Slovak University of Technology in Bratislava
Faculty of Chemical and Food Technology
Institute of Information Engineering, Automation, and Mathematics

Real-Time Explicit Model Predictive Control of Processes

Ing. Martin Herceg

Dissertation Thesis

Supervisors: Prof. Ing. Miroslav Fikar, DrSc.
Ing. Michal Kvasnica, PhD.

2009

Acknowledgment

It would not been possible to write the thesis without a tight collaboration from my colleagues. I start by expressing my deepest gratitude to my nearest supervisor, Michal Kvasnica. From the early beginnings of my research activities, he has been guiding me in systematic way towards practical solutions and even in the cases when I thought it would be impossible ... Thank you for instilling this non giving up temperament in me and continuous guidance in every aspect of the academic life.

Secondly, my thanks goes to my direct supervisor Miroslav Fikar who has taken care of my scientific activities since I was an early stage student. He has opened the doorway to the Institute where I could spend long time by "playing with things" in laboratories, experiencing the principles of real-time control. I'm very grateful for his teaching guidance when I met with Linux stuff for the first time as well as with LaTeX, Matlab, C, and many more. Furthermore, he allowed me to meet and visit the top level research groups for which I'm utmost grateful.

My former PhD colleagues Michal Čižniar and Tomáš Hirmajer deserve a special thanks for instantaneous discussions and support with software implementation of MPC. It was really special time of year when we could share the ideas in one office. Without both of you, I would not finish very important parts of my research regarding nonlinear MPC. Thank you guys!

I must also mention the colleagues from the Institute, especially Ľuboš Čirka, Marián Podmajerský, Juraj Vörös, Radoslav Paulen, Stanislav Vagač, Janka Závacká, Katka Vaneková, Lenka Blahová, Andrea Kalmárová, Mária Karšaiová, Anna Vasičkaninová as well as Professors Alojz Mészáros, Ján Mikleš, Ján Dvoran, and Monika Bakošová for creating an unique working atmosphere at the Institute where it was a pleasure to work in. Especially, my thanks goes to Monika Cepeková for her friendly and easy going attitude to process heaps of the paper work.

Throughout my studies I met a lot of exceptional people. Many of them come from the chemical engineering group at STU Bratislava, namely: Matúš Chlebovec, Pavol Laššák, Katarína Vaňková, Martin Hucík, Juraj Sláva, and from the Automatic Control Laboratory in Zurich: Sébastien Mariéthoz, Colin Jones, and many more I have met on my journeys and conferences. Thank you all for sharing the research experiences and all the great times!

Last but not least I wish to thank to my family and nearest friends, for their support and

patience throughout my studies. Without your help it would be very difficult to finish the studies. Thank you very much for everything.

Martin Herceg
Bratislava, November 2009

Abstract

The aim of the thesis is to provide techniques for computing and application of explicit model predictive control of processes in real-time. Explicit model predictive control is a method for optimal control of processes with constraints where the control law is given in an explicit form. It is an optimization based approach where a process model is used to predict the future. In the presented approach the model is derived using a hybrid modeling framework. It will be shown how that hybrid models are capable of modeling wide class of processes with a sufficient precision. Based on the hybrid model, it is possible to formulate an optimal control problem which considers varying dynamical properties of the plant and moreover, takes the operating constraints into account. If the optimal control problem is solved for the whole set of initial conditions, an explicit solution is obtained. The explicit solution characterizes the optimal control law as a function of given initial conditions. This allows the control law to be easily realized in practice, just by evaluating the function for given value of initial conditions. Therefore, the explicit model predictive controller designed in this way is especially suitable for applications with fast dynamical changes which require low implementation cost. The thesis describes essentials for understanding ingredients of explicit model predictive control. In particular, the main part concerns with construction of hybrid models and their deployment in formulation of optimization problems. Consequently, efficient algorithms for computing explicit solutions to time optimal control problems are proposed. Obtained results are applied in real-time experiments that show desired optimal performance and the controller satisfies all operating constraints.

Abstrakt

Cieľom dizertačnej práce je prezentovať metódy výpočtov explicitného prediktívneho riadenia a ich aplikácie na procesoch v reálnom čase. Explicitné prediktívne riadenie je spôsob optimálneho riadenia procesov s obmedzeniami, kde zákon riadenia je daný priamo, čiže v explicitnej forme. Základom prediktívneho riadenia je model, ktorý slúži na získanie predpovede do budúcnosti a následnej formulácie problému optimálneho riadenia. V tejto práci sa využíva hybridný prístup k modelovaniu, ktorý dokáže opísať dynamické vlastnosti procesov s dostatočnou presnosťou. Na základe hybridného opisu je potom možné naformulovať problém optimálneho riadenia, v ktorom vystupujú meniace sa dynamické vlastnosti procesu, a navyše je možné zahrnúť aj operačné obmedzenia. Ak sa takýto problém rieši pre celú množinu začiatočných podmienok, výsledok sa nazýva explicitné riešenie. Explicitné riešenie charakterizuje zákon riadenia ako funkciu začiatočných podmienok, čo umožňuje veľmi jednoduchú aplikáciu zákona riadenia v praxi, danú evaluáciou funkcie pre konkrétnu hodnotu začiatočných podmienok. Preto sú práve explicitné riešenia veľmi vhodné na riadenie procesov s rýchlou dynamikou a vyžadujú nenáročné podmienky na implementáciu. Práca vysvetľuje základné princípy potrebné na porozumenie výpočtov explicitných riešení v prediktívnom riadení. Zaoberá sa hlavne modelovaním hybridných systémov a ich zakomponovaním do formulácie optimalizačných problémov. Navrhnuté sú algoritmy na získanie explicitných riešení časovo-optimálnych úloh riadenia, ktoré sú aplikované a verifikované v praxi. Dosiahnuté výsledky riadenia potvrdzujú, že meniace vlastnosti procesu sú rešpektované, kvalita riadenia a všetky operačné obmedzenia sú splnené.

Contents

List of Abbreviations

CFTOC	–	constrained finite time optimal control
CITOC	–	constrained infinite time optimal control
CPU	–	central processing unit
CSTR	–	continuously stirred tank reactor
DMC	–	dynamic matrix control
ELC	–	extended linear complementarity (system)
FLOPS	–	floating point operations
GPC	–	generalized predictive control
HYSDEL	–	hybrid system description language
IMC	–	internal model control
LC	–	linear complementarity (system)
LED	–	light emitting diode
LMI	–	linear matrix inequality
LP	–	linear program
MILP	–	mixed-integer linear program
MLD	–	mixed logical dynamical (system)
MMPS	–	max-min-plus-scaling systems
MPC	–	model predictive control
mpLP	–	multiparametric linear program
mpMILP	–	multiparametric mixed-integer linear program
mpQP	–	multiparametric quadratic program
NLP	–	nonlinear program
PDC	–	parallel distributed compensation
PID	–	proportional integral derivative (controller)
PWA	–	piecewise affine
QP	–	quadratic program
RTW	–	real-time workshop
SDP	–	semi-definite program
SOS	–	sum of squares
TS	–	Takagi-Sugeno (system)

Introduction

One of the developing fields in process control is model predictive control (MPC). MPC is an optimization-based approach which exploits the knowledge of dynamical behavior of the plant, represented by a process model. Based upon the information from the model, the optimal control problem is constructed where specific requirements regarding performance as well as constraints satisfaction can be met [68]. MPC has spread out in the industry with an increasing grow in computer-aided engineering, because it mostly relies on optimization routines.

Traditionally, MPC approach solves the optimal control problem repeatedly in every sampling period. At each computation the controller predicts the process behavior over some time horizon to the future by using the process inputs as degrees of freedom. The aim is to find a future sequence of these inputs such that certain goals are attained. MPC therefore searches such sequence by solving an optimization problem. If a solution is found, MPC actually uses only the first element of the computed sequence and this is applied to the plant. This is how MPC implements feedback action in the closed-loop system.

Computational power is often required because advanced optimization routines are needed to solve the optimal control problem. In practice, the measurements are usually collected periodically, and the time lag between two consecutive updates refers to sampling time. For a safe operation of the plant, the optimization algorithm should terminate before next sampling instant. Therefore, MPC has been mostly applied in the chemical industry where processes have large settling times, and thus the sampling time is sufficiently large. Hence, the popularity of MPC grew mainly in this field and garnered large attention due to its major advantage, to deal with operating constraints. Several commercial packages are currently available which implement MPC in the on-line fashion, i.e. solving the optimization problem as the process is running. An excellent survey of these methods is documented in [87]. The disadvantage is, however, this way for implementation of MPC is very demanding on computational resources and it is also reason for expensive hardware & software setup. These incentives have motivated development of *explicit* MPC.

In the explicit MPC setup, the optimization problem is solved *parametrically*, i.e. for the whole set of variables which satisfy design constraints. This alternative approach ensures that the solution to the optimal control problem exists before the plant actually starts operation. In the implementation phase there is no need for repetitive optimization, only the explicit solution is evaluated at every sampling instant. The approach has thus computational advantage on the implementation side, hence it is suitable for processes

with very small sampling times. Furthermore, evaluation of the explicit solution can be done with low computational power which makes explicit MPC attractive for industrial purposes.
Quality of predictions in explicit MPC can be improved by finding a process model that has a sufficient accuracy in the desired operating regime. Therefore, the progress of explicit MPC has evolved from using linear models [11, 14, 16] toward hybrid models with higher accuracy [12, 23, 32] motivated by the fact, that hybrid models can approximate the original system with arbitrary accuracy [94]. However, to deploy hybrid models for MPC synthesis, they need to converted to a form suitable for optimization purposes [15] and there are few software tools that can provide such translation, e.g. [107]. The explicit MPC approach has been developed by [11, 16, 23] and recent progress in the field is summarized in book collections [83, 84].

The structure of the thesis is as follows:

Part I gives an overview of the current state in the literature. The computational principles of explicit MPC are explained, in particular, the concepts of polytopic sets and optimization problems. Next, the multiparametric technique for solving the optimization problems is detailed which creates the core of explicit MPC. General MPC approach is explained in Chapter 4 where concrete ingredients as well as feedback implementation are discussed. The main building block for the remainder of the thesis is Chapter 5, where original algorithms of [5, 11, 16, 23, 46] are presented.

Part II presents the main theoretical contributions of the thesis. Firstly, a software for modeling and simulation of linear hybrid systems is presented. This part builds on results from HYSDEL 3.0 project was established between STU Bratislava and ETH Zurich between 2008 – 2009. HYSDEL 3.0 is a modeling language for hybrid systems and allows easy setup of optimization problems in MPC. HYSDEL 3.0 is an extension of [107] which offers more flexibility and compatibility with MPC setups.
Secondly, the control part deals with the time optimal control of hybrid systems. The presented results are extensions of the approach [46] for a tracking problem and for a regulation problem of uncertain hybrid systems. The same idea used in time optimal control of uncertain hybrid system has been applied to fuzzy systems. Concretely, these results are given as:

Time optimal tracking of a time varying reference signal for PWA systems. Published in:

- Herceg, M., Kvasnica, M., Fikar, M.: Minimum-time predictive control of a servo engine with deadzone, *Control Engineering Practice*, 17(11):1349–1357, 2009.

Adaptive time optimal control of PWA systems. Published in:

Main Goals

The main theme is to provide techniques for computing and efficient application of explicit MPC to real-time equipments. This aim is tackled in three phases: modeling, control design, and real-time application.

- The first objective is a software implementation for modeling of linear hybrid systems. Since process models are inherent part of MPC, generation of numerically sound models helps to formulate an optimization problem more efficiently. It will be shown that the developed software is capable of producing compact process models from simple logical rules that are specially tailored for optimization purposes. Furthermore, the software allows composition of large models and analysis of their dynamic behavior in a user friendly manner.

- The control design part follows three main goals:

  - Explicit solution to time optimal MPC tracking problem for linear hybrid systems. The proposed solution is an extension of the explicit MPC approach to cases for tracking an unknown (and possibly time varying) reference. It is demonstrated that such a solution guarantees convergence to the desired reference signal in a minimal number of steps and maintains all design constraints.
  - Explicit solution to time optimal MPC problem for uncertain linear hybrid systems. The objective of the scheme is to extend the results of explicit MPC to cases with uncertain hybrid models. In particular, the proposed scheme is capable of incorporating parametric uncertainties present in dynamical matrix to explicit control design if certain assumptions on process model are satisfied. Furthermore, the approach is applicable to a class of Takagi-Sugeno fuzzy systems which is a significant contribution to the control theory of fuzzy systems.
  - Design of a low-complexity explicit MPC scheme. The suggested approach dramatically reduces the requirements for data storage and processing power needed to implement explicit MPC. Motivated by practical needs to meet the hardware constraints the scheme offers alternative implementation of explicit MPC. The scheme thus suits applications with very cheap implementation cost.

- The aim of the practical part is to study the proposed solutions to explicit MPC in real-time experiments. Provided results shall give an immediate answer for practical applicability and one can verify theoretical conclusions directly from the experiments.

Part I

BACKGROUND

Chapter 1

Mathematical Basics

To understand the concepts of MPC, one has to take a deeper look for the concrete building blocks used in optimization theory, such as functions and sets. This chapter thus introduces standard mathematical concepts used throughout the whole thesis. Especially, the part devoted to geometrical operations with sets will be emphasized and important geometrical operations are illustrated to simplify the presentation.

1.1 Basic Definitions

Fundamental definitions from set and function theory are given in this section. The notions of convexity and the abbreviation PWA, meaning piecewise affine, will be the most frequently used throughout the thesis. Notations and definitions are based on [31, 63].

1.1.1 Sets

Definition 1.1 (ϵ-Ball [31]) *The open n-dimensional ϵ-ball in $\mathbb{R}^n$ around a given point (center) $\boldsymbol{x}_c$ is the set*

$$\mathcal{B}_\epsilon(\boldsymbol{x}_c) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{x}_c\| < \epsilon\},$$

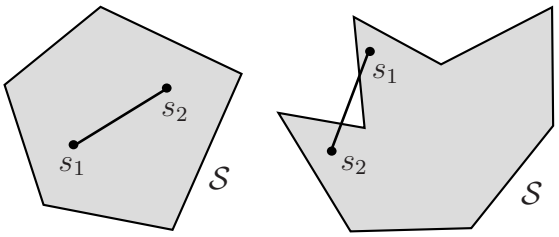
where the radius $\epsilon > 0$ and $\|\cdot\|$ denotes any vector norm (usually the Euclidean vector norm $\|\cdot\|_2$).

Definition 1.2 (Neighborhood [31]) *The neighborhood of a subset $\mathcal{S}$ of $\mathcal{X} \subseteq R^n$ is defined as a set $\mathcal{N}(\mathcal{S})$ with $\mathcal{S} \subset \mathcal{N}(\mathcal{S}) \subseteq \mathcal{X}$ such that for each $s \in \mathcal{S}$ there exist an n-dimensional ϵ-ball with $\mathcal{B}_\epsilon(s) \subseteq \mathcal{N}(\mathcal{S})$ and $\epsilon > 0$.*

Definition 1.3 (Convex set [31]) *A set $\mathcal{S} \subseteq R^{n_s}$ is convex if the line segment connecting any pair of points of $\mathcal{S}$ lies entirely in $\mathcal{S}$, i.e. if for any $s_1, s_2 \in \mathcal{S}$ and any α with $0 \leq \alpha \leq 1$, we have*

$$\alpha s_1 + (1 - \alpha) s_2 \in \mathcal{S}.$$

See Figs. 1.1(a), 1.1(b).



(a) (b)

Figure 1.1: Convex (a) and non-convex (b) set.

Definition 1.4 (Convex hull [31]) *The convex hull of a set $\mathcal{S} \subseteq \mathcal{R}^{n_s}$ is the smallest convex set containing $\mathcal{S}$, i.e.*

$$hull(\mathcal{S}) := \left\{ \sum_{i=1}^{k} \alpha_i s_i \,\middle|\, \forall s_i \in \mathcal{S}\ \exists \alpha_i,\ \alpha_i \geq 0,\ \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

Definition 1.5 (Closed set [31]) *A set $\mathcal{S} \subseteq R^{n_s}$ is closed if every point not in $\mathcal{S}$ has a neighborhood disjoint from $\mathcal{S}$, i.e.*

$$\forall \boldsymbol{x} \notin \mathcal{S}, \exists \epsilon > 0 \quad \textit{such that} \quad \mathcal{B}_\epsilon(\boldsymbol{x}) \cap \mathcal{S} = \emptyset.$$

Definition 1.6 (Bounded set [31]) *A set $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ is bounded if it is contained inside some ball $\mathcal{B}_r(\cdot)$ of finite radius r, i.e.*

$$\exists r < \infty,\ s \in \mathbb{R}^{n_s} \quad \textit{such that} \quad \mathcal{S} \subseteq \mathcal{B}_r(s).$$

Definition 1.7 (Compact set [31]) *A set $\mathcal{S}$ is compact if it is closed and bounded.*

Definition 1.8 (Set collection [31]) *$\mathcal{S}$ is called a set collection (in $\mathbb{R}^{n_s}$) if it is a collection of finite number of n_s-dimensional sets $\mathcal{S}_i$, i.e.*

$$\mathcal{S} := \{\mathcal{S}_i\}_{i=1}^{N_\mathcal{S}},$$

where $\dim(\mathcal{S}_i) = n_s$ and $\mathcal{S}_i \subseteq \mathbb{R}^{n_s}$ for $i = 1, \ldots, N_\mathcal{S}$ with $N_\mathcal{S} < \infty$. A set collection of sometimes also referred to as family of sets.

Definition 1.9 (Partition [31]) *A collection of sets $\{\mathcal{S}_i\}_{i=1}^{N_\mathcal{S}}$ is a partition of a set $\mathcal{S}$ if $\mathcal{S} = \cup_{i=1}^{N_\mathcal{S}} \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j$ for all $i \neq j$, where $i, j \in \{1, \ldots, N_\mathcal{S}\}$.*

1.1.2 Functions

Definition 1.10 (Vector 1-, ∞-, 2-norm [31]) *The vector 1-, ∞-, and 2-norm of $\boldsymbol{x} \in \mathbb{R}^n$ is defined as*

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|, \quad \|\boldsymbol{x}\|_\infty := \max_{1 \le i \le n} |x_i|, \quad \|\boldsymbol{x}\|_2 := \sum_{i=1}^{n} x_i^2$$

respectively, where x_i is the i-th element of $\boldsymbol{x}$.

Definition 1.11 (Convex/concave function [31]) *A real-valued function $\boldsymbol{f} : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ is convex if its domain $\mathcal{X} \subseteq R^n$ is a convex set and*

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \quad 0 \le \alpha \le 1 \quad \Longrightarrow \quad \boldsymbol{f}(\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2) \le \alpha \boldsymbol{f}(\boldsymbol{x}_1) + (1-\alpha)\boldsymbol{f}(\boldsymbol{x}_2)$$

where $\le$ is to be considered component wise. $\boldsymbol{f}(\cdot)$ is strictly convex if the last inequality above is replaced by strict inequality. $\boldsymbol{f}(\cdot)$ is concave if $-\boldsymbol{f}(\cdot)$ is convex.

Definition 1.12 (Positive/negative (semi-)definite function [31]) *A real-valued function $\boldsymbol{f} : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ with $\mathcal{X} \subseteq \mathbb{R}^n$ is positive definite if $\boldsymbol{f}(\boldsymbol{0}) = \boldsymbol{0}$ and $\boldsymbol{f}(\boldsymbol{x}) > \boldsymbol{0}, \forall \boldsymbol{x} \neq \boldsymbol{0}$. $\boldsymbol{f}(\cdot)$ is positive semi-definite if $\boldsymbol{f}(\boldsymbol{0}) = \boldsymbol{0}$ and $\boldsymbol{f}(\boldsymbol{x}) \ge \boldsymbol{0}, \forall \boldsymbol{x} \neq \boldsymbol{0}$. $\boldsymbol{f}(\cdot)$ is called negative (semi-)definite if $-\boldsymbol{f}(\cdot)$ is positive (semi-)definite.*

Definition 1.13 (Affine function [31]) *A real-valued function $\boldsymbol{f} : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ with $\mathcal{X} \subseteq \mathbb{R}^n$ is affine if it is of the form*

$$\boldsymbol{f}(\boldsymbol{x}) := \boldsymbol{F}\boldsymbol{x} + \boldsymbol{g},$$

where $\boldsymbol{F} \in \mathbb{R}^{n_f \times n}$ and $\boldsymbol{g} \in \mathbb{R}^{n_f}$.

Definition 1.14 (Piecewise affine function [31]) *A real-valued function $\boldsymbol{f}_{PWA} : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ with $\mathcal{X} \subseteq \mathbb{R}^n$ is piecewise affine (PWA), if $\{\mathcal{X}_i\}_{i=1}^{N_\mathcal{X}}$ is a set partition of $\mathcal{X}$ and*

$$\boldsymbol{f}_{PWA}(\boldsymbol{x}) := \boldsymbol{F}_i \boldsymbol{x} + \boldsymbol{g}_i \quad \forall \boldsymbol{x} \in \mathcal{X}_i,$$

where $\boldsymbol{F}_i \in \mathbb{R}^{n_f \times n}$, $\boldsymbol{g}_i \in \mathbb{R}^{n_f}$, and $1, \ldots, N_\mathcal{X}$.

1.2 Polytopes

Polytopes are a special type of sets that are very important in the optimization theory. This section reviews the basic notation used in this field. For a detailed view, the reader is referred to [47].

Definition 1.15 (Hyperplane [31]) *A hyperplane $\mathcal{P}$ in $\mathbb{R}^n$ is a set of the form*

$$\mathcal{P} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^T \boldsymbol{x} = l\},$$

where $\boldsymbol{h} \in \mathbb{R}^n$, $l \in \mathbb{R}$.

Definition 1.16 (Half-space [31]) *A (closed) half-space $\mathcal{P}$ in $\mathbb{R}^n$ is a set of the form*

$$\mathcal{P} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^T\boldsymbol{x} \leq l\},$$

where $\boldsymbol{h} \in \mathbb{R}^n$, $l \in \mathbb{R}$.

Definition 1.17 (Polyhedron [31]) *A polyhedron is the intersection of a finite number of half-spaces.*

One fundamental property of polyhedrals is stated by the following lemma.

Lemma 1.1 ([31]) *A polyhedron $\mathcal{P} \in \mathcal{R}^n$ is a convex set and can always be represented in the form*

$$\mathcal{P} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{l}\},$$

where $\boldsymbol{H} \in \mathbb{R}^{m\times n}$, $\boldsymbol{l} \in \mathbb{R}^m$.

Definition 1.18 (Face, facet, ridge, edge, vertex [63]) *A linear inequality $\boldsymbol{h}^T\boldsymbol{x} \leq l$ is called valid for a polyhedron $\mathcal{P}$ if $\boldsymbol{h}^T\boldsymbol{x} \leq l$ holds $\forall \boldsymbol{x} \in \mathcal{P}$. A subset $\mathcal{F}$ of $\mathcal{P}$ of a polyhedron is called a face of $\mathcal{P}$ if it can be represented as*

$$\mathcal{F} := \mathcal{P} \cap \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{h}^T\boldsymbol{x} = l\},$$

for some valid inequality $\boldsymbol{h}^T\boldsymbol{x} = l$. The faces of the polyhedron $\mathcal{P}$ with dimension 0, 1, (n-2), and (n-1) are called vertices, edges, ridges, and facets, respectively.

Definition 1.19 (Polytope [31]) *A polytope is a bounded polyhedron.*

Following theorem defines two representation of a polytope: H- and V- polytope. For graphical illustration, see Figs. 1.2(a), 1.2(b).

Theorem 1.1 (Polytope representation [31]) *$\mathcal{P} \subset \mathbb{R}^n$ is the convex hull of a finite point set $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{N_v}\}$ for some $\boldsymbol{v}_i \in \mathbb{R}^n$ with $1, \ldots, N_v$ (V-polytope)*

$$\mathcal{P} := hull(\mathcal{V}) = \left\{ \boldsymbol{x} \in \mathbb{R}^n \,\middle|\, \boldsymbol{x} = \sum_{i=1}^{N_v} \alpha_i \boldsymbol{v}_i,\ 0 \leq \alpha_i \leq 1,\ \sum_{i=1}^{N_v} \alpha_i = 1 \right\},$$

if and only if it is bounded intersection of finitely many half-spaces (H-polytope)

$$\mathcal{P} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{l}\},$$

for some $\boldsymbol{H} \in \mathbb{R}^{m\times n}$, $\boldsymbol{l} \in \mathbb{R}^m$.

Definition 1.20 (Polytope collection [31]) *$\mathcal{P}$ is called a polytope collection or P-collection (in $\mathbb{R}^n$) if it is a set collection of a finite number of n-dimensional polytopes $\mathcal{P}_i$, i.e.*

$$\mathcal{P} := \{\mathcal{P}_i\}_{i=1}^{N_\mathcal{P}}$$

where $\mathcal{P}_i := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{l}\}$, $dim(\mathcal{P}_i) = n$, $i = 1, \ldots, N_\mathcal{P}$ with $N_\mathcal{P} < \infty$. A P-collection is also referred to as family of polytopes.

See Fig. 1.3.

1.3 Geometric Operations with Polytopes

Definition 1.21 (Intersection of Polytopes) *The intersection of two polytopes $\mathcal{P} \in \mathbb{R}^n$ and $\mathcal{Q} \in \mathbb{R}^n$ is defined by*

$$\mathcal{P} \cap \mathcal{Q} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in \mathcal{P}, \boldsymbol{x} \in \mathcal{Q}\}.$$

See Fig. 1.4.

Definition 1.22 (Set Difference [31]) *The set difference of two polytopes $\mathcal{P} \in \mathbb{R}^n$ and $\mathcal{Q} \in \mathbb{R}^n$ is defined by*

$$\mathcal{P} \backslash \mathcal{Q} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \in \mathcal{P},\ \boldsymbol{x} \notin \mathcal{Q}\}.$$

See Figs. 1.5(a), 1.5(b).

Definition 1.23 (Pontryagin Difference [63]) *The Pontryagin difference of two polytopes $\mathcal{P} \in \mathbb{R}^n$ and $\mathcal{Q} \in \mathbb{R}^n$ is a polytope*

$$\mathcal{P} \ominus \mathcal{Q} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} + \boldsymbol{q} \in \mathcal{P},\ \forall \boldsymbol{q} \in \mathcal{Q}\}.$$

See Figs. 1.6(a), 1.6(b).

Definition 1.24 (Chebyshev Ball [31]) *The Chebyshev Ball of a polytope $\mathcal{P} \subset \mathbb{R}^n$ is the largest Euclidean ball*

$$\mathcal{B}_r(\boldsymbol{x}_c) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{x}_c\|_2 \leq r\}$$

where r is the Chebyshev radius and $\boldsymbol{x}_c$ the Chebyshev center, such that $\mathcal{B}_r(\boldsymbol{x}_c) \subseteq \mathcal{P}$.

Definition 1.25 (Projection [31]) *Given a set $\mathcal{Q} \subseteq \mathbb{R}^{n_\mathcal{Q}}$ and a set $\mathcal{P} \subseteq \mathbb{R}^{n_\mathcal{P}}$ with $n_\mathcal{Q} \leq n_\mathcal{P} < \infty$, the (affine) projection of $\mathcal{P}$ onto $\mathcal{Q}$ is defined as*

$$proj_\mathcal{Q}(\mathcal{P}) := \{\boldsymbol{q} \in \mathcal{Q} \mid \exists \boldsymbol{p} \in \mathcal{P} \text{ with } \boldsymbol{q} = \boldsymbol{A}\boldsymbol{p} + \boldsymbol{f}\}$$

for some given $\boldsymbol{A} \in \mathbb{R}^{n_\mathcal{Q} \times n_\mathcal{P}}$ and $\boldsymbol{f} \in \mathbb{R}^{n_\mathcal{Q}}$.

Definition 1.26 (Bounding Box [97]) *A bounding box Bbox($\mathcal{P}$) of a set $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{l}\}$ is the smallest hyperrectangle which contains the set $\mathcal{P}$.*

See Fig. 1.7

Definition 1.27 (Affine Transformation) *An affine transformation of a polytope $\mathcal{P} \in \mathbb{R}^n$ is a polytope $\mathcal{R} \in \mathbb{R}^n$*

$$\mathcal{R} := \{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{f},\ \boldsymbol{x} \in \mathcal{P}\}$$

for some given $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{f} \in \mathbb{R}^n$.

See Fig. 1.8.

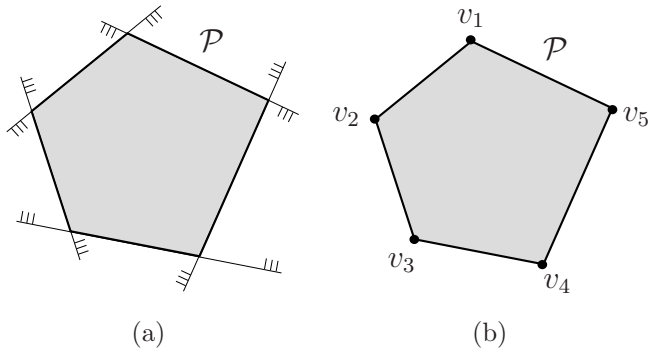


Figure 1.2: H-(a) and V-(b) representation of a polytope.

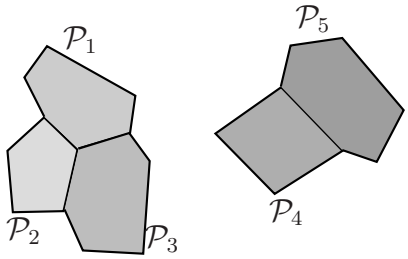


Figure 1.3: Polytope collection (or P-collection) is a (possibly) non-convex union of polytopes, i.e. $\mathcal{P} = \bigcup_i \mathcal{P}_i$.

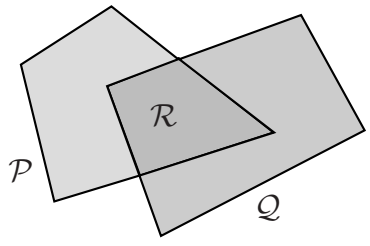


Figure 1.4: Intersection of two polytopes, $\mathcal{R} = \mathcal{P} \cap \mathcal{Q}$.

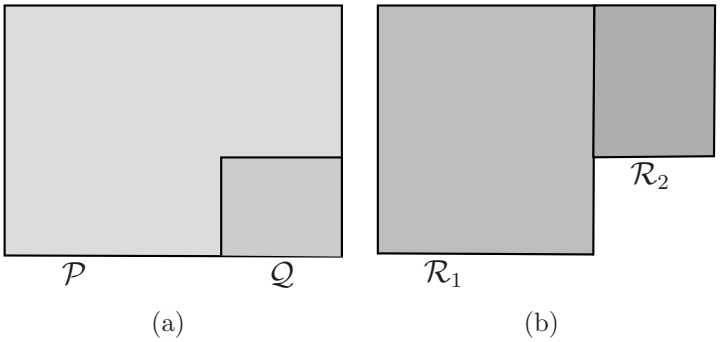


Figure 1.5: Set difference $\mathcal{R} = \mathcal{P} \backslash \mathcal{Q} = \mathcal{R}_1 \cup \mathcal{R}_2$.

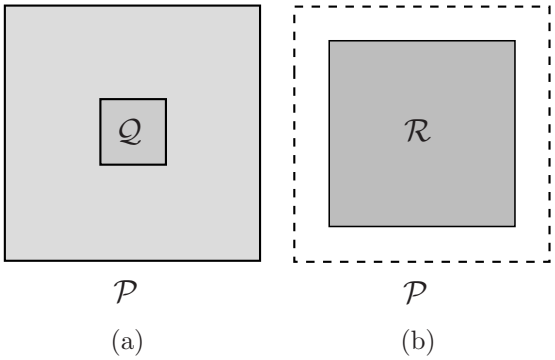


Figure 1.6: Pontryagin difference $\mathcal{R} = \mathcal{P} \ominus \mathcal{Q}$.

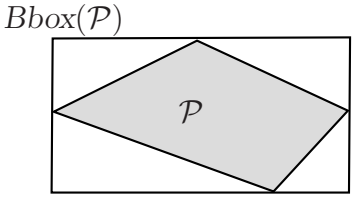


Figure 1.7: Bounding box of a polytope $\mathcal{P}$.

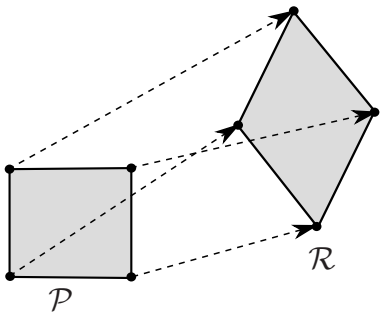


Figure 1.8: Affine transformation of polytope $\mathcal{P}$, $\mathcal{R} = \boldsymbol{A}\mathcal{P} + \boldsymbol{f}$ for given $\boldsymbol{A}$, $\boldsymbol{f}$.

Chapter 2

Optimization Problems

In this chapter basic optimization problems are reviewed. Some of the problems will be used later in the thesis to characterize the proposed approaches to MPC. Formulation of the optimization problems follows the convention used in textbooks on optimization where the unknown variables are usually denoted as $\boldsymbol{x} \in \mathbb{R}^n$, i.e. $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$. Interested reader may find out more in excellent books [1, 79].

2.1 General Formulation

A general optimization problem can be written as follows

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}) \tag{2.1a}$$

$$\text{s.t.} \quad \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \tag{2.1b}$$

$$\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \tag{2.1c}$$

where $F(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function, $\boldsymbol{g}(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}^{n_g}$, $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}^{n_f}$ are inequality and equality constraints, respectively. The *vector of variables* $\boldsymbol{x} \in \mathbb{R}^n$ represents the parameters to be determined. Solution $\boldsymbol{x}^*$ of the problem (2.1) is optimal if $F(\boldsymbol{x}) \geq F(\boldsymbol{x}^*)$ holds for any $\boldsymbol{x}$ from sets (2.1b) and (2.1c). A discipline encompassing the theory and practice of numerically solving problems like (2.1) is known as *mathematical programming* [1, Section 1.1]. Depending on the form of the objective function (2.1a) and sets (2.1b), (2.1c) two main groups are distinguished:

1. convex optimization [24],
2. non-convex optimization [7, 17, 37].

Convex optimization is basically a subgroup of more general, non-convex optimization, and its specification is that objective function (2.1a) is a convex function and constraints (2.1b), (2.1c) are convex sets. This fundamental property plays an important role in determining the strategy which should be applied in order to efficiently solve the problem. In the sequel some basic properties of both groups will be outlined and some methods for solving the problems will be reviewed.

2.2 Convex Problems

Convex optimization problems have the property that (2.1b) and (2.1c) are convex functions according to Definition 1.11. Furthermore, any local optimal point $\boldsymbol{x}^*$ (i.e. with respect to neighborhood of a point $\boldsymbol{x}^*$) is also globally optimal (i.e. with respect to all points in constraint set (2.1b) and (2.1c)). In other words, there is guarantee that absolute minimum of the objective function will be found. Another aspect of convex optimization problems is that there exist very efficient methods to solve it. This section lists basic problems which will be frequently used in the main part of the thesis.

2.2.1 Linear Programming

Linear Program (LP) is a convex optimization problem where the objective function and constraints are linear, i.e.

$$\min_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x} \tag{2.2a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} \tag{2.2b}$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.2c}$$

where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{b} \in \mathbb{R}^p$. In fact, the inequalities in (2.2b) and (2.2c) create a polyhedron, which is called the constraint set. Geometrically, LP can be interpreted as searching the minimum of a linear function over a given polyhedral set. The solution of LP has one important property that it always lies on a boundary of the set (2.2b). Complexity of the problem can be expressed in the number of variables n and constraints m entering the problem. Currently the most efficient methods to solve LP are *simplex methods* with exponential complexity and *interior point methods* with polynomial complexity [110].

2.2.2 Quadratic Programming

A Quadratic Program (QP) is a convex optimization problem where the objective function is quadratic and constraints are linear. Precisely,

$$\min_{\boldsymbol{x}} 0.5\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} \tag{2.3a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} \tag{2.3b}$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.3c}$$

where the quadratic term $0.5\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} \geq 0$ is restricted to have only nonnegative values, i.e. the matrix $\boldsymbol{P} \succeq 0$ is positive semidefinite. Moreover, $\boldsymbol{P} \in \mathbb{R}^{n\times n}$, $\boldsymbol{q} \in \mathbb{R}^n$, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{b} \in \mathbb{R}^p$. From a geometrical point of view, a minimum of a convex quadratic function is searched over a polyhedral set (2.3b), (2.3c). QP can be solved using *active set methods*, or *interior point methods* [79].

2.2.3 Semidefinite Programming

A Semidefinite Program (SDP) is a convex optimization problem of the form

$$\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \tag{2.4a}$$

$$\text{s.t.} \quad \boldsymbol{F}_0 + \sum_i \boldsymbol{x}_i \boldsymbol{F}_i \succeq 0, \tag{2.4b}$$

where $\boldsymbol{F}_i \in \mathbb{R}^{n \times n}$ are given symmetric matrices and $\boldsymbol{c} \in \mathbb{R}^n$. The fundamental property of SDP is that the set (2.4b) is a convex set (Definition 1.3) and the constraint $\succeq 0$ means positive semidefinite. SDP covers wider scope of problems, including LP, linear matrix inequalities (LMI), and has interesting geometrical interpretation. In particular, the constraints create a convex conic set over which a given convex function is minimized. Solution can be found by employing one of the interior point methods with polynomial complexity. Practically, SDP solvers are roughly one order of magnitude slower than LP or QP solvers [45]. For a more detailed view in SDP, the reader is referred to [24].

2.2.4 Sum of Squares Decomposition

One of the basic problems in control theory is checking global non-negativity of a function of several variables, which in general can be shown to be a very difficult problem [80, Section 4.1]. A Sum of Squares (SOS) decomposition is a convex optimization problem where the non-negativity of a polynomial $p(\boldsymbol{x}) \geq 0$ with degree d is certified by searching for an existence of a monomial $v(\boldsymbol{x})$ of degree less than or equal to $d/2$ such that $\boldsymbol{p} = \sum_{i=1} v_i^2(\boldsymbol{x})$. Mathematically, the problem can be written as SDP, i.e.

$$\text{find } \boldsymbol{Q} \tag{2.5a}$$

$$\text{s.t.} \quad \boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{v}^T(\boldsymbol{x}) \boldsymbol{Q} \boldsymbol{v}(\boldsymbol{x}) \tag{2.5b}$$

$$\boldsymbol{Q} \succeq 0 \tag{2.5c}$$

where $\boldsymbol{Q}$ is a symmetric and positive definite matrix. Note that there can be multiple solutions to problem (2.5), thus some objective function is usually added. The choice of objective function depends on the particular problem. For instance, the problem of finding the lowest bound for polynomial $p(\boldsymbol{x}) \geq 0$ can be written as

$$\min_{\boldsymbol{Q}} \gamma \tag{2.6a}$$

$$\text{s.t.} \quad \boldsymbol{p}(\boldsymbol{x}) - \gamma = \boldsymbol{v}^T(\boldsymbol{x}) \boldsymbol{Q} \boldsymbol{v}(\boldsymbol{x}) \tag{2.6b}$$

$$\boldsymbol{Q} \succeq 0 \tag{2.6c}$$

where $\gamma \in \mathbb{R}$, $\gamma > 0$ is the objective function. Problems with SOS decomposition have been extensively studied in [80] and have wide applications in systems and control theory. Since SOS decomposition are embedded into SDP framework, efficient solvers are available.

