization and the optimal control of constrained linear systems with bounded state disturbances.* In *In Proc. 2003 European Control Conference*, Cambridge, UK, September 2003.

[KM04a] KERRIGAN, E. C. and J. M. MACIEJOWSKI: *Feedback min-max model predictive control using a single linear program: Robust stability and the explicit solution.* Int. J. Robust Nonlinear Control, 14:395–413, 2004.

[KM04b] KERRIGAN, E.C. and J.M. MACIEJOWSKI: *Properties of a new parameterization for the control of constrained systems with disturbances.* In *Proc. of the American Control Conference*, Boston, USA, June 2004.

[KS90] KEERTHI, S. S. and K. SRIDHARAN: *Solution of parametrized linear inequalities by fourier elimination and its applications.* J. Opt. Theory and Applications, 65(1):161–169, 1990.

[LHWB04] LAZAR, M., W.P.M.H. HEEMELS, S. WEILAND and A. BEMPORAD: *Stabilization conditions for model predictive control of constrained PWA systems.* In *Proc. 43th IEEE Conf. on Decision and Control*, Bahamas, December 2004.

[Löf03] LÖFBERG, J.: *Minimax approaches to robust model predictive control.* PhD thesis, Linköping University, Linköping, Sweden, 2003. Diss. No. 812.

[Löf04] LÖFBERG, J.: *YALMIP : A Toolbox for Modeling and Optimization in MATLAB.* In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. Available from http://control.ee.ethz.ch/~joloef/yalmip.php.

[LR03] LINCOLN, B. and A. RANTZER: *Relaxed Optimal Control of Piecewise Linear Systems.* In *IFAC Conference on Analysis and Design of Hybrid Systems*, Saint-Malo, France, 2003.

[LS95] LEWIS, F.L. and V.L. SYRMOS: *Optimal Control - Second Edition.* John Wiley & Sons, 1995.

[LTS99] LYGEROS, J., C. TOMLIN and S. SASTRY: *Controllers for reachability specifications for hybrid systems.* Automatica, 35(3):349–370, 1999.

[MA98] MOUNT, D. and S. ARYA: *ANN: Library for Approximate Nearest Neighbor Searching*, June 1998. http://www.cs.umd.edu/~mount/ANN/.

[Mac02] MACIEJOWSKI, J.M.: *Predictive Control with Constraints.* Prentice Hall, 2002.

[Mak01] MAKHORIN, A.: *GLPK - GNU Linear Programming Kit*, 2001. http://www.gnu.org/directory/libs/glpk.html.

[May01] MAYNE, D. Q.: *Control of Constrained Dynamic Systems.* European Jornal of Control, 7:87–99, 2001.

[MFTM00] MIGNONE, D., G. FERRARI-TRECATE and M. MORARI: *Stability and stabilization of piecewise affine and hybrid systems: An LMI approach.* In *Proc. 39th IEEE Conf. on Decision and Control*, December 2000.

[Mig02] MIGNONE, D.: *Control and Estimation of Hybrid systems via Mathematical Optimization.* PhD thesis, Automatic Control Labotatory – ETH, Zurich, 2002.

[Mit99] MITTELMANN, D.: *Benchmarking Interior Point LP/QP Solvers.* Opt. Methods & Software, 12:655–670, 1999.

[Mit04] MITTELMANN, H.: *Benchmarks for Optimization Software*, 2004. http://plato.asu.edu/topics/benchm.html.

[MKR00] MENDEZ, J. A., B. KOUVARITAKIS and J. A. ROSSITER: *State Space approach to interpolation in MPC.* International journal of robust nonlinear control, 10:27–38, 2000.

[ML99] MORARI, M. and J.H. LEE: *Model predictive control: past, present and future.* Computers & Chemical Engineering, 23(4–5):667–682, 1999.

[MR02] MAYNE, D. Q. and S. RAKOVIĆ: *Optimal control of constrained piecewise affine discrete-time systems using reverse transformation.* In *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002.

[MR03] MAYNE, D. Q. and S. RAKOVIĆ: *Model predictive control of constrained piecewise affine discrete-time systems.* Int. J. of Robust and Nonlinear Control, 13(3):261–279, April 2003.

[MRRS00] MAYNE, D. Q., J.B. RAWLINGS, C.V. RAO and P.O.M. SCOKAERT: *Constrained model predictive control: Stability and Optimality.* Automatica, 36(6):789–814, June 2000.