2.3 Non-convex Problems

To tackle the general Non-Linear Program (NLP) (2.1), an insight view into the problem's structure must be taken because there exists no unique approach in selection of optimization algorithms. Based on the structure, NLPs are categorized into several groups where concrete methods can be applied. One special class of NLP is a mixed-integer linear program which is of main interest in the following.

2.3.1 Mixed Integer Linear Programming

A Mixed Integer Linear Program (MILP) is a non-convex optimization problem where the objective function and constraints are linear. The non-convexity stems from the fact that the optimized variables $\boldsymbol{x}$, $\boldsymbol{\delta}$ belong to the real set $\mathbb{R}$ and binary set $\{0,1\}$, respectively. In particular,

$$\min_{\boldsymbol{x},\boldsymbol{\delta}} \ \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{\delta} \tag{2.7a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} + \boldsymbol{T}\boldsymbol{\delta} \leq \boldsymbol{h} \tag{2.7b}$$

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{E}\boldsymbol{\delta} = \boldsymbol{b} \tag{2.7c}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\delta} \in \{0,1\}^q$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{d} \in \mathbb{R}^q$, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, $\boldsymbol{T} \in \mathbb{R}^{m\times q}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{p\times n}$, $\boldsymbol{E} \in \mathbb{R}^{p\times q}$, and $\boldsymbol{b} \in \mathbb{R}^p$. The only difference comparing to convex LP is a presence of binary variables $\boldsymbol{\delta}$ which can take only values $\{0,1\}$ and this is also a reason why the set (2.7b) is non-convex. Fortunately, if the binary variable is fixed or relaxed, a convex set is obtained and the problem can be solved using methods for convex optimization. This principle is deployed in *branch and bound* and *branch and cut* methods [66, 92] that avoid complete enumeration of 0-1 integer values. Even with this enhancement, the algorithms have exponential complexity depending on the cardinality of binary vector $\boldsymbol{\delta}$.

Chapter 3

Multiparametric Programming

The aim of the multiparametric programming is to characterize the optimal solution $\boldsymbol{x}^*$ of a given optimization problem for a full range of parameters $\boldsymbol{\theta} \subset \mathbb{R}^s$. Such result is referred to as *explicit solution* because it is expressed as a map in an explicit form, i.e. $\boldsymbol{x}^*(\boldsymbol{\theta}) : \mathbb{R}^s \mapsto \mathbb{R}^n$. If the parameters $\boldsymbol{\theta}$ are fixed numbers, the multiparametric problem will reduce to (2.1). Multiparametric programming is thus another way of solving optimization problems which gives exactly the same result as if the problem has been solved using any of the methods reviewed in Chapter 2. The main advantage of explicit solutions is their geometric interpretation which has a strong application in theory of optimal control. Some basic multiparametric problems will be introduced in the sequel.

3.1 Multiparametric Problems

For a sake of completeness, multiparametric versions of LP, QP and MILP problems will be introduced in this section. Multiparametric programs are emerging field in system engineering and recent developments in theory and applications are summarized in collection [83, 84].

3.1.1 Multiparametric Linear Programming

A multiparametric Linear Program (mpLP) is a convex optimization problem where the objective function and constraints are linear, and moreover, constraints depend linearly on a given vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^s$. More precisely,

$$\min_{\boldsymbol{x}} \ \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{\theta} \tag{3.1a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} + \boldsymbol{S}\boldsymbol{\theta} \tag{3.1b}$$

$$\boldsymbol{M}\boldsymbol{\theta} \leq \boldsymbol{l} \tag{3.1c}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a vector of optimization variables. Furthermore, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{d} \in \mathbb{R}^s$, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, $\boldsymbol{S} \in \mathbb{R}^{m\times s}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{M} \in \mathbb{R}^{p\times s}$, and $\boldsymbol{l} \in \mathbb{R}^p$. The only difference comparing to LP is that the solution of the problem (3.1) is searched for whole possible values of

parameters $\boldsymbol{\theta} \in \Theta$, $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^s \mid \boldsymbol{M\theta} \leq \boldsymbol{l}\}$, i.e. (3.1c). Let $\mathcal{X} \subset \mathbb{R}^n$ denote the set of $\boldsymbol{x}^*$ for which the problem (3.1) was feasible. Then, the explicit solution will be a mapping $\boldsymbol{x}^*(\boldsymbol{\theta}) : \Theta \mapsto \mathcal{X}$. The attribute multiparametric refers to presence of multiple parameters, otherwise the problem boils down to LP.

3.1.2 Multiparametric Quadratic Programming

A multiparametric Quadratic Program (mpQP) is a convex optimization problem where the objective function is quadratic, constraints are linear, and the right hand side of constraints contains a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^s$ which depend linearly on $\boldsymbol{\theta}$, i.e.

$$\min_{\boldsymbol{x}} \; 0.5\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + 0.5\boldsymbol{\theta}^T\boldsymbol{Q}\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{x} \tag{3.2a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} + \boldsymbol{S}\boldsymbol{\theta} \tag{3.2b}$$

$$\boldsymbol{M}\boldsymbol{\theta} \leq \boldsymbol{l} \tag{3.2c}$$

where $\boldsymbol{P} \succ 0$, $\boldsymbol{Q} \succeq 0$, $\boldsymbol{R} \succ 0$, $\boldsymbol{P} \in \mathbb{R}^{n\times n}$, $\boldsymbol{Q} \in \mathbb{R}^{s\times s}$, $\boldsymbol{R} \in \mathbb{R}^{s\times n}$, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{S} \in \mathbb{R}^{m\times s}$ $\boldsymbol{M} \in \mathbb{R}^{p\times s}$ and $\boldsymbol{l} \in \mathbb{R}^p$. Contrary to QP, the solution is searched for the whole range of parameters $\boldsymbol{\theta} \in \Theta$, $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^s \mid \boldsymbol{M\theta} \leq \boldsymbol{l}\}$, i.e (3.2c). Let $\mathcal{X} \subset \mathbb{R}^n$ denote the set of $\boldsymbol{x}^*$ for which the problem (3.2) was feasible. Then, the solution will be a mapping $\boldsymbol{x}^*(\boldsymbol{\theta}) : \Theta \mapsto \mathcal{X}$. Equation (3.2b) express the polyhedral set as a function of $\boldsymbol{\theta}$ over which the feasible solution is to be found.

3.1.3 Multiparametric Mixed Integer Linear Programming

A multiparametric Mixed Integer Linear Program (mpMILP) is a non-convex optimization problem where the objective function and constraints are linear, and moreover, the variables belong to real and binary set. Moreover, the right hand side of constraints contains a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^s$ which depend linearly on $\boldsymbol{\theta}$. Precisely,

$$\min_{\boldsymbol{x},\boldsymbol{\delta}} \; \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{\delta} \tag{3.3a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{x} + \boldsymbol{T}\boldsymbol{\delta} \leq \boldsymbol{h} + \boldsymbol{S}\boldsymbol{\theta} \tag{3.3b}$$

$$\boldsymbol{M}\boldsymbol{\theta} \leq \boldsymbol{l} \tag{3.3c}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\delta} \in \{0,1\}^q$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{d} \in \mathbb{R}^q$, $\boldsymbol{G} \in \mathbb{R}^{m\times n}$, $\boldsymbol{T} \in \mathbb{R}^{m\times q}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{S} \in \mathbb{R}^{m\times s}$, $\boldsymbol{M} \in \mathbb{R}^{p\times s}$ $\boldsymbol{l} \in \mathbb{R}^p$. However, for a given combination of fixed binary variables $\boldsymbol{\delta}$, the problem (3.3) boils down to mpLP. For fixed $\boldsymbol{\delta}$'s, there exist up to 2^q possible combinations of binary couples has to be explored but due to branch and bound techniques for solving MILP, this number is gradually reduced.

3.2 Solving Multiparametric Problems

The aim of this section is to present one of the methods for solving multiparametric problems. More details about the efficient strategies for solving mpLP, mpQP and mpMILP

problems are given in [11, 16, 23]. An efficient version of the multiparametric algorithm appeared in [105] and more recently in [57]. In fact, the overall principle is the same thus only mpQP case is detailed here.
Given $\boldsymbol{P} \succ 0$ in mpQP problem (3.2), it is convenient to introduce a new variable $\boldsymbol{z} \in \mathbb{R}^n$

$$\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{P}^{-1}\boldsymbol{R}\boldsymbol{\theta}. \tag{3.4}$$

By substitution of (3.4) in (3.2), the problem is transformed to a form

$$\min_{\boldsymbol{z}} \quad 0.5\boldsymbol{z}^T\boldsymbol{P}\boldsymbol{z} \tag{3.5a}$$
$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{h} + \boldsymbol{T}\boldsymbol{\theta} \tag{3.5b}$$

where $\boldsymbol{T} = \boldsymbol{S} + \boldsymbol{G}\boldsymbol{P}^{-1}\boldsymbol{R}$. It is very important to note that parameter $\boldsymbol{\theta}$ enters the constraints (3.5b) linearly. The transformed problem (3.5) is tackled by investigating the necessary Karush-Kuhn-Tucker optimality conditions which provide the basic apparatus for solving mpQP. Expressing these conditions for problem (3.5) gives

$$\boldsymbol{P}\boldsymbol{z} + \boldsymbol{G}^T\boldsymbol{\lambda} = \boldsymbol{0}, \quad \boldsymbol{\lambda} \in \mathbb{R}^q \tag{3.6a}$$
$$\boldsymbol{\lambda}_i(\boldsymbol{G}_i\boldsymbol{z} - \boldsymbol{h}_i - \boldsymbol{T}_i\boldsymbol{\theta}) = \boldsymbol{0}, \quad i = 1, \ldots, q \tag{3.6b}$$
$$\boldsymbol{\lambda} \geq \boldsymbol{0} \tag{3.6c}$$
$$\boldsymbol{G}\boldsymbol{z} - \boldsymbol{h} - \boldsymbol{T}\boldsymbol{\theta} \leq \boldsymbol{0} \tag{3.6d}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. From (3.6a) one immediately obtains

$$\boldsymbol{z}^* = -\boldsymbol{P}^{-1}\boldsymbol{G}^T\boldsymbol{\lambda} \tag{3.7}$$

which turns out to be a linear relation between $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ through the substitution (3.4). As the relation (3.4) is linear in $\boldsymbol{\theta}$, relation (3.7) is also linear with respect to $\boldsymbol{\theta}$. Solving the problem (3.5) for a particular value of $\boldsymbol{\theta}$ gives expression for $\boldsymbol{\lambda}$ and since $\boldsymbol{\lambda}$ is constraint-dependent, denote the part $\boldsymbol{\lambda}_{\mathcal{A}}$ as the vector of Lagrange multipliers corresponding to active constraints $\boldsymbol{\lambda}_{\mathcal{A}} > \boldsymbol{0}$, and $\boldsymbol{\lambda}_{\mathcal{I}}$ to inactive constraints $\boldsymbol{\lambda}_{\mathcal{I}} = \boldsymbol{0}$. Similarly, define

$$\mathcal{A}(\boldsymbol{\theta}) = \{i \,|\, \boldsymbol{G}_i\boldsymbol{z}(\boldsymbol{\theta}) = \boldsymbol{h}_i + \boldsymbol{T}_i\boldsymbol{\theta}\} \tag{3.8}$$

which is the set of indices of active constraints at the optimum. The set (3.8) is obtained by solving QP for a feasible initial value of $\boldsymbol{\theta}$. Subsequently, rows indexed by the active constraints $\mathcal{A}(\boldsymbol{\theta})$ are extracted from the constraint matrices $\boldsymbol{G}$, $\boldsymbol{h}$, $\boldsymbol{T}$ to form matrices $\boldsymbol{G}_{\mathcal{A}}$, $\boldsymbol{h}_{\mathcal{A}}$, $\boldsymbol{T}_{\mathcal{A}}$ considering only this limited set of constraints. Substituting (3.7) into equality constraints (3.6b) gives

$$-\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T\boldsymbol{\lambda}_{\mathcal{A}} + \boldsymbol{h}_{\mathcal{A}} + \boldsymbol{T}_{\mathcal{A}}\boldsymbol{\theta} = \boldsymbol{0} \tag{3.9}$$

which in turn defines set of active multipliers

$$\boldsymbol{\lambda}_{\mathcal{A}} = (\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T)^{-1}(\boldsymbol{h}_{\mathcal{A}} + \boldsymbol{T}_{\mathcal{A}}\boldsymbol{\theta}). \tag{3.10}$$

Combining (3.10) with equation (3.7) the expression for $\boldsymbol{z}^*$ is

$$\boldsymbol{z}^* = -\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T(\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T)^{-1}(\boldsymbol{h}_{\mathcal{A}} + \boldsymbol{T}_{\mathcal{A}}\boldsymbol{\theta}) \tag{3.11}$$

which is a solution to (3.5). In the back-transformation, the minimizer $\boldsymbol{x}^*$ of the problem (3.2) is given as

$$\boldsymbol{x}^* = \boldsymbol{F}_r\boldsymbol{\theta} + \boldsymbol{g}_r \tag{3.12}$$

which is an affine function in $\boldsymbol{\theta}$ where

$$\boldsymbol{F}_r = \boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T(\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T)^{-1}\boldsymbol{T}_{\mathcal{A}} - \boldsymbol{P}^{-1}\boldsymbol{R}^T \tag{3.13}$$

$$\boldsymbol{g}_r = \boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T(\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T)^{-1}\boldsymbol{h}_{\mathcal{A}}. \tag{3.14}$$

Moreover, as the active constraints are defined over the set $\mathcal{A}(\boldsymbol{\theta})$, conditions (3.6c) and (3.6d) must be satisfied. Therefore by plugging (3.10) in (3.6c) and (3.11) in (3.6d) one obtains a polyhedral description in the parameter space where $\boldsymbol{x}^*$ is valid

$$\mathcal{P}_r = \{\boldsymbol{\theta} \in \mathbb{R}^s \,|\, \boldsymbol{H}_r\boldsymbol{\theta} \leq \boldsymbol{l}_r\} \tag{3.15}$$

with

$$\boldsymbol{H}_r = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{F}_r + \boldsymbol{P}^{-1}\boldsymbol{R}^T) - \boldsymbol{T} \\ (\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T)\boldsymbol{T}_{\mathcal{A}} \end{pmatrix} \tag{3.16}$$

$$\boldsymbol{l}_r = \begin{pmatrix} \boldsymbol{h} - \boldsymbol{G}\boldsymbol{g}_r \\ -(\boldsymbol{G}_{\mathcal{A}}\boldsymbol{P}^{-1}\boldsymbol{G}_{\mathcal{A}}^T)\boldsymbol{h}_{\mathcal{A}} \end{pmatrix}. \tag{3.17}$$

Note that the optimizer (3.12) is actually associated with the region of active constraints (3.15). To cover the whole feasible area, the algorithm traverses through the parameter space and iteratively detects the active sets (3.8). By this way a sequence of affine functions (3.12) is generated, each corresponding to given region (3.15). The final result of the multiparametric problem forms $r = 1, \ldots, n_R$ partitions of PWA function defined over $\mathcal{P}_f = \bigcup \mathcal{P}_r$ partitions where $\mathcal{P}_f$ is called a *feasible set*.
In summary, mpQP algorithm can be summarized into following steps:

1. **Active constraint identification:** For a given initial feasible $\boldsymbol{\theta}$ search a set of active constraints (3.8) by solving (3.5). Extract the particular rows from matrices $\boldsymbol{G}$, $\boldsymbol{h}$, $\boldsymbol{T}$ and form matrices $\boldsymbol{G}_{\mathcal{A}}$, $\boldsymbol{h}_{\mathcal{A}}$, $\boldsymbol{T}_{\mathcal{A}}$ to obtain optimizer (3.11).

2. **Region computation:** Obtain explicit representation of the optimizer $\boldsymbol{x}^*$ (3.12) and the corresponding region (3.15).

3. **State space exploration:** If the first region is found, continue the search by change in parameters $\boldsymbol{\theta}$ and repeating steps 1. – 2. until the whole feasible domain is not covered. Partitioning the remainding space is done via reversing the sign in one of the inequality (3.15) and removing redundant inequalities, as suggested in [16]. Alternatively, one can use approach of [3].

This method for solving multiparametric problem mpLP, mpQP, and mpMILP are implemented in Multi-Parametric Toolbox (MPT) by [64] which is freely available on the internet.

3.3 Properties of the Explicit Solutions

The properties of the explicit solutions are stated by following theorems.

Theorem 3.1 (Properties of mpLP, [23]) *Consider the mpLP (3.1) with a linear objective function $\mathcal{J}(\boldsymbol{x},\boldsymbol{\theta}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{\theta}$. Then, the set of feasible parameters Θ is convex, there exists an optimizer $\boldsymbol{x}(\boldsymbol{\theta}) : \Theta \mapsto \mathbb{R}^n$ which is continuous and piecewise affine, i.e.*

$$\boldsymbol{x}^*(\boldsymbol{\theta}) = \boldsymbol{F}_r\boldsymbol{\theta} + \boldsymbol{g}_r \quad \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_r, \tag{3.18}$$

where $\mathcal{P}_r = \{\boldsymbol{\theta} \in \mathbb{R}^s \mid \boldsymbol{H}_r\boldsymbol{\theta} \leq \boldsymbol{l}_r\}, r = 1,\dots,n_R$ and the optimal value function $\mathcal{J}(\boldsymbol{x}^*,\boldsymbol{\theta}) : \Theta \mapsto \mathbb{R}$ is continuous, convex and piecewise affine.

Example 3.1 *Consider the following mpLP*

$$\begin{array}{ll} \underset{x}{min} & x \\ s.t. & \theta_1 - \theta_2 - x \leq 0.2 \\ & -0.5\theta_1 + \theta_2 - x \leq 0.4 \\ & -1 \leq x \leq 1 \\ & 0 \leq \theta_1 \leq 1 \\ & 0 \leq \theta_2 \leq 1 \end{array} \tag{3.19}$$

where x is the optimization variable and $\boldsymbol{\theta} = (\theta_1,\theta_1)^T$ is the vector of parameters. Solving the problem (3.19) parametrically, with the help of optimization tools such as YALMIP and MPT [64, 67], gives

$$x^* = \begin{cases} -0.5\theta_1 + \theta_2 - 0.4 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_1 \\ \theta_1 - \theta_2 - 0.2 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_2 \end{cases} \tag{3.20}$$

where

$$\mathcal{P}_1 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0.6 & -0.8 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} 0 \\ 1 \\ 1 \\ -0.08 \end{pmatrix} \right\},$$

$$\mathcal{P}_2 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ -0.6 & 0.8 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0.08 \end{pmatrix} \right\}.$$

Partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ are depicted in Fig. 3.1 (a). Optimizer x^* of the problem (3.19) is shown in Fig. 3.2 (a) and value function $J(x^*,\boldsymbol{\theta})$ is illustrated in Fig. 3.3 (a).

Theorem 3.2 (Properties of mpQP, [23]) *Consider mpQP (3.2). Then, the set of feasible parameters Θ is convex, the optimizer $\boldsymbol{x}^* : \Theta \mapsto \mathbb{R}^n$ is continuous and piecewise affine, i.e.*

$$\boldsymbol{x}^*(\boldsymbol{\theta}) = \boldsymbol{F}_r\boldsymbol{\theta} + \boldsymbol{g}_r \quad \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_r, \tag{3.21}$$

where $\mathcal{P}_r = \{\boldsymbol{\theta} \in \mathbb{R}^s \mid \boldsymbol{H}_r\boldsymbol{\theta} \leq \boldsymbol{l}_r\}, r = 1, \ldots, n_R$ and the optimal value function $\mathcal{J}(\boldsymbol{x}^*, \boldsymbol{\theta}) : \Theta \mapsto \mathbb{R}$ is continuous, convex and piecewise quadratic.

Example 3.2 *Consider the following mpQP*

$$\begin{aligned} \min_{x} \quad & x^2 \\ s.t. \quad & \theta_1 - \theta_2 - x \leq 0.2 \\ & -0.5\theta_1 + \theta_2 - x \leq 0.4 \\ & -1 \leq x \leq 1 \\ & 0 \leq \theta_1 \leq 1 \\ & 0 \leq \theta_2 \leq 1 \end{aligned} \tag{3.22}$$

where x is the optimization variable and $\boldsymbol{\theta} = (\theta_1, \theta_2)^T$ is the vector of parameters. Problem (3.22) differs from (3.19) only in the objective function, the constraint set is the same. Solving the problem (3.22) parametrically gives

$$x^* = \begin{cases} 0 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_1 \\ \theta_1 - \theta_2 - 0.2 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_2 \\ -0.5\theta_1 + \theta_2 - 0.4 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_3 \end{cases} \tag{3.23}$$

where

$$\mathcal{P}_1 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0.707 & -0.707 \\ -0.447 & 0.894 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0.141 \\ 0.358 \end{pmatrix} \right\},$$

$$\mathcal{P}_2 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ -0.707 & 0.707 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} 0 \\ 1 \\ -0.141 \end{pmatrix} \right\},$$

$$\mathcal{P}_3 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0.447 & -0.894 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} 0 \\ 1 \\ 1 \\ -0.358 \end{pmatrix} \right\}.$$

Partitions $\mathcal{P}_1$, $\mathcal{P}_3$, and $\mathcal{P}_3$ are depicted in Fig. 3.1 (b). Optimizer x^* of the problem (3.19) is shown in Fig. 3.2 (b) and value function $J(x^*, \boldsymbol{\theta})$ is illustrated in Fig. 3.3 (b).

Theorem 3.3 (Properties of mpMILP, [23]) *Consider the mpMILP (3.3). Then the set of feasible parameters Θ is a (possibly) non-convex polyhedral set and the value function $J(\boldsymbol{z}^*, \boldsymbol{\theta})$, $\boldsymbol{z}^*(\boldsymbol{\theta}) = (\boldsymbol{x}^*(\boldsymbol{\theta}), \boldsymbol{\delta}^*(\boldsymbol{\theta}))^T$, is piecewise affine on polyhedra. If the optimizer $\boldsymbol{z}^*(\boldsymbol{\theta})$ is unique for all $\boldsymbol{\theta} \in \Theta$, then the optimizer function $\boldsymbol{z}^*(\boldsymbol{\theta})$ is piecewise affine on polyhedra. Otherwise, it is always possible to define a piecewise affine optimizer function $\boldsymbol{z}^*(\boldsymbol{\theta}) \in Z(\boldsymbol{\theta})$, $Z(\boldsymbol{\theta}) := \Theta \mapsto 2^{\mathbb{R}^n} \times 2^{\{0,1\}^q}$ which describes for all $\boldsymbol{\theta} \in \Theta$ the set of optimizers $\boldsymbol{x}(\boldsymbol{\theta})$ related to $J(\boldsymbol{z}^*)$.*
Moreover, if the cost function $J(\boldsymbol{z}^*)$ is convex in $\boldsymbol{z}$ for each fixed $\boldsymbol{\theta}$, then it is always possible to define a piecewise affine and continuous optimizer function $\boldsymbol{z}^*(\boldsymbol{\theta}) \in Z(\boldsymbol{\theta})$.

Example 3.3 *Consider the following mpMILP*

$$\begin{aligned} \min_{x} \quad & \delta \\ s.t. \quad & \theta_2 + 0.5\delta \leq 1 \\ & -\theta_2 - 0.5\delta \leq -0.5 \\ & \theta_1 + \theta_2 + \delta \leq 2 \\ & 0 \leq \theta_1 \leq 1 \\ & 0 \leq \theta_2 \leq 1 \end{aligned} \tag{3.24}$$

where δ is the binary optimization variable and $\boldsymbol{\theta} = (\theta_1, \theta_2)^T$ is the vector of parameters. Solving the problem (3.24) parametrically gives

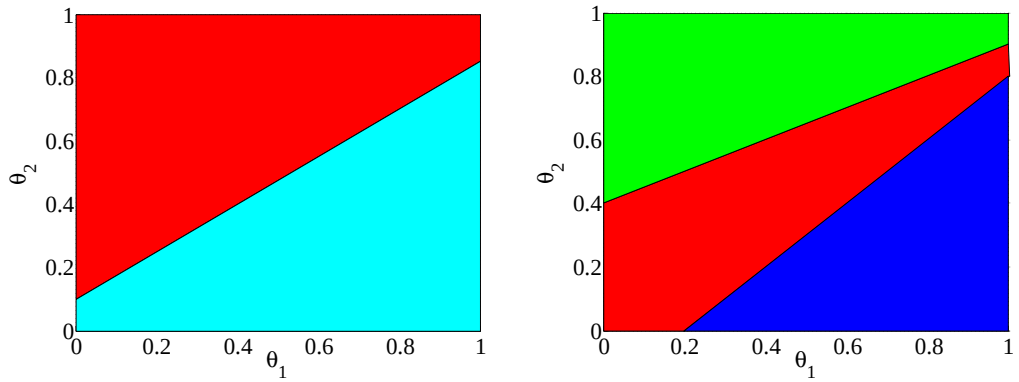
$$\delta^* = \begin{cases} 0 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_1 \\ 1 & \text{if} \quad \boldsymbol{\theta} \in \mathcal{P}_2 \end{cases} \tag{3.25}$$

where

$$\mathcal{P}_1 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} 0 & -1 \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} -0.5 \\ 1 \\ 1 \\ 0 \end{pmatrix} \right\},$$

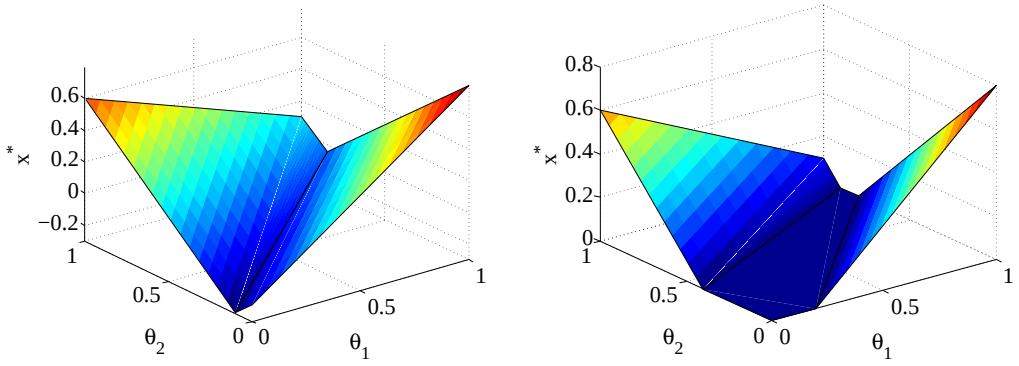
$$\mathcal{P}_2 := \left\{ \boldsymbol{\theta} \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} 0 & 1 \\ 0.707 & 0.707 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \boldsymbol{\theta} \leq \begin{pmatrix} 0.5 \\ 0.707 \\ 0 \\ 0 \end{pmatrix} \right\}.$$

Partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ are shown in Fig. 3.4(a). Note that the union of both partitions is non-convex. Optimal values of δ^* are shown in Fig. 3.4(b), one representing the value of 0 and the other one value of 1.



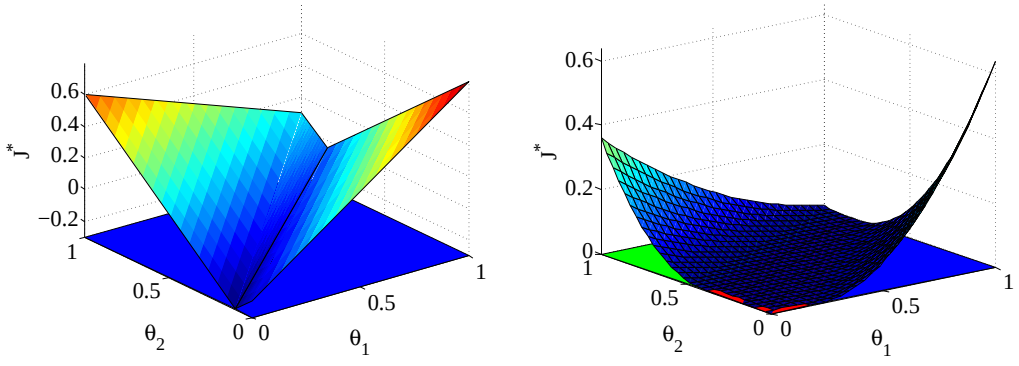
(a) (b)

Figure 3.1: Partitions of the explicit solution for mpLP (a) and mpQP (b).



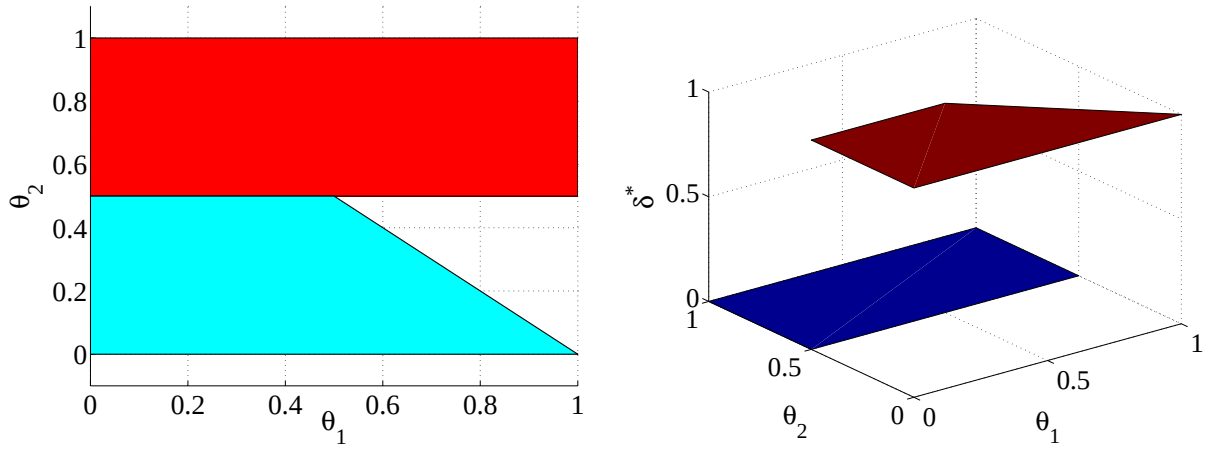
(a) (b)

Figure 3.2: Corresponding optimizer for mpLP (a) and mpQP (b).



(a) (b)

Figure 3.3: Value function for mpLP (a) and mpQP (b).



(a) (b)

Figure 3.4: Explicit solution to mpMILP problem in Example 3.3 consist of two partitions which create a non-convex union (a). The value function is equal to the optimizer in this case (b).

Chapter 4

Model Predictive Control

Model Predictive Control (MPC) is one of successful methodologies which is especially suitable for control of processes in the presence of physical constraints [25, 68]. Its applicability to industrial processes is excellently documented in [87] where an overview of available commercial packages is given. Currently, the notion of MPC covers a wide area of methods, which can be categorized depending on various criteria. In this chapter the principles of MPC will be explained and an overview of existing approaches will be given. It should be stressed here that the symbols used in this Chapter follow the convention of textbooks on control theory. In particular, $\boldsymbol{x}$ will be used to denote the plant's states, $\boldsymbol{u}$ are the plant's inputs and $\boldsymbol{y}$ are the plant's outputs. The optimization variables and parameters will be specified accordingly.

4.1 Predicting the Future

As the adverb predictive suggests, the idea of MPC is to somehow foresee the future and use this prediction to consequently make a decision. This behavior is typical in human reasoning and no wonder that it has been applied to process control. In fact, the first ever predictive approach is a standard proportional-integral-derivative (PID) controller. The prediction part is built by the derivative term which foresees the process evolution over an infinite small time horizon. The overall effect of PID controller is then composed of weighted past and future information. So far PID control has been the best possible solution for industry, providing cheap and reliable process management. However, in human reasoning the predictions are realized over larger horizons. This approach is called decision planning and one may remember such strategy e.g. from playing chess game. Forecasting the future affects our everyday life. For instance, one can imagine a situation when driving a car as given in Fig. 4.1. The future information for the driver is the shape of the road ahead with possible obstacles. Current information gives the front panel with the actual speed and engine revolutions, whereas past information can be seen in the rear mirror. Depending on these information the driver can decide which action should be taken in order to manage the left turn. In this situation the prior information comes from the future and the resulting driver's actions is turning the wheel to the left.

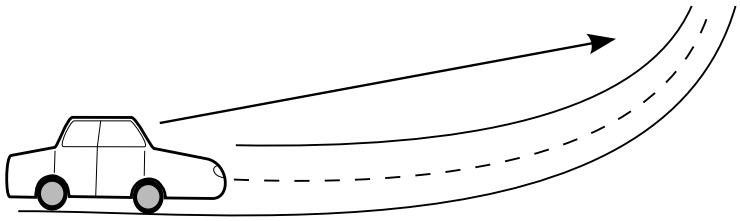


Figure 4.1: Driving a car is a predictive control strategy.

In order to carry over this decision making process for automatic control, its necessary to introduce the concept of *process model*. A model is a mathematical abstraction of real process which is usually described using input-output relations. Inputs are usually signals to the plant which can be manipulated (e.g. gas pedal) or can not (e.g. side wind, slope of the road). The actual information about the plant is given by *state variables*, such as car speed or engine revolutions, and one can read some output variables (e.g. data on the front panel). Usually, the updates about plant outputs are provided by measurements. Denoting $\boldsymbol{u}$ in general as plant inputs with dimension m, $\boldsymbol{x}$ as states with dimension n, $\boldsymbol{y}$ as outputs with dimension r, one such model can be given by

$$\boldsymbol{x}(t+T_s) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{4.1a}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{4.1b}$$

where $\boldsymbol{f}$, $\boldsymbol{g}$ are corresponding mathematical functions of $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$. The variable t represents time and T_s is a discrete sample time unit. The process model (4.1) gives the state prediction one step ahead, i.e. in time $t+T_s$ based on the current information from $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$. It is assumed that variables in (4.1) are functions of time, i.e. $\boldsymbol{u} = \boldsymbol{u}(t)$, $\boldsymbol{x} = \boldsymbol{x}(t)$, $\boldsymbol{y} = \boldsymbol{y}(t)$, but the time dependence notation will be mostly deprecated throughout the thesis.

Assuming that the process model (4.1) is given, this information can be utilized to obtain future predictions from current time t to $t + kT_s$, where $k = 1, 2, \ldots, N$ and N denotes the prediction horizon. To simplify the presentation, it is more convenient to replace the prediction interval $[t, t + kT_s]$ with time instances as $\{t_0, \ldots, t_N\}$ where $t_0 = t$ and $t_N = t + NT_s$. For dynamical systems it is always important to know initial condition $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$ which gives the initial constraint on state variables. Given $\boldsymbol{x}_0$, one can compute the time evolution of system (4.1), that is, for particular values of input $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}$ to obtain states $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and outputs $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_N$ at given prediction times $t_0, \ldots, t_N$. Future inputs $\boldsymbol{U} = (\boldsymbol{u}_0^T, \boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_{N-1}^T)^T$ can be taken to play the role of decision variables which influence the outputs, thus the model outputs can be managed to follow a certain goal. Now the question which arises is, how to manage the future inputs? The answer is, to formulate an optimal control problem.

4.2 Formulation of a General Optimal Control Problem

Due to increasing needs for constraints treatment in the process control, there is a strong motivation to express control problems as problems of constrained optimization. The optimization theory allows a suitable translation between the linguistic representation of a control problem to an equivalent mathematical form. A very general optimal control problem might be formulated as follows

Problem 4.1 (General Formulation of Optimal Control Problem) *Find optimal control inputs* $\boldsymbol{U}^* = (\boldsymbol{u}_0^T, \boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_{N-1}^T)^T$ *which drive the system from the current state* $\boldsymbol{x}_0$ *at time* t_0 *towards origin such that*

$$\begin{aligned} \textit{min} \quad & F(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} L(\boldsymbol{x}_k, \boldsymbol{u}_k) && \text{(4.2a)} \\ \textit{subject to:} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), && \text{(4.2b)} \\ & \boldsymbol{x}_k \in \mathcal{X} && \text{(4.2c)} \\ & \boldsymbol{u}_k \in \mathcal{U}. && \text{(4.2d)} \end{aligned}$$

Expression (4.2a) defines the cost function, (4.2b) is the process model and $\mathcal{X}$, $\mathcal{U}$ are the constraints on states (4.2c) and inputs (4.2d), respectively. Problem 4.1 is often referred to as constrained finite time optimal control (CFTOC) because of the presence of constraints and finite horizon N. If the optimization problem is formulated over an infinite horizon, it is referred to as constrained infinite time optimal control problem (CITOC). Predictions from the process model are taken into account up to N steps to the future and control inputs $\boldsymbol{U}^*$ are the optimized degrees of freedom. Optimal control problems, such as Problem 4.1, have been formulated in late 50's of the 20th century and the main motivation that time was driven by military and aerospace needs with applications in flight control. This research stream is often referred to as classical theory of optimal control and it has provided a very powerful tool for solving numerous practical problems [9, 68].
Assume that Problem 4.1 is feasible for $\boldsymbol{x}_0 = \boldsymbol{x}(t)$. Depending on how the control input $\boldsymbol{U}^*$ are expressed, the optimal solution to Problem 4.1 is characterized as follows:

implicit solution: The computed input $\boldsymbol{U}^*$ is given as a sequence of numerical values $\boldsymbol{u}_0^*$, $\boldsymbol{u}_1^*$, …, $\boldsymbol{u}_{N-1}^*$ which depend on the particular values of $\boldsymbol{x}_0$ at specific times within the interval $[t_0, t_N]$.

explicit solution: The control input $\boldsymbol{U}^*$ is given as a sequence of functions typically with plant state as its argument, i.e. $\boldsymbol{u}_0^* = \boldsymbol{\pi}_0(\boldsymbol{x}_0)$, $\boldsymbol{u}_1^* = \boldsymbol{\pi}_1(\boldsymbol{x}_0)$, etc.

In fact, the solution to CFTOC Problem 4.1 comprises of future optimal instances of inputs, states, outputs, and they form input/state/output trajectories. These trajectories represent the optimal evolution of a given process model from the current time t_0 up to

time t_N in the predicted future. Obtained sequence of inputs $\boldsymbol{U}$ can be fed to the system in an open-loop manner and the system states will move from $\boldsymbol{x}_0$ towards origin. However, it can happen that $\boldsymbol{x}_N$ may not be at the origin at t_N due to finite horizon N. One may try to recompute the solution to Problem 4.1 again, using $\boldsymbol{x}_N$ as the initial point, but it may cause CFTOC Problem 4.1 to become infeasible. Moreover, since the model is a simplification of real behavior, there is always mismatch between the real plant and the process model. The consequence of the mismatch can cause the real final state to differ from the prediction. Ignoring this fact may cause unwanted performance loss, and in the worst, case violation of constraints as well as instability of the closed loop [63]. Therefore, additional corrections via feedback implementation are necessary.

4.3 Closed-Loop Implementation of MPC

Applying the optimal solution to CFTOC problem in the closed loop setup does not guarantee convergence toward origin since the predicted and real trajectories may differ. The computed sequence of inputs is based on the actual information given by the state $\boldsymbol{x}_0$. Note that the actual feedback part of $\boldsymbol{U}$ is only the first element $\boldsymbol{u}_0$, and the rest of the computed trajectory is an open loop sequence. Thus, in order to introduce the feedback, $\boldsymbol{u}_0$ should be applied repetitively.

This way of implementation is called *receding horizon principle* and can be illustrated as shown in Fig. 4.2. Denote t_0 as the current time and t_1 as one time step forehead and t_{-1} as one step backward. Analogously, the states and inputs are defined with respect to their corresponding time. Consider the prediction horizon $N = 3$. Fig. 4.2(a) shows the predictions of state $\boldsymbol{x}$ with the corresponding input signal $\boldsymbol{u}$ up to time t_3 from given initial condition $\boldsymbol{x}_0$. The red axis refers to the current time t_0. Optimization algorithm searches the future evolution of the state up to time t_3 with manipulating the future inputs $\boldsymbol{U} = (\boldsymbol{u}_0^T, \boldsymbol{u}_1^T, \boldsymbol{u}_2^T)^T$ until there is no improvement in the optimization objective of Problem 4.1. When optimization terminates, the first element $\boldsymbol{u}_0$ is selected and applied to the plant. Fig. 4.2(b) shows the situation where new updates $\boldsymbol{x}_0$ arrived and the prediction is shifted one step ahead. The red axis has moved indicating that the red lines are now referring to the past. Note that there is a difference between the real evolution of the plant state (solid red trajectory) contrary to the prediction (dashed trajectory). The new state update $\boldsymbol{x}_0$ creates now initial condition for new predictions. The optimization problem needs to be recomputed again based on the new initial condition. If the optimal solution is available, the first element $\boldsymbol{u}_0$ is picked from the computed trajectory, fed back to the plant, and the same principle repeats as shown in Fig. 4.2(c).

The implementation scheme differs for the implicit and explicit type of solution. In the implicit case, the optimal inputs are given as sequence of numbers, thus it suffices to use the element from this sequence and apply it to the plant. However, the optimization problem needs to be recomputed again in the next sample to obtain a new sequence of inputs. The implementation of the implicit solution thus solves CFTOC every time as new state updates are collected. This scheme is shown in Fig. 4.3.

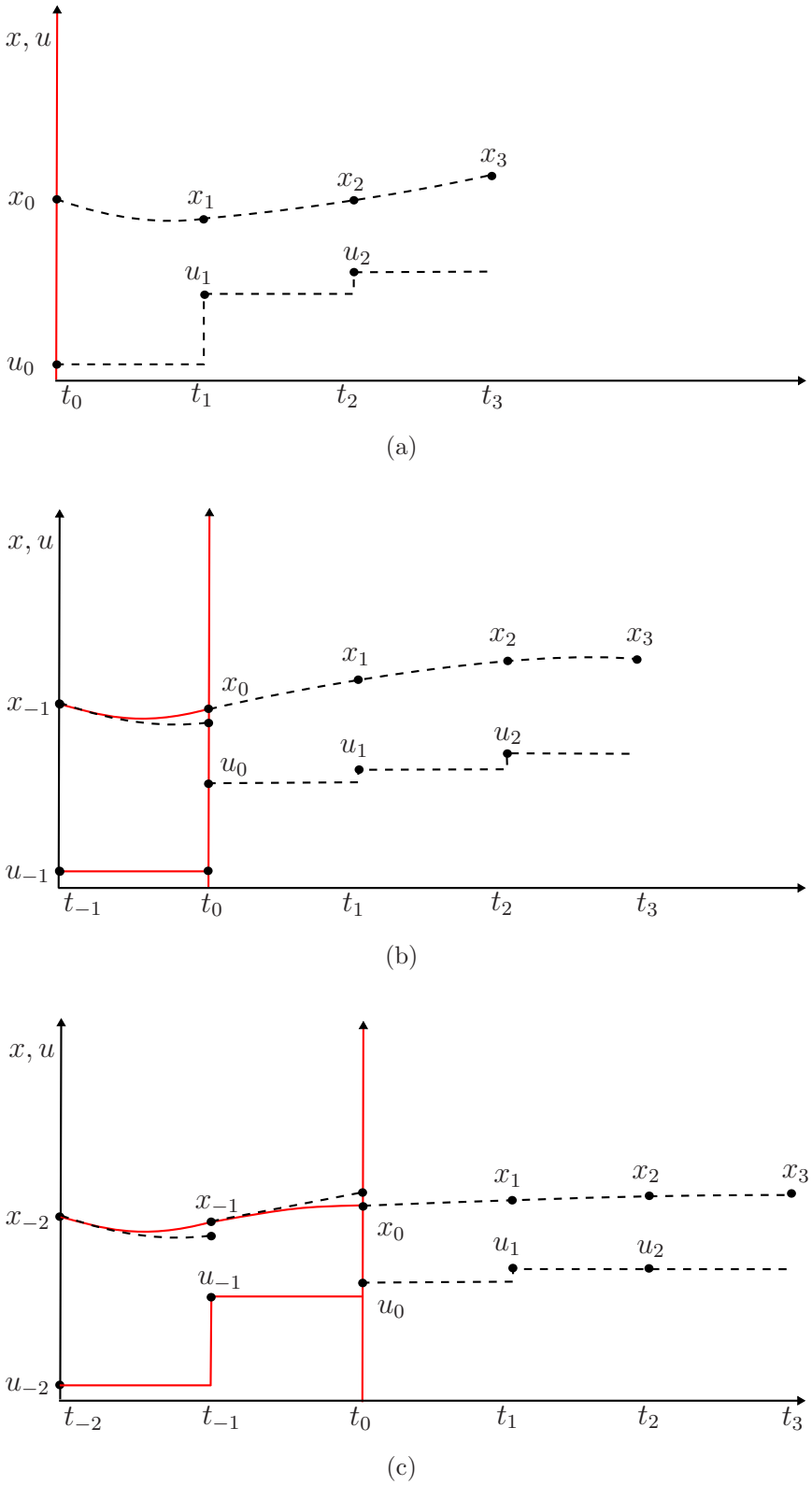


Figure 4.2: Receding horizon principle.

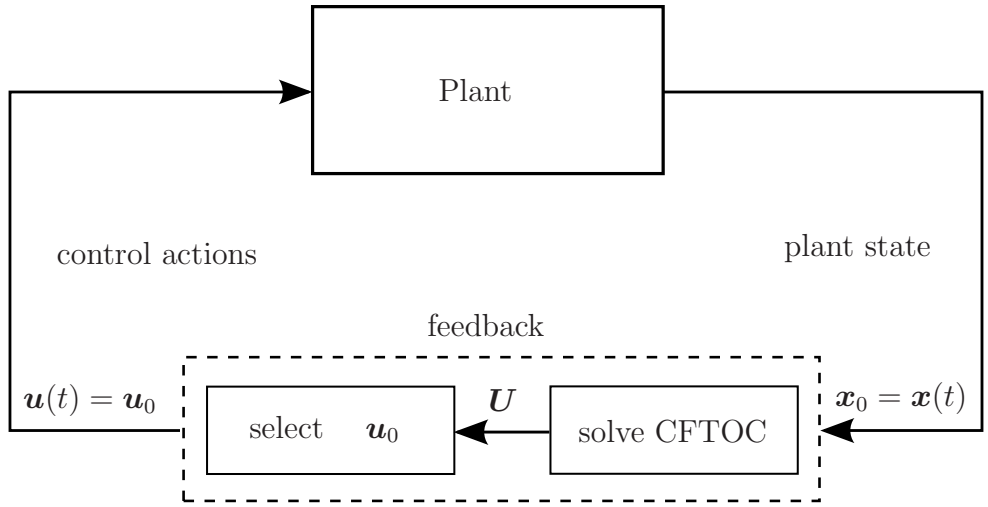


Figure 4.3: A feedback control scheme with implicit solution.

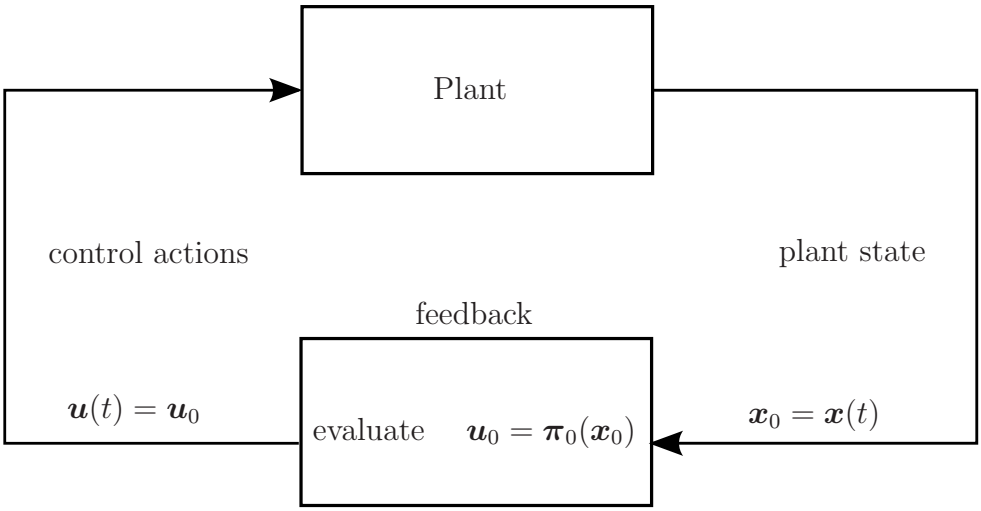


Figure 4.4: A feedback control scheme with explicit solution.

In the explicit case, the optimal inputs are expressed as functions of states, i.e. $\boldsymbol{u}_0^* = \boldsymbol{\pi}_0(\boldsymbol{x}_0)$, $\boldsymbol{u}_1 = \boldsymbol{\pi}_1(\boldsymbol{x}_0)$,..., $\boldsymbol{u}_{N-1}^* = \boldsymbol{\pi}_{N-1}(\boldsymbol{x}_0)$. To apply the receding horizon principle, the first function is evaluated for the actual value of state $\boldsymbol{x}_0$ and the value $\boldsymbol{u}_0^*$ is applied to the plant. Remaining functions are not needed since in the next sampling instant the same function $\boldsymbol{u}_0^* = \boldsymbol{\pi}_0(\boldsymbol{x}_0)$ is evaluated again, but this time with a new state. The scheme is depicted in Fig. 4.3.

There are advantages and disadvantages of both implementation schemes. The feedback control scheme using the implicit solution translates CFTOC problem to one of the optimization problems stated in Chapter 2 that can be efficiently solved with existing optimization tools. The scheme thus relies on the optimization solver at each sampling time. Typically, the solver is capable of dealing with large problems but it needs some time to

terminate. Therefore, this approach is suitable for plants with high number of optimized variables and constraints where the sampling time is long enough for the solver to terminate. The closed-loop scheme using explicit solution is much simpler to implement, as it only evaluates the given function for a changing state. CFTOC problem is transformed to a corresponding multiparametric program presented in Chapter 3 and solved explicitly for the whole range of operating parameters. However, such computation is much more difficult to obtain if the number of optimized parameters is large. The explicit MPC approach is thus suitable for plants with small number of variables and fast sampling.
Implementation of the optimal control law via receding horizon policy is based on the assumption that the solution to Problem 4.1 exists for all time $t > 0$. Therefore, one should formulate the optimal control problem with some feasibility guarantees. Stability of the closed-loop system is yet another issue of the feedback implementation. Both, feasibility and stability issues will be detailed later in Section 4.5. In the following sections the attention will be given to particular ingredients of CFTOC problem, namely, the cost function, the process model and the constraints.

4.4 Ingredients of MPC

This section explains the ingredients needed to formulate the optimal control Problem 4.1. In particular, a quick overview in the selection of cost functions and constraints is given. The main attention is paid to process models that will be further described in Part II.

4.4.1 Cost Function

The cost function in Problem 4.1 characterizes the performance criterion to be minimized with the predicted behavior of a process model up to horizon N. Typically, the cost function accounts the predicted trajectories of states $\boldsymbol{x}$ and future control moves $\boldsymbol{U}$ with respect to the desired reference point. Without loss of generality, assume that the given reference point is the origin, which is a typical goal for a regulation problem. In order to reach the maximum performance, the predictions should be counted up to infinite horizon $N \to \infty$. Such objective can be expressed as follows

$$J(\boldsymbol{x}, \boldsymbol{U}) = \sum_{k=0}^{\infty} L(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.3}$$

where $L(\boldsymbol{x}, \boldsymbol{u}) : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}$ is a function of states and inputs at k-th step during the prediction. Although there exist approaches that can handle CITOC problems [5, 78, 86], it is standard practice to split the objective function (4.3) to a finite sum and terminal cost, i.e.

$$J(\boldsymbol{x}, \boldsymbol{U}) = F(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} L(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.4}$$

where $F(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ is a terminal cost function. The truncated cost function (4.4) approximates the cost function with infinite horizon (4.3) but it requires a finite number of optimization variables. The choice of $F(\cdot)$ and $L(\cdot)$ depends on the particular control problem. A very common formulation of cost function (4.4) is as follows

$$F(\boldsymbol{x}) = \|\boldsymbol{P}\boldsymbol{x}\|_p \tag{4.5a}$$

$$L(\boldsymbol{x}, \boldsymbol{u}) = \|\boldsymbol{Q}\boldsymbol{x}\|_p + \|\boldsymbol{R}\boldsymbol{u}\|_p \tag{4.5b}$$

where $\|\cdot\|_p$ is a p-norm with $p = \{1, 2, \infty\}$ and $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$ are weighting factors.
Time optimal control problems use the cost function in the form

$$J = N \tag{4.6}$$

which accounts only prediction horizon. The control objective is to minimize the number of predicted steps.

4.4.2 Process Models

Models play a very important role in MPC, as they are the only source of predicting future. Mathematical modeling of a real physical process utilizes usually 1) theoretical aspects such as laws of mass and energy conservation, mechanical principles, etc., 2) empirical aspects such as observation, measurement, or 3) combination of both. The result of mathematical modeling is a process model describing a process behavior with respect to time and it can be given in one of the following form:

Input-output models: Predicted outputs $\boldsymbol{y}_0$, $\boldsymbol{y}_1$, …, $\boldsymbol{y}_n$ are function of past outputs and past inputs $\boldsymbol{u}_0$, $\boldsymbol{u}_1$, …, $\boldsymbol{u}_n$, i.e.

$$\boldsymbol{y}_n = \boldsymbol{f}(\boldsymbol{y}_{n-1}, \ldots, \boldsymbol{y}_0, \boldsymbol{u}_n, u_{n-1}, \ldots, \boldsymbol{u}_0). \tag{4.7}$$

Input-output models are easily obtained in practice, e.g. by investigating step or input responses of the plant. Therefore, several approaches to MPC has been developed which use input-output models, e.g. dynamic matrix control (DMC) approach [36, 85], internal model control (IMC) [42], generalized predictive control (GPC) [35]. In fact, GPC approach turned to be as one of the very successful control methodologies and the original paper became the 3rd most cited article in Automatica [109]. A detailed overview on GPC with industrial applications in given in [25].

State-space models: The state-space formulation is given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{4.8a}$$

$$\boldsymbol{y}_k = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.8b}$$

where the subscript k denotes the discrete time instance. Equation (4.8a) represents the plant dynamics and (4.8b) is the output equation. The structure of state-space models provides useful information for investigating closed loop properties, e.g. stability. Derivations of MPC approaches with state space models is evident in [69, 71, 90] and later in [68].

In principle, input-output models (4.7) and state-space models (4.8) are equivalent, as they can be translated from one form to another. State-space models have the advantage that they contain more information about the plant due to the presence of state variables $\boldsymbol{x}$ which is missing in the input-output representation. Moreover, most of optimal control theory is based on state-space models, hence in the remainder of the thesis only state-space representation will be considered.
Goal of the modeling process is to obtain the simplest representation of the real plant while capturing its most important dynamical properties. However, the model will never be perfect, thus some discrepancy will be always present. To account this difference, uncertain models have been introduced. In particular,

Models with additive disturbances: Dynamic equation of the model is affected by disturbance term $\boldsymbol{d}$ from a bounded set $\mathcal{D}$, i.e.

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{d}_k,\ \boldsymbol{d}_k \in \mathcal{D}. \tag{4.9}$$

Note that the uncertainty $\boldsymbol{d}$ enters the equation (4.9) linearly which has important consequences in robust control design [89].

Models with parametric uncertainties: The uncertainty affects the parameters $\boldsymbol{p}$ which are part of the function $\boldsymbol{f}$, i.e.

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{p}_k, \boldsymbol{x}_k, \boldsymbol{u}_k),\ \boldsymbol{p}_k \in \mathcal{P}. \tag{4.10}$$

Parameters $\boldsymbol{p}$ are not constants but they are allowed to vary in bounded set $\mathcal{P}$.

To meet the main goals of the thesis, an overview of hybrid models needs to be introduced. Specifically, the following models are considered:

Linear Time Invariant (LTI) models: The dynamic equation (4.8a) has a specific form where the functions $\boldsymbol{f}$, $\boldsymbol{g}$ are given as a linear combination of states and inputs, i.e.

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{4.11a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k \tag{4.11b}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$, $\boldsymbol{C} \in \mathbb{R}^{r\times n}$, $\boldsymbol{D} \in \mathbb{R}^{r\times m}$ are constant matrices. LTI models are widely used in MPC due to their relatively simple structure, defined by matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$, which is valid over the whole space of interest. Moreover, LTI models can be easily obtained in practice, e.g. using techniques from experimental identification [72, Chapter 6]. Formulation and solution to optimal control problems involving LTI models has been extensively studied in many textbooks, for instance [58, 68, 72].

Hybrid models: Hybrid models are a special class of systems where the equation for system dynamics is combined with logical rules. The consequence of logical operations is that variables describing the model are defined not only in the class of real numbers,

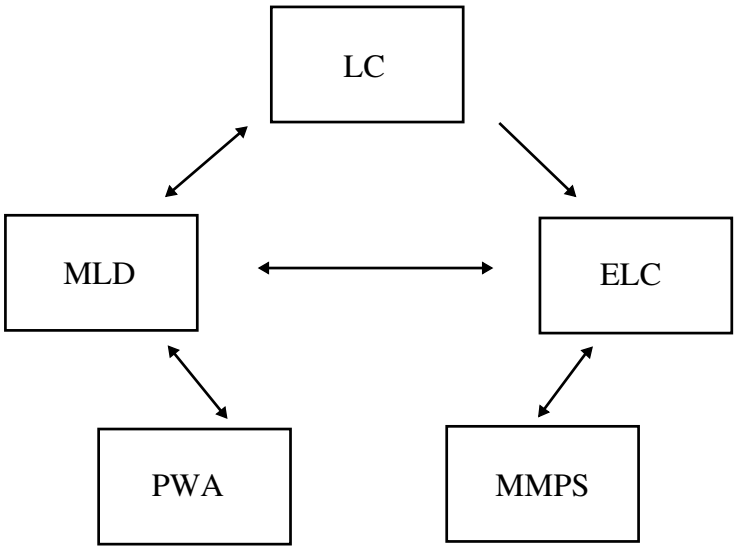


Figure 4.5: Equivalence of hybrid dynamical models [48].

but also Boolean variables are present. A hybrid model can be given in the form of (4.8) whereas additional conditions have to be provided to specify the relation between binary and real variables. Fundamentals of integrating logical elements into predictive control framework are documented in [15] and the interested reader may find more in [65]. It has been shown in [48] that the behavior of hybrid systems can be captured using many forms, in particular

- Mixed Logical Dynamics (MLD) systems
- Piecewise Affine (PWA) systems
- Linear complementarity (LC) systems
- Extended linear complementarity (ELC) systems
- Max-min-plus-scaling (MMPS) systems

and their dynamical behavior is equivalent. The equivalence principle allows one to transform from one representation to another and opens the possibility to select the desired hybrid modeling framework. Illustration of the equivalence principle is shown in Fig. 4.5. For the use in this thesis, only MLD and PWA systems will be reviewed.

Piecewise Affine (PWA) models: PWA systems are a special class of hybrid systems which can approximate the dynamic behavior of general nonlinear systems with arbitrary precision [94]. PWA systems have strong advantages for the use in optimal control. From a practical point of view, they can be easily obtained by linearization technique at various operating points. On a theoretical side,

they maintain a piecewise linear structure which is very useful for optimization. Their dynamical behavior is given by in the state space form (4.8) and are composed of multiple linear models, i.e.

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.12a}$$

$$= \boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i \quad \text{if} \quad \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} \in \mathcal{D}_i \tag{4.12b}$$

where matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{c}_i$, $i = 1, \ldots, n_{\mathcal{D}}$ define the local linear model and $n_{\mathcal{D}}$ is the total number of dynamics. The specificity of PWA model (4.12) lies in IF-THEN logical rules, that is, whenever state-input vector $(\boldsymbol{x}_k^T, \boldsymbol{u}_k^T)^T$ lies inside a polytopic set

$$\mathcal{D}_i := \left\{ \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{pmatrix} \in \mathbb{R}^{n+m} \,\middle|\, \boldsymbol{H}_i \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{pmatrix} \leq \boldsymbol{l}_i \right\}, \tag{4.13}$$

local dynamics i is selected. In order to avoid multiple updates $\boldsymbol{x}_{k+1}$, there can be only one dynamics associated to given partition which comes with the following assumption:

Assumption 4.1 (Well-posedness) *PWA model in (4.12) is well-posed according to Definition 1 in [15] if the regions (4.13) are not overlapping, i.e.* $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ *for* $i \neq j$.

PWA model thus acts as dynamical selection from table of local models. Comparing to LTI systems the evolution of PWA model can be non-smooth due to switching, thus MPC based on PWA models has to deal also with these issues. An overview of PWA based MPC approaches is given in [3, 23, 32].

Mixed Logical Dynamical (MLD) Models: Mixed Logical Dynamical (MLD) systems describe in general the behavior of linear discrete-time systems with integrated logical rules. MLD systems have been introduced by [15], and the model is given by a set of following equations

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}_u \boldsymbol{u}_k + \boldsymbol{B}_{\text{aux}} \boldsymbol{w}_k + \boldsymbol{B}_{\text{aff}} \tag{4.14a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}_u \boldsymbol{u}_k + \boldsymbol{D}_{\text{aux}} \boldsymbol{w}_k + \boldsymbol{D}_{\text{aff}} \tag{4.14b}$$

$$\boldsymbol{E}_x \boldsymbol{x}_k + \boldsymbol{E}_u \boldsymbol{u}_k + \boldsymbol{E}_{\text{aux}} \boldsymbol{w}_k \leq \boldsymbol{E}_{\text{aff}} \tag{4.14c}$$

where (4.14a) is the state-update equation, (4.14b) is the output equation, and linear inequalities (4.14c) describe the switching conditions expressed as inequalities. MLD model (4.14) uses following notations: $\boldsymbol{x} \in \mathbb{R}^{n_{x_r}} \times \{0,1\}^{n_{x_b}}$ is a vector of real and binary states, $\boldsymbol{u} \in \mathbb{R}^{n_{u_r}} \times \{0,1\}^{n_{u_b}}$ are the (real and binary) inputs, $\boldsymbol{y} \in \mathbb{R}^{n_{y_r}} \times \{0,1\}^{n_{y_b}}$ vector of (real and binary) outputs, $\boldsymbol{w} \in \mathbb{R}^{n_z} \times \{0,1\}^{n_d}$ represent auxiliary real, and binary variables, respectively, and $\boldsymbol{A}$, $\boldsymbol{B}_u$, $\boldsymbol{B}_{\text{aux}}$, $\boldsymbol{B}_{\text{aff}}$, $\boldsymbol{C}$, $\boldsymbol{D}_u$, $\boldsymbol{D}_{\text{aux}}$, $\boldsymbol{D}_{\text{aff}}$, $\boldsymbol{E}_x$, $\boldsymbol{E}_u$, $\boldsymbol{E}_{\text{aux}}$, $\boldsymbol{E}_{\text{aff}}$ are matrices of suitable dimensions. For a given state $\boldsymbol{x}_k$ at discrete time instant k and input $\boldsymbol{u}_k$ the evolution of the MLD system (4.14) is determined by solving $\boldsymbol{w}_k$ from (4.14c) and updating

$\boldsymbol{x}_{k+1}$, $\boldsymbol{y}_k$. In order to avoid multiple solutions to (4.14), MLD system must be *well-posed*, i.e. there must exist only unique solution $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_k$ once $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are specified [15].

The specificity of MLD model (4.14) contrary to general case (4.8) comes from its internal structure. That is, relations between variables are linear and operating constraints (4.20) are integral part of the model. The practical advantage of MLD models is that they can be generated from basic logical rules used very common in industry.

Takagi-Sugeno (TS) fuzzy models: Fuzzy sets and systems have been introduced by [114] and they represent objects with different grades of membership. A fuzzy system is characterized by a base of rules with associated membership functions and its properties are determined by a combination of these rules. A great advantage of fuzzy systems is their ability to approximate almost any dynamic systems with arbitrary precision [100].

The class of discrete-time TS models [98] is described by fuzzy "IF ... THEN" rules which specify interpolating conditions between $i = 1, \ldots, n_{\mathcal{D}}$ local dynamics. Contrary to PWA models more than one local model can be valid at the same time. This is the fundamental difference comparing to PWA models where only one model can be selected. Generally, TS model is given as

$$\text{IF} \quad \boldsymbol{x}_{1,k} \text{ is } \mu_{i1} \text{ and } \ldots \ \boldsymbol{x}_{n,k} \text{ is } \mu_{in} \quad \text{THEN} \quad \boldsymbol{x}_{k+1} = \boldsymbol{A}_i\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \tag{4.15}$$

where μ_{ij} are fuzzy sets for $i = 1, \ldots, n_{\mathcal{D}}$ dynamics and $j = 1, \ldots, n$ with n being the dimension of $\boldsymbol{x}$. Matrices $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{c}_i$ define the local dynamics and $\mu_{ij} \in [0, 1]$ is the fuzzy membership function. The aggregated system output is inferred as

$$\boldsymbol{x}_{k+1} = \frac{\sum_{i=1}^{n_{\mathcal{D}}} w_i(\boldsymbol{x}_k)(\boldsymbol{A}_i\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i)}{\sum_{i=1}^{n_{\mathcal{D}}} w_i(\boldsymbol{x}_k)} \tag{4.16}$$

with

$$w_i(\boldsymbol{x}_k) = \prod_{j=1}^{n} \mu_{ij}(\boldsymbol{x}_{j,k}) \tag{4.17}$$

where the membership function $\mu_{ij}(\boldsymbol{x}_{j,k})$ measures the activation of the fuzzy set j in the rule i. Using the definition

$$\alpha_i(\boldsymbol{x}_k) = \frac{w_i(\boldsymbol{x}_k)}{\sum_{i=1}^{n_{\mathcal{D}}} w_i(\boldsymbol{x}_k)}, \ \alpha_i(\boldsymbol{x}_k) \geq 0 \tag{4.18}$$

the overall system model can be described as

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{n_{\mathcal{D}}} \alpha_i(\boldsymbol{x}_k)(\boldsymbol{A}_i\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i). \tag{4.19}$$

The use of fuzzy models in control is excellently documented in a survey paper by [40].

4.4.3 Constraints

Physical, safety, environmental, or economical constraints are often entering the formulations of optimal control problems. Presence of constraints in optimization problems is usually a consequence of real world limitations to a mathematical abstraction represented by process model. In fact, the model of the form (4.1) is an equality constraint, restricting the optimal solution to be searched along the predicted trajectory. Besides equality constraints there exist also inequality constraints which define the operating space of variables. Examples of such constraints are for instance the non-negativity of liquid flow, finiteness of equipment performance, boundedness of supply resources etc. Constraints are thus very important ingredient of MPC as they can be directly considered in the control design. Practically, this ability makes MPC superior to other control approaches.
Constraints on state and control variables can be separated according to their validity over the prediction horizon $[t_0, t_N]$, into following categories:

Interior constraints: The constraints are imposed at each predicted times t_1, t_2, ..., t_{N-1} inside the prediction interval, i.e.

$$\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{u}_i) = \mathbf{0}, \tag{4.20a}$$
$$\boldsymbol{\psi}(\boldsymbol{x}_i, \boldsymbol{u}_i) \leq \mathbf{0}. \tag{4.20b}$$

Initial constraints: These constraints actually determine the feasible set of initial conditions at time t_0, i.e.

$$\boldsymbol{\phi}(\boldsymbol{x}_0) = \mathbf{0}, \tag{4.20c}$$
$$\boldsymbol{\psi}(\boldsymbol{x}_0) \leq \mathbf{0}. \tag{4.20d}$$

Terminal constraints: The constraints are imposed only at the final time t_N, i.e.

$$\boldsymbol{\phi}(\boldsymbol{x}_N) = \mathbf{0}, \tag{4.20e}$$
$$\boldsymbol{\psi}(\boldsymbol{x}_N) \leq \mathbf{0}. \tag{4.20f}$$

Constraints have a direct consequence on the feasibility of the resulting optimization problem. For instance, investigation of the initial constraint set can help to determine the feasibility of the optimization problem even without solving it. If the initial constraint set is empty, then the problem does not have any solution. Therefore, the optimization problem should be formulated in such a manner that there are feasibility guarantees which in turn help to ensure stability of the closed loop [93]. Further issues regarding the closed-loop stability are discussed in the next section.

4.5 Stability Requirements

Due to safety reasons it is often required that the closed loop system remains in the given operating range and does not uncontrollable drifts away. Theoretically speaking, the

closed loop system must be stable, i.e. resistant to small perturbations of the initial state. Stability thus plays an important role in the control design and has also to be considered in MPC approach. Due to model/plant mismatch, finite horizon, unaccountable disturbances, the predicted trajectories may differ from real ones. Thus, the computed optimal action may be based on wrong future assessment and it may cause instability. Instability can originate also from the process model, for instance if it contains unstable poles.

The problem of ensuring stability in the closed loop implementation of MPC has been studied very extensively in the literature and various modifications in the construction of the optimization problem have been proposed. A common feature of these approaches lies in incorporation of some correction factor in the predictions such that given goal is truly achieved. Looking at the structure of general optimal control Problem 4.1 this factor can be part of objective function or constraints, as the model is usually given. An excellent survey paper [70] deals with this topic exhaustively in details and the main results are presented here.

Since the early development of predictive control in 1970's – 1990's, stability was mostly achieved by extensive tuning. This was the easiest way until Keerthi and Gilbert [59] (1988) established the connection with Lyapunov stability theory for discrete-time systems and Mayne and Michalska [69] (1990) for continuous-time systems. Since then the value function was almost universally employed as a natural Lyapunov function for investigating stability [70, sec. 2.4.1].

To clarify the approach, consider that the controlled plant is modeled by a set of equations defined by a state-space model (4.8) in discrete time. Moreover, the process is under state constraints $\mathcal{X}$ and input constraints $\mathcal{U}$ which can take one of the form given in (4.20). Without loss of generality assume that the optimal control is given as a regulation problem with general cost function (4.4), that is, from any initial condition $\boldsymbol{x}_0 \in \mathcal{X}$ the goal is to find a sequence of control moves which drive states and inputs toward the point $(\mathbf{0}, \mathbf{0})$, which is the origin of the given coordinate system and a stabilizing solution to (4.8), i.e. $\mathbf{0} = \boldsymbol{f}(\mathbf{0}, \mathbf{0})$. The sets $\mathcal{U}$, $\mathcal{X}$ are assumed to be compact and contain the origin in their interior. The optimal control problem can be cast as follows

$$\min_{\boldsymbol{U}} J(\boldsymbol{x}, \boldsymbol{U}) = F(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} L(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.21a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.21b}$$

$$\boldsymbol{x}_k \in \mathcal{X} \tag{4.21c}$$

$$\boldsymbol{u}_k \in \mathcal{U} \tag{4.21d}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{4.21e}$$

where (4.21a) is the cost function, (4.21b) represents the process model and (4.21c), (4.21d) and (4.21e) are the constraints. Assuming that the cost function (4.21a) is a Lyapunov function candidate, then there exists MPC control law which asymptotically stabilizes the controlled system and the selected cost function corresponds to Lyapunov function if the following assumptions hold [70]:

A1: There exists a terminal set $\Omega \subset \mathcal{X}$, such that Ω is closed and $\mathbf{0} \in \Omega$ (state constraints are satisfied in Ω).

A2: There exists a terminal controller $\boldsymbol{\kappa}(\boldsymbol{x}_k) \in \mathcal{U}$, $\forall \boldsymbol{x}_k \in \Omega$, $k = N, \ldots, \infty$ (input constraints are satisfied in Ω).

A3: The set Ω is positively invariant for $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}(\boldsymbol{x}))$, i.e. for any $\boldsymbol{x}_0 \in \Omega$ all subsequent updates of $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\kappa}(\boldsymbol{x}_k))$ remain in the set Ω.

A4: $F(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}(\boldsymbol{x}))) - F(\boldsymbol{x}) + L(\boldsymbol{x}, \boldsymbol{\kappa}(\boldsymbol{x})) \leq 0$, $\forall \boldsymbol{x} \in \Omega$, $F(\cdot)$ is a local Lyapunov function in Ω.

According to assumptions A1–A4, various predictive control schemes exist which guarantee asymptotic stability and a brief summary is revealed in the sequel.

MPC using terminal state constraints: Here the stability of the closed loop control is achieved by imposing

$$\boldsymbol{x}_N = \mathbf{0} \tag{4.22}$$

which is the terminal equality constraint (4.20e). Equation (4.22) induces that $\Omega = \{\mathbf{0}\}$, Ω is invariant and terminal controller $\boldsymbol{\kappa}(\boldsymbol{x}) = \mathbf{0}$ maintains the states at the stable origin. Obviously, the left hand side of inequality in A4 is equal to 0 for all $\boldsymbol{x}(t)$, $t > t_N$. This approach was one of the earliest methods of ensuring closed loop stability. Adding the relatively simple requirement (4.22) to the optimization problem forces the predicted trajectories to reach the terminal state, thus the given goal will always be met if the problem (4.21) is feasible. However, from the computational point of view, the terminal state equality constraint causes that the resulting optimization problem might be difficult to solve. In the case that the initial condition $\boldsymbol{x}_0$ is far from the origin, one has to choose prediction horizon long in order to retain the optimization problem feasible. Simultaneously, with longer prediction horizon also the number of decision variables grows, and this increases the complexity of the optimization problem.

MPC using terminal cost: This approach introduces a non-zero terminal cost in (4.21a) without imposing terminal state constraints. However, for general systems (4.8) stability holds only if the prediction horizon is very large, otherwise this approach is valid for linear unconstrained systems, or constrained but stable linear systems. Nevertheless, this methodology was introduced by [20] and later explored by [90].

MPC using terminal constraint set: In this MPC setup, the stabilizing ingredient is the terminal set constraint Ω, i.e.

$$\boldsymbol{x}_N \in \Omega \tag{4.23}$$

with an active terminal controller $\boldsymbol{\kappa}(\boldsymbol{x})$ inside the region Ω. The terminal controller (A2) ensures that all states from Ω are steered to the stable origin rendering the set Ω positively invariant (A3). Such scheme is sometimes referred to as *dual-mode*

MPC due to use of two controllers. The first controller is based on the receding horizon principle and the second $\boldsymbol{\kappa}(\boldsymbol{x})$ takes care of stabilizing effect of the closed loop. Assumptions A1-A3 can be easily checked to hold and A4 is satisfied by a proper choice of $F(\cdot)$ in (4.21a).

MPC using terminal cost and terminal constraint set: This modification of MPC is widely adopted in the literature, since it can handle a large variety of control problems. Both terminal constraint (4.23) and terminal cost in (4.21a) are employed to guarantee stability. These ingredients satisfy exactly the assumptions A1-A4 and the related approach will be detailed in the next section.

Infinite horizon MPC: In this variation of MPC, the optimal control problem is solved with infinite horizon. Thus, all possible future evolution of the system is taken into account and the assumptions A1-A4 are not needed. However, the disadvantage of this approach is that the resulting optimization problem can be of high complexity due to infinite number of decision variables. Despite this fact, there are approaches for solving it [5, 78, 86] .

Contractive MPC: In this approach, as suggested by [38], stability is enforced using contractive constraints defined as

$$\|\boldsymbol{x}_N\|_p \leq \beta \|\boldsymbol{x}_0\|_p \tag{4.24}$$

where $\beta \in (0, 1) \subset \mathbb{R}$ is a positive constant and p denotes the selected norm. Constraint (4.24) enforces predicted states to lie closer to the origin in some p-norm measure with decay β. Applying the approach in the receding horizon fashion, the contractive constraint (4.24) causes that the set of states $\boldsymbol{x}_N$ is shrinking towards origin, thus stability can be established.

Suboptimal MPC offers stability guarantees even if there is no need for optimal solutions. This approach is investigated in [93] where the authors retain decreasing property of the Lyapunov function across closed loop trajectory as a natural consequence of imposing either terminal equality constraints (4.22) or terminal inequality constraints (4.23). The MPC problem is then recast as feasibility problem (i.e. without objective function).