[MRTT53] MOTZKIN, T.S., H. RAIFFA, G.L. THOMPSON and R.M. THRALL: *The double description method.* Contributions to the Theory of Games, II(28):51–73, 1953.

[MS97] MAYNE, D. Q. and W. R. SCHROEDER: *Robust Time-Optimal Control of Constrained Linear Systems.* Automatica, 33(12):2103–2118, 1997.

[Neu04] NEUMAIER, A.: *Complete Search in Continuous Global Optimization and Constraint Satisfaction.* In ISERLES, A. (editor): *Acta Numerica*, Lecture Notes in Control and Information Sciences. Cambridge University Press, 2004.

[Num02] NUMERICAL ALGORITHMS GROUP, LTD.: *NAG Foundation Toolbox for MATLAB 6.* Oxford, UK, 2002. http://www.nag.co.uk/.

[OSS95] OTTMANN, TH., S. SCHUIERER and S. SOUNDARALAKSHMI: *Enumerating extreme points in higher dimensions.* In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 900, pages 562–570, 1995.

[Par03] PARRILO, P.A.: *Semidefinite programming relaxations for semialgebraic problems.* Mathematical Programming Ser. B, 96(2):293–320, 2003.

[PK03] PANNOCCHIA, G. and E.C. KERRIGAN: *Offset-free control of constrained linear discrete-time systems subject to persistent unmeasured disturbances.* In *42nd Conference on Decision and Control*, Maui, Hawaii, USA, December 2003.

[PP03] PRAJNA, S. and A. PAPACHRISTODOULOU: *Analysis of Switched and Hybrid Systems - Beyond Piecewise Quadratic Methods.* In *American Control Conference*, Denver, Colorado, USA, June 2003.

[PPSP04] PRAJNA, S., A. PAPACHRISTODOULOU, P. SEILER and P.A. PARRILO: *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004.

[PY93] POLAK, E. and T.H. YANG: *Moving horizon control of linear systems with input saturation and plant uncertainty: Part 1 & 2.* Int. J. Control, 58(3):613–663, 1993.

[QB97] QIN, S.J. and T.A. BADGWELL: *An overview of industrial model predictive control technology.* In *Chemical Process Control – V*, volume 93, no. 316, pages 232–256. AIChe Symposium Series – American Institute of Chemical Engineers, 1997.

[Raw00] RAWLINGS, J. B.: *Tutorial Overview of Model Predictive Control.* IEEE Control Sys. Magazine, 20(3):38–52, June 2000.

[RBC03] ROSSITER, J. A., B.KOUVARITAKIS and M. CANNON: *Stability proof for computationally efficient predictive control in the uncertain case.* In *Proc. of the American Control Conference*, Denver, Colorado, USA, June 2003.

[RBW90] R.R. BITMEAD, M. GEVERS and V. WERTZ: *Adaptive Optimal Control, The Thinking Man's GPC.* Prentice Hall, 1990.

[RG04a] RAKOVIĆ, S. V. and P. GRIEDER: *Approximations and Properties of the Disturbance Response Set of PWA Systems.* Technical Report AUT04-02, Automatic Control Lab, ETHZ, Switzerland, 2004. http://control.ee.ethz.ch/.

[RG04b] ROSSITER, J. A. and P. GRIEDER: *Using Interpolation to Simplify Explicit Model Predictive Control.* In *American Control Conference*, Boston, Massachusetts, June 2004.

[RG05] ROSSITER, J. A. and P. GRIEDER: *Using Interpolation to Simplify Explicit Model Predictive Control.* Automatica, 41(4):637–643, April 2005.

[RGJ04] RAKOVIĆ, S. V., P. GRIEDER and C. JONES: *Computation of Voronoi Diagrams and Delaunay Triangulation via Parametric Linear Programming.* Technical Report AUT04-03, Automatic Control Lab, ETHZ, Switzerland, 2004. http://control.ee.ethz.ch/.

[RGK+04a] RAKOVIĆ, S. V., P. GRIEDER, M. KVASNICA, D. Q. MAYNE and M. MORARI: *Computation of Invariant Sets for Piecewise Affine Discrete Time Systems subject to Bounded Disturbances.* In *IEEE Conference on Decision and Control*, December 2004.