4.6 Methods for Computing Terminal Sets

Construction of the terminal set Ω is of crucial importance in MPC because it is one of the ingredients commonly used to ensure stability of the closed loop system. The question which arises is, how to compute the terminal set Ω and terminal controller $\boldsymbol{\kappa}(\boldsymbol{x})$ such that assumptions A1-A4 hold. This comes with the notion of invariance as follows:

Definition 4.1 (Control Invariant Set) *A control invariant set Ω is the set of states for which there exist a controller $\boldsymbol{u} = \boldsymbol{\kappa}(\boldsymbol{x}) \in \mathcal{U}$ such that if $\boldsymbol{x}_0 \in \Omega$, then all subsequent state updates $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\kappa}(\boldsymbol{x}_k)$ also belong to the set Ω for $k = 0, \ldots, \infty$.*

Definition 4.1 suggests that the terminal set can be found by investigating the autonomous behavior of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}(\boldsymbol{x}))$ for given set of initial conditions. This is one of the many ways in the literature for computing terminal sets, for further references see for instance [29, 58, 61, 89]. The terminal set is interesting from a computational point of view, because the size of the set influences the complexity of the optimal control Problem 4.1. If the model is linear, stable, and no constraints are imposed, it can be shown that the terminal set is the whole feasible space. As input/state constraints are present, the terminal set may be only a subset of the state space [29]. In some cases the terminal region may reduce to only one point, which is basically achieved by MPC using terminal state constraint approach. Therefore one would like to have possibly the largest terminal region. However, it is very difficult, if not impossible, to find the largest terminal region for a given nonlinear system [28]. The main unresolved difficulty at this point is the determination of the region Ω which appears to require that some global test is satisfied which again may not be trivial except for academic examples [77].
In the following some of the methods for computing terminal sets for LTI and PWA systems are reviewed. The approaches generating Ω in a polytopic form are preferred, since they enter the formulation of optimization problem linearly and thus can be employed in explicit MPC.
Theoretical concepts of polytopic invariant sets originate from the work of [44] and were extended later to cover cases with additive disturbances in [21, 61]. Consider a class of LTI systems (4.11) which evolve under given linear control law $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x}$ with known matrix $\boldsymbol{K}$, i.e. $\boldsymbol{x}_{k+1} = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{x}_k$ and $\boldsymbol{x} \in \mathcal{X}$. The state constraint set $\mathcal{X}$ as in a polytopic form (1.1), i.e. $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{l}\}$. By defining the set Ω_k as

$$\Omega_k := \left\{\boldsymbol{x} \in \mathbb{R}^n \,\middle|\, \boldsymbol{\Phi}^k\boldsymbol{x} \in \mathcal{X},\ k = 0, 1, \ldots\right\} \tag{4.25}$$

the maximal positively invariant set for a controller $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x}$ can be found via following algorithm:

Algorithm 4.1 (Maximal invariant set for LTI systems [44])

Step 1: *Set k=0.*

Step 2: *If the set Ω_{k+1} generated by (4.25) equals Ω_k, jump to step 4, otherwise set $k = k + 1$ and continue.*

Step 3: *Replace $k = k + 1$ and go back to Step 2.*

Step 4: *The maximal invariant set is given by $\Omega = \Omega_k$. The maximal number of iteration is $k^* = k$.*

The maximal invariant set Ω can be found after k^* iterations. The proof for finite convergence as well as other detail of the Algorithm 4.1 can be found in [44].
The result for LTI systems have been further extended to the framework of PWA systems by [46]. Later, the algorithm has been modified to cover PWA models with additive

disturbances in [88]. Here, the approach of [46] is reviewed which aims at finding the maximal invariant set for PWA systems of the form (4.12). It is assumed that given PWA system is stabilizable at the origin, i.e. those local dynamics containing origin must have the affine term $\boldsymbol{c}_i$ equal zero. Furthermore, input contraints $\mathcal{U}$ and state constraints $\mathcal{X}$ are present and are given as polytopic sets. The stabilizing terminal controller $\boldsymbol{u} = \boldsymbol{\kappa}(\boldsymbol{x})$ can be found via solving the following LMI optimization problem

$$\min_{\boldsymbol{Y}_i, \boldsymbol{Z}, \gamma} \ \gamma \tag{4.26a}$$

$$\text{subj. to} \ \ \boldsymbol{Z} \succ 0, \tag{4.26b}$$

$$\begin{pmatrix} \boldsymbol{Z} & * & * & * \\ (\boldsymbol{A}_i\boldsymbol{Z} + \boldsymbol{B}_i\boldsymbol{Y}_i)^T & \boldsymbol{Z} & * & * \\ (\boldsymbol{Q}^{0.5}\boldsymbol{Z}) & \boldsymbol{0} & \gamma\boldsymbol{I} & * \\ (\boldsymbol{R}^{0.5}\boldsymbol{Y}_i) & \boldsymbol{0} & \boldsymbol{0} & \gamma\boldsymbol{I} \end{pmatrix} \succeq 0, \quad \forall i \in \mathcal{I}_0 \tag{4.26c}$$

where the set $\mathcal{I}_0$ is a selection of indexes corresponding to dynamics that contain the origin, $\boldsymbol{Q}$, $\boldsymbol{R}$ are weighing matrices, $\boldsymbol{Y}_i = \boldsymbol{K}_i\boldsymbol{Z}$, $\boldsymbol{Z} = (1/\gamma)\boldsymbol{P}^{-1}$ are unknown matrices and the scalar γ accounts for the worst case switching sequence within $\mathcal{I}_0$. Solution to the problem (4.26) results in a state feedback law $\boldsymbol{u} = \boldsymbol{K}_i\boldsymbol{x}$ with gains $\boldsymbol{K}_i$ for which there exist a quadratic Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}$. Subsequently, for the terminal controller $\boldsymbol{u} = \boldsymbol{K}_i\boldsymbol{x}$ and given constraint set $\mathcal{X}, \mathcal{U}$ one can apply techniques for computing the maximal invariant set as follows:

Algorithm 4.2 (Maximal invariant set for PWA systems [88])

Step 1: *Select the initial region Ω_0 as union of all partitions containing origin, i.e $\Omega_0 = \bigcup_{i\in\mathcal{I}_0} \mathcal{D}_i$, and set $k = 0$.*

Step 2: *Compute the reachable set Ω_{k+1} for the set Ω_k assuming that the system (4.12) evolves according to piecewise linear feedback law $\boldsymbol{u} = \boldsymbol{K}_i\boldsymbol{x}$ such that constraints on states $\mathcal{X}$ and inputs $\mathcal{U}$ are satisfied, i.e.*

$$\Omega_{k+1} = \{\boldsymbol{x} \in \mathcal{X} \mid \exists \boldsymbol{u} = \boldsymbol{K}_i\boldsymbol{x} \in \mathcal{U} \ s.t. \ \boldsymbol{f}_{PWA}(\boldsymbol{x}, \boldsymbol{K}_i\boldsymbol{x}) \in \Omega_k, \ i \in \mathcal{I}_0\} \tag{4.27}$$

where $\boldsymbol{f}_{PWA}$ is given by (4.12).

Step 3: *If the generated set Ω_{k+1} equals Ω_k, jump to Step 4, otherwise set $k = k + 1$ and repeat step 2.*

Step 4: *Maximal invariant set is given by $\Omega = \Omega_k$.*

Important is, that the terminal set Ω can be represented as a polytope

$$\Omega := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}\boldsymbol{x} \leq \boldsymbol{l}\} \tag{4.28}$$

and enters the optimization problem linearly in states. The triple $\boldsymbol{P}$, Ω together with the terminal controller $\boldsymbol{u} = \boldsymbol{K}_i\boldsymbol{x}$ with $i \in \mathcal{I}_0$ can be used to design a stabilizing MPC controller for PWA systems (4.12) according to scheme using terminal cost and terminal constraint set.

Chapter 5

Explicit MPC

Introduction of explicit MPC has been a very significant step in control theory. Due to the progress of multiparametric programming, the solution to some optimization problems can be found in an explicit form where the control law is given as a function of parameters. As seen from MPC view, such a solution avoids repetitive optimization and allows MPC to be implemented on systems with rapid sampling. This chapter reviews some of the effective methods in explicit MPC and details the implementation issues. Outlined methods are based on original publications by [11, 16, 23, 46].

5.1 Main Features

The idea behind the explicit approach is transformation of the optimal control Problem 4.1 to its parametric form (3.1), (3.2) or (3.3). The parameter $\boldsymbol{\theta}$ is usually given by the state vector of initial conditions $\boldsymbol{x}_0$ which represents the feedback information from the plant. Solving the optimization problem parametrically gives the expression for optimal control actions as a sequence of functions $\boldsymbol{U}^* = (\boldsymbol{\pi}_0(\boldsymbol{x}_0)^T, \boldsymbol{\pi}_1(\boldsymbol{x}_0)^T, \ldots, \boldsymbol{\pi}_{N-1}(\boldsymbol{x}_0)^T)^T$. Recalling the receding horizon principle in MPC, only the first element is used, i.e. the function $\boldsymbol{\pi}_0(\boldsymbol{x}_0)$. The benefit of the explicit MPC is that it solves the optimal control problem only once. The function $\boldsymbol{\pi}_0(\boldsymbol{x}_0)$ is then applied in the feedback connection as already shown in Fig. 4.3 on page 34.
Another very important feature of explicit solutions is that the result can be further simplified and stored in a very compact format. This format is typically a look-up table which comprises multiple sorted elements. Nowadays, various techniques exist which provide fast searching and evaluation of such structures, giving the effectiveness of the explicit solutions not only on speed advantages but also on ease of implementation. Design and implementation of explicit MPC can be summarized into following points:

Off-line: Solve the optimal control Problem 4.1 for all possible values of initial conditions $\boldsymbol{x}_0$.

On-line: Compute the optimal control $\boldsymbol{u}^* = \boldsymbol{\pi}_0(\boldsymbol{x}_0)$ law by evaluating the corresponding function, as shown in Fig. 4.3.

However, the explicit MPC has limitations. In fact, the number of partitions that define the control law may grow in the worst case exponentially. This has motivated for development of new methods that provide less complex solutions, see e.g. [115] for references, and it is one of the goals to be addressed in Part II of the thesis.

5.2 Multiparametric Forms of Optimal Control Problems

In the sequel, multiparametric versions of optimal control problems based on linear and PWA models are formulated. The presented technique shows how to translate the optimal control problems to their respective multiparametric forms reviewed in Chapter 3.

5.2.1 Linear Model, 1/∞-Norm

Consider a regulation problem of a dynamical system described by a set of equations in the state space and in discrete time (4.8) with a special linear form (4.11). To derive the multiparametric problem involving model (4.11), only the dynamic equation (4.11a) is considered, i.e. $\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the dynamic matrix, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ is the input matrix, $\boldsymbol{x} \in \mathbb{R}^n$ denotes the state vector and $\boldsymbol{u} \in \mathbb{R}^m$ is the input vector. Assume that both input and state variables are subjected to hard constraints, i.e.

$$\mathcal{X} := \{\boldsymbol{x} \in \mathbb{R}^n \mid \underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}}\} \tag{5.1}$$
$$\mathcal{U} := \{\boldsymbol{x} \in \mathbb{R}^m \mid \underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \overline{\boldsymbol{u}}\} \tag{5.2}$$

where $\underline{\boldsymbol{x}}$, $\underline{\boldsymbol{u}}$ and $\overline{\boldsymbol{x}}$, $\overline{\boldsymbol{u}}$ denote the lower and upper bounds, respectively. The constraints (5.1), (5.2) restrict the initial feasible set for state $\boldsymbol{x} \in \mathbb{R}^n$ and input $\boldsymbol{u} \in \mathbb{R}^m$ variables. Furthermore, it is required that these constraint are satisfied throughout the whole prediction interval. The goal of the optimal control problem is to determine a sequence of future inputs $\boldsymbol{U} = (\boldsymbol{u}_0^T, \boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_{N-1}^T)^T$ by solving the following optimization problem

$$\min_{\boldsymbol{U}} J(\boldsymbol{x}, \boldsymbol{U}) = \|\boldsymbol{P}\boldsymbol{x}_N\|_p + \sum_{k=0}^{N-1} \|\boldsymbol{Q}\boldsymbol{x}_k\|_p + \|\boldsymbol{R}\boldsymbol{u}_k\|_p \tag{5.3a}$$
$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{5.3b}$$
$$\boldsymbol{x}_k \in \mathcal{X} \tag{5.3c}$$
$$\boldsymbol{u}_k \in \mathcal{U} \tag{5.3d}$$
$$\boldsymbol{x}_N \in \Omega \tag{5.3e}$$
$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{5.3f}$$

accounting the predictions of states and inputs at sampling instants $k = 0, 1, \ldots, N$ where N is the prediction horizon and p denotes 1- or ∞- norm. It is assumed that matrices $\boldsymbol{Q}$, $\boldsymbol{R}$ are non-singular, and $\boldsymbol{P}$ has full column rank, and the pair $\boldsymbol{A}$, $\boldsymbol{B}$ is stabilizable.

Equation (5.3e) is the terminal polytopic set constraint (4.28) due stability requirements. The initial condition (5.3f) is considered as a vector of parameters in the multiparametric approach. Problem (5.3) is transformed to standard form of mpLP (3.1) by introducing auxiliary vector $\boldsymbol{z}$ depeding on the norm p [24]. If $p = 1$, then

$$\boldsymbol{z} = \left((\boldsymbol{\epsilon}_0^x)^T, \cdots, (\boldsymbol{\epsilon}_N^x)^T, (\boldsymbol{\epsilon}_1^u)^T, \cdots, (\boldsymbol{\epsilon}_{N-1}^u)^T, \boldsymbol{U}^T\right)^T \tag{5.4}$$

and it relates to the following set of constraints

$$-\boldsymbol{\epsilon}_k^x \leq \boldsymbol{Q}\boldsymbol{x}_k \tag{5.5a}$$
$$-\boldsymbol{\epsilon}_k^x \leq -\boldsymbol{Q}\boldsymbol{x}_k \tag{5.5b}$$
$$-\boldsymbol{\epsilon}_k^u \leq \boldsymbol{R}\boldsymbol{u}_k \tag{5.5c}$$
$$-\boldsymbol{\epsilon}_k^u \leq -\boldsymbol{R}\boldsymbol{u}_k. \tag{5.5d}$$

For $p = \infty$,

$$\boldsymbol{z} = (\epsilon_0^x, \cdots, \epsilon_N^x, \epsilon_0^u, \cdots, \epsilon_{N-1}^u, \boldsymbol{U}^T)^T \tag{5.6}$$

and it must satisfy

$$-\mathbf{1}\epsilon_k^x \leq \boldsymbol{Q}\boldsymbol{x}_k \tag{5.7a}$$
$$-\mathbf{1}\epsilon_k^x \leq -\boldsymbol{Q}\boldsymbol{x}_k \tag{5.7b}$$
$$-\mathbf{1}\epsilon_k^u \leq \boldsymbol{R}\boldsymbol{u}_k \tag{5.7c}$$
$$-\mathbf{1}\epsilon_k^u \leq -\boldsymbol{R}\boldsymbol{u}_k \tag{5.7d}$$

where $\mathbf{1}$ is column vector of ones. Both cases (5.4)–(5.5), (5.6)–(5.7) are very similar, thus in the sequel only case for $p = \infty$ will be considered. With the help of $\boldsymbol{z}$ the objective function in (5.3a) is replaced by

$$J(\boldsymbol{z}) = \epsilon_0^x + \cdots + \epsilon_N^x + \epsilon_1^u + \cdots + \epsilon_{N-1}^u \tag{5.8}$$

which is an upper bound on $J(\boldsymbol{U})$. Using the linear relation in (5.3b) the predictions can be written as a function of initial conditions $\boldsymbol{x}_0$ and vector $\boldsymbol{U}$, i.e.

$$\boldsymbol{x}_k = \boldsymbol{A}^k\boldsymbol{x}_0 + \sum_{j=0}^{k-1} \boldsymbol{A}^j\boldsymbol{B}\boldsymbol{u}_{k-1-j}, \quad k = 1, \ldots, N \tag{5.9a}$$

$$\begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{N-1} \end{pmatrix} \boldsymbol{x}_0 + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \boldsymbol{B} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \mathbf{0} \end{pmatrix} \boldsymbol{U} \tag{5.9b}$$

$$= \tilde{\boldsymbol{A}}\boldsymbol{x}_0 + \tilde{\boldsymbol{B}}\boldsymbol{U} \tag{5.9c}$$

Using the expression (5.9) the optimization problem (5.3) can be rewritten as

$$\min_{\boldsymbol{z}} J(\boldsymbol{z}) \tag{5.10a}$$

s.t.

$$\tilde{\boldsymbol{A}}\boldsymbol{x}_0 + \tilde{\boldsymbol{B}}\boldsymbol{U} \leq \overline{\boldsymbol{x}} \tag{5.10b}$$

$$-\tilde{\boldsymbol{A}}\boldsymbol{x}_0 - \tilde{\boldsymbol{B}}\boldsymbol{U} \leq -\underline{\boldsymbol{x}} \tag{5.10c}$$

$$\boldsymbol{U} \leq \overline{\boldsymbol{u}} \tag{5.10d}$$

$$-\boldsymbol{U} \leq -\underline{\boldsymbol{u}} \tag{5.10e}$$

$$\tilde{\boldsymbol{H}}\tilde{\boldsymbol{A}}\boldsymbol{x}_0 + \tilde{\boldsymbol{H}}\tilde{\boldsymbol{B}}\boldsymbol{U} \leq \boldsymbol{l} \tag{5.10f}$$

$$-\mathbf{1}\epsilon^x \leq \pm\tilde{\boldsymbol{Q}}\left(\tilde{\boldsymbol{A}}\boldsymbol{x}_0 + \tilde{\boldsymbol{B}}\boldsymbol{U}\right) \tag{5.10g}$$

$$-\mathbf{1}\epsilon^u \leq \pm\tilde{\boldsymbol{R}}\boldsymbol{U} \tag{5.10h}$$

where $\boldsymbol{H}$, $\boldsymbol{l}$, $\tilde{\boldsymbol{H}} = (\mathbf{0}, \ldots, \mathbf{0}, \boldsymbol{H})$ is the H-representation of the terminal set Ω (4.28) and $\tilde{\boldsymbol{Q}} = \mathrm{diag}(\boldsymbol{Q}, \ldots, \boldsymbol{Q}, \boldsymbol{P})$, $\tilde{\boldsymbol{R}} = \mathrm{diag}(\boldsymbol{R}, \ldots, \boldsymbol{R})$. Subsequently, the overall formulation (5.10) can be written in a vector form as follows

$$\min_{\boldsymbol{z}} J(\boldsymbol{z}) = \mathbf{1}^T\boldsymbol{z} \tag{5.11a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{h} + \boldsymbol{S}\boldsymbol{x}_0 \tag{5.11b}$$

where $\boldsymbol{z}$ is the optimized variable and $\boldsymbol{x}_0$ is a vector of parameters. Variables $\boldsymbol{G}$, $\boldsymbol{h}$, $\boldsymbol{S}$ are obtained by appropriate substitution from problem (5.10), i.e.

$$\begin{aligned}
\boldsymbol{G} &= (\tilde{\boldsymbol{B}}^T, -\tilde{\boldsymbol{B}}^T, \boldsymbol{I}, -\boldsymbol{I}, (\tilde{\boldsymbol{H}}\tilde{\boldsymbol{B}})^T, (\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{B}})^T, -(\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{B}})^T, -\tilde{\boldsymbol{R}}^T, \tilde{\boldsymbol{R}}^T)^T \\
\boldsymbol{h} &= (\overline{\boldsymbol{x}}^T, -\underline{\boldsymbol{x}}^T, \overline{\boldsymbol{u}}^T, -\underline{\boldsymbol{u}}^T, \boldsymbol{l}^T, \epsilon^x\mathbf{1}^T, \epsilon^x\mathbf{1}^T, \epsilon^u\mathbf{1}^T, \epsilon^u\mathbf{1}^T)^T \\
\boldsymbol{S} &= (-\tilde{\boldsymbol{A}}^T, \tilde{\boldsymbol{A}}^T, \mathbf{0}, \mathbf{0}, -(\tilde{\boldsymbol{H}}\tilde{\boldsymbol{A}})^T, (\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{A}})^T, -(\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{A}})^T, \mathbf{0}, \mathbf{0})^T.
\end{aligned}$$

Formulation (5.11) is now in a standard form as given by (3.1) and can be solved using multiparametric techniques presented in Chapter 3. For further details regarding this technique the reader is referenced to [11].

5.2.2 Linear Model, 2-Norm

The use of one or infinity norms in cost function have the advantage that the resulting optimization problem is transformed to mpLP where the relations are linear. Replacing the objective function with 2-norm creates a quadratic function and the problem thus shifted into QP field. Consider a regulation problem of a linear discrete time system (5.3b) subject to input and state constraints (5.1), (5.2) respectively. Mathematically, the optimal

control problem can be formulated as

$$\min_{\boldsymbol{U}} J(\boldsymbol{x},\boldsymbol{U}) = \|\boldsymbol{P}\boldsymbol{x}_N\|_2 + \sum_{k=0}^{N-1} \|\boldsymbol{Q}\boldsymbol{x}_k\|_2 + \|\boldsymbol{R}\boldsymbol{u}_k\|_2 \tag{5.12a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{5.12b}$$

$$\boldsymbol{x}_k \in \mathcal{X} \tag{5.12c}$$

$$\boldsymbol{u}_k \in \mathcal{U} \tag{5.12d}$$

$$\boldsymbol{x}_N \in \Omega \tag{5.12e}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{5.12f}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0$, $\boldsymbol{R} = \boldsymbol{R}^T \succ 0$, and $\boldsymbol{P} = \boldsymbol{P}^T \succeq 0$. It is assumed that the system (5.12b) is stabilizable and the stabilizing terminal set Ω in (5.12e) has been determined. To express the problem (5.12) in the parametric version, the predictions of the linear model are constructed as in (5.9) and plugged into respective equations (5.12). Consequently, the problem can be rewritten as follows

$$\min_{\boldsymbol{U}} J(\boldsymbol{x},\boldsymbol{U}) \tag{5.13a}$$

$$\text{s.t.}$$

$$\tilde{\boldsymbol{A}}\boldsymbol{x}_0 + \tilde{\boldsymbol{B}}\boldsymbol{U} \leq \overline{\boldsymbol{x}} \tag{5.13b}$$

$$-\tilde{\boldsymbol{A}}\boldsymbol{x}_0 - \tilde{\boldsymbol{B}}\boldsymbol{U} \leq -\underline{\boldsymbol{x}} \tag{5.13c}$$

$$\boldsymbol{U} \leq \overline{\boldsymbol{u}} \tag{5.13d}$$

$$-\boldsymbol{U} \leq -\underline{\boldsymbol{u}} \tag{5.13e}$$

$$\tilde{\boldsymbol{H}}\tilde{\boldsymbol{A}}\boldsymbol{x}_0 + \tilde{\boldsymbol{H}}\tilde{\boldsymbol{B}}\boldsymbol{U} \leq \boldsymbol{l} \tag{5.13f}$$

where

$$J(\boldsymbol{x},\boldsymbol{U}) = \boldsymbol{U}^T(\tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{B}} + \tilde{\boldsymbol{R}})\boldsymbol{U} + 2\boldsymbol{x}_0^T\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{B}}\boldsymbol{U} + \boldsymbol{x}_0^T\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{A}}\boldsymbol{x}_0 \tag{5.14}$$

and $\tilde{\boldsymbol{Q}} = \text{diag}(\boldsymbol{Q},\ldots,\boldsymbol{Q},\boldsymbol{P})$, $\tilde{\boldsymbol{R}} = \text{diag}(\boldsymbol{R},\ldots,\boldsymbol{R})$ and $\tilde{\boldsymbol{H}} = (\boldsymbol{0},\ldots,\boldsymbol{0},\boldsymbol{H})^T$. Since constant term appears in (5.14), objective function can be modified to a simpler form

$$J'(\boldsymbol{x},\boldsymbol{U}) = 0.5\boldsymbol{U}^T\boldsymbol{M}\boldsymbol{U} + \boldsymbol{x}_0^T\boldsymbol{N}\boldsymbol{U} \tag{5.15}$$

where $\boldsymbol{M} = \tilde{\boldsymbol{B}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{B}} + \tilde{\boldsymbol{R}}$ and $\boldsymbol{N} = \tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{B}}$. By introducing an auxiliary vector

$$\boldsymbol{z} = \boldsymbol{U} + \boldsymbol{M}^{-1}\boldsymbol{N}^T\boldsymbol{x}_0 \tag{5.16}$$

and plugging in (5.15) the objective function becomes $J(\boldsymbol{z}) = 0.5\boldsymbol{z}^T\boldsymbol{M}\boldsymbol{z}$. Using this simplified notation, the optimization problem (5.13) is transformed to

$$\min_{\boldsymbol{z}} J(\boldsymbol{z}) = 0.5\boldsymbol{z}^T\boldsymbol{M}\boldsymbol{z} \tag{5.17a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{h} + \boldsymbol{S}\boldsymbol{x}_0 \tag{5.17b}$$

where $\boldsymbol{G}$, $\boldsymbol{h}$, $\boldsymbol{S}$ are given by

$$\boldsymbol{G} = (\tilde{\boldsymbol{B}}^T, -\tilde{\boldsymbol{B}}^T, \boldsymbol{I}, -\boldsymbol{I}, (\tilde{\boldsymbol{H}}\tilde{\boldsymbol{B}})^T)^T$$
$$\boldsymbol{h} = (\overline{\boldsymbol{x}}^T, -\underline{\boldsymbol{x}}^T, \overline{\boldsymbol{u}}^T, -\underline{\boldsymbol{u}}^T, \boldsymbol{l}^T)^T$$
$$\boldsymbol{S} = ((\tilde{\boldsymbol{B}}\boldsymbol{M}^{-1}\boldsymbol{N} - \tilde{\boldsymbol{A}})^T, (\tilde{\boldsymbol{A}} - \tilde{\boldsymbol{B}}\boldsymbol{M}^{-1}\boldsymbol{N})^T, (\boldsymbol{M}^{-1}\boldsymbol{N})^T, -(\boldsymbol{M}^{-1}\boldsymbol{N})^T, (\tilde{\boldsymbol{H}}\tilde{\boldsymbol{B}}\boldsymbol{M}^{-1}\boldsymbol{N} - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{A}})^T)^T.$$

The problem (5.17) is a standard form of mpQP (3.2) with $\boldsymbol{x}_0$ as a parameter and $\boldsymbol{z}$ as optimization variable. Further details regarding the multiparametric approach to MPC with quadratic objective are given in [16].

5.2.3 PWA Model, 1/∞-Norm

Optimal control problem involving PWA model (4.12) can be formulated as follows

$$\min_{\boldsymbol{U}} J(\boldsymbol{x}, \boldsymbol{U}) = \|\boldsymbol{P}\boldsymbol{x}_N\|_p + \sum_{k=0}^{N-1} \|\boldsymbol{Q}\boldsymbol{x}_k\|_p + \|\boldsymbol{R}\boldsymbol{u}_k\|_p \tag{5.18a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.18b}$$
$$\boldsymbol{x}_k \in \mathcal{X} \tag{5.18c}$$
$$\boldsymbol{u}_k \in \mathcal{U} \tag{5.18d}$$
$$\boldsymbol{x}_N \in \Omega \tag{5.18e}$$
$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{5.18f}$$

where $p = \{1, \infty\}$, $\mathcal{X}$, $\mathcal{U}$ are interior constraint sets for states (5.1) and inputs (5.2), respectively, and Ω is a terminal set constraint (4.28). The specificity of problem (5.18) is due to presence of switching rules in PWA model (4.12) which select the active dynamics i if the condition (4.13) is fulfilled. Formally, such a rule is associated to a binary variable $\delta_i = \{0, 1\}$ which is activated if all inequalities of (4.13) are satisfied, i.e.

$$\delta_{i,k} = 1 \iff \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} \in \mathcal{D}_i, \quad \mathcal{D}_i = \left\{ \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{pmatrix} \in \mathbb{R}^{n+m} \,\middle|\, \boldsymbol{H}_i \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{pmatrix} \leq \boldsymbol{l}_i \right\} \tag{5.19}$$

where the variable $\delta_{i,k}$ denotes a selector which is activated if the state-input vector $(\boldsymbol{x}_k^T, \boldsymbol{u}_k^T)^T$ belongs to a polytopic set $\mathcal{D}_i$. Note that the variable $\delta_{i,k}$ is a function of discrete time k. Evolution of such system posses hybrid behavior because it comes to switching whenever state-input vector $(\boldsymbol{x}_k, \boldsymbol{u}_k)^T$ moves from region $\mathcal{D}_i$ to $\mathcal{D}_j$. Incorporation of discrete logic into MPC framework has been studied in [15] and it is referred to as *big-M* technique. The principle relies on transformation of logical rules into set of inequalities which contain both, real and binary variables. Assuming that the regions $\mathcal{D}_i$ do not overlap, i.e. $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$, (5.19) can be rewritten as

$$\boldsymbol{H}_i \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} - \boldsymbol{l}_i \leq \boldsymbol{M}_i(1 - \delta_{i,k}), \quad \forall i = 1, \ldots, n_{\mathcal{D}} \tag{5.20a}$$

$$\sum_{i=1}^{n_{\mathcal{D}}} \delta_{i,k} = 1 \tag{5.20b}$$

where $\boldsymbol{M}_i = \max_{\boldsymbol{x}_k, \boldsymbol{u}_k} \left(\boldsymbol{H}_i \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} - \boldsymbol{l}_i \right)$. Note that (5.20b) represents an exclusive or condition, that is, if $\delta_{i,k} = 1$ other delta's different from i are equal zero. With the help of (5.20) PWA model (4.12) is given by

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{n_\mathcal{D}} \delta_{i,k} (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i) \tag{5.21}$$

which is nonlinear due to product between $\delta_{i,k}$ and states/inputs. But, using the big-M technique, (5.21) can be converted to linear form as follows

$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i) \leq \overline{\boldsymbol{x}}(1 - \delta_{i,k}), \tag{5.22a}$$
$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i) \geq \underline{\boldsymbol{x}}(1 - \delta_{i,k}). \tag{5.22b}$$

It's easy to verify that if $\delta_{i,k} = 1$, the terms in the right-hand-side of constraints (5.22) vanish, and the double-sided inequalities reduce to and equality constraint $\boldsymbol{x}_{k+1} = \boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i$. If, on the other hand, $\delta_{i,k} = 0$, then (5.22) simply become redundant and the i-th local model doesn't contribute to $\boldsymbol{x}_{k+1}$. Hence, PWA model (4.12) is transformed to a set of inequalities given by (5.20) and (5.22). Collecting equations (5.20), (5.22) with input/state constraints (5.1), (5.2), and terminal constraint (4.28), the optimal control problem can be formulated as follows:

$$\min_{\boldsymbol{U}} \; J(\boldsymbol{x}, \boldsymbol{U}) = \|\boldsymbol{P}\boldsymbol{x}_N\|_p + \sum_{k=0}^{N-1} \|\boldsymbol{Q}\boldsymbol{x}_k\|_p + \|\boldsymbol{R}\boldsymbol{u}_k\|_p \tag{5.23a}$$

$$\text{s.t.} \quad \boldsymbol{H}_i \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} - \boldsymbol{l}_i \leq \boldsymbol{M}_i (1 - \delta_{i,k}) \tag{5.23b}$$

$$\sum_{i=1}^{n_\mathcal{D}} \delta_{i,k} = 1 \tag{5.23c}$$

$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_i + \boldsymbol{c}_i) \leq \overline{\boldsymbol{x}}(1 - \delta_{i,k}) \tag{5.23d}$$

$$\boldsymbol{x}_{k+1} - (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_i + \boldsymbol{c}_i) \geq \underline{\boldsymbol{x}}(1 - \delta_{i,k}) \tag{5.23e}$$

$$\underline{\boldsymbol{x}} \leq \boldsymbol{x}_k \leq \overline{\boldsymbol{x}} \tag{5.23f}$$

$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}_k \leq \overline{\boldsymbol{u}} \tag{5.23g}$$

$$\boldsymbol{x}_N \in \Omega \tag{5.23h}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{5.23i}$$

where the optimization variables are $\boldsymbol{U}$, $\boldsymbol{\delta}_k$, $\boldsymbol{x}_k$ for $k = 0, \ldots, N$ and $\boldsymbol{x}_0$ is the vector of parameters. Formulation (5.23) is further transformed to its mpMILP form (3.3) similarly as outlined for LTI systems in Section 5.2.1.

Alternatively, one may exploit the equivalence between PWA models and mixed-logical MLD systems (4.14) to derive a multiparametric problem formulation. The advantage of MLD representation is that the binary variables $\boldsymbol{\delta} \in \{0, 1\}^{n_d}$ and constraints are internal

part of the model and one can use it to obtain state predictions as functions of $\boldsymbol{x}_0$, similarly as with LTI model (5.9), i.e.

$$\boldsymbol{x}_k = \boldsymbol{A}^k \boldsymbol{x}_0 + \sum_{j=0}^{k-1} \boldsymbol{A}^j \left(\boldsymbol{B}_u \boldsymbol{u}_{k-1-j} + \boldsymbol{B}_{\text{aux}} \boldsymbol{w}_{k-1-j} + \boldsymbol{B}_{\text{aff}} \right), \quad k = 1, \ldots, N. \tag{5.24}$$

where $\boldsymbol{w} \in \mathbb{R}^{n_w}$ are auxiliary variables. Assuming $\boldsymbol{x}_0$ as parameters and $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}$, $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{N-1}$ as optimization variables, one can reformulate MPC problem for PWA model (4.12) as a standard mpMILP [4, 13].

5.3 Multiparametric Problems in MPC for PWA Systems

This section reviews the multiparametric problems for PWA systems and approaches for solving them. The same theory can be applied to LTI systems, as they are subclass of PWA systems. The main aim is to characterize the properties of explicit solutions and to present algorithms needed to efficiently compute the result.

5.3.1 Constrained Finite Time Optimal Control

The CFTOC problem for PWA systems (5.18) searches for an optimal control input $\boldsymbol{U} \in \mathcal{U}$ which drives the system states $\boldsymbol{x} \in \mathcal{X}$ from any initial condition $\boldsymbol{x}_0 \in \mathcal{X}$ towards neighborhood of the origin Ω using finite number of steps. CFTOC problems for PWA systems have been studied by [3, 23, 32] and the main results are extracted here.
Explicit solution to CFTOC problem (5.18) is stated by the following theorem:

Theorem 5.1 (Solution to CFTOC [23]) *The solution to the optimal control problem (5.18) with $p \in \{1, \infty\}$ and a linear state-update in (5.18b) is a time-varying piecewise affine state feedback control law of the form*

$$\boldsymbol{u}^*(\boldsymbol{x}_k) = \boldsymbol{F}_{k,i} \boldsymbol{x}_k + \boldsymbol{g}_{k,i} \quad \text{if} \quad \boldsymbol{x}_k \in \mathcal{P}_{k,i} \tag{5.25}$$

and the optimal value function is a time-varying piecewise affine function of the state

$$J^*(\boldsymbol{x}_k) = \boldsymbol{\Phi}_{k,i} \boldsymbol{x}_k + \Gamma_{k,i} \quad \text{if} \quad \boldsymbol{x}_k \in \mathcal{P}_{k,i} \tag{5.26}$$

where $\mathcal{P}_{k,i} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{H}_{k,i} \boldsymbol{x} \leq \boldsymbol{l}_{k,i}\}$ is a polyhedral partition of the set $\mathcal{X}_k$ of feasible states $\boldsymbol{x}_k$ at time k with $k = 0, \ldots, N-1$.

Theorem 5.1 is powerful as it provides the tool for obtaining optimal feedback law (5.25) in the form of a piecewise affine function defined over finite number of polytopes $\mathcal{P}_{k,i}$. Moreover, Theorem 5.1 provides information about the cost function (5.26) which can be further utilized. Notation in Theorem 5.1 should be interpreted as at k-th instance of the

prediction horizon $i = 1, \ldots, n_{\mathcal{P}}$ regions are generated. For the use in MPC and due to receding horizon principle, only the first element from the vector $\boldsymbol{U}$ is of interest, i.e. $\boldsymbol{u}_{0,i}$. It is therefore convenient to denote the solution to CFTOC problem (5.25) as $\boldsymbol{u} = \boldsymbol{F}_i\boldsymbol{x}_0+\boldsymbol{g}_i$ if $\boldsymbol{x}_0 \in \mathcal{P}_i$, and (5.26) as $J(\boldsymbol{x}_0) = \boldsymbol{\Phi}_i\boldsymbol{x}_0 + \Gamma_i$ if $\boldsymbol{x}_0 \in \mathcal{P}_i$, which will be mostly used in the remainder of the manuscript.
Explicit solution to (5.18) in the form of (5.25) can be obtained in the dynamic programming fashion, utilizing the backward propagation principle. That is, starting from the given known set Ω and solving smaller subproblems in the backward time. Algorithm for computing explicit solution to CFTOC via dynamic programming can be summarized as follows:

Algorithm 5.1 (Solution to CFTOC via Dynamic Programming [3])

1. *Calculate a stabilizing terminal set* Ω*, terminal cost* $\boldsymbol{P}$ *and terminal controller.*
2. *Set* $\mathcal{J}_0 = \|\boldsymbol{P}\boldsymbol{x}_0\|_p$*, and denote the initial feasible set* $\mathcal{S}_0 = \Omega$*.*
3. *For* $k = 1$ *to* N*, solve parametrically*

$$\min_{\boldsymbol{u}_k} J_k = \|\boldsymbol{Q}\boldsymbol{x}_k\|_p + \|\boldsymbol{R}\boldsymbol{u}_k\|_p + J_{k-1} \tag{5.27a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{PWA}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.27b}$$

$$\boldsymbol{x}_{k+1} \in \mathcal{S}_{k-1} \tag{5.27c}$$

$$\boldsymbol{x}_k \in \mathcal{X} \tag{5.27d}$$

$$\boldsymbol{u}_k \in \mathcal{U} \tag{5.27e}$$

and generate new regions $\mathcal{S}_k = \bigcup_i \mathcal{P}_{k,i}$ *by intersection and comparison.*

At each step of the algorithm a subproblem (5.27) is solved parametrically which yields an expression for state feedback control law with corresponding local region according to Theorem 5.1. More specifically, the Algorithm 5.1 solves mpLP iteratively through $n_{\mathcal{D}}$ dynamics and through $n_{\mathcal{P}_k}$ terminal sets at k-th iteration, which gives $n_{\mathcal{D}}n_{\mathcal{P}_k}$ operations in total. As the generated local regions may be overlapping, the geometric operations of intersection and comparisons are performed to deal with the possibly non-convex shape of regions $\mathcal{S}_k$. The intersection operation is such that redundant polytopes are removed (i.e. those which are completely covered by other polytopes). Under comparison operation one can understand selection of regions, to which the minimal cost function is assigned [3]. Moreover, if the cost function J_k is expressed as a function of $\boldsymbol{x}_0$, the algorithm allows to spot whether it differs in two consecutive steps, opening a way towards finding explicit solution to CITOC problem. For more details about the algorithm, the reader is referred to [5].

5.3.2 Time Optimal Control

In this section a multiparametric version of time optimal control [46] of PWA systems is presented. The attribute time optimal refers to a control policy which drives the system

states/inputs towards given terminal set Ω in the least possible number of steps. The related optimization problem can be formulated as follows

$$\min_{\boldsymbol{U}} \; J = N \tag{5.28a}$$
$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.28b}$$
$$\boldsymbol{x}_k \in \mathcal{X} \tag{5.28c}$$
$$\boldsymbol{u}_k \in \mathcal{U} \tag{5.28d}$$
$$\boldsymbol{x}_N \in \Omega \tag{5.28e}$$
$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{5.28f}$$

where $\mathcal{X}$, $\mathcal{U}$ are sets of state, input constraints respectively, and Ω is a terminal set constraints. The idea for solving the time optimal control lies in partitioning the problem (5.28) into a sequence of horizon $N = 1$ subproblems, and solving them sequentially. Basically, the algorithm consists of two steps:

1. Finding a stabilizing terminal set Ω around origin.
2. Solving a sequence of 1-step optimal control problems that drive the system states towards Ω.

The terminal set Ω is associated with a state feedback law which guarantees the invariance property. That is, once the states enter the set Ω, there exist a local terminal controller which guarantees that states remain in this set for all time. Construction of the invariant set is of crucial importance due stability reasons [70] and this has influence on the stability of overall approach since all states are driven to this set. Hence, if one finds such a set, it is then used in the subsequent horizon 1 control problem as a terminal constraint. Reformulating the optimal control problem into sequences of 1-step control problems has several advantages, namely:

- Closed loop stability is guaranteed by construction of the invariant terminal set and requiring that all states are pushed towards this set in finite time.
- The resulting look-up table is sequentially constructed which simplifies the implementation task.

The stabilizing terminal set Ω can be computed using approaches in [61, 88] as it was presented in Section 4.6. The set Ω is used as a terminal constraint in the consequent 1-step CFTOC problem given by

$$\min_{\boldsymbol{u}_k} \quad \|\boldsymbol{R}\boldsymbol{u}_k\|_p + \|\boldsymbol{Q}\boldsymbol{x}_k\|_p + \|\boldsymbol{P}\boldsymbol{x}_{k+1}\|_p \tag{5.29a}$$
$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{PWA}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.29b}$$
$$\boldsymbol{x}_{k+1} \in \mathcal{S}_k \tag{5.29c}$$
$$\boldsymbol{x}_k \in \mathcal{X} \tag{5.29d}$$
$$\boldsymbol{u}_k \in \mathcal{U} \tag{5.29e}$$

Solving the problem (5.29) parametrically gives the result in a form of PWA state feedback law, which is associated with polytopic regions $\mathcal{P}_{k,i}$, as stated by Theorem 5.1. One important implication of Theorem 5.1 is that the set of states $\mathcal{P}_{k,i}$ for which the problem (5.29) is feasible can be represented as a union of convex polytopes. This allows one to formulate the algorithm for time optimal approach as follows

Algorithm 5.2 (The Time Optimal Algorithm [46])

1. *Calculate an initial stabilizing terminal set using reachability techniques as discussed by [88]. Denote the set by* $\mathcal{S}_0 = \Omega$. *Set the iteration counter* $k = 0$.
2. *Solve parametrically the optimization problem* (5.29) *with the terminal set* $\mathcal{S}_k$ *constraint in* (5.29c). *Denote the feasible set of the problem* (5.29) *as the union of all* i *regions as* $\mathcal{S}_{k+1} = \bigcup_i \mathcal{P}_{k+1,i}$.
3. *If* $\mathcal{S}_{k+1} = \mathcal{S}_k$, *abort, the algorithm has converged.*
4. *Otherwise increment the iteration counter* $k = k + 1$ *and jump back to Step 2.*
5. *The total number of iterations is given by* $k^* = k$.

The explicit solution generated in Step 2 of Algorithm 5.2 may contain overlapping regions due to MILP nature of (5.29). In this case the operations of intersection and comparison are performed as in CFTOC approach. The algorithm is illustrated in Fig. 5.1 for $k = 0, 1, 2, 3$ iterations. The initial terminal set $\mathcal{S}_0 = \Omega$, associated to the iteration counter $k = 0$, is depicted with the darkest gray. The consequent terminal sets $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ have brighter shadows. Because the optimization problem (5.29) is solved for 1-step ahead in each iteration the overall control effect is that the system states are pushed from the set $\mathcal{S}_{k+1}$ to $\mathcal{S}_k$ in one time step, hence the initial stabilizing set is reached at most k^* steps where k^* denotes the number of iterations at which the algorithm converged. One can notice that the whole feasible domain $\mathcal{D}$ might not be covered by the PWA control law (3.21) and white areas in Fig. 5.1 denote parts of the state-space for which the Problem 5.29 is not feasible at any iteration. For further details regarding the computations, see [46] as well as [45, 63].

5.4 On-line Implementation

Once the explicit solution to MPC is computed, the feedback control law is given as PWA function, i.e.

$$\boldsymbol{u}^* = \boldsymbol{F}_i \boldsymbol{x} + \boldsymbol{g}_i \quad \text{if} \quad \boldsymbol{x} \in \mathcal{P}_i \tag{5.30}$$

where $i = 1, \ldots, n_{\mathcal{P}}$ is the number of regions $\mathcal{P}_i$ and $\boldsymbol{F}_i$, $\boldsymbol{g}_i$ are computed via (3.12). The control law (5.30) is defined over $n_{\mathcal{P}}$ polytopes and in the on-line implementation phase of explicit MPC one has to evaluate such a function in real-time. The feedback control scheme is shown in Fig. 4.3 whereas the control law $\boldsymbol{\pi}_0(\boldsymbol{x}_0)$ is given as PWA function (5.30).

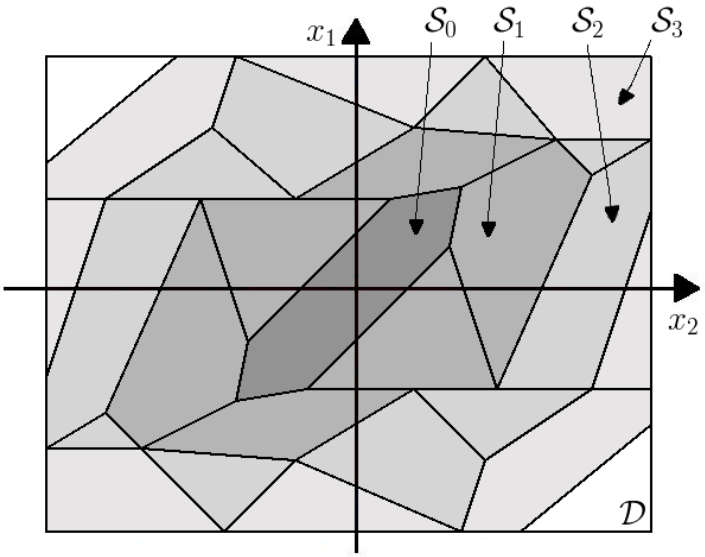


Figure 5.1: Explicit solution to the time optimal control problem. The algorithm evolves from the terminal set $\mathcal{S}_0$ outwards and the union of the sets is the maximum controllable domain.

If there is enough time for evaluation, one may proceed with a *sequential approach*. That is, for given state $\boldsymbol{x}$ loop through regions $\mathcal{P}_i$ until the corresponding region index j is found where $\boldsymbol{x} \in \mathcal{P}_j$. Then evaluate the feedback law (5.30). Such scheme has linear complexity in the number of regions, i.e. $O(n_{\mathcal{P}})$. However there exist cases, where the control scheme is implemented with very high sampling rates, e.g. control of electrical drives, voltage converters [8, 43], and a sequential search can be prohibitive. To deal with this issue, [106] suggested to translate PWA function (5.30) to a *binary search tree* where the complexity of on-line evaluation is logarithmic in number of regions, i.e. $O(\log_2 n_{\mathcal{P}})$. The idea behind is to index the look-up table to a special tree form which allows faster searching through union of regions $\mathcal{P}_i$. The algorithm for construction of the binary tree partitions the union of regions at each step by half, according to *leading hyperplanes*. A leading hyperplane is searched among H-representation of regions $\mathcal{S}_i$ which separates their union in two halfspaces such that they both contain approximately the same number of regions. An illustration of a search tree construction in shown in Fig. 5.2. At each stage the leading hyperplane (shown as red dashed line) is selected and the remaining space is partitioned until individual regions are reached. Moreover, the algorithm of [106] tries to optimize the search tree in a way such that the total number of separating hyperplanes is minimized and the tree contains the minimum number of nodes. The total number of hyperplanes defines the tree depth D, and it expresses the total number of operations needed to traverse from a main root to a particular leaf.

Once the binary search tree has been constructed, it has very attractive properties for evaluation of PWA function in the on-line phase of MPC. In particular, its running time

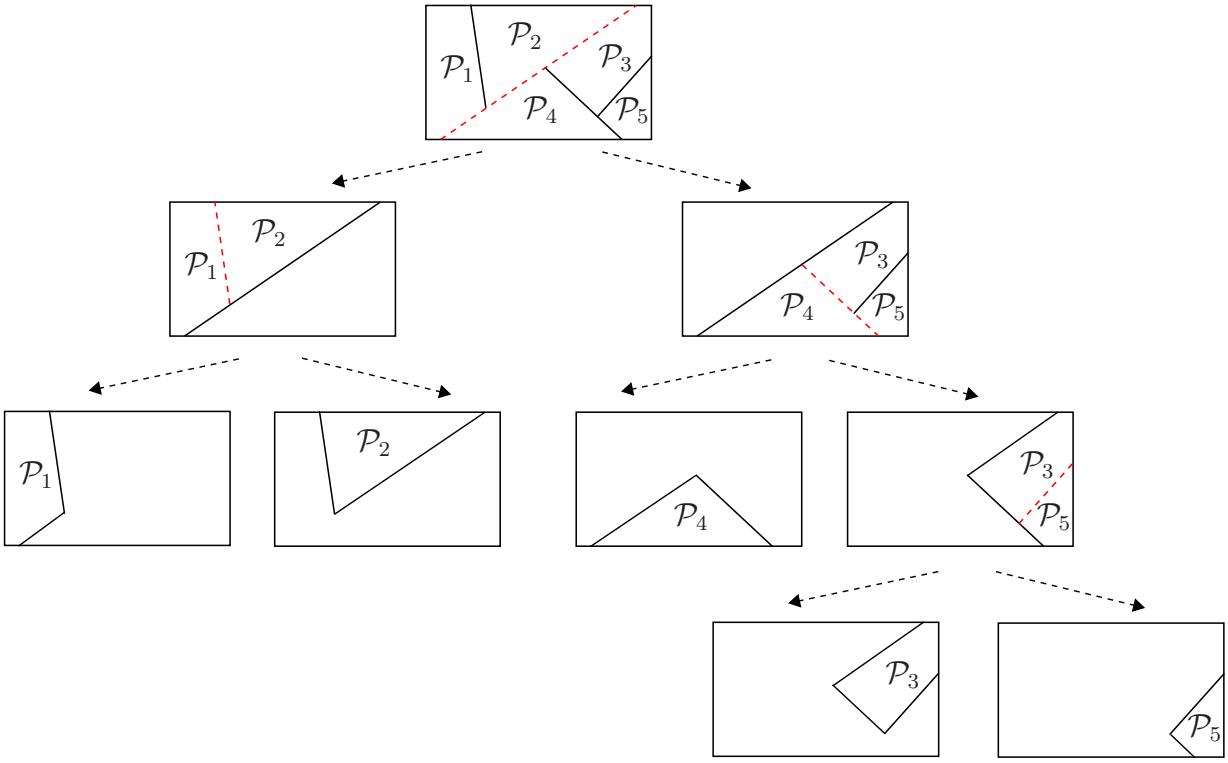


Figure 5.2: Construction of a binary search tree.

is influenced by the tree depth, which is approximately $D = \lceil 1.7 \log_2 n_{\mathcal{P}} \rceil$. Memory and storage requirements depend on the total number of nodes, i.e. $n_{\mathcal{P}}^{1.7}$.

Part II

MODELING AND CONTROL OF HYBRID SYSTEMS

Chapter 6

Modeling of Hybrid Processes

Model predictive control utilizes information from the process model to predict future. Ideally, the model should capture most of the relevant dynamical properties in order to design the best possible control. From MPC point of view, the process model enters the formulation of optimal control problems as equality constraint and influences the complexity of the optimization problem. The problem of finding a suitable model for MPC thus reduces to investigating the difficulty of solving the optimization problem for a concrete application.

It was shown in Sections 5.2.1 and 5.2.2 that if the provided model (4.11) is linear in states and inputs, this structure can be exploited to formulate mpLP or mpQP and solved explicitly. Linear models are thus very suitable for explicit MPC. Furthermore, linear models can be obtained easily in practice, e.g. using the well-established techniques for experimental identification [72]. Many of the identification algorithms are available as software packages that can provide linear models directly from the measurement data, e.g. IDTOOL [34].

However, for many industrial processes the use of linear models is often related to some neighborhood of the operating point and as the plant drifts away from this point, the linear model looses accuracy. Furthermore, many applications can contain logical parts, such as on/off switches, mechanical gears or if-then rules, where the use of LTI models (4.11) is inappropriate. To overcome this limitation, hybrid models have been introduced, and are reviewed in Section 4.4.2. Hybrid models, and in particular PWA systems, have been proved to provide sufficient accuracy when approximating the dynamical behavior of general nonlinear systems [94]. Moreover, as it was shown in Section 5.2.3, hybrid models are suitable for explicit MPC due to their piecewise linear structure. However, when it comes to practice, there's a lack of software tools that can generate hybrid models without much programming effort. Although there are tools capable of programming almost arbitrary dynamics, the resulting model often contains elements from the programming language or it is represented as 'black box'. Especially, when it comes to formulations of optimal control problems, such models are unsuitable for the use in explicit MPC.

Therefore, the contribution of this part is to present a software tool for modeling of hybrid systems which is capable of generating models suitable for explicit MPC. In particular, the

software produces MLD models (4.14) that combine

- linear dynamics
- logical statements
- constraints

in one compact representation. The structure of MLD models is given by a set of linear equations/inequalities in real and binary variables which is specifically tailored for the optimization purposes. Due to equivalence between hybrid models [48] MLD models can be further transformed to other forms [10], such as PWA systems (4.12), and one can deploy existing algorithms for synthesis and implementation of explicit MPC.

6.1 HYSDEL

HYbrid System DEscription Language (HYSDEL) is a software tool for modeling of hybrid systems. The original idea has been introduced by [107] in order to simplify the task of generating MLD models (4.14) from simple language statements. HYSDEL was created to automatize the translation from logical expressions such as (5.21) into equivalent mathematical representation (5.20) using the big-M translation technique of [15].

6.1.1 General Properties

HYSDEL allows modeling of wide class of hybrid systems described by interconnections of linear dynamic systems, automata, if-then-else and propositional logic rules. It allows hybrid models to be formulated on a very low level which is very appealing for processor programming languages. Based on this elementary description, HYSDEL translates the code into equivalent MLD form (4.14) which is suitable for further analysis, including control design. Unlike general-purpose optimization modeling languages, HYSDEL is a specialized language for describing a dynamic behavior combined with logical conditions. Tailored to the specific class of problems, algorithms implemented in the HYSDEL compiler generate models which are more compact and which render the optimization problems using such models more efficient compared to formulations obtained by general-purpose modeling software. The process of generating a model with the help of HYSDEL can be described using three main components:

- HYSDEL source file (input)
- compiler
- mathematical model (output).

At the input side user creates a HYSDEL source code, represented by HYSDEL file, which is consequently passed to a compiler. The compiler executes the big-M translation technique on a given script and outputs MLD model. The operational principle is shown in Fig. 6.1.

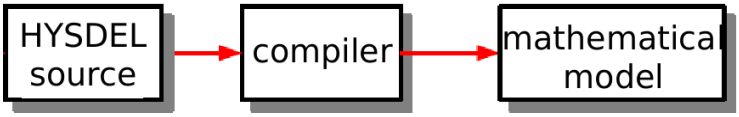


Figure 6.1: Principle of model generation in HYSDEL.

6.1.2 HYSDEL Language Syntax

The source file is created using HYSDEL programming language which allows to use difference equations, on/off switches, IF-THEN-ELSE rules, and finite state automata in a programming-typical way. Structure of HYSDEL source file can be categorized in two main parts

- interface part
- implementation part.

The interface part serves as a place for declaration of variables and in the implementation part the relations between variables are defined. Each of the mode is delimited by curly brackets and a typical structure of a HYSDEL source file looks as follows

```
SYSTEM name {
 /* example of HYSDEL file structure */
    INTERFACE {
       /* declaration of variables */
    }
    IMPLEMENTATION {
       /* relations between declared variables */
    }
 }
```

HYSDEL language is similar to C programming language, however, the flexibility of the code is different. In particular, the HYSDEL version 2.0.5 by [107] does not support vectors and matrices as variables and lacks important language constructs like loops. These drawbacks make the description of large models cumbersome and prone to errors. For instance, if one wants to write a code for a linear time-invariant model, the possible HYSDEL script might look as shown in the following example, where each variable must be written as scalar, i.e.

```
SYSTEM model {
 INTERFACE {
   STATE { REAL x1, x2; }
   INPUT { REAL u1, u2; }
   OUTPUT { REAL y; }
   PARAMETER {  REAL a11=1;    REAL a12=-0.2;
```

```
                REAL a21=-0.3; REAL a22= 0.5; }
  }
 IMPLEMENTATION {
  CONTINUOUS {
    x1=a11*x1 + a12*x2 + u1;
    x2=a21*x1 + a22*x2 + u2;
  }
  OUTPUT {
    y = x1 + x2;
  }
 }
}
```

Moreover, during the last 2 years the development of HYSDEL has not been following the progress of research in the topic, i.e. the generation of MLD models from logic conditions uses a relatively limited set of algorithms originally implemented in HYSDEL compiler, while recently several tools have emerged providing means for more advanced MLD and PWA model formulations, most notably MPT [64] and YALMIP [67].
From the usability point of view, the current version of HYSDEL has some major drawbacks. Thus, to overcome all of the shortcomings, HYSDEL 3.0 has been developed, which offers more enhanced modeling, code flexibility and graphical interface.

6.2 HYSDEL 3.0

The new developed version of HYSDEL 3.0 includes several enhancements, in particular,

- extension of HYSDEL syntax
- new improved compiler
- generation of higher quality models
- merging of models
- graphical modeling of hybrid systems.

The extended HYSDEL syntax can be characterized by use of variables in vectorized/matrix form, indexed access to declared variables, nested FOR loops, and structured model merging. Enhanced syntax allows now code writing in a rationalized form which is more easier to follow and revise. Returning to the previous example of LTI model, new HYSDEL 3.0 syntax looks as follows

```
SYSTEM model {
 INTERFACE {
   STATE { REAL x(2); }
```

```
   INPUT { REAL u(2); }
   PARAMETER {  REAL A = [1, -0.2; -0.3,0.5]; }
   }
 IMPLEMENTATION {
   CONTINUOUS { x = A*x + u;  }

 }
}
```

Note that the size of code is reduced due to vectorized syntax. Further code reductions is obtained by simplifying the earlier commands. For instance, PWA model comprising of two dynamics can be written as follows

```
SYSTEM model {
 INTERFACE {
   STATE { REAL x(2); }
   INPUT { REAL u(2); }
   PARAMETER {  REAL A1 = [1, -0.2; -0.3, 0.5];
                REAL A2 = [0.9, -0.8; 1.2, 0.7]; }
   }
 IMPLEMENTATION {
   AUX { REAL z(2); }
   DA { z = { IF x(1) >= 0 THEN A1*x + u ELSE A2*x + u;}  }
   CONTINUOUS { x = z; }
 }
}
```

The compiler in HYSDEL 3.0 uses YALMIP to generate MLD model. YALMIP is a MATLAB toolbox for rapid prototyping of optimization problems [67] and it allows generation of numerically well conditioned models. The operation principle of HYSDEL 3.0 is shown in Fig. 6.2. Introduction of YALMIP extended the possibilities of HYSDEL 3.0 language.

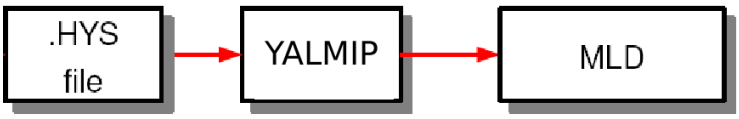


Figure 6.2: Operational principle of HYSDEL 3.0.

In particular, HYSDEL 3.0 offers compositional modeling, i.e. generation of models by merging of submodels. This has a strong advantage for modeling of large scale systems because it is allowed to write the description in multiple source files and merge them together directly on a language level. In addition, HYSDEL 3.0 incorporates some of the tricks investigated in [63] for efficient implementation of big-M technique needed for generation of high quality models. The produced models can be further optimized to get numerically sound models as best as possible. HYSDEL 3.0 thus features various enhancements and its code extensions will be reviewed in the following.

6.2.1 MLD System Formulation

Arising from shortcomings of MLD model used by HYSDEL 2.0.5, new version of HYSDEL 3.0 offers more flexibilities to describe the behavior of hybrid systems. In order to clearly distinguish between real/binary variables and equalities/inequalities in MLD formulation, sets of indices are added to MLD form and new notations are adopted. More precisely, MLD description is given by

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}_u\boldsymbol{u}_k + \boldsymbol{B}_{\text{aux}}\boldsymbol{w}_k + \boldsymbol{B}_{\text{aff}} && \text{(6.1a)} \\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}_u\boldsymbol{u}_k + \boldsymbol{D}_{\text{aux}}\boldsymbol{w}_k + \boldsymbol{D}_{\text{aff}} && \text{(6.1b)} \\ & \boldsymbol{E}_x\boldsymbol{x}_k + \boldsymbol{E}_u\boldsymbol{u}_k + \boldsymbol{E}_{\text{aux}}\boldsymbol{w}_k \;\leq\; \boldsymbol{E}_{\text{aff}} && \text{(6.1c)} \\ \{\text{sets of indices}\} & \quad J_x,\ J_u,\ J_w,\ J_{\text{eq}},\ J_{\text{ineq}} && \text{(6.1d)} \end{aligned}$$

where the auxiliary vector $\boldsymbol{w}_k$ comprises of two elements $\boldsymbol{w}_k = (\boldsymbol{z}_k^T,\ \boldsymbol{\delta}_k^T)^T$. Comparing to the previous description (4.14), the model contains indices (6.1d) that indicate which variables in the vector correspond to real/binary for states J_x, inputs J_u, and auxiliaries J_w. Additionally, J_{eq} corresponds to a set of indices that define rows of matrices (6.1c) with equality constraints and J_{ineq} with inequalities, i.e.

$$\begin{aligned} \boldsymbol{E}_x^{\text{eq}}\boldsymbol{x}_k + \boldsymbol{E}_u^{\text{eq}}\boldsymbol{u}_k + \boldsymbol{E}_{\text{aux}}^{\text{eq}}\boldsymbol{w}_k &= \boldsymbol{E}_{\text{aff}}^{\text{eq}} \\ \boldsymbol{E}_x^{\text{ineq}}\boldsymbol{x}_k + \boldsymbol{E}_u^{\text{ineq}}\boldsymbol{u}_k + \boldsymbol{E}_{\text{aux}}^{\text{ineq}}\boldsymbol{w}_k &\leq \boldsymbol{E}_{\text{aff}}^{\text{ineq}} \end{aligned}$$

Structure of the indexed sets J_x, J_u, J_w is vector-wise and it distinguishes between binary and real variables with strings 'r', 'b'. Precisely, string 'r' refers to real and string 'b' refers to Boolean variable, e.g.

$$\{J_x\} = \begin{pmatrix} \texttt{'r'} \\ \texttt{'r'} \\ \texttt{'b'} \\ \texttt{'r'} \end{pmatrix} \quad \text{corresponds to} \quad \begin{pmatrix} x_1 \text{ is of type } \texttt{REAL} \\ x_2 \text{ is of type } \texttt{REAL} \\ x_3 \text{ is of type } \texttt{BOOL} \\ x_4 \text{ is of type } \texttt{REAL} \end{pmatrix}$$

To be able to determine the position of a given real/binary variable in a vector, these sets contain also a numerical information. For instance, the location of equality constraints in matrices (6.1c) is given by J_{eq} and refers to rows in (6.1c) which form equalities e.g.

$$J_{\text{eq}} = \begin{pmatrix} 1 \\ 5 \\ 8 \end{pmatrix} \quad \text{corresponds to} \quad \begin{pmatrix} 1\text{st row is an equality constraint} \\ 5\text{th row is an equality constraint} \\ 8\text{th row is an equality constraint} \end{pmatrix}$$

6.2.2 Using HYSDEL 3.0

Procedure for generation of the MLD system (6.1) starts in writing a corresponding source in some text editor using HYSDEL language and specifying the suffix `.hys`. Assume, that a file with name `my_file.hys` was created and written in HYSDEL language. To get an appropriate MLD representation (6.1), the file needs to be processed in MATLAB environment as follows:

```
>> hysdel3('my_file.hys')
```

where `hysdel3` is the main routine for executing the compilation. By invoking this command in MATLAB, an m-file equivalent of the `my_file.hys`, i.e. `my_file.m` is generated in the current directory. This new generated m-file is basically YALMIP code which contains whole information given by `my_file.hys`. The m-file represents now MLD model (6.1) and by typing simply the same name at MATLAB prompt

```
>> my_file
```

the script automatically generates a structure called S. The structure contains all details about MLD model (6.1), i.e. matrices, dimensions, indexes of constraints, etc. If the structure S is available, each particular information about the MLD model (6.1) can be extracted using MATLAB "dot" syntax, i.e.

```
S.nx  /* dimension of states */
 .nxr /* dimension of real states */
 .nxb /* dimension of binary states */
 .nu  /* dimension of inputs */
 .nur /* dimension of real inputs */
 .nub /* dimension of binary inputs */
 .ny  /* dimension of outputs */
 .nyr /* dimension of real outputs */
 .nyb /* dimension of binary outputs */
 .nw  /* dimension of auxiliary variables */
 .nz  /* dimension of auxiliary real variables */
 .nd  /* dimension of auxiliary binary variables */
 .nc  /* dimension of constraints (equalities + inequalities) */
```

Matrices introduced in MLD model (6.1) are stored as fields with the same name, i.e.

```
S.A
 .Bu
 .Baux
 .Baff
 .C
 .Du
 .Daux
 .Daff
 .Ex
 .Eu
 .Eaux
 .Eaff
```

and indexed sets are accessible through substructure `J`

```
S.J.X    /* string of indices ('r' or 'b') for states */
 .J.U    /* string of indices ('r' or 'b') for inputs */
 .J.Y    /* string of indices ('r' or 'b') for outputs */
 .J.W    /* string of indices ('r' or 'b') for auxiliary variables */
```

Information about the numerical position of variables in a vector is given by a substructure `j`, i.e.

```
S.j.xr    /* indices of REAL states */
 .j.xb    /* indices of BOOL states */
 .j.ur    /* indices of REAL inputs */
 .j.ub    /* indices of BOOL inputs */
 .j.yr    /* indices of REAL outputs */
 .j.yb    /* indices of BOOL outputs */
 .j.d     /* indices of BOOL auxiliary variables */
 .j.z     /* indices of REAL auxiliary variables /*
 .j.eq    /* indices of equality constraints */
 .j.ineq  /* indices of inequality constraints */
```

Further information about MLD model are stored in remaining substructures of the variable S. Here belong the names, types, and dimensions of declared variables, i.e.

```
S.InputName     /* names of input variables */
 .InputKind     /* types of the input variables (real, binary) */
 .InputLength   /* dimensions of the input variables */
 .StateName     /* names of state variables */
 .StateKind     /* types of the state variables (real, binary) */
 .StateLength   /* dimensions of the state variables */
 .OutputName    /* names of output variables */
 .OutputKind    /* types of the output variables (real, binary) */
 .OutputLength  /* dimensions of the output variables */
 .AuxName       /* names of auxiliary variables */
 .AuxKind       /* types of the auxiliary variables (real, binary) */
 .AuxLength     /* dimensions of the auxiliary variables */
```

upper and lower bounds

```
S.xl     /* lower bound on state variables */
 .xu     /* upper bound on state variables */
 .ul     /* lower bound on input variables */
 .uu     /* upper bound on input variables */
 .wl     /* lower bound on auxiliary variables */
 .wu     /* upper bound on auxiliary variables */
```

The number of total constraints is given by the field

```
S.nc     /* total number of constraints (equalities + inequalities) */
```

Information about possible symbolic variables in HYSDEL file is stored in the field

```
S.symtable     /* information about declared variables */
```

as a substructure with additional fields. Since HYSDEL 3.0 offers merging of several MLD structures on a syntactical level, the information about the interconnections between the local models is stored in field

```
S.connections   /* table of interconnections between HYSDEL modules */
```

Next section describes the particular extensions in HYSDEL 3.0 language.

6.2.3 Language Elements

INTERFACE Section

The INTERFACE section defines main variables which appear for the given modeled SYSTEM. More precisely, this section defines input, state and output variables distinguished by strings INPUT, STATE, and OUTPUT. Moreover, additional variables which do not belong to these classes are supposed to be declared in the section called PARAMETER. The MODULE section is to be present where subsystems are declared.
Syntactical structure of the INTERFACE section is of the following form

```
INTERFACE { interface_item }
```

which comprises of curly brackets and *interface_item*. The *interface_item* may take only following forms

```
/* allowed INTERFACE items */
   MODULE { /* module_item */ }
   INPUT { /* input_item */ }
   STATE { /* state_item */  }
   OUTPUT { /* output_item */ }
   PARAMETER { /* parameter_item */ }
```

where each item appears only once in this section. Each *interface_item* is separated at least with one space character and may be omitted, if it is not required. The order of each item can be arbitrary, it does not play a role for further processing.

Example 6.1 *A structure of a standard HYSDEL file is shown here, where the INTERFACE items are separated by paragraphs, and curly brackets denote visible start- and endpoints of each section.*

```
SYSTEM name {
/* example of HYSDEL file */
   INTERFACE {
      /* declaration of variables, subsystems */
      MODULE {
         /* declaration subsystems */
         ...
      }
      INPUT {
         /* declaration input variables */
         ...
      }
      STATE {
         /* declaration state variables */
         ...
      }
      OUTPUT {
         /* declaration output variables */
         ...
      }
      PARAMETER {
         /* declaration of parameters */
         ...
      }
   }
   IMPLEMENTATION {
      /* relations between declared variables */
         ...
   }
}
```

The main change comparing to previous version is that input variables can be defined as vectors, whereas the dimension of the input vector of real variables is n_{u_r} and n_{u_b} for binary variables. This allows to define vectorized variables in the meaning of $\boldsymbol{u}_r \in \mathbb{R}^{n_{u_r}}$, $\boldsymbol{u}_b \in \{0,\ 1\}^{n_{u_b}}$ and the corresponding syntax is as follows

```
REAL var(nur) [var_1_min, var_1_max;
               var_2_min, var_2_max;
                     ...,       ...;
               var_nur_min, var_nur_max];
```

for variables of type REAL and

```
BOOL var(nub);
```

for Boolean variables where `var` is the name of the variable. Note that according to expression in parentheses "(" ")", which denotes the dimension of the vector, the lower and upper bounds must be specified for each variable in this vector. The above syntax of variable declaration is valid for each of the *interface_item* except the case, if the variable to be defined is symbolic parameter. For symbolic parameters the declaration reads

```
PARAMETER {
 type var;       /* symbolic scalar */
 type var(n);    /* symbolic vector */
 type var(n,m);  /* symbolic matrix */
}
```

where the string `type` is either REAL or BOOL and the variable `var` can be scalar, vector with n rows, or matrix with dimensions n and m.

Example 6.2 *We want to declare constants* $a = 1$ *as Boolean variable,* $\boldsymbol{b} = (-1,\ 0.5,\ -7.3)^T$ *and* $\boldsymbol{D} = \begin{pmatrix} -1 & 0 & 0.23 \\ 0.12 & -0.78 & 2.1 \end{pmatrix}$ *as real variables. This can be done as follows*

```
PARAMETER {
  BOOL a = 1; REAL b = [-1; 0.5; -7.3];
  REAL D = [-1, 0, 0.23; 0.12, -0.78, 2.1];
}
```

IMPLEMENTATION Section

Relations between variables are determined in the IMPLEMENTATION section. According to type of variables, this section is further partitioned into subsections, which remain the same as in previous version. Syntactical structure of the IMPLEMENTATION section has the following form

```
IMPLEMENTATION { implementation_item }
```

which comprises of curly brackets and *implementation_item*. The *implementation_item* may take only following forms

```
AUX { /* aux_item */ }
CONTINUOUS { /* continuous_item */ }
AUTOMATA { /* automata_item */ }
LINEAR { /* linear_item  */ }
LOGIC { /* logic_item  */ }
AD {  /* ad_item */ }
DA {  /* da_item */ }
MUST { /* must_item */ }
OUTPUT { /* output_item */ }
```

where each item appears only once in this section. Each *implementation_item* is separated at least with one space character and may be omitted, if it is not required. The order of each item can be arbitrary, it does not play a role for further processing. Note that OUTPUT section is also present as in the INTERFACE part but here it has different syntax and semantics.

Example 6.3 *A structure of the standard HYSDEL file is given next where the meaning of each subsection of the IMPLEMENTATION part is briefly explained*

```
SYSTEM name {
    INTERFACE {
       /* declaration of variables, subsystems */
    }
    IMPLEMENTATION {
       /* relations between declared variables */
      AUX {
           /* declaration of auxiliary variables, needed for
              calculations in the IMPLEMENTATION section */
      }
      CONTINUOUS {
           /* state update equation for variables of type REAL */
      }
      AUTOMATA {
           /* state update equation for variables of type BOOL */
      }
      LINEAR {
           /* linear relations between variables of type REAL */
      }
      LOGIC {
           /* logical relations between variables of type BOOL */
      }
      AD {
           /* analog-digital block, specifying relations between
              variables of type REAL to BOOL */
      }
      DA {
           /* digital-analog block, specifying relations between
              variables of type BOOL to REAL */
      }
      MUST {
           /* specification of input/state/output constraints */
      }
      OUTPUT {
```

```
            /* selection of output variables which can be of type
               REAL or BOOL) */
      }
    }
}
```

HYSDEL 3.0 enhancements in the IMPLEMENTATION section can be applied in any of the *implementation_item* and the common features are listed as follows:

- indexing
- FOR and nested FOR loops
- operators and built-in functions

Indexing

Introducing vectors and matrices induced the extension of the HYSDEL 3.0 language to use indexed access to internal variables. The syntax is different for vectors and matrices since it depends on the dimension of the variable. Access to vectorized variables has the following syntax

```
new_var = var(ind);
```

where `new_var` denotes the name of the auxiliary variable (must be defined in AUX section), `var` is the name of the internal variable and `ind` is a vector of indices, referring to position of given elements from a vector. Indexing is based on a MATLAB syntax, where the argument `ind` must contain only $\mathbb{N}^+ = \{1,\ 2,\ \ldots\}$ valued elements and its dimension is less or equal to dimension of the variable `var`. Syntax of the `ind` vector can be one of the following:

- increasing/decreasing sequence

  ```
  ind_start:increment:ind_end
  ```

  where `ind_start` denotes the starting position of indexed element, `increment` is the value of which the starting value increases/decreases, and `ind_end` indicates the end position of indexed element.

- increasing by one sequence

  ```
  ind_start:ind_end
  ```

  where the value `increment` is now omitted and HYSDEL 3.0 automatically treats the value as +1

- particular positions

  ```
  [pos_1, pos_2, ..., pos_n]
  ```

  where `pos_1`, ..., `pos_n` indicates the particular position of elements

- nested indices

  ```
  ind(sub_ind)
  ```

  where the vector `ind` is sub-indexed via the aforementioned ways by vector `sub_ind` with $\mathbb{N}^+$ values

Example 6.4 *In the parameter section were defined two variables. The first variable is a constant vector* $\boldsymbol{h} = (-0.5,\ 3,\ 1,\ \pi,\ 0)^T$ *and the second variable is a symbolical expression* $\boldsymbol{g} \in \mathbb{R}^3$, $g_1 \in [-1,\ 1]$, $g_2 \in [-2,\ 2]$, $g_3 \in [-3,\ 3]$. *We want to assign new variables* $\boldsymbol{z}$ *and* $\boldsymbol{v}$ *for particular elements of these vectors. Examples are:*

- *increasing sequence, e.g.* $\boldsymbol{z} = (-0.5,\ 1,\ 0)^T$

  ```
  z = h(1:2:5);
  ```

- *decreasing sequence, e.g.* $\boldsymbol{v} = (g_3,\ g_2)^T$

  ```
  v = g(3:-1:2);
  ```

- *increasing by one, e.g.* $\boldsymbol{z} = (1,\ \pi,\ 0)^T$

  ```
  z = h(3:5);
  ```

- *particular positions, e.g.* $\boldsymbol{v} = (g_1,\ g_3)^T$

  ```
  v = g([1,3]);
  ```

- *nested indexing, e.g.* $\boldsymbol{z} = (3,\ \pi)^T$, $\boldsymbol{k} = (2,\ 3,\ 4)$

  ```
  z = h(k([1, 3]));
  ```

  where the variable $\boldsymbol{k}$ *has to be declared first.*

FOR Loops

FOR loops are another important feature of HYSDEL 3.0 version. To create a repeated expression, one has to first define an iteration counter in the AUX section according to syntax

```
AUX {
  INDEX iter;
}
```

where the prefix INDEX denotes the class, and `iter` is the name of the iteration variable. If there are more iteration variables required, the additional variables are separated by commas ",", i.e.

```
AUX {
  INDEX iter1, iter2, iter3;
}
```

As the iteration variable is declared, the FOR syntax takes the form of

```
FOR ( iter = ind ) { repeated_expr }
```

where the string FOR is followed by expression in normal brackets "(", ")" and expression in curly brackets "{", "}". The expression in normal brackets is characterized by assignment `iter = ind` where the iteration variable `iter` incrementally follows the set defined by variable `ind` and this variable takes one of the form shown in Section 6.2.3 Indexing. The expression in curly brackets named *repeated_expr* is recursively evaluated for each value of iterator `iter` and can take the form of

Operators and Built-in Functions

Throughout the whole HYSDEL source file various relations between variables can be defined. Because the variables may be also vectors (matrices), the list of supported operators is distinguished by

- element-wise operations (Tab. 6.1)
- and vector functions (Tab. 6.2).

For Boolean variables HYSDEL supports operators, as summarized in Tab. 6.3. Additionally, syntactical functions are available, which allow condition checking and are summarized in Tab. 6.4.

Relation	Operator	HYSDEL representation
addition	$+$	`+`
subtraction	$-$	`-`
multiplication	$.$	`*`
elementwise multiplication	$.$	`.*`
division	$/$	`/`
elementwise division	$/$	`./`
absolute value	$\|a\|$	`abs(a)`
a to the power of b	a^b	`a.^b`
e^b	$\exp b$	`exp(b)`
square root	$\sqrt{a}$	`sqrt(a)`
common logarithm (with base 10)	$\log a$	`log10(a)`
natural logarithm (with base e)	$\ln a$	`log(a)`
binary logarithm (with base 2)	$\log_2 a$	`log2(a)`
cosine	$\cos a$	`cos(a)`
sine	$\sin a$	`sin(a)`
tangent	$\tan a$	`tan(a)`
rounding function	round	`round`(a)
rounding function	ceil	`ceil`(a)
rounding function	floor	`floor`(a)

Table 6.1: List of supported element-wise operators on variables of type REAL.

Relation	Operator	HYSDEL representation
matrix/vector addition	$+$	`+`
matrix/vector subtraction	$-$	`-`
matrix/vector multiplication	$.$	`*`
matrix power	A^b	`A^b`
vector sum	$\sum_i a_i$	`sum(a)`
1-norm of a vector	$\sum_i \|a_i\|$	`norm_1(a)`
∞-norm of a vector	$\max\|a_i\|$	`norm_inf(a)`

Table 6.2: List of supported vector/matrix operators on variables of type REAL.

Relation	Operator	HYSDEL representation
or	$\vee$	`|` or `||`
and	$\wedge$	`&` or `&&`
one way implication	$\Rightarrow$	`->`
one way implication	$\Leftarrow$	`<-`
equivalence	$\Leftrightarrow$	`<->`
negation	$\neg$	`~` or `!`

Table 6.3: List of supported operators on variables of type BOOL.

Function	HYSDEL representation
true if all elements of a vector are nonzero	`all(a)`
true if any element of a vector is nonzero	`any(a)`

Table 6.4: List of functions for condition checking.