[RGK+04b] RAKOVIĆ, S. V., P. GRIEDER, M. KVASNICA, D. Q. MAYNE and M. MORARI: *Computation of Invariant Sets for Piecewise Affine Discrete Time Systems subject to Bounded Disturbances.* Technical Report EEE/C&P/SVR/tbf, Imperial College London, UK, 2004.

[RKC01] ROSSITER, J. A., B. KOUVARITAKIS and M. CANNON: *Computationally efficient algorithms for constraint handling with guaranteed stability and near optimality.* Int. J. Control, 74(17):1678–1689, 2001.

[RKKM03] RAKOVIĆ, S. V., E. C. KERRIGAN, K. I. KOURAMAS and D. Q. MAYNE: *Approximation of the minimal robustly positive invariant set for discrete-time LTI systems with persistent state disturbances.* In *Proc. of the Conf. on Decision and Control, Maui, Hawaii, USA*, pages 3917–3918, December 2003.

[RKM03] RAKOVIĆ, S. V., E. C. KERRIGAN and D. Q. MAYNE: *Reachability computations for constrained discrete-time systems with state- and input-dependent disturbances.* In *Proc. of the Conf. on Decision and Control, Maui, Hawaii, USA*, pages 3905–3910, December 2003.

[RM93] RAWLINGS, J.B. and K.R. MUSKE: *The stability of constrained receding-horizon control.* IEEE Trans. Automatic Control, 38:1512–1516, 1993.

[Ros03] ROSSITER, J. A.: *Model-based predictive control, a practical approach.* CRC Press, 2003.

[RR00] RAO, C.V. and J.B. RAWLINGS: *Linear programming and model predictive control.* J. Process Control, 10:283–289, 2000.

[RRK91] ROSSITER, J. A., M.J. RICE and B. KOUVARITAKIS: *A numerically robust state-space approach to stable predictive control strategies.* Automatica, 38(1):65–73, 1991.

[Ryb99] RYBNIKOV, K.: *Stresses and Liftings of Cell Complexes.* Discrete and Computational Geometry, 21(4):481 – 517, June 1999.

[Sak04] SAKIZLIS, V.: *Design of Model Based Controllers via Parametric Programming.* PhD thesis, Imperial College, London, UK, March 2004.

[Sch87] SCHECHTER, M.: *Polyhedral functions and multiparametric linear programming.* Journal of Optimization Theory and Applications, 53(2):269–280, May 1987.

[SD87] SZNAIER, M. and M.J. DAMBORG: *Suboptimal control of linear systems with state and control inequality constraints.* In *Proc. 26th IEEE Conf. on Decision and Control*, volume 1, pages 761–762, Los Angeles, CA, December 1987.

[Sei86] SEIDEL, R.: *Constructing higher-dimensional convex hulls at logarithmic cost per face.* In *In Proc. of the 18th ACM Symposium on the Theory of Computing*, pages 404–413, 1986.

[Ser88] SERRA, J.: *Image Analysis and Mathematical Morphology, Vol II: Theoretical advances.* Academic Press, 1988.

[SLG+04] SUARD, R., J. LÖFBERG, P. GRIEDER, M. KVASNICA and M. MORARI: *Efficient Computation of Controller Partitions in Multi-Parametric Programming.* In *Proc. 43th IEEE Conf. on Decision and Control*, Bahamas, December 2004.

[Son81] SONTAG, E.D.: *Nonlinear regulation: The piecewise linear approach.* IEEE Trans. Automatic Control, 26(2):346–358, April 1981.

[Son96] SONTAG, E.D.: *Interconnected automata and linear systems: A theoretical framework in discrete-time.* In ALUR, R., T.A. HENZINGER and E.D. SONTAG (editors): *Hybrid Systems III – Verification and Control*, number 1066 in *Lecture Notes in Computer Science*, pages 436–448. Springer–Verlag, 1996.

[SP96] SKOGESTAD, S. and I. POSTLETHWAITE: *Multivariable Feedback Control.* John Wiley & Sons, 1996.

[SR98] SCOKAERT, P.O.M. and J.B. RAWLINGS: *Constrained linear quadratic regulation.* IEEE Trans. Automatic Control, 43(8):1163–1169, August 1998.

[Stu99] STURM, J.F.: *Using SeDuMi 1.02, A MATLAB Toolbox for Optimization over Symmetric Cones.* Optimization Methods and Software, 11-12(1-4):625–653, October 1999.