6.2.4 Compiler

HYSDEL 3.0 incorporates compiler that uses YALMIP [67] for rapid algorithm development. The language of YALMIP is consistent with standard MATLAB syntax, thus making it extremely simple to use for anyone familiar with MATLAB. Another benefit of YALMIP is that it implements many advanced techniques allowing the user to concentrate on the high-level model, while YALMIP takes care of the low-level modeling to obtain as efficient and numerically sound models as possible.
The YALMIP's ability to produce a numerically well-conditioned models stands for generation of higher quality MLD models, which at the end serve for better conditioned optimization. Another important aspects of YALMIP are its easy maintainability, independence from operating system platforms and its relatively easy way of code extension.

6.2.5 Graphical Modeling

HYSDEL 3.0 offers a graphical layer of modeling which includes also model merging. This layer comprises of "HYSDEL 3.0 Model" block which is a basic HYSDEL 3.0 library block. In addition, this block can be further converted into arbitrary library unit, depending on which source file it refers to. By this way the user can create its own library of standard units (e.g. valves, pumps, burners, etc.) which can be dragged and dropped to a graphical interface and connected thereafter. HYSDEL 3.0 offers the possibility to transform the whole graphical layer to a single HYSDEL source file to create the corresponding MLD model. By this way the user does not have to take care about the code writing which is done automatically. An example of a simple block scheme is depicted in Fig. 6.3. Each

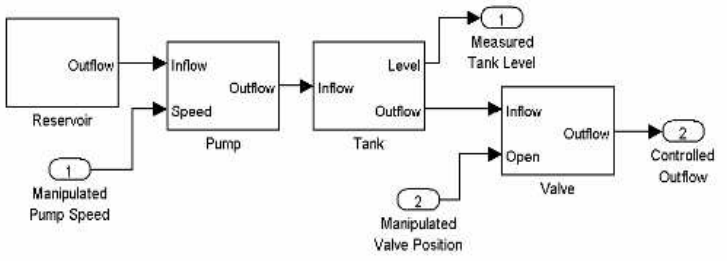


Figure 6.3: Graphical layer of HYSDEL 3.0.

of the library's block is basically a separate HYSDEL file with its own parameters. This approach allows user to adjust the dynamical properties of given block by specifying the model parameter and increases the modularity of the overall model.

Creating Subsystems

Subsystems are declared in the MODULE section according to their file's name and containing parameters. Structurally, subsystems are independent HYSDEL 3.0 source files

with given names, inputs, outputs, states, parameters and are contained in the same directory as the *master* file. Consider a simple production system which is built by two storage tanks, one conveyor belt and a packaging unit, as illustrated in Fig. 6.4. The task is to model the whole production unit using MLD system. First, the production unit is virtually separated into subsystems, as shown in Fig. 6.5 and the connections between the subsystems are clarified.

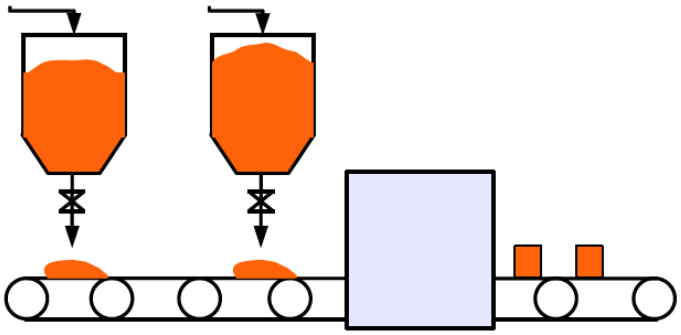


Figure 6.4: Illustrative example of a production system.

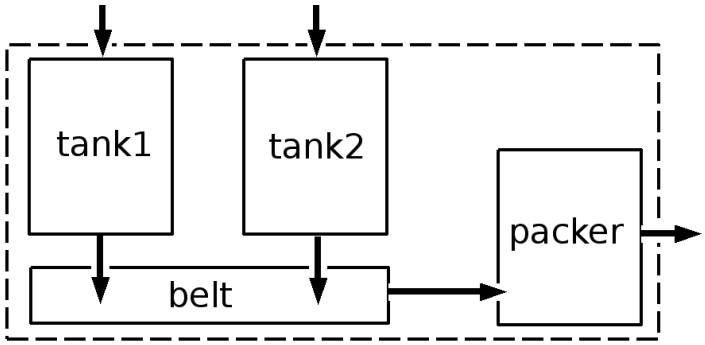


Figure 6.5: Partitioning the overall process into subsystems.

The master file can be denoted as `production` and at first sight it seems that one would declare four slave files, e.g.

1. `tank1`
2. `tank2`
3. `belt`
4. `packer`

but because tanks `tank1` and `tank2` have the same dynamics, it is better to treat these objects as parameters of one file. Subsequently, it is possible to group these both tanks into one file and end up in only three subsystems, e.g.

1. `storage_tank`
2. `conveyor_belt`
3. `packaging`

where each of the objects is now an individual hys-file and can contain arbitrary number of parameters. Declaration of these subsystems in HYSDEL 3.0 takes a form

```
MODULE {
  storage_tank tank1, tank2;
  conveyor_belt belt;
  packaging packer;
}
```

and it corresponds to three HYSDEL 3.0 source files which will be included, i.e.

1. `storage_tank.hys`
2. `conveyor_belt.hys`
3. `packaging.hys`

Each of these file is a *slave* file. For instance, the file `storage_tank.hys` may be written in HYSDEL 3.0 language as follows

```
SYSTEM storage_tank {
  INTERFACE {
    STATE  { REAL level; }
    INPUT  { REAL u; }
    OUTPUT { REAL y; }
    PARAMETER {
       REAL diameter;
       REAL k=1e-2;
    }
  }
  IMPLEMENTATION {
    CONTINUOUS { level = 0.9*level+1/diameter*(u-k*level);}
    OUTPUT { y = level; }
  }
 }
```

where one parameter as kept as symbolic. This is important, as it allows to define only one hys-file which can be used several times for different values of this parameter, thus referring to multiple objects. Similarly, one can create other slave files, in particular

```
SYSTEM conveyor_belt {
  INTERFACE {
    STATE  { REAL speed [0, 6]; }
    INPUT  { BOOL sw; }
    OUTPUT { REAL y; }
    PARAMETER {
       REAL k=[0.5, 0.1];
    }
  }
  IMPLEMENTATION {
    AUX { REAL z; }
    DA { z = {IF sw THEN k(1)*speed+2*(REAL sw) ELSE k(2)*speed }; }
    CONTINUOUS { speed = z; }
    OUTPUT { y = speed; }
  }
 }
```

and

```
SYSTEM packaging {
 INTERFACE {
   INPUT  { REAL inflow [0, 50]; }
   OUTPUT { REAL outflow; }
 }
 IMPLEMENTATION {
   OUTPUT { outflow = 0.5*inflow; }
 }
}
```

which does not contain any symbolic parameters. As slave files have been declared, each of them can be compiled individually and used for simulation. However, since the interest is to have a single MLD file, next section gives an outline how to do this.

Model Merging

Having the subsystems declared, it is now possible to link them together in a single source file using LINEAR or LOGIC section (depending on the class of connected variables). Here it is important to know the "dot" syntax of MATLAB, which is used to access the inputs/outputs of subsystem in a hierarchy tree. For instance, we have declared parameters `tank1` and `tank2` in a MODULE section of a slave file `storage_tank`, which has input variable denoted as `u` and output variable `y`. Access to these variables is done via "dot" syntax, i.e.

```
tank1.y
tank1.u
```

```
tank2.y
tank2.u
```

and it can be extended to vector notation, e.g. `tank1.y(1)` etc. Using this syntax, one can now merge the slave files to one single master file with name `production.hys`

```
SYSTEM production {
   INTERFACE {
      MODULE {
        storage_tank tank1, tank2;
        conveyor_belt belt;
        packaging packer;
      }
      STATE { REAL time [0, 1000]; }
      INPUT { REAL raw_flow(2) [0, 5; 0, 5]; }
      OUTPUT { REAL packages; }
    }
   IMPLEMENTATION {
      AUX { BOOL d;}
      LINEAR {
       tank1.u = raw_flow(1);
       tank2.u = raw_flow(2);
       packer.inflow  = 0.1*tank1.y + 0.5*tank2.y;
      }
      LOGIC { belt.sw = d; }
      AD { d = packer.inflow > 0; }
      CONTINUOUS { time = time + 1; }
      OUTPUT { packages = packer.outflow; }
      MUST {
       tank1.level <= 100;
       tank2.level <= 80;
      }
    }
}
```

where all the subconnections are declared in LINEAR and LOGIC section. Note that the master file `production.hys` has its own inputs and outputs and these are assigned to subsystems via "dot" syntax. Furthermore, one can access the state variables of single subsystems and do additional operations with them, e.g. adding constraints as given in MUST section of `production.hys` file.
After all the involved sub-connections in the master file are declared, one can proceed with compilation of the master file

```
>> hysdel3('production')
```

```
Creating instance "tank1" of "storage_tank"...
Creating instance "tank2" of "storage_tank"...
Creating instance "belt" of "conveyor_belt"...
Creating instance "packer" of "packaging"...
```

and HYSDEL 3.0 generates an m-file equivalent of the input file (here `production.m`). The generated m-file is a YALMIP code, given as a script which returns MLD model (6.1) in a symbolical form. The values of symbolical parameter have the same names as they were defined in the source file (i.e. `tank1.diameter`, `tank2.diameter`). The final representation can be obtained via command

```
>> production
```

and it returns MLD model (6.1) for the given production system depicted in Fig. 6.4. The procedure for generating MLD form of the master file can be done simple on a graphical level as long the source master file `production.hys` is available. For this purpose, the standard HYSDEL 3.0 block is copied from HYSDEL 3.0 Simulink library to a new scheme and all the required fields as shown in Fig. 6.6 are filled. Here, the values of symbolical parameters for tank diameter have been assigned in `par` structure. After confirming the OK option, HYSDEL 3.0 automatically compiles the source file and generates a subsystem with corresponding input/output ports. Consequently, one can add other blocks to the scheme and the result might look as given in Fig. 6.6. By pressing START button in Simulink, the simulation of MLD system should output the number of packages generated for given input raw flow, which is illustrated in Fig. 6.7.

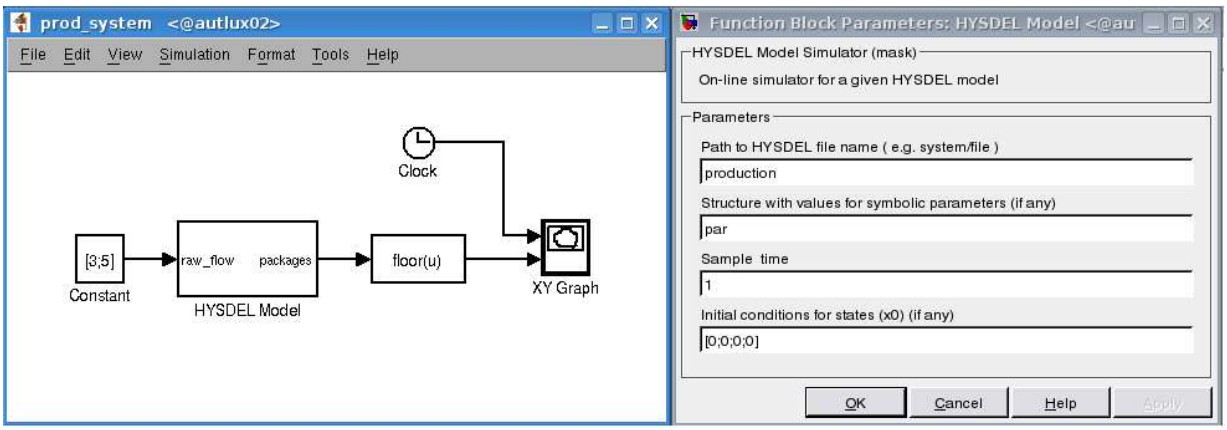


Figure 6.6: Production system using HYSDEL 3.0 graphical layer.

6.3 Translation to PWA System

Once MLD structure is available in MATLAB workspace, one can use it in several ways, including simulation of hybrid systems, or control design using MPT toolbox [64]. First,

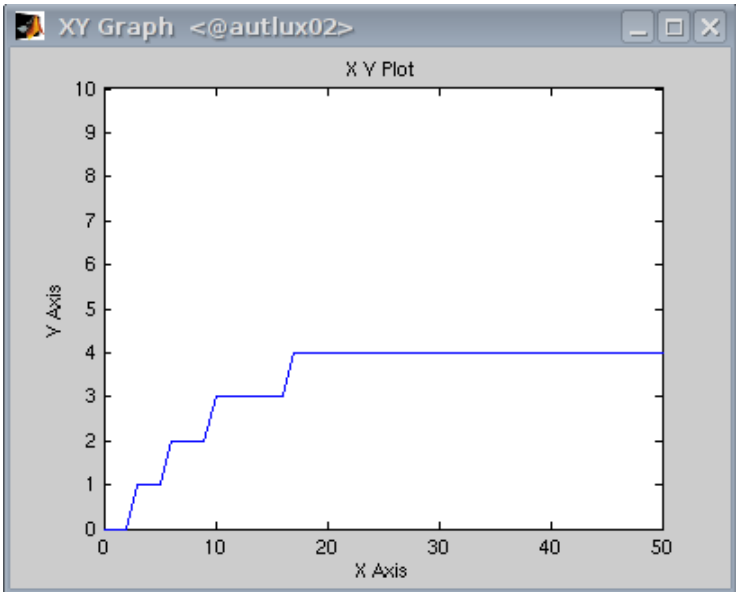


Figure 6.7: Output from the "production.hys" system.

however, MLD model needs to be transformed to PWA model required by MPT toolbox. The principle of conversion is based on an equivalence between different forms of hybrid system [48] and an efficient transformation from MLD description to PWA system has been shown in [10]. The algorithm is implemented in HYSDEL 3.0 under function **h3_mld2pwa**. The use of this function is as follows

```
>> sysStruct = h3_mld2pwa(S)
```

where the input argument is MLD structure S and the output `sysStruct` is PWA model described in MPT format.

Example 6.5 *The aim is to obtain PWA representation of a simple hybrid model of car. The dynamics of a car is given by three states* ***position***, ***velocity***, ***turbocount*** *and the switching between two hybrid modes (normal mode/turbo mode) is given by an external binary input* ***turbo***. *The source file might be given as follows:*

```
SYSTEM turbo_car {
  INTERFACE {
    STATE {
      REAL position [-50, 50];
      REAL velocity [-10, 10];
      REAL turbocount [-10, 10];
    }
    INPUT {
      REAL acc [-1, 1];
      BOOL turbo;
```

```
    }
    OUTPUT {
      REAL y;
    }
  }
  IMPLEMENTATION {
    AUX {
      REAL aux_acc;
    }
    DA {
     aux_acc = {IF turbo THEN 2*acc ELSE acc};
    }
    CONTINUOUS {
      position = position + velocity + aux_acc;
      velocity = velocity + 0.5*aux_acc;
      turbocount = turbocount - (REAL turbo);
    }
    OUTPUT {
      y = position;
    }
  }
}
```

To get MLD model, the source file ***turbo_car.hys*** *is compiled first*

```
>> hysdel3('turbo_car')
```

and evaluated

```
>> turbo_car
```

Secondly, PWA form can be obtained by

```
>> sysStruct = h3_mld2pwa(S)
```

which can be used for control design in MPT Toolbox. For instance, to obtain an explicit controller which drives the car to a position 30, can be achieved by defining a problem structure ***probStruct*** *in MATLAB (see MPT manual for help)*

```
>> probStruct.N = 3;
>> probStruct.Q = [1 0.5 0];
>> probStruct.R = [1 0];
>> probStruct.norm = 1;
>> probStruct.subopt_lev = 0;
>> probStruct.xref = [30; 0; 0];
```

and invoking the main routine

```
>> ctrl = mpt_control(sysStruct, probStruct);
```

Resulting controller is stored in a MPT-structure and to simulate a closed loop we need to use other functions of MPT Toolbox, such as

```
>> mpt_plotTimeTrajectory(ctrl,[0;0;0],20)
```

which returns a plot shown in Fig. 6.8.

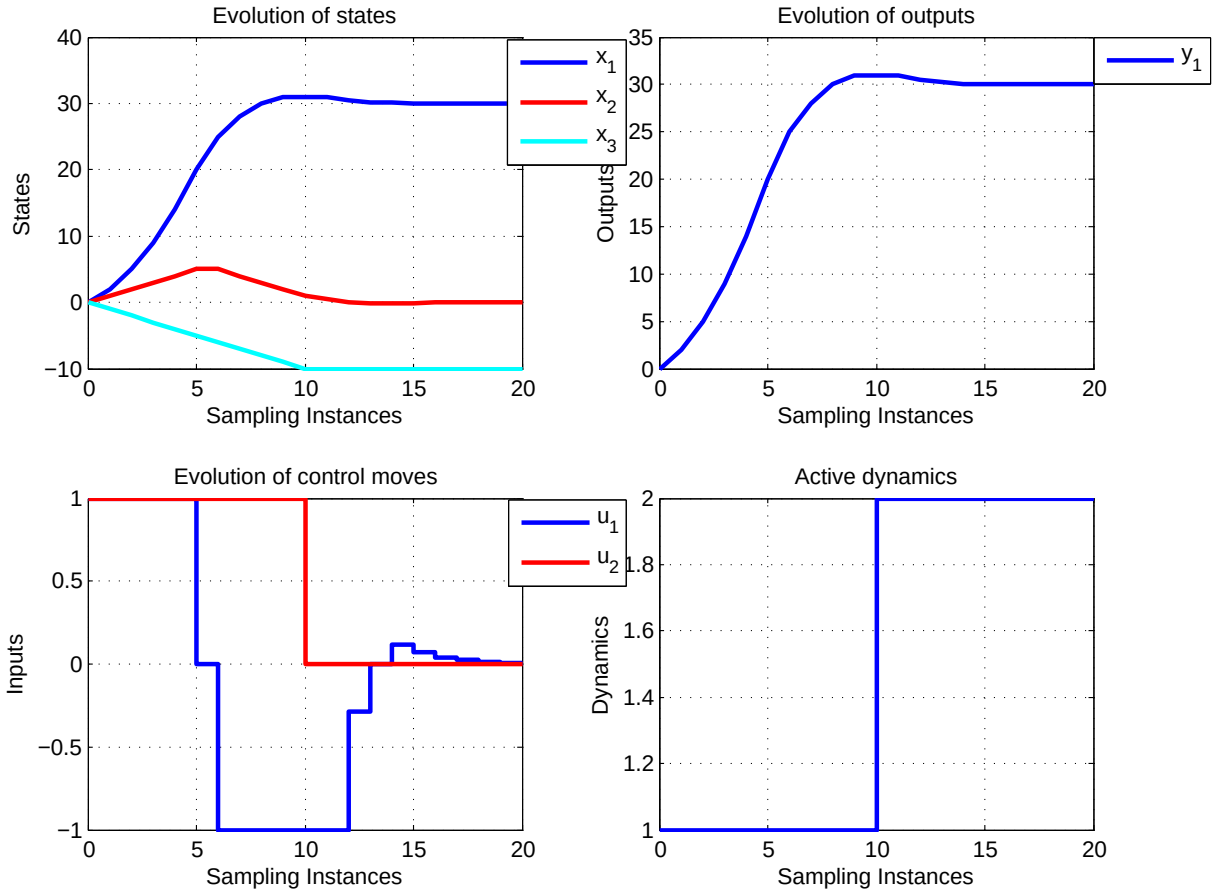


Figure 6.8: A closed loop simulation for a `turbo_car` model as returned by MPT Toolbox.

Conclusions

The chapter presents HYSDEL 3.0, a tool for modeling and composition of hybrid systems. The software allows automatic generation of high quality models from simple language statements defined between real and binary variables. The tool offers high flexibility with

low level modeling language, especially due to extended syntax and possibility to define symbolical variables. Furthermore, HYSDEL 3.0 incorporates a graphical level of modeling which is suitable for modeling and composition of large scale models. The software is integrated for the use with MPT Toolbox, thus the generated MLD models can be transformed to PWA models and readily deployed into MPC framework. Having addressed the first aim of the thesis, the focus in the next chapter is given to tackle time optimal MPC problems for PWA systems and to present low complexity MPC scheme.

Chapter 7

Explicit MPC for PWA Systems

7.1 Time Optimal Tracking of a Varying Reference

The technique for computing explicit solution to time optimal control problem for PWA systems has been published in [46]. However, the proposed methodology deals with a regulation problem towards origin and does not guarantee that the plant will follow the desired set-point. Problem of driving the states toward given reference is a standard requirement in industry and this field is traditional application for PID controllers. However, if the plant operates under hard constraints, such as saturation of an actuator, the task of maintaining constraints may be overwhelming for PID controllers and may cause severe problems. Not surprisingly that [19] reports 30% of all control loops in Canadian paper mills were oscillating because of valve problems. In order to tackle this problem, plant dynamics with operating constraints must be considered for controller design. This field is especially suited for hybrid modeling framework because the plant dynamics are naturally captured by MLD model (4.14) or PWA model (4.12). Due to equivalence of hybrid models, PWA model is preferred because formulation of MPC problem (5.23) is more suitable for multiparametric programming as with MLD model. In order to guarantee real-time implementability of the approach, the time optimal MPC problem is solved explicitly for all possible values of plant states. This section thus presents a novel way of synthesizing an explicit controller which achieves time optimal tracking of a varying reference for PWA systems.

7.1.1 Problem Formulation

The task of designing a control law which drives system states to a desired target set in the minimum number of steps has been reviewed in Section 5.3.2. The approach uses an algorithm which first designs a control invariant set around the origin and subsequently iteratively constructs the control law which steers the system to such set in a minimal number of steps. A drawback of such approach stems from the fact that it can only be used for regulation problems, i.e. for tasks of driving the system states to the origin. However, from practical reasons it is often required for states to converge to non-zero time-

varying reference signals. Such requirement differs from a standard setup in control theory and one is therefore motivated to extend the existing results of [46] to cover this case. The corresponding MPC problem can be formulated as follows

$$\min_{\boldsymbol{U}} J(\boldsymbol{U}) = N \tag{7.1a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{7.1b}$$

$$\boldsymbol{x}_k \in \mathcal{X} \tag{7.1c}$$

$$\boldsymbol{u}_k \in \mathcal{U} \tag{7.1d}$$

$$\boldsymbol{x}_N = \boldsymbol{x}_{\text{ref}}(t) \tag{7.1e}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{7.1f}$$

which requires that the predicted states must equal the given time-varying reference signal $\boldsymbol{x}_{\text{ref}}$ (7.1e) at the end of prediction horizon N. Objective function (7.1a) expresses the number of steps to reach the reference $\boldsymbol{x}_{\text{ref}}$ to be minimized. Sets $\mathcal{X}, \mathcal{U}$ denote the state/input constraints, usually given as polytopes with lower/upper bounds (cf. (5.1), (5.2)). PWA model (7.1b) corresponds to the hybrid model of the form (4.12) with regions defined in the joint $\boldsymbol{x}$–$\boldsymbol{u}$ space (4.13). In order to establish a feasible solution to (7.1), PWA model must be well-defined as given by Assumption 4.1. It is further assumed that the constraint sets $\mathcal{X}$, $\mathcal{U}$ are linear in $\boldsymbol{x}$ and $\boldsymbol{u}$ in the problem (7.1), thus it is possible to employ the multiparametric techniques presented in Section 5.2.3 to obtain an explicit solution in the form of (5.25). Once such a solution is computed, it can be implementable in real-time and with very fast sampling rates as shown in Section 5.4.
In the next section the modification of the algorithm by [46] is performed, which relies on design of a stabilizing terminal set. Once the suitable terminal set is found, the rest of the Algorithm 5.2 remains unchanged.

7.1.2 Design of the Stabilizing Terminal Set

The aim of this part is to construct the stabilizing terminal set Ω, which drives the states to desired reference in a dead-beat fashion. The terminal set has to be designed such that once states $\boldsymbol{x} \in \mathbb{R}^n$ are contained in such set, there is a control law which drives them to the time varying reference $\boldsymbol{x}_{\text{ref}} \in \mathbb{R}^n$ in a finite number of steps. One possibility of including the time varying reference for the use in multiparametric programming is to augment the state vector as follows

$$\tilde{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{v}_{\text{ref}} \end{pmatrix} \tag{7.2}$$

where $\boldsymbol{v}_{\text{ref}} \in \mathbb{R}^r$ are time dependent elements of the vector $\boldsymbol{x}_{\text{ref}}$ with $r \leq n$. Here can be mentioned that not all elements of $\boldsymbol{x}_{\text{ref}}$ must be included as only time varying elements are of interest. Augmentation (7.2) causes reformulation of PWA model (4.12) to

$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}_{\text{PWA}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) \tag{7.3a}$$

$$= \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k + \tilde{\boldsymbol{B}}_i \boldsymbol{u}_k + \tilde{\boldsymbol{c}}_i \quad \text{if} \quad \begin{pmatrix} \tilde{\boldsymbol{x}}_k \\ \boldsymbol{u}_k \end{pmatrix} \in \tilde{\mathcal{D}}_i \tag{7.3b}$$

where

$$\tilde{\boldsymbol{A}}_i = \begin{pmatrix} \boldsymbol{A}_i & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}, \qquad \tilde{\boldsymbol{B}}_i = \begin{pmatrix} \boldsymbol{B}_i \\ \boldsymbol{0} \end{pmatrix}, \qquad \tilde{\boldsymbol{c}}_i = \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{0} \end{pmatrix}, \tag{7.4}$$

are the augmented state update matrices with $i = 1, \ldots, n_{\tilde{\mathcal{D}}}$. Additionally, one has to consider also extended state constraint set

$$\tilde{\mathcal{X}} := \{\tilde{\boldsymbol{x}} \in \mathbb{R}^{n+r} \mid \underline{\tilde{\boldsymbol{x}}} \leq \boldsymbol{x} \leq \overline{\tilde{\boldsymbol{x}}}\}. \tag{7.5}$$

The suitable terminal set Ω should contain a terminal control law $\boldsymbol{\kappa}(\tilde{\boldsymbol{x}}) \in \mathcal{U}$ which fulfills the condition (7.1e) for all possible states within $\tilde{\mathcal{X}}$. To obtain such a set, it is suggested to solve the following feasibility problem:

$$\begin{aligned} \text{find} \quad & \boldsymbol{U} && \text{(7.6a)} \\ \text{s.t.} \quad & \boldsymbol{x}_{k+N} = \boldsymbol{x}_{\text{ref}} && \text{(7.6b)} \\ & \tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}_{\text{PWA}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) && \text{(7.6c)} \\ & \tilde{\boldsymbol{x}}_k \in \tilde{\mathcal{X}} && \text{(7.6d)} \\ & \boldsymbol{u}_k \in \mathcal{U} && \text{(7.6e)} \end{aligned}$$

where (7.6b) points to a terminal equality constraint (7.1e). By solving the problem (7.6) parametrically, according to Theorem 5.1, one obtains the feasible set Ω of parameters $\tilde{\boldsymbol{x}}_0$ which satisfy given constraints. The set takes a form of a (possibly non-convex) union of finite number of convex polytopes, i.e. $\Omega = \bigcup_i \mathcal{P}_i$ over which a PWA feedback function $\boldsymbol{u} = \boldsymbol{F}_i\tilde{\boldsymbol{x}}_0 + \boldsymbol{g}_i$ if $\tilde{\boldsymbol{x}}_0 \in \mathcal{P}_i$ with, $i = 1, \ldots, n_{\mathcal{P}}$ is defined as in (5.25). The equality constraint (7.6b) assures that the control law will be in a form such that once the state resides in the set Ω, it will be steered to the respective reference in, at most, N steps. It is therefore desired to choose the minimal value of N in order to attain dead-beat properties. Selection of N depends on the dimension n for the original PWA system (7.1b). For $N = 1$ the set Ω will automatically guarantee that in one time step ahead the reference $\boldsymbol{x}_{\text{ref}}$ will be reached. In a general case where $n > 1$, the dead-beat controller requires at least n steps to reach the reference. As $N > 1$ it is required that the target set Ω must satisfy the invariance property, as given by Definition 4.1.
To achieve the invariance property, it is suggested to calculate an invariant subset $\Omega_{\text{inv}} \subseteq \Omega$ using the algorithm of [88] presented in section 4.6. Basically, this approach uses reachability tools to iteratively remove subsets of states from the set Ω which may drive the states away from this region and the result is the control invariant set Ω_{inv}.

7.1.3 The Time Optimal Algorithm for Reference Tracking

Once the control invariant set Ω_{inv} is available from (7.6), it can be used to design a time optimal controller according to Algorithm 5.2. In fact, the skeleton of Algorithm 5.2 remains unchanged, only the target set is replaced with Ω_{inv}. To recapitulate the principle

of Algorithm 5.2, the problem (7.1) is partitioned into N subproblems with horizon one which are solved sequentially. At each stage of the Algorithm, the control law is constructed by solving the following problem parametrically

$$\min_{\boldsymbol{u}_k} \quad \|\boldsymbol{R}\boldsymbol{u}_k\|_p + \|\boldsymbol{Q}\tilde{\boldsymbol{x}}_k\|_p + \|\boldsymbol{P}\tilde{\boldsymbol{x}}_{k+1}\|_p \tag{7.7a}$$

$$\text{s.t.} \quad \tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}_{\text{PWA}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) \tag{7.7b}$$

$$\tilde{\boldsymbol{x}}_{k+1} \in \mathcal{S}_k \tag{7.7c}$$

$$\tilde{\boldsymbol{x}}_k \in \tilde{\mathcal{X}} \tag{7.7d}$$

$$\boldsymbol{u}_k \in \mathcal{U} \tag{7.7e}$$

where (7.7c) represents terminal set constraint (4.20f) for the augmented system (7.7b). The modified algorithm of [46] can now be reformulated in order to steer the states toward reference as follows:

Algorithm 7.1 (Time Optimal Algorithm for Reference Tracking)

1. *For given $\boldsymbol{P}$, $\boldsymbol{R}$, $\boldsymbol{Q}$ solve the problem (7.7) parametrically and denote by Ω the set of states for which the problem is feasible.*

2. *Apply the procedure of [88] to find a subset $\Omega_{inv} \subseteq \Omega$ which is invariant according to Definition 4.1.*

3. *Set the iteration counter $k = 0$ and set $\mathcal{S}_k = \Omega_{inv}$.*

4. *Solve (7.7) parametrically by considering $\tilde{\boldsymbol{x}}$ as the parameter. Denote the feasible set of the problem (7.7) by $\mathcal{S}_{k+1}$.*

5. *If $\mathcal{S}_{k+1} = \mathcal{S}_k$, stop, the algorithm has converged. Otherwise, increase k by 1 and jump back to Step 4.*

6. *The total number of iterations is given by $k^* = k$.*

At every run of Step 4 of the Algorithm 7.1, a control law of the form $\boldsymbol{u}_k = \boldsymbol{F}_{k,i}\tilde{\boldsymbol{x}}_k + \boldsymbol{g}_{k,i}$ is generated according to Theorem 5.1. Since the dynamics in (7.7b) is hybrid, the feasible set at the iteration k, $\mathcal{S}_k = \bigcup_i \mathcal{P}_{k,i}$, will be a non-convex union of finite number of convex polytopes. The set is then used at the next iteration as a new target set constraint. The algorithm terminates if at two subsequent iterations no new states have been added to $\mathcal{S}_k$ and total number of iterations k^* denotes the number of steps needed for states $\tilde{\boldsymbol{x}}$ to enter the terminal set Ω_{inv}.

Theorem 7.1 *The feedback laws $\boldsymbol{u}(\tilde{\boldsymbol{x}}_k)$, $k = \{0, \ldots, k^*\}$ calculated by Algorithm 7.1 are such that PWA system (7.1b) is pushed towards reference $\boldsymbol{x}_{ref}$ in, at most, $k^* + N$ steps for any $\tilde{\boldsymbol{x}} \in \bigcup_k \mathcal{S}_k$.*

PROOF: At iteration k for any $\tilde{\boldsymbol{x}}_0 \in \mathcal{S}_k$ the feedback law obtained in Step 1 of Algorithm 7.1 is such that the one-step prediction $\tilde{\boldsymbol{x}}_1 = \boldsymbol{f}_{\text{PWA}}(\tilde{\boldsymbol{x}}_0, \boldsymbol{u}_k(\tilde{\boldsymbol{x}}_0))$ is pushed into $\mathcal{S}_{k-1}$ in one time step. The iterative nature of the algorithm guarantees that for any $\tilde{\boldsymbol{x}}_1 \in \mathcal{S}_{k-1}$ we have $\tilde{\boldsymbol{x}}_2 = \boldsymbol{f}_{\text{PWA}}(\tilde{\boldsymbol{x}}_1, \boldsymbol{u}_{k-1}(\tilde{\boldsymbol{x}}_1)) \in \mathcal{S}_{k-2}$. By consecutively applying the feedback laws $\boldsymbol{u}_{k-2}(\tilde{\boldsymbol{x}}_2), \boldsymbol{u}_{k-3}(\tilde{\boldsymbol{x}}_3), \ldots, \tilde{\boldsymbol{u}}_0(\tilde{\boldsymbol{x}}_{k^*})$ one therefore get $\tilde{\boldsymbol{x}}_{k^*} \in \mathcal{S}_0$ (note that $\mathcal{S}_0 = \Omega_{\text{inv}}$). Hence all states of PWA system (7.1b) are pushed towards $\mathcal{S}_0$ in, at most, k^* due to the stopping criterion in Step 5. Inside $\mathcal{S}_0$ there are N steps required to reach the reference $\boldsymbol{x}_{\text{ref}}$, as given by feasible solution of (7.6). Hence, the maximum number of steps to reach the reference is $k^* + N$. □

Iterative property of Algorithm 7.1 causes that the generated sets $\mathcal{S}_k$ are overlapping, that is, there exist multiple control laws $\boldsymbol{u}_k(\tilde{\boldsymbol{x}}_0)$ with different indices k. In order to guarantee the time optimal property, one has to select the lowest index from an available set. This is ensured by the following implementation algorithm:

Algorithm 7.2 (On-line implementation)

1. *Find the minimal value of the index k_{min} for which $\tilde{\boldsymbol{x}}_0 \in \mathcal{S}_{k_{min}}$.*
2. *From the k_{min}-th feedback law of the form (5.25) find the region index r for which $\tilde{\boldsymbol{x}}_0 \in \mathcal{P}_r$.*
3. *Calculate the control action $\boldsymbol{u}^* = \boldsymbol{F}_{k_{min},r}\tilde{\boldsymbol{x}}_0 + \boldsymbol{g}_{k_{min},r}$.*

Theorem 7.2 *The time optimal controller calculated by Algorithm 7.1 and applied to PWA system (7.1b) in a receding horizon fashion according to Algorithm 7.2 guarantees that all states are pushed towards reference $\boldsymbol{x}_{ref}$ in the minimal possible number of steps.*

PROOF: Assume that the initial state $\tilde{\boldsymbol{x}}_0$ is contained in the set $\mathcal{S}_k$ from which it takes at most $k + N$ steps to reach the reference $\boldsymbol{x}_{\text{ref}}$ according to Theorem 7.1. The control law identified by Algorithm 7.1 will drive the states into the set $\mathcal{S}_{k-1}$ in one time step. Therefore, the states will enter Ω_{inv} in k steps when Algorithm 7.2 is called repeatedly at each sampling instance. Inside Ω_{inv} there exist controller which requires N steps to reach the reference $\boldsymbol{x}_{\text{ref}}$, as given by feasible solution of (7.6). □

7.1.4 The Robust Time Optimal Algorithm for Reference Tracking

The Algorithm 7.1 together with on-line implementation of Algorithm 7.2 guarantees that the reference will be reached with minimal number of steps. However, this result is valid only with the assumption that PWA model (7.1b) exactly matches with the real plant which is not always true in practice. Therefore, in order to include possible uncertainties in PWA model, [88] proposed a robust modification of the algorithm by [46]. The plant

to be controlled (7.7b) is affected by additive disturbances of the form (4.9). For the case here, the formulation of the robust time optimal control is as follows

$$\min_{\boldsymbol{U}} J(\boldsymbol{U}) = N \tag{7.8a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \tag{7.8b}$$

$$\boldsymbol{x}_k \in \mathcal{X} \tag{7.8c}$$

$$\boldsymbol{u}_k \in \mathcal{U} \tag{7.8d}$$

$$\boldsymbol{w}_k \in \mathcal{W} \tag{7.8e}$$

$$\boldsymbol{x}_N = \boldsymbol{x}_{\text{ref}}(t) \tag{7.8f}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{7.8g}$$

where $\boldsymbol{w} \in \mathcal{W}$ is additive noise bounded by the polytopic set $\mathcal{W}$, i.e.

$$\mathcal{W} := \{\boldsymbol{w} \in \mathbb{R}^n \mid \underline{\boldsymbol{w}} \leq \boldsymbol{w} \leq \overline{\boldsymbol{w}}\}. \tag{7.9}$$

Instances of $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, $\boldsymbol{w}_k$ are supposed to belong to their respective sets $\mathcal{X}$, $\mathcal{U}$, $\mathcal{W}$ for all prediction time $k = 0, \ldots, N$. As the aim is to obtain reference tracking, the model (7.8b) is augmented as in (7.3), i.e.

$$\tilde{\boldsymbol{x}}_{k+1} = \tilde{\boldsymbol{f}}_{\text{PWA}}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) + \tilde{\boldsymbol{w}}_k \tag{7.10a}$$

$$= \tilde{\boldsymbol{A}}_i \tilde{\boldsymbol{x}}_k + \tilde{\boldsymbol{B}}_i \boldsymbol{u}_k + \tilde{\boldsymbol{c}}_i + \tilde{\boldsymbol{w}}_k \quad \text{if} \quad \begin{pmatrix} \tilde{\boldsymbol{x}}_k \\ \boldsymbol{u}_k \end{pmatrix} \in \tilde{\mathcal{D}}_i \tag{7.10b}$$

where the set $\tilde{\mathcal{W}}$ is an augmented form of (7.9). Though the time varying reference might not be affected with additive noise but due to computational reasons it is assumed so. Otherwise, the Pontryagin difference could not be obtained which is required by the approach [88].
Overall principle of the robust time optimal algorithm is the same as Algorithm 5.2 but the invariant set Ω_{inv} has be computed for all possible realizations of $\tilde{\boldsymbol{w}}_k$ in the sense of the following definition.

Definition 7.1 (Robust Control Invariant Set, [88]) *A set $\Psi \in \tilde{\mathcal{X}}$ is said to be a robustly controlled invariant set for PWA system (7.10) subject to constraints $\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}$, $\boldsymbol{u} \in \mathcal{U}$, $\tilde{\boldsymbol{w}} \in \tilde{\mathcal{W}}$ if for every $\tilde{\boldsymbol{x}} \in \Psi$ there exist $\boldsymbol{u} \in \mathcal{U}$ such that $\tilde{\boldsymbol{f}}_{PWA}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) + \tilde{\boldsymbol{w}} \in \Psi$, $\forall \tilde{\boldsymbol{w}} \in \tilde{\mathcal{W}}$.*

The robust control invariant set can be computed with the help of geometric operations over polytopes. To get an insight, one has to interpret the evolution of (7.10) under constraints (7.8c), (7.8d), (7.8e) as affine transformation of polytopes (Definition 1.27). Consider a reachable set $\mathcal{R}$ as a set of states for which evolution of (7.10) in one time step can be obtained under set constraints $\tilde{\mathcal{X}}$ and closed loop control $\mathcal{U}$ and disturbance $\tilde{\mathcal{W}}$. In general, the reachable set $\mathcal{R}$ can be expressed as a nonconvex union of polytopes which

are affected by disturbances $\tilde{\boldsymbol{w}}$ from the set $\tilde{\mathcal{W}}$. As the disturbance enters PWA function (7.10) linearly, by subtracting the set $\tilde{\mathcal{W}}$ from $\mathcal{R}$ using the Pontryagin difference operation in Definition 1.23 an robustly invariant subset can be obtained. This idea is elaborated in the robust time optimal algorithm by [88] and the extension for reference tracking can be summarized as follows:

Algorithm 7.3 (Robust Time Optimal Algorithm for Reference Tracking)

1. *For given $\boldsymbol{P}$, $\boldsymbol{R}$, $\boldsymbol{Q}$ solve the problem (7.7) parametrically and denote by Ω the set of states for which the problem is feasible.*
2. *Apply the procedure of [88] to find a subset $\Omega_{inv} \subseteq \Omega$ which is invariant according to Definition 4.1. Compute the robust invariant subset $\Psi = \Omega_{inv} \ominus \tilde{\mathcal{W}}$ using the Pontryagin difference operation according to Definition 1.23.*
3. *Set the iteration counter $k = 0$ and set $\mathcal{S}_k = \Psi$.*
4. *Solve (7.7) parametrically by considering $\tilde{\boldsymbol{x}}$ as the parameter. Denote the feasible set of the problem (7.7) by $\mathcal{S}_{k+1}$.*
5. *If $\mathcal{S}_{k+1} = \mathcal{S}_k$, stop, the algorithm has converged. Otherwise, increase k, $\mathcal{S}_k = \mathcal{S}_k \ominus \tilde{\mathcal{W}}$, and jump back to Step 4.*
6. *The total number of iterations is given by $k^* = k$.*

The Algorithm 7.3 generates at each run of the step 4 a nonconvex union of sets $\mathcal{S}_k$ which is shrunken by the disturbance set $\tilde{\mathcal{W}}$ using Pontryagin difference operation. To each of the set $\mathcal{S}_k$ a control law $\boldsymbol{u}_k = \boldsymbol{F}_{k,i}\tilde{\boldsymbol{x}}_k + \boldsymbol{g}_{k,i}$ is computed according to Theorem 5.1. If the Algorithm 7.3 is implemented on-line according to Algorithm 7.2, then the time optimal controller robustly drives the states toward reference in the time optimal fashion. Argumentation follows the proof concept for Theorem 7.1 with that difference that algorithm for computing maximal robustly controlled invariant set of [88] is used for convergence to Ψ which is already computed from (7.6).

7.1.5 Examples

One Dimensional Case

For illustration of Algorithm 7.1 for reference tracking, consider the following 1D PWA system:

$$x_{k+1} = \begin{cases} -0.5x_k + 0.5u_k & \text{if} \quad x_k \leq 0, \\ 0.9x_k + 0.5u_k & \text{if} \quad x_k \geq 0. \end{cases} \tag{7.11}$$

The control objective is to follow a time-varying reference x_{ref} in the least number of steps while $-5 \leq u_k \leq 5$ and $-10 \leq x_k \leq 10$ for all k must be satisfied. PWA model (7.11) has been augmented with varying reference and the time optimal algorithm has been applied.

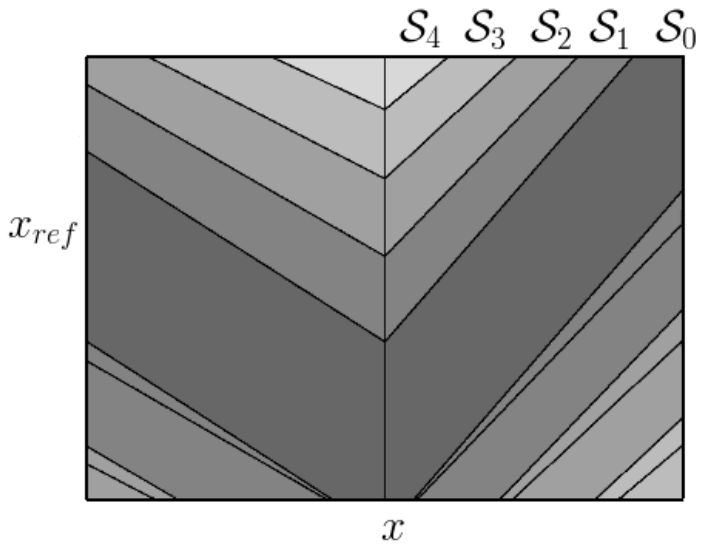


Figure 7.1: Solution of the time optimal algorithm for 1D example (7.11).

Illustration of the algorithm is depicted in Fig. 7.1 for 5 iterations. Regions associated to each iteration are equally shadowed, the darkest region $\mathcal{S}_0$ corresponds to the terminal set. Since one-step problems are solved at each step, the controller guarantees that all states will enter the initial terminal set Ω in the least possible number of steps. Subsequently, once the states arrive to the set Ω, the control laws which have been obtained as a solution to (7.6) will drive them to the respective reference in a dead-beat fashion.
Figures 7.2 and 7.3 show the state and input trajectories of the closed loop control from initial condition $x_0 = 10$. The reference signal x_{ref} varies in discrete time instances k as follows:

$$x_{\text{ref}} = \begin{cases} 5\sin(0.4k - \pi/2) + 5 & \text{for} \quad k \leq 30 \\ 6 & \text{for} \quad k > 30. \end{cases}$$

From the simulation on discrete PWA model (7.11) in Fig. 7.2 is evident that evolution of the state x follows the desired reference in time optimal manner and the imposed state constraints are fulfilled for all time. Similarly, the control profile in Fig. 7.3 satisfy input constraints and behaves in a dead-beat fashion.

Two Dimensional Case

For illustration of the robust time optimal Algorithm 7.3 for reference tracking, consider the example "car on the road" shown in Fig. 4.1. Assume that the car is perfectly described by the following equations of motion [99]

$$\dot{x}_1 = \frac{v}{l}\tan u, \tag{7.12a}$$

$$\dot{x}_2 = v \sin x_1 \tag{7.12b}$$

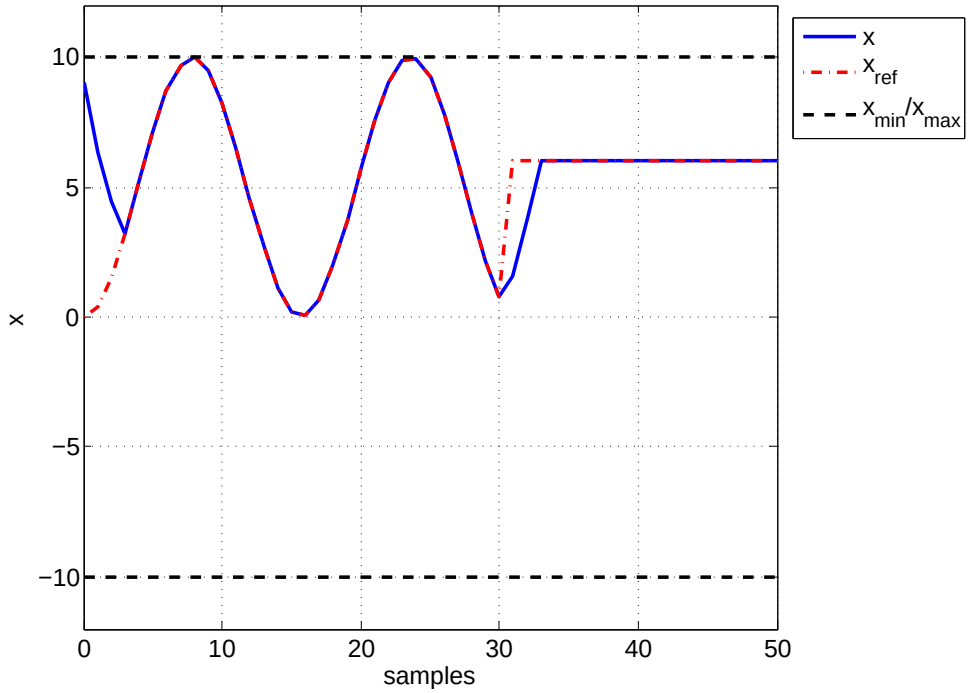


Figure 7.2: State profiles for the closed loop time optimal control.

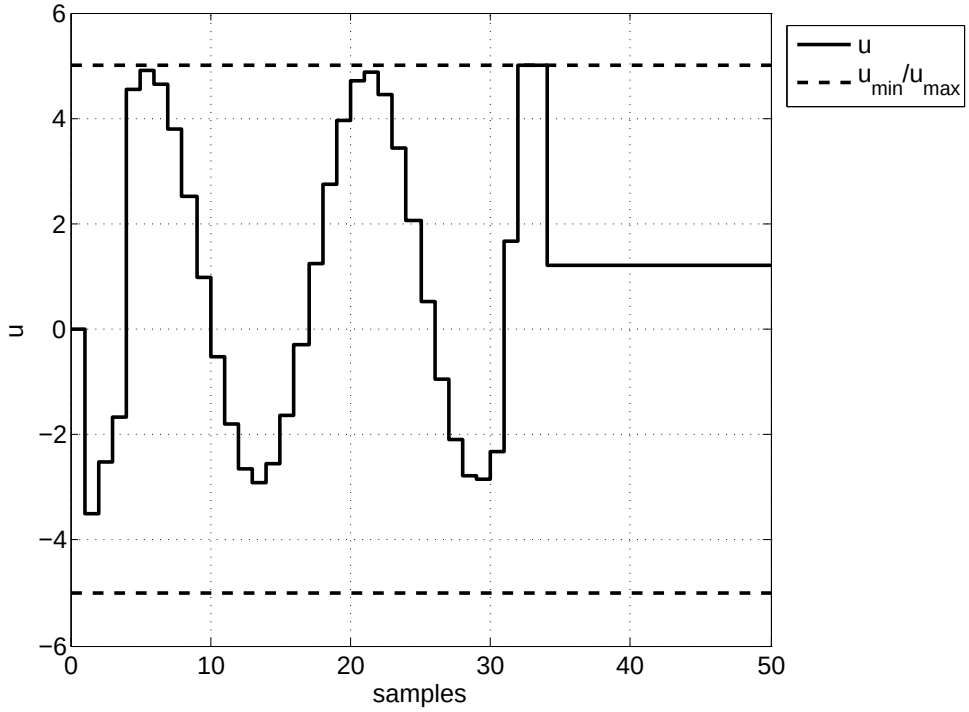


Figure 7.3: Input profile for the closed loop time optimal control.

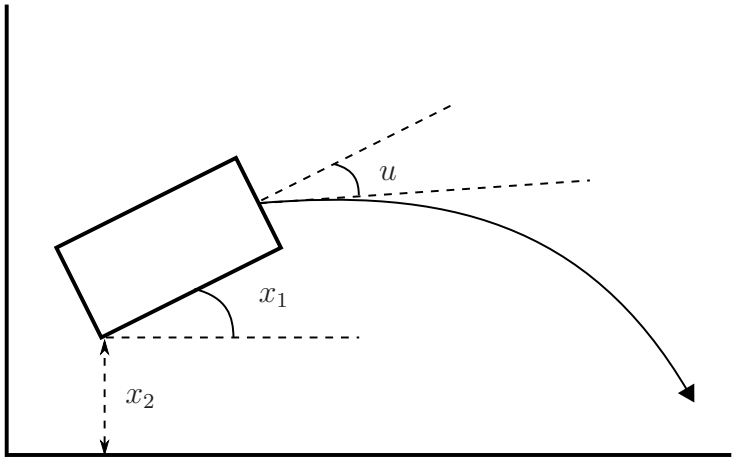


Figure 7.4: Coordinate system of a car example.

where x_1 is the angle of the car, x_2 is the vertical position of the rear end, u is the steering angle, v is constant speed, and l is the truck length. The coordinate system is shown in Fig. 7.4. Equations (7.12) are described in continuous time and are nonlinear due to trigonometric terms $\tan u$, $\sin x_1$.
In order to design explicit solution, it is required to construct an equivalent model in discrete-time. It is obtained by applying finite difference approximation to time derivations in (7.12), i.e.

$$x_{1,k+1} = x_{1,k} + \frac{vT_s}{l} \tan u_k, \tag{7.13a}$$

$$x_{2,k+1} = x_{2,k} + vT_s \sin x_{1,k} \tag{7.13b}$$

where $T_s = 1\,\mathrm{s}$ is the sampling time and k denotes the discrete time. Consequently, the linear model is derived from (7.13) by first order Taylor series approximation of nonlinear terms around origin. In particular $\tan u_k \approx u_k$ for $|u_k| \leq 0.5$, $\sin x_{1,k} \approx x_{1,k}$ if $x_{1,k}$ is close to 0, and the model reads

$$x_{1,k+1} = x_{1,k} + \frac{vT_s}{l} u_k, \tag{7.14a}$$

$$x_{2,k+1} = x_{2,k} + vT_s x_{1,k}. \tag{7.14b}$$

PWA model is obtained by approximating the $\sin x_1$ function with three lines $y_1(x_1)$, $y_2(x_1)$, $y_3(x_1)$, as shown in Fig. 7.5, and by collecting the linear models (7.14). The resulting model gives

$$x_{1,k+1} = x_{1,k} + \frac{vT_s}{l} u_k, \tag{7.15a}$$

$$x_{2,k+1} = \begin{cases} x_{2,k} - 0.6vT_s x_{1,k} - 1.94vT_s & \text{if} \quad -\pi \leq x_{1,k} \leq -\pi/2 \\ x_{2,k} + 0.64vT_s x_{1,k} & \text{if} \quad -\pi/2 \leq x_{1,k} \leq \pi/2 \\ x_{2,k} - 0.6vT_s x_{1,k} + 1.94vT_s & \text{if} \quad \pi/2 \leq x_{1,k} \leq \pi \end{cases} \tag{7.15b}$$

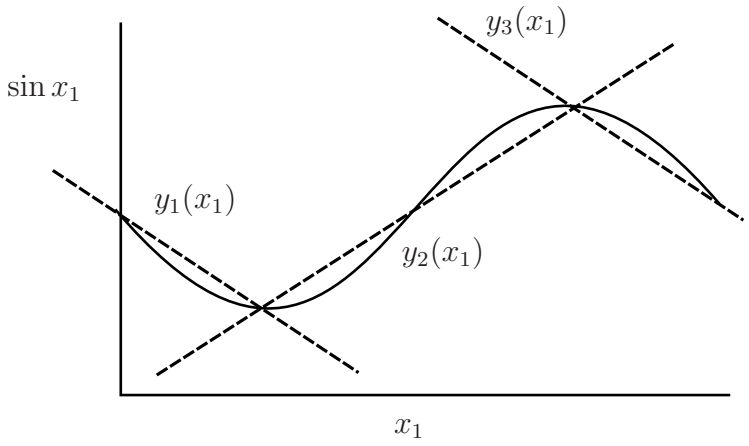


Figure 7.5: Linear approximations of $\sin x_1$ function.

where the operator "if" represents the switching function. Depending on the location of state $x_{1,k}$ the corresponding local approximation is chosen and PWA model thus acts as selection from table (7.15) of local models.
To verify the accuracy of PWA model, the following scenario is considered. The car starts from a zero initial position and the driver performs a full right turn of a driving wheel, then immediately turns left and consequently returns to zero position. The corresponding profile of the steering angle is shown in Fig. 7.6. Applying this maneuver to different car models gives prediction of state trajectories depicted in Figs. 7.7(a), 7.7(b). A comparison of trajectories for x_1 in Fig. 7.7(a) does not say much, except that PWA model is accurate only at the very beginning of prediction. As the prediction horizon increases, PWA models suffer from finite difference approximation and do not copy the behavior of continuous time models precisely. This observation is more visible in Fig. 7.7(b) where bigger differences between trajectories are present. This fact is due to rough approximation, as shown in Fig. 7.5. PWA models could be constructed with higher precision if $\sin x_1$ nonlinearity would be approximated with more lines which corresponds to more switching rules.
As shown in Figs. 7.7(a), 7.7(b) PWA model is not perfect and there is some uncertainty present. To account for this inaccuracy, additive uncertainty is assumed on both states, i.e.

$$\mathcal{W} := \{\boldsymbol{w} \in \mathbb{R}^2 \mid \|\boldsymbol{w}\|_\infty \leq 0.02\}.$$

To recall the physical meaning of states, the first state x_1 represents the speed of the car and the state x_2 is the position. Both states are constrained

$$\mathcal{X} := \{\boldsymbol{x} \in \mathbb{R}^2 \mid |x_1| \leq \pi,\ |x_2| \leq 1\}$$

and the objective is to drive the position of a car to a time-varying reference $x_{\text{ref}} = (x_{1,\text{ref}}, x_{2,\text{ref}})^T$, where $x_{1,\text{ref}} = 0$ and $x_{2,\text{ref}} = x_{2,\text{ref}}(t)$. The manipulated input (steering angle) must remain within the constraint set

$$\mathcal{U} := \{u \in \mathbb{R} \mid |u| \leq 0.5\}.$$

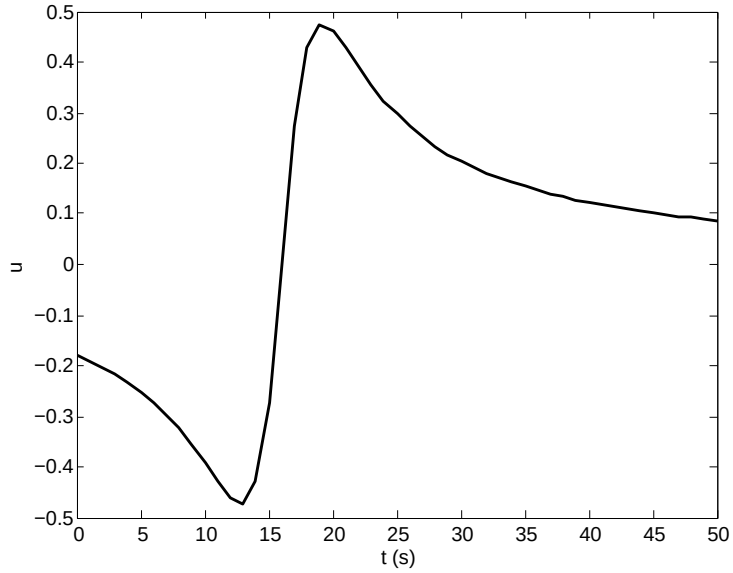
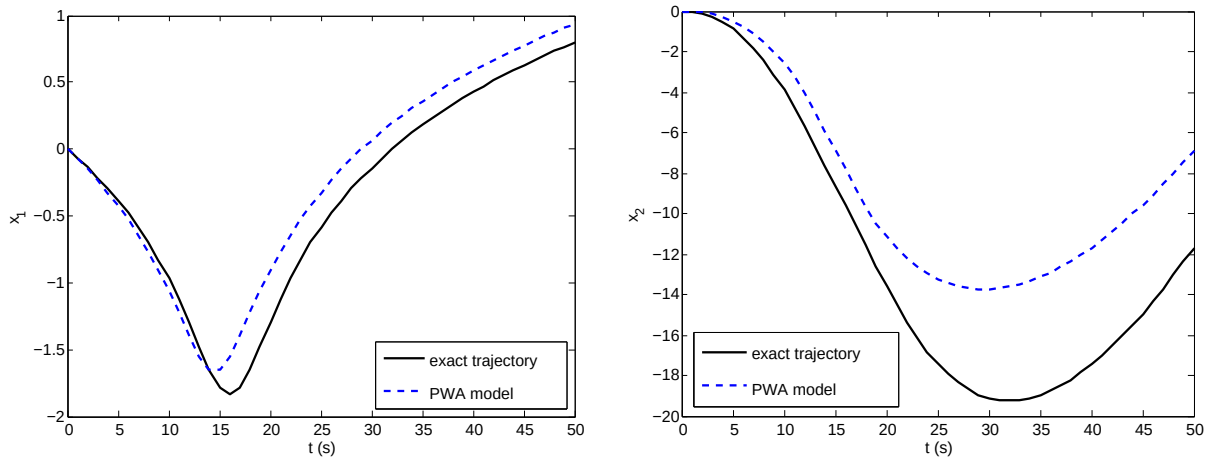


Figure 7.6: Maneuver for testing accuracy.



(a) (b)

Figure 7.7: State predictions of x_1 (a) and x_2 (b) obtained with PWA model of a car.

In the first step, PWA system is augmented with the reference $-1 \leq x_{2,\text{ref}} \leq 1$ to get (7.3). Consequently, the disturbance set $\mathcal{W}$ is enlarged to account for possible uncertainty in x_{ref}

$$\tilde{\mathcal{W}} := \{\tilde{\boldsymbol{w}} \in \mathbb{R}^3 \mid |\tilde{w}_1| \leq 0.02,\ |\tilde{w}_2| \leq 0.02,\ |\tilde{w}_3| \leq 0.001\ \}.$$

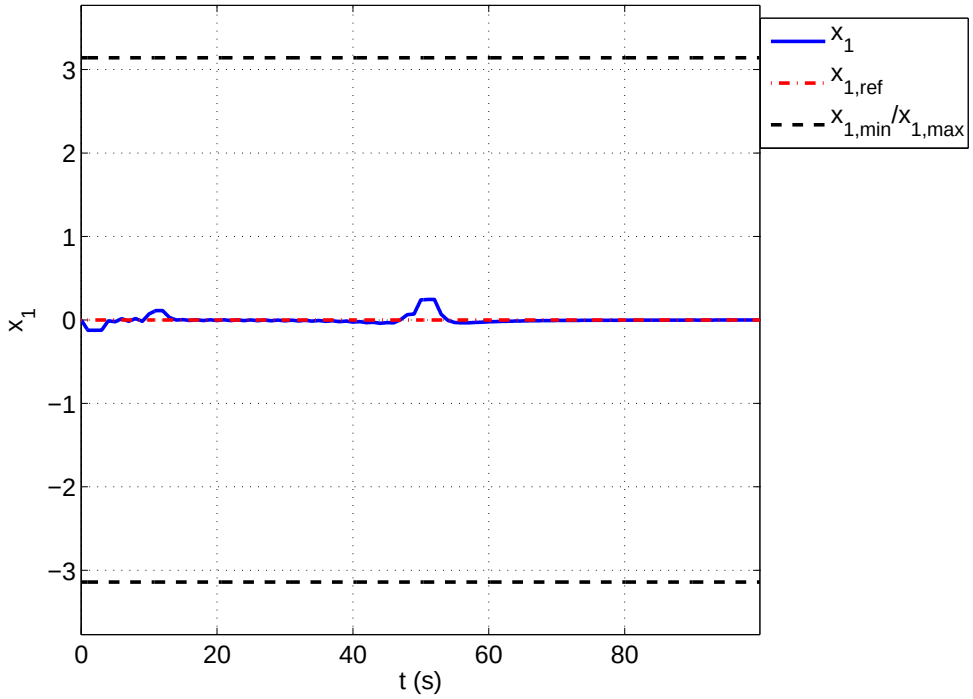
The augmented set $\tilde{\mathcal{W}}$ is introduced in order to compute the Pontryagin difference at Step 2 of the Algorithm 7.3 which requires the dimensions of Ω_{inv} and $\tilde{\mathcal{W}}$ to be equal. The stabilizing terminal set has been computed with $N = 2$ and the procedure of [88] for computing maximal invariant set has been applied. Executing the Algorithm 7.3 with weights $\boldsymbol{P} = \boldsymbol{0}$, $\boldsymbol{Q} = \boldsymbol{I}$, $R = 1$ one obtains the explicit solution in the form of (5.25). The control law has been applied in the closed loop simulation for the continuous-time model (7.15). State trajectories are shown in Figs. 7.8(a), 7.8(b) with the time varying reference x_{ref} given as the testing maneuver on Fig. 7.6. Car's position is steered to the reference, even if the simulation is performed on a continuous time model. Input profile is shown in Fig. 7.9 and it is evident that input constraints are satisfied.

Conclusions

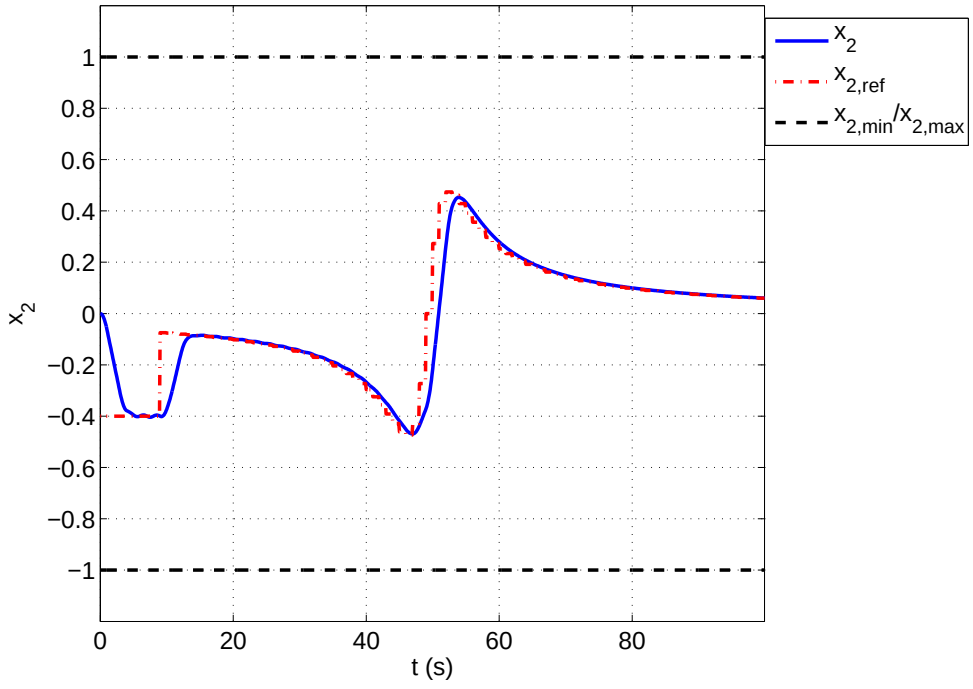
The section addresses the goal of finding the explicit solution to time optimal tracking problem for PWA systems. It has been shown that the proposed approach comprises of two steps. First, the control invariant set Ω_{inv} has to be designed by solving the feasibility problem (7.6). Secondly, the set Ω_{inv} is deployed as a terminal set constraint in the recursive Algorithm 7.1 that generates explicit solution to the time optimal tracking MPC problem. For a proper implementation of the explicit solution, Algorithm 7.2 is applied. In case that given PWA model is affected by additive disturbances, the robust time optimal Algorithm 7.3 for reference tracking is suggested. The approach has been demonstrated on two examples and the real-time application will be studied in Part III.

7.2 Adaptive Time Optimal Control

The objective of this section is to present an adaptive approach for time optimal control of PWA systems. Contrary to previous section, where the goal was to attain reference tracking in the minimum number of steps, the objective here is to drive the system states toward given terminal set. Thus, the overall task is considered as a regulation problem. The reasoning beyond the adaptive approach is to enhance the closed loop performance for the cases where the information about PWA model is uncertain or, it changes throughout the operation of closed loop control. Thus, the argumentation follows the classical concept of robust control where the controller is calculated such that it robustly stabilizes the whole family of PWA models.



(a)



(b)

Figure 7.8: State profiles of x_1 (a) and x_2 (b) obtained under time optimal control toward reference x_{ref}.

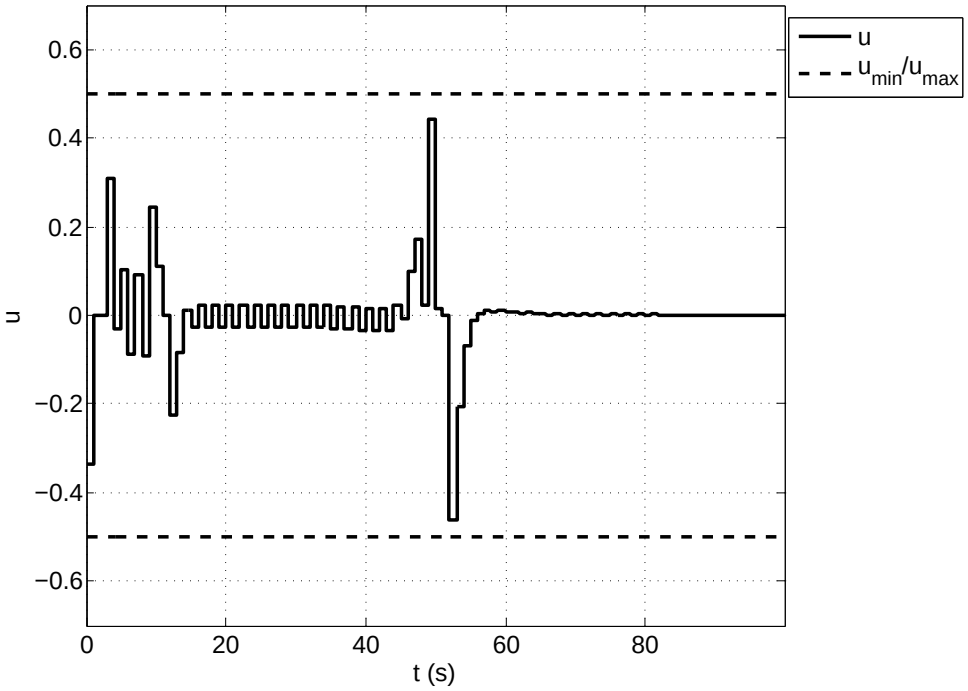


Figure 7.9: Input profile for the closed loop time optimal control of a car example.

7.2.1 Problem Formulation

The problems with uncertain models have been introduced in Section 4.4.2 where two cases are distinguished: models with additive uncertainties (4.9) and models with parametric uncertainties (4.10). Time optimal control of PWA systems with additive uncertainties has been resolved in [88] for convergence to a desired target set and a modified algorithm for reference tracking is presented in Section 7.1.4. The problem considered here concerns PWA systems affected with parametric uncertainties of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{\lambda}_k, \boldsymbol{x}_k, \boldsymbol{u}_k) \tag{7.16a}$$

$$= \boldsymbol{A}_i(\boldsymbol{\lambda}_k)\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \quad \text{if} \quad \boldsymbol{x}_k \in \mathcal{D}_i \tag{7.16b}$$

where $\boldsymbol{\lambda}_k \in \Lambda$ is convex and compact set and $\mathcal{D}_i$, $i = 1, \ldots, n_{\mathcal{D}}$, is a partition of the state space, $\mathcal{D} = \bigcup_i \mathcal{D}_i$. The relation $\boldsymbol{A}_i(\boldsymbol{\lambda}_k)$ is given by

$$\boldsymbol{A}_i(\boldsymbol{\lambda}_k) = \sum_{l=1}^{n_\lambda} \lambda_{k,j} \boldsymbol{A}_{i,l} \tag{7.17}$$

where $0 \leq \lambda_{k,l} \leq 1$ and vertices $\boldsymbol{A}_{i,l}$ can be obtained by evaluating $\boldsymbol{A}_i(\boldsymbol{\lambda}_k)$ for all vertices $\boldsymbol{\lambda}_{k,l} \in \text{vert}(\Lambda)$, i.e. $\boldsymbol{A}_{i,l} = \boldsymbol{A}_i(\boldsymbol{\lambda}_{k,l})$. Expression (7.17) is a general form of convex hull as given by Definition 1.4. The related problem of time optimal control for PWA systems

affected with parametric uncertainties can be formulated as follows

$$\min_{\boldsymbol{U}} J = N \tag{7.18a}$$
$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{\lambda}_k, \boldsymbol{x}_k, \boldsymbol{u}_k) \tag{7.18b}$$
$$\boldsymbol{x}_k \in \mathcal{X} \tag{7.18c}$$
$$\boldsymbol{u}_k \in \mathcal{U} \tag{7.18d}$$
$$\boldsymbol{\lambda}_k \in \Lambda \tag{7.18e}$$
$$\boldsymbol{x}_N = \Omega \tag{7.18f}$$
$$\boldsymbol{x}_0 = \boldsymbol{x}(t) \tag{7.18g}$$

where Ω is the desired terminal set defined as polytope. Sets $\mathcal{X}, \mathcal{U}$ represent the state and input constraints usually given as (5.1) and (5.2). Solution to the problem (7.18) follows the max-min concept of [6] and utilizes the notion of *information vector* $\boldsymbol{z}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$ which gives actual knowledge about the plant. The variables $\boldsymbol{x}_k$, $\boldsymbol{\lambda}_k$ are assumed to be known at the current time k, thus the adaptive feedback law must respect these in the form of $\boldsymbol{u}(\boldsymbol{z}_k)$. These considerations are elaborated by the adaptive time optimal algorithm presented in the next section.

7.2.2 Adaptive Time Optimal Algorithm

The adaptive time optimal algorithm is constructed in such a manner that it iteratively pushes the states towards desired target set in time optimal fashion. If there is additive disturbance present as in (7.10) which enters PWA dynamic equations linearly, the algorithm is modified to account for the uncertainty set by Pontryagin difference operation. However, as PWA system is affected with parametric uncertainty of the form (7.17), equation (7.16) contains a product $\boldsymbol{A}_i(\boldsymbol{\lambda}_k)\boldsymbol{x}_k$. To retain the linearity of PWA model, [6, 18] suggests to employ the information vector

$$\boldsymbol{z}_k = \boldsymbol{A}_i(\boldsymbol{\lambda}_k)\boldsymbol{x}_k \tag{7.19}$$

which aggregates the knowledge of local PWA mode i, uncertain parameter $\boldsymbol{\lambda}_k$ and state vector $\boldsymbol{x}_k$ in one. Such modification transforms the equation (7.16) to

$$\boldsymbol{x}_{k+1} = \boldsymbol{z}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \tag{7.20}$$

which is now linear in variables $\boldsymbol{z}_k$ and $\boldsymbol{u}_k$ for a given mode i. Transformation of (7.16) to (7.20) is especially suited for use in adaptive control since information about all instances of vector (7.19) will be known at the time of executing the feedback law. Reformulation of (7.16) has immediate advantage for the use in multiparametric programming, as (7.19) enters the equation (7.18b) linearly and can be treated as parameter. To account for all variations of the information variable (7.19), it is suitable to include this term in the objective function directly, or via relation (7.20). Now the task is to express the 1-step horizon subproblem which should be solved at each iteration of Algorithm 5.2. This can

be done by considering extremal variations of (7.17) into account as follows

$$\begin{aligned}
\min_{\boldsymbol{u}_k} \quad & \|\boldsymbol{Q}\boldsymbol{x}_{k+1}\|_p + \|\boldsymbol{R}\boldsymbol{u}_k\|_p && \text{(7.21a)}\\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{z}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i && \text{(7.21b)}\\
& \boldsymbol{x}_{k+1} \in \mathcal{S}_k && \text{(7.21c)}\\
& \boldsymbol{\lambda}_k \in \Lambda && \text{(7.21d)}\\
& \boldsymbol{x}_k \in \mathcal{X} && \text{(7.21e)}\\
& \boldsymbol{u}_k \in \mathcal{U} && \text{(7.21f)}
\end{aligned}$$

where $p = \{1, \infty\}$ and $i = 1, \ldots, n_{\mathcal{D}}$. Note that constraints (7.21c), (7.21d) are given as

$$\boldsymbol{A}_{i,1}\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \in \mathcal{S}_k \quad \text{(7.22a)}$$

$$\vdots$$

$$\boldsymbol{A}_{i,n_\lambda}\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \in \mathcal{S}_k \quad \text{(7.22b)}$$

which accounts all vertices of the convex hull (7.17) for fixed local dynamics i. If the terminal set $\mathcal{S}_k$ is given as one polytope, the problem (7.21) is given as mpLP for fixed i. Thus, for $n_{\mathcal{D}}$ dynamics of PWA model (7.16) one has to solve $n_{\mathcal{D}}$-times mpLP. Explicit solution to the problem (7.21) for fixed i is expressed according to Theorem 5.1 and is given by

$$\begin{aligned}
\boldsymbol{u}_k &= \boldsymbol{F}_{i,j}\boldsymbol{z}_k + \boldsymbol{g}_{i,j} \quad \text{if} \quad \boldsymbol{z}_k \in \mathcal{P}_{k,i,j}, && \text{(7.23a)}\\
J_k &= \boldsymbol{\Phi}_{i,j}\boldsymbol{z}_k + \Gamma_{i,j} \quad \text{if} \quad \boldsymbol{z}_k \in \mathcal{P}_{k,i,j}, && \text{(7.23b)}
\end{aligned}$$

where $j = 1, \ldots, n_{\mathcal{P}_i}$ and $\mathcal{P}_{k,i,j}$ is the polytope defined in the $\boldsymbol{z}$-space. Subscript notation should be interpreted as the one step prediction (7.20) for the current value of $\boldsymbol{z}$ at time k is driven by the feedback law $\boldsymbol{u}_k$ toward terminal set Ω for i-th dynamics of uncertain PWA model (7.16). In addition, the feedback law is given as PWA function defined over $j = 1, \ldots, n_{\mathcal{P}_i}$ regions. For the whole model, the optimal feedback law $\boldsymbol{u}^*(\boldsymbol{z}_k)$ is given as the one with minimum cost [23], i.e.

$$\boldsymbol{u}^*(\boldsymbol{z}_k) = \arg\min_{i,j} J(\boldsymbol{z}_k) \quad \text{(7.24)}$$

and is defined as PWA function in the form

$$\boldsymbol{u}^*(\boldsymbol{z}_k) = \boldsymbol{F}_r\boldsymbol{z}_k + \boldsymbol{g}_r \quad \text{if} \quad \boldsymbol{z}_k \in \mathcal{P}_{k,r} \quad \text{(7.25)}$$

where $r = 1, \ldots, n_{\mathcal{P}_k}$. Note that the explicit solution is defined over a collection of polytopes $\mathcal{P}_{k,r}$ defined in the $\boldsymbol{z}$-space. However, for the iterative use according to Algorithm 5.2 these sets must be given in $\boldsymbol{x}$-space as required by the terminal set constraint (7.21c). This can be achieved by geometric projection of the sets

$$\mathcal{S}_k = \bigcup_{r=1}^{n_{\mathcal{P}_k}} \bigcup_{i=1}^{n_{\mathcal{D}}} proj_{\boldsymbol{x}}(\mathcal{Z}_{k,r,i}) \quad \text{(7.26)}$$

onto the $\boldsymbol{x}$-space of according to Definition 1.25 where

$$\mathcal{Z}_{k,r,i} := \left\{ \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} \;\middle|\; \begin{array}{c} \boldsymbol{u}_k \in \mathcal{U}, \\ \boldsymbol{x}_k \in \mathcal{D}_i, \\ \boldsymbol{A}_{i,1}\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \in \mathcal{S}_k, \\ \vdots \\ \boldsymbol{A}_{i,n_\lambda}\boldsymbol{x}_k + \boldsymbol{B}_i\boldsymbol{u}_k + \boldsymbol{c}_i \in \mathcal{S}_k \end{array} \right\} \tag{7.27}$$

The set $\mathcal{S}_k$, which is given as a union

$$\mathcal{S}_k = \bigcup_{r=1}^{n_\mathcal{S}} \mathcal{S}_{k,r}, \tag{7.28}$$

is used as a target set for subsequent iteration according to general principle of Algorithm 5.2. However, (7.28) is frequently a non-convex set. Therefore only a convex subset can be given as terminal set constraint in (7.21). This can be explained by looking at Figures 7.10(a) and 7.10(b) where an example of a target set with tree polytopes $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ is depicted. Assume that the initial point $\boldsymbol{x}_0$ is contained in a mode i of the uncertain PWA model (7.16) with two extremal realizations $\boldsymbol{A}_{i,1}$ and $\boldsymbol{A}_{i,2}$. The point $\boldsymbol{x}_0$ can be driven using local dynamics 1 to the point $\boldsymbol{x}_1$ and using dynamics 2 to the point $\boldsymbol{x}_2$. Figure 7.10(a) shows the possibility when the target set is built by a convex subset of $\{\mathcal{S}_2, \mathcal{S}_3\}$ and $\boldsymbol{x}_1 \in \mathcal{S}_2$, $\boldsymbol{x}_2 \in \mathcal{S}_3$. Due to convexity of $\{\mathcal{S}_2, \mathcal{S}_3\}$ all possible combinations of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, represented by the connecting line, lie inside this set, thus a control law can be found which stabilizes all combinations of dynamics 1 and 2. However, Figure 7.10(b) shows the case when the target set is a nonconvex, i.e. $\{\mathcal{S}_1, \mathcal{S}_3\}$. Here, the connecting line between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ does not remain inside the set $\{\mathcal{S}_1, \mathcal{S}_3\}$, thus there cannot be found controller which can stabilize all combinations of dynamics 1 and 2.
Taking the aforementioned ideas into consideration, the adaptive time optimal algorithm is given as follows.