[TB04] TORRISI, F.D. and A. BEMPORAD: *HYSDEL — A Tool for Generating Computational Hybrid Models.* IEEE Trans. Contr. Sys. Techol., 12:235–249, March 2004. http://control.ee.ethz.ch/~hybrid/hysdel/hysdel.msql.

[The03] THE MATHWORKS, INC.: *MATLAB Users Manual.* Natick, MA, US, 2003. http://www.mathworks.com.

[TJ02] TØNDEL, P. and T.A. JOHANSEN: *Complexity reduction in explicit model predictive control.* In *IFAC World Congress*, Spain, Barcelona, 2002.

[TJB01] TØNDEL, P., T.A. JOHANSEN and A. BEMPORAD: *An Algorithm for Multi–Parametric Quadratic Programming and Explicit MPC Solutions.* In *Proc. 40th IEEE Conf. on Decision and Control*, Orlando, Florida, December 2001.

[TJB03a] TØNDEL, P., T.A. JOHANSEN and A. BEMPORAD: *An algorithm for multiparametric quadratic programming and explicit MPC solutions.* Automatica, 39(3):489–497, 2003.

[TJB03b] TØNDEL, P., T.A. JOHANSEN and A. BEMPORAD: *Evaluation of piecewise affine control via binary search tree.* Automatica, 39(5):945–950, 2003.

[TJB03c] TØNDEL, P., T.A. JOHANSEN and A. BEMPORAD: *Further results on Multi-parametric quadratic programming.* In *Proceedings 42nd IEEE*

Conference on Decision and Control, Maui, Hawaii, USA, December 2003.

[TLS00] TOMLIN, C.J., J. LYGEROS and S. S. SASTRY: *A Game Theoretic Approach to Controller Design for Hybrid Systems.* Proc. of the IEEE, 88, July 2000.

[Tøn00] TØNDEL, P.: *Constrained Optimal Control via Multiparametric Quadratic Programming.* PhD thesis, Department of Engineering Cybernetics, NTNU, Trondheim, Norway, 2000.

[Tor03] TORRISI, F.D.: *Modeling and Reach-Set Computation for Analysis and Optimal Control of Discrete Hybrid Automata.* PhD thesis, Automatic Control Laboratory, ETH Zurich, 2003.

[TTT99] TOH, K. C., M. J. TODD and R. H. TÜTÜNCÜ: *SDPT3 - a Matlab software package for semidefinite programming, version 2.1.* Optimization Methods and Software, 11-12(1-4):545–581, 1999.

[UGT03] UNNELAND, K., P. GRIEDER and F.D. TORRISI: *Analysis of Multi Parametric Quadratic Programming Algorithms at the Automatic Control Lab.* Technical Report AUT03-12, Automatic Control Lab, ETHZ, Switzerland, 2003.

[Van01] VANDERBEI, R.J.: *Linear Programming: Foundations and Extensions.* Kluwer Academic Publishers, second edition, 2001. http://www.princeton.edu/~rvdb/LPbook/onlinebook.pdf.

[VB96] VANDENBERGHE, L. and S. BOYD: *Semidefinite programming.* SIAM Review, 38(1):49–95, March 1996.

[Ver03] VERES, S. M.: *Geometric Bounding Toolbox (GBT) for* MATLAB. http://www.sysbrain.com, 2003.

[Vid93] VIDYASAGAR, M.: *Nonlinear Systems Analysis.* Prentice Hall, 2nd edition, 1993.

[Wei] WEISSTEIN, ERIC W.: *MathWorld-A Wolfram Web Resource.* http://mathworld.wolfram.com/.

[WK03a] WAN, Z. and M. V. KOTHARE: *Robust model predictive control with a time varying terminal constraint set.* Systems and Control Letters, 48(5):375–383, April 2003.

[WK03b] WAN, Z. and M. V. KOTHARE: *A Two-Level Model Predictive Control Formulation for Stabilization and Optimization.* In *Proc. of the American Control Conference*, pages 5294–5299, Denver, Colorado, June 2003.

[WSV00] WOLKOWICZ, H., R. SAIGAL and L. VANDENBERGHE (editors): *Handbook of Semidefinite Programming.* Kluwer, 2000.

[Zie94] ZIEGLER, G. M.: *Lectures on Polytopes.* Springer, 1994.

[ZM95] ZHENG, A. and M. MORARI: *Stability of model predictive control with mixed constraints.* IEEE Trans. Automatic Control, 40:1818–1823, 1995.

Index

L

M

N

P

R

S

T

V