Algorithm 7.4 (Time Optimal Adaptive Algorithm)

1. *Given optimization weights $\boldsymbol{Q}$, $\boldsymbol{R}$, a stabilizing terminal set Ω, denote this set by $\mathcal{S}_0$ and set the iteration counter $k = 0$.*

2. *(a) For each local dynamics $i = 1, \ldots, n_\mathcal{D}$ of PWA system (7.16) and for each combination of indices $r = 1, \ldots, n_\mathcal{S}$ which built convex terminal set constraint $\mathcal{S}_k$ in (7.28) solve the optimization problem (7.21) parametrically.*

   (b) Store the optimizer $\boldsymbol{u}_k^* = \boldsymbol{F}_{k,r}\boldsymbol{z}_k + \boldsymbol{g}_{k,r}$ if $\boldsymbol{z}_k \in \mathcal{P}_{k,r}$ by taking the minimum per (7.24).

   (c) Compute next terminal constraint by (7.26) and denote their union as $\mathcal{S}_{k+1} = \bigcup_r \mathcal{S}_{k,r}$.

3. *If $\mathcal{S}_{k+1} = \mathcal{S}_k$, abort, the algorithm has converged.*

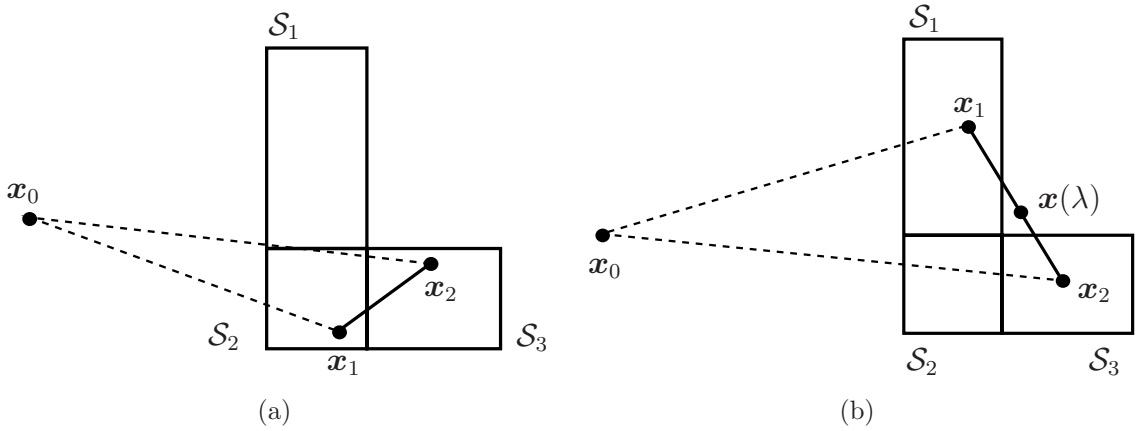


Figure 7.10: Convex combination of polytopes (a) and non-convex combination of polytopes (b) for specification of a target set.

4. *Otherwise increment the iteration counter* $k = k + 1$ *and jump back to Step 2.*

5. *The total number of iterations is given by* $k^* = k$.

Theorem 7.3 *The feedback laws* $\boldsymbol{u}^*(\boldsymbol{z}_k)$, $k = (0, \ldots, k^*)$ *calculated by Algorithm 7.4 are such that the PWA system (7.16) can robustly be pushed to* Ω $\forall \boldsymbol{\lambda} \in \Lambda$ *in, at most,* k^* *steps for any* $\boldsymbol{x} \in \mathcal{S}_{k^*}$. *Moreover, measurements of* $\boldsymbol{\lambda}$ *are taken into account by* $\boldsymbol{u}^*(z_k)$ *via (7.21a) to further optimize for performance.*

PROOF: At iteration k for any $\boldsymbol{x}_0 \in \mathcal{S}_k$ the feedback law obtained in Step 2(b) is such that the one-step prediction $\boldsymbol{x}_1 = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{u}(\boldsymbol{x}_0))$ is pushed into $\mathcal{S}_{k-1}$ in one time step $\forall \boldsymbol{\lambda}_0 \in \Lambda$. Robustness is ensured by taking $\mathcal{S}_k$ as given by (7.26) and (7.28). The iterative nature of the algorithm guarantees that for any $\boldsymbol{x}_1 \in \mathcal{S}_{k-1}$ we have $\boldsymbol{x}_2 = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_1, \boldsymbol{\lambda}_1, \boldsymbol{u}_{k-1}(\boldsymbol{x}_1)) \in \mathcal{S}_{k-2}$, $\forall \boldsymbol{\lambda}_1 \in \Lambda$. By consecutively applying the feedback laws $\boldsymbol{u}_{k-2}(\boldsymbol{x}_2), \boldsymbol{u}_{k-3}(\boldsymbol{x}_3), \ldots, \boldsymbol{u}_0(\boldsymbol{x}_k)$ we therefore get $\boldsymbol{x}_{k+1} \in \mathcal{S}_0$ (note that $\mathcal{S}_0 = \Omega$ by Step 1). Hence all states of PWA system (7.16) are pushed towards Ω in, at most, k^* steps due to the stopping criterion in Step 3. By employing (7.19) in the performance objective (7.21a) the knowledge of $\boldsymbol{\lambda}_k$ is taken into account when optimizing for $\boldsymbol{u}_k^*$ at each iteration. □

The regions $\mathcal{S}_k$ may overlap due to iterative property of Algorithm 7.4. For a proper implementation of PWA control law, one has to identify at which iteration the information variable $\boldsymbol{z}_k$ lies, then the control law is taken as the one where $\boldsymbol{z}_k \in \mathcal{P}_{k,r}$. This routine is specified by the following algorithm

Algorithm 7.5 (On-line implementation)

1. *Given the state vector* $\boldsymbol{x}_0$, *identify the mode* i *of PWA dynamics (7.16).*

2. *Find all regions where* $\boldsymbol{x}_0 \in \mathcal{S}_k$ *and take the one with minimal index* k_{min}.
3. *For given* $\boldsymbol{x}_0$, $\boldsymbol{\lambda}_0$ *and* i *evaluate* $\boldsymbol{z}_0$ *per* (7.19).
4. *For given* k_{min} *find the region index* r *where* $\boldsymbol{z}_0 \in \mathcal{P}_{k_{min},r}$.
5. *Evaluate control action* $\boldsymbol{u} = \boldsymbol{F}_{k_{min},r}\boldsymbol{z}_0 + \boldsymbol{g}_{k_{min},r}$.

Theorem 7.4 *The time optimal adaptive controller calculated by Algorithm 7.4 and applied to PWA system* (7.16) *in a receding horizon control fashion according to Algorithm 7.5 guarantees that all states are pushed towards* Ω *in the minimal possible number steps.*

PROOF: Assume the initial state $\boldsymbol{x}_0$ is contained in the set $\mathcal{S}_k$ from which it takes k steps to reach Ω according to Theorem 7.3. The control law identified by Algorithm 7.5 will drive the states into the set $\mathcal{S}_{k-1}$ in one time step. Therefore, the states will enter Ω in k steps when Algorithm 7.5 called repeatedly at each sampling instance. □

The control law calculated by Algorithm 7.4 is interpreted as the adaptive controller, because the effect of varying $\boldsymbol{\lambda}_k$ is taken into account via (7.19) and the control law is given by (7.25). Since the result is given as PWA function, it can be stored and implemented in the real-time control.

7.2.3 Example

In this section adaptive time optimal Algorithm 7.4 is applied to a modified version of the periodic PWA system of [15]. The dynamics of such a system is given by

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{A}_1\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if} \quad x_{1,k} < 0 \\ \boldsymbol{A}_2\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if} \quad x_{1,k} \geq 0, \end{cases} \tag{7.29}$$

where $x_{1,k}$ denotes the first coordinate of the state vector $\boldsymbol{x}_k$. State-update matrices for each of the two modes of the PWA system are given by

$$\boldsymbol{A}_i = \lambda \begin{pmatrix} \cos(\alpha_i) & -\sin(\alpha_i) \\ \sin(\alpha_i) & \cos(\alpha_i) \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{7.30}$$

with $\alpha_1 = -\pi/3$ and $\alpha_2 = \pi/3$, respectively. It is assumed that the value of the parameter λ is unknown at the time of the synthesis of the control law, but it is bounded by $0.7 \leq \lambda \leq 1$. The vertices $\boldsymbol{A}_{i,1}$, $\boldsymbol{A}_{i,2}$ in (7.17) can be obtained by evaluating $\boldsymbol{A}_i$ from (7.30) for boundary values of this interval. The objective of the control design is to drive states of the model (7.29) to the terminal set

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^2 \mid -0.5 \leq \boldsymbol{x} \leq 0.5\} \tag{7.31}$$

while satisfying state/input constraints

$$\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid -5 \leq \boldsymbol{x} \leq 5\}, \quad \mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}. \tag{7.32}$$

The adaptive time optimal Algorithm 7.4 has been implemented using routines from MPT Toolbox [64] and YALMIP [67]. Algorithm 7.4 has converged at iteration 8, generating 5 PWA feedback laws of the form (7.25), which are parametrized in the information variable $\boldsymbol{z}$. Regions over which these control laws are defined are depicted in Fig. 7.11.
The adaptive time optimal controller has been applied in the closed loop simulation for varying uncertainty λ. For simplicity, the parameter λ is not estimated on-line from the process, but it is assumed that the value is perfectly known at each sampling instant. A random sequence of λ_k has been generated and it is depicted in Fig. 7.12.
The initial condition for closed loop simulation has been set to $\boldsymbol{x}_0 = (0, \ -5)^T$. Corresponding evolution of adaptive control actions is shown in Fig. 7.14, and the state profiles are shown in Fig. 7.13. As can be seen from the plots, the time optimal controllers adapts itself to the current measurements of the parameter λ and robustly drives all system states towards the given terminal set Ω.

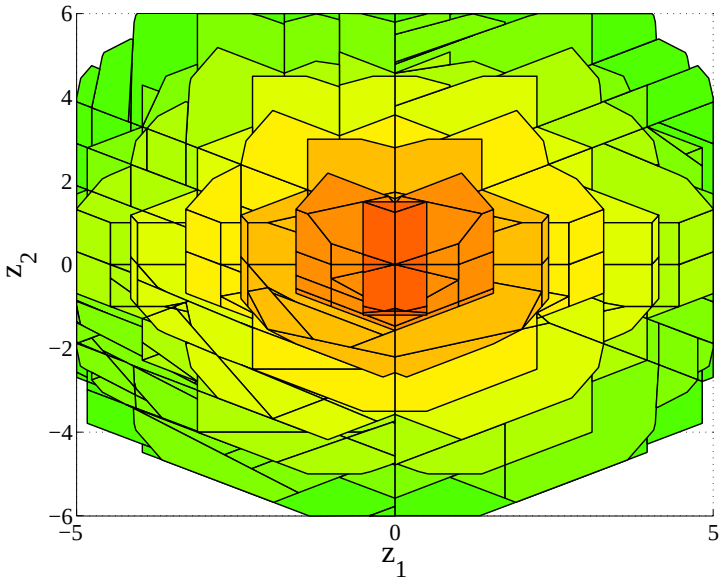


Figure 7.11: Regions of the PWA feedback laws $u(\boldsymbol{z}_k)$. By the same colors are depicted regions from the same iteration k.

Conclusions

The section proposes the explicit solution to time optimal MPC problem for PWA systems where the parameters of the dynamical matrix model are allowed to vary in a given bounded set Λ. The approach is based on the adaptive time optimal Algorithm 7.4 which solves at each step the optimal control problem (7.21) explicitly by considering (7.19) as parameter. The control law is characterized in the parameter $\boldsymbol{z}$ and the implementation mechanism is given by Algorithm 7.5. Since the whole information from the plant is given by the variable

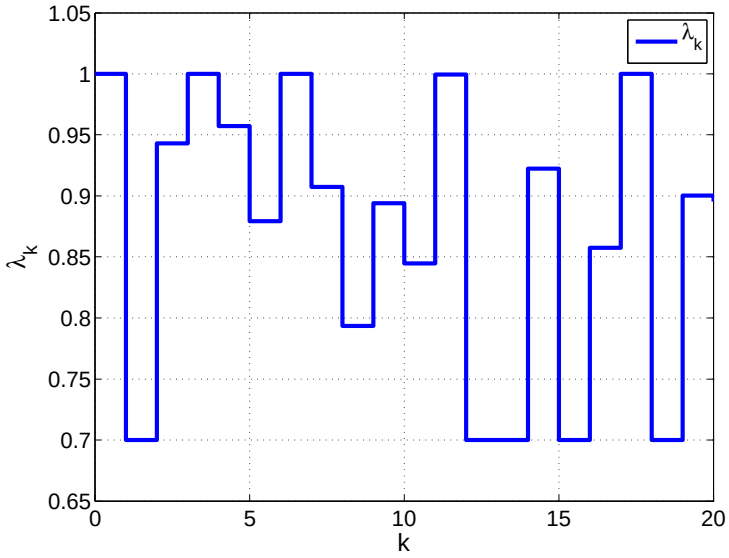


Figure 7.12: Time-varying fluctuations of the value of the uncertainty λ_k.

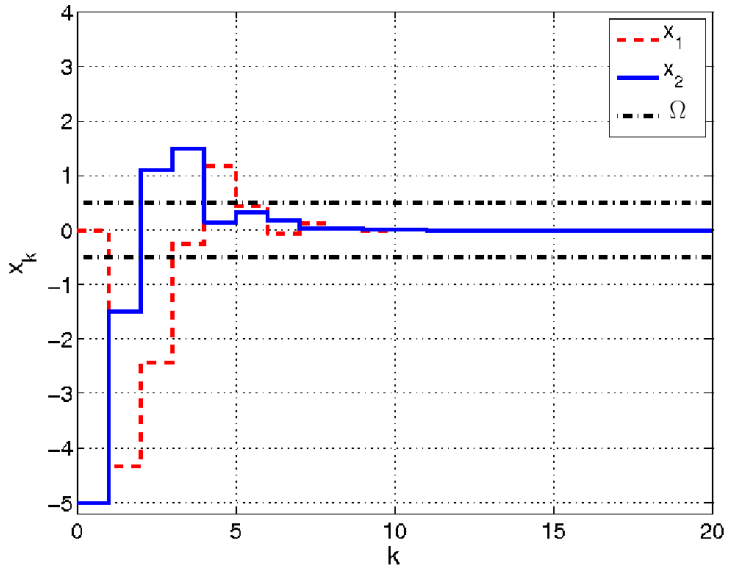


Figure 7.13: Closed-loop profiles of $\boldsymbol{x}$.

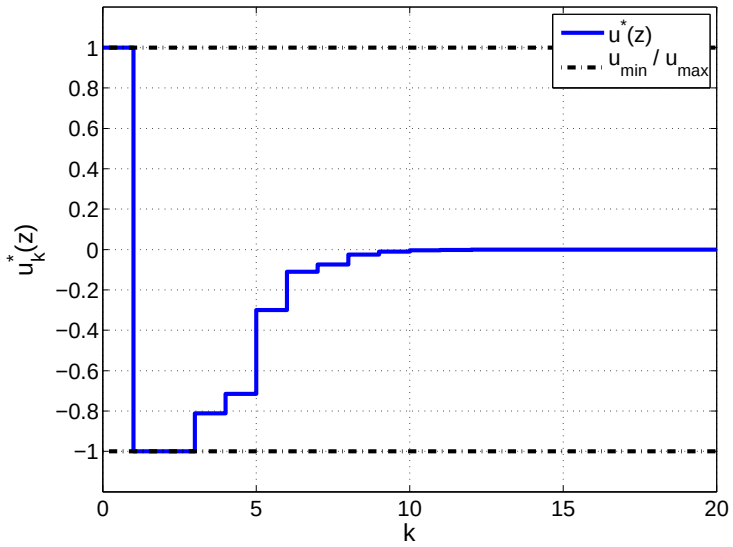


Figure 7.14: Closed-loop profiles of $u(\boldsymbol{z})$.

$\boldsymbol{z}$, the adaptive time optimal controller is capable to compensate the varying uncertainties from the set Λ. The approach is shown on a simple example which indicates the robust nature of the controller. Furthermore, the uncertain characteristic of the process can be applied to design a time optimal controller for the class of Takagi-Sugeno models which will be presented in the next section.

7.3 Time Optimal Control of Takagi-Sugeno Fuzzy Systems

The class of Takagi-Sugeno (TS) models can be viewed as universal approximators of general nonlinear systems [101]. Since the original contribution by [98], the attraction of fuzzy systems has gained much attention in the control society as the way of expressing the complex nonlinear behavior can be done in a human-friendly way. Because the TS model was easily understood and more precise, it was then consequently deployed in the model-based control strategies. An excellent survey [41] offers a detailed insight on recent developments in control of TS fuzzy systems.

This section is aimed to establish a bridge between the fuzzy TS models and uncertain PWA models. The main motivation lies in the explicit solutions to PWA model as this approach might nicely avoid the computational burden involved in on-line optimization. Although efficient on-line optimization of fuzzy MPC is suggested in [75], the solution might still suffer from suboptimality and reliability of the solver. Secondly, the stability guarantees can be given for the class of PWA systems by construction, as proved in [46]. This is simpler than derivation of stability constraints [76] or exploring the interaction between local fuzzy sets [112].

The initial connection between TS and PWA models has been investigated in [56] where the authors distinguish between operating and interpolating regions. The operating regions are those, where only one local model is valid and interpolating regions refer to areas where the system dynamics is given by convex combination of numerous local models. Following these ideas an initial transformation procedure appeared in [51] and consequently, it was applied to a time optimal control [50]. However, the main disadvantage of the proposed approach is that the uncertainty, which stems from modeling the interpolating regions, may become high and the controller acts conservatively. In this section an extension of these ideas is elaborated and the transformation of TS fuzzy systems to uncertain PWA systems is presented. Based on the uncertain PWA model, the explicit time optimal controller is constructed which provides stability and feasibility guarantees.

7.3.1 Relation Between TS Model and PWA Model

Both, discrete-time Takagi-Sugeno model (4.19) and PWA model (4.12) have been introduced in Section 4.4.2. To link these models together, assume that switching in PWA model is driven only by change in states, i.e.

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i \quad \text{if} \quad \boldsymbol{x}_k \in \mathcal{D}_i \tag{7.33}$$

where the set $\mathcal{D}_i \subset \mathbb{R}^n$ is a polytopic partition of the state space $\mathcal{D} = \bigcup_i \mathcal{D}_i$. The switching rule is associated with a binary variable $\delta_{i,k} = \{0, 1\}$ which is true if $\boldsymbol{x}_k \in \mathcal{D}_i$, as shown in (5.19). Using $\delta_{i,k}$, PWA model (7.33) can be written as

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{n_\mathcal{D}} \delta_{i,k} (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i) \tag{7.34}$$

where the condition

$$\sum_{i=1}^{n_\mathcal{D}} \delta_{i,k} = 1 \tag{7.35}$$

ensures that only one local dynamics i is selected.
TS model can be described using the fuzzy membership functions $\alpha_i(\boldsymbol{x}_k) \in [0, 1]$ which are defined in (4.18). Actually, the membership functions α_i represent the switching condition between the local dynamics given in fuzzy logic. Since the fuzzy logic is defined over the whole interval $[0, 1]$, TS model is given as interpolation between local models, i.e.

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{n_\mathcal{D}} \alpha_i(\boldsymbol{x}_k) (\boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i \boldsymbol{u}_k + \boldsymbol{c}_i). \tag{7.36}$$

The common factor for the models (7.34), (7.36) is the structure of the dynamical equation and the difference appears in the multiplication factors $\delta_i = \{0, 1\}$, $\alpha_i \in [0, 1]$. The immediate answer is, that both models give the same update if $\delta_i = \alpha_i$, which can only happen when α_i takes any of the boundary value from the interval $[0, 1]$. Investigation

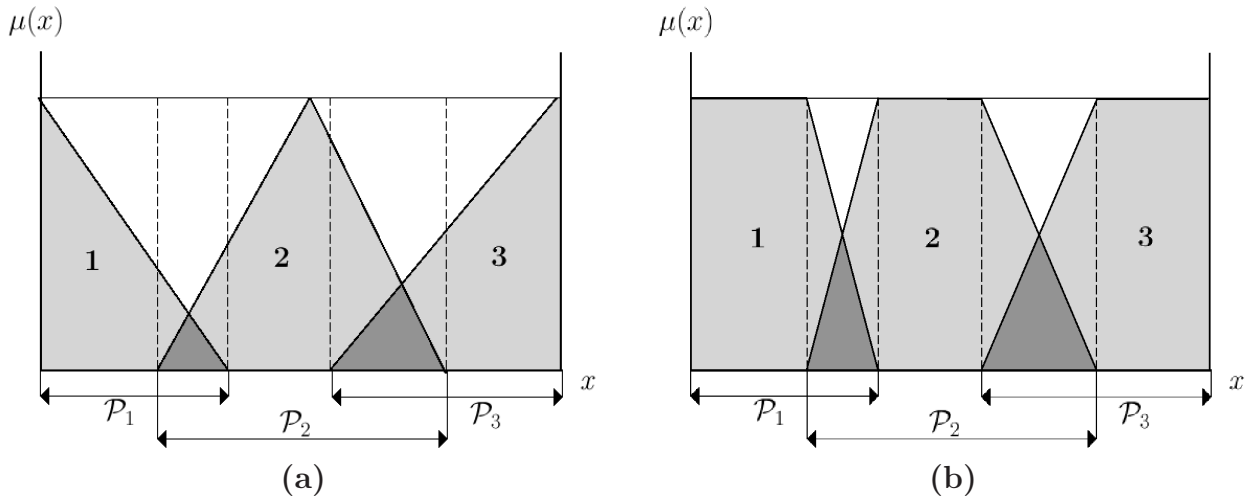


Figure 7.15: Interpolating regions (dark gray) and operating regions (gray) for linear (a) and trapezoidal (b) type of membership functions.

of the boundary values for α_i leads to the analysis of the membership functions μ_{ij}. The analysis shows that the exact shape of the membership functions μ_{ij} is not important as the space over which the function is defined (i.e. support of the function $supp(\mu_{ij}(\boldsymbol{x})) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \mu_{ij}(\boldsymbol{x}) > 0\}$). From (4.17) it follows that interpolation between local models takes place only if the supports of the fuzzy membership functions intersect, otherwise only one local model is valid. This principle suggests to partition the space to interpolation regions and operating regions as shown in [56]. This idea is nicely illustrated in Fig. 7.15 using three rules with triangular and trapezoidal membership functions. The support is denoted as $supp(\mu_i) = \mathcal{P}_i$ and interpolating regions are colored by dark gray. The reason why the linear or trapezoidal types of membership functions are preferred is because the support can be expressed as a polytope given by Definition 1.19, i.e.

$$\mathcal{P}_i := \{\boldsymbol{x}_k \in \mathbb{R}^n \mid \boldsymbol{H}_i \boldsymbol{x}_k \leq \boldsymbol{l}_i\} \tag{7.37}$$

where $\boldsymbol{H}_i \in \mathbb{R}^{q \times n}$ and $\boldsymbol{l}_i \in \mathbb{R}^q$ are matrices with q defining the number of half-spaces surrounding $\mathcal{P}$ [47].

Distinction between the operating and interpolating regions gives an answer where both models are equal or not. In the operating regions, only one model is active, thus both models are equal. In the interpolation regions, the update is given as linear combination of active dynamics, hence it acts as PWA systems with a parametric uncertainty. Hence, the main idea lies in translating TS system to PWA model with parametric uncertainties and to use explicit MPC to control the original TS system. As it will be seen in the next section, all involved operations in the transformation procedure will consist of geometry of polyhedral sets, hence the approach is more geometrical-based.

7.3.2 Transformation to Uncertain PWA System

The aim of this section is to transform TS model (7.36) to PWA model of the form (4.12). In the first step of the transformation one has to determine crisp boundaries between the interpolating regions and operating regions. This can be done a straightforward manner, by defining new regions for each intersection of the neighboring supports of fuzzy sets (7.37), i.e.

$$\mathcal{D}_a = \mathcal{P}_i \setminus \bigcup_{j \neq i} \mathcal{P}_j \tag{7.38}$$

where $\setminus$ is the set-difference operation given by Definition 1.22. Because the supports $\mathcal{P}_i$ are polytopes, their intersection will be also a polytope. Searching through the whole universe $\mathcal{P} = \bigcup_i \mathcal{P}_i$ one finds a number of polytopes. The remaining regions can be obtained again by a set-difference operation

$$\mathcal{D}_b = \mathcal{P} \setminus \mathcal{D}_a. \tag{7.39}$$

The resulting partitioning is given as union of $\mathcal{D}_a$ and $\mathcal{D}_b$, i.e.

$$\mathcal{D} = \mathcal{D}_a \cup \mathcal{D}_b = \bigcup_{j}^{n_\mathcal{D}} \mathcal{D}_j. \tag{7.40}$$

The partitioning is illustrated in Fig. 7.16 and Fig. 7.17. Fig. 7.16 illustrates the membership functions $\mu_i(x_k)$ for TS model with their supports $\mathcal{P}_i$. The polytopes $\mathcal{P}_i$ in Fig. 7.16 do overlap, hence the set-difference operation (7.38) is performed, which gives regions $\mathcal{D}_1$, $\mathcal{D}_3$, and $\mathcal{D}_5$ in Fig. 7.17. By performing the second set-difference operation (7.39) the regions $\mathcal{D}_2$, $\mathcal{D}_4$ are generated. Important is, that the generated regions $\mathcal{D}_j$ have crisp boundaries and do not intersect. In addition, since the supports $\mathcal{P}_i$ are represented as polytopes (7.37), the overall calculation of intersections can be performed using standard algebraic manipulations for which efficient routines exist [64].
Once the strictly separated regions $\mathcal{D}_j$ are obtained, the second step is to associate one local model to each such region. This step requires introduction of the following two assumptions:

Assumption 7.1 *The membership functions in* (4.17) *are trapezoidal functions, i.e.*

$$\sum_{i}^{n_\mathcal{D}} \alpha_i(\boldsymbol{x}_k) = 1. \tag{7.41}$$

Assumption 7.2 *In TS model* (4.15)*, the matrices* $\boldsymbol{B}_i$ *and* $\boldsymbol{c}_i$ *are constant* $\forall i = 1, \ldots, n_\mathcal{D}$*, i.e.* (4.19) *can be written as*

$$\boldsymbol{x}_{k+1} = \Big(\sum_{i}^{n_\mathcal{D}} \alpha_i(\boldsymbol{x}_k) \boldsymbol{A}_i \Big) \boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{c} \tag{7.42}$$

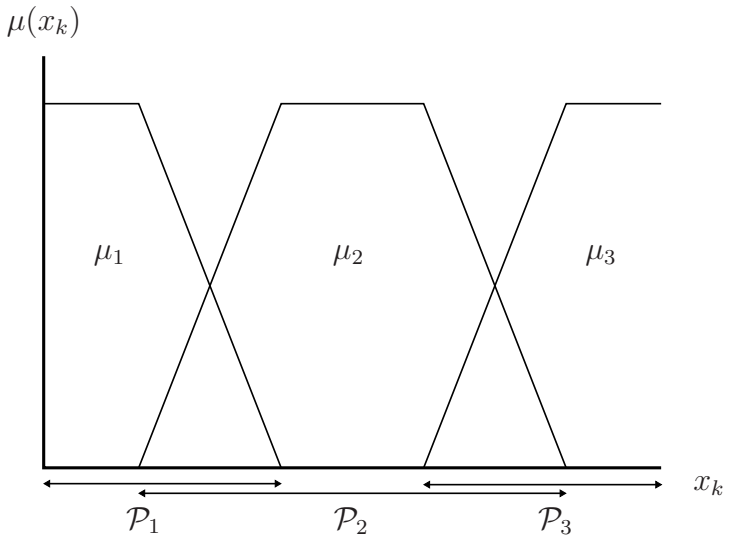


Figure 7.16: Fuzzy membership functions and their supports $\mathcal{P}_i$.

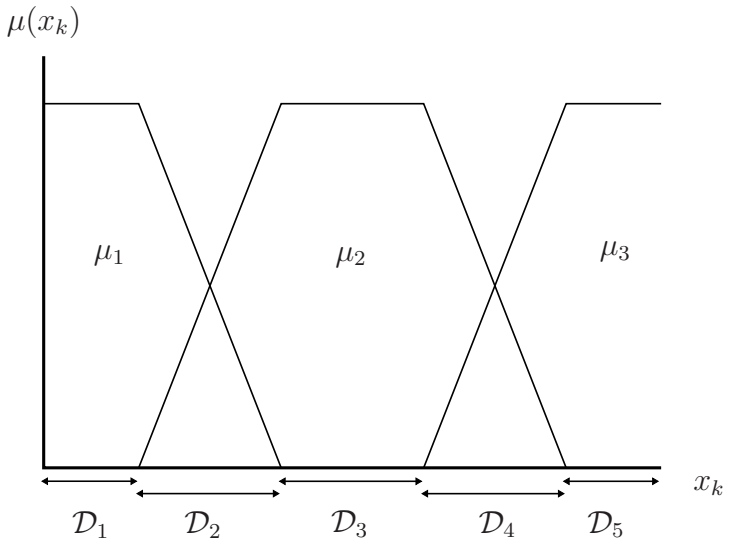


Figure 7.17: Intersections between supports of membership functions defines PWA partitioning.

Assumption 7.1 is stated in order to obtain a combination of interpolating dynamics given as convex hull. Assumption 7.2 is more restrictive because it requires matrices $\boldsymbol{B}$ and $\boldsymbol{c}$ to be constants. It is imposed in order to obtain PWA model suitable for adaptive time optimal scheme in explicit MPC.
Association of local dynamics to given partitions $\mathcal{D}_j$ is done for each j, by rewriting (7.42) into the form of (7.16) in such a way that the cross-term $\alpha_j(\boldsymbol{x}_k)\boldsymbol{A}_j$ is replaced by the linear combination of the form of (7.17). The vertices $\boldsymbol{A}_{j,1}, \ldots, \boldsymbol{A}_{j,n_\alpha}$ of the uncertainty set (7.17) can be easily obtained from (7.42) by evaluating $\alpha_j(\boldsymbol{x}_k)\boldsymbol{A}_j$ for $\boldsymbol{x}_k$ being the vertices of the region $\mathcal{D}_j$, i.e.

$$\boldsymbol{A}_{j,t} = \alpha(\boldsymbol{v}_t)\boldsymbol{A}_j \quad \forall \boldsymbol{v}_t \in vert(\mathcal{D}_j). \tag{7.43}$$

Simply speaking, using this procedure the original TS model (7.42) is over-approximated by a PWA system with parametric uncertainties in a way such that the set of state evolutions governed by the PWA model (7.16) is in fact a superset of the evolutions of the original TS model.
Once the non-overlapping regions $\mathcal{D}_j$ and the corresponding vertices $\boldsymbol{A}_{j,1}, \ldots, \boldsymbol{A}_{j,n_\alpha}$ of (7.17) are computed, the connection between PWA form (7.16) and the TS description (4.19) can be established by the following lemma.

Lemma 7.1 *PWA model (7.16) is equivalent to the TS model (7.42) if*

$$\lambda_{j,k} = \alpha_j(\boldsymbol{x}_k) \tag{7.44}$$

is known (can be measured or estimated) at each time k.

PROOF: Follows directly from Assumptions 7.1, 7.2, and from the definition of $\mathcal{D}_j$ and $\boldsymbol{A}_{j,t}$ as in (7.40) and (7.43), respectively. □
Once TS model has been expressed in uncertain PWA form, it is possible to apply techniques for synthesis of predictive controlled based on robust control approaches. In particular, if TS model is transformed to PWA system with parametric uncertainties, the adaptive time optimal Algorithm 7.4 described in Section 7.2.2 can be employed. Explicit solution in both cases ensures the real-time implementation with very low computational effort.

7.3.3 Example

Transformation procedure to PWA system with parametric uncertainty has been applied to design a time optimal controller for a continuously stirred tank reactor (CSTR), where the reaction $A \rightarrow B$ takes place. By considering the normalized conversion rate as x_1 and the normalized mixture temperature as x_2, and the coolant temperature as the system input u, the model of the reactor is given by [27]

$$\dot{x}_1 = f_1(\boldsymbol{x}) + \left(\frac{1}{\eta} - 1\right) x_1 \tag{7.45a}$$

$$\dot{x}_2 = f_2(\boldsymbol{x}) + \left(\frac{1}{\eta} - 1\right) x_2 + \beta u \tag{7.45b}$$

where

$$f_1(\boldsymbol{x}) = -\frac{1}{\eta}x_1 + D_\alpha(1-x_1)e^{\left(\frac{x_2}{1+x_2/\gamma_0}\right)}$$
$$f_2(\boldsymbol{x}) = -\left(\frac{1}{\eta}+\beta\right)x_2 + HD_\alpha(1-x_1)e^{\left(\frac{x_2}{1+x_2/\gamma_0}\right)}$$
$$\gamma_0 = 20, \quad H = 8, \quad \beta = 0.3, \quad D_\alpha = 0.072, \quad \eta = 0.8.$$

The states are subject to constraints $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 10$, respectively. The control signal is bounded by $-20 \leq u \leq 20$.
The nonlinear model (7.45) is approximated by a Takagi-Sugeno fuzzy system with three membership functions. Assuming the sampling time $T_s = 0.1$ minutes, the TS model is given by

$$\boldsymbol{x}_{k+1} = \left(\sum_{i=1}^{3}\alpha_i(\boldsymbol{x}_k)\boldsymbol{A}_i\right)\boldsymbol{x}_k + \boldsymbol{B}u_k \tag{7.46}$$

where $\alpha_i(\boldsymbol{x}_k)$ can be calculated from $\mu_i(\boldsymbol{x}_k)$ by (4.18). The numerical values of $\boldsymbol{A}_i$ and $\boldsymbol{B}$ are given by

$$\boldsymbol{A}_1 = \begin{pmatrix} 0.8665 & 0.0067 \\ -0.1260 & 0.9094 \end{pmatrix}, \quad \boldsymbol{A}_2 = \begin{pmatrix} 0.8029 & 0.0388 \\ -0.6278 & 1.1623 \end{pmatrix},$$
$$\boldsymbol{A}_3 = \begin{pmatrix} 0.6041 & 0.0265 \\ -2.1938 & 1.0652 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0.0001 \\ 0.0286 \end{pmatrix}.$$

Dynamics $\boldsymbol{A}_1$ and $\boldsymbol{A}_3$ capture the stable operating points of the CSTR, while dynamics $\boldsymbol{A}_2$ corresponds to the unstable mode of the reactor. The membership functions $\mu_1(\boldsymbol{x})$, $\mu_2(\boldsymbol{x})$, and $\mu_3(\boldsymbol{x})$ are given as trapezoidal functions with centers around respective linearization points:

$$\mu_1(\boldsymbol{x}) = \begin{cases} 1 & \text{if } 0 \leq x_2 \leq 2.5 \\ 1 - \frac{x_2-2.5}{2.652-2.5} & \text{if } 2.5 \leq x_2 \leq 2.652 \\ 0 & \text{otherwise} \end{cases}$$
$$\mu_2(\boldsymbol{x}) = \begin{cases} \frac{x_2-2.5}{2.652-2.5} & \text{if } 2.5 \leq x_2 \leq 2.652 \\ 1 & \text{if } 2.652 \leq x_2 \leq 2.852 \\ 1 - \frac{x_2-2.852}{3-2.852} & \text{if } 2.852 \leq x_2 \leq 3 \\ 0 & \text{otherwise} \end{cases}$$
$$\mu_3(\boldsymbol{x}) = \begin{cases} 1 & \text{if } x_2 \geq 3 \\ \frac{x_2-2.852}{3-2.852} & \text{if } 2.852 \leq x_2 \leq 3 \\ 0 & \text{otherwise.} \end{cases}$$

In order to synthesize the time optimal controller for such a TS system, TS model (7.46) needs to be converted to a corresponding PWA form (7.16) by using the procedure described in Section 7.3.2. The partitioning $\mathcal{D}_j$ results directly from the respective domains

of individual μ_i's, i.e.

$$\begin{aligned}
\mathcal{D}_1 :=& \{\boldsymbol{x} \in R^2 \mid 0 \leq x_2 \leq 2.5\} \\
\mathcal{D}_2 :=& \{\boldsymbol{x} \in R^2 \mid 2.5 \leq x_2 \leq 2.652\} \\
\mathcal{D}_3 :=& \{\boldsymbol{x} \in R^2 \mid 2.652 \leq x_2 \leq 2.852\} \\
\mathcal{D}_4 :=& \{\boldsymbol{x} \in R^2 \mid 2.852 \leq x_2 \leq 3\} \\
\mathcal{D}_5 :=& \{\boldsymbol{x} \in R^2 \mid 3 \leq x_2 \leq 10\}.
\end{aligned} \tag{7.48}$$

What remains to be done is to assign dynamics to each element of $\mathcal{D}_j$. This can be performed by (7.43) and results in

$$\boldsymbol{x}_{k+1} = \begin{cases}
\boldsymbol{A}_1\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } x_2 \in \mathcal{D}_1 \\
\{\boldsymbol{A}_1, \boldsymbol{A}_2\}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } x_2 \in \mathcal{D}_2 \\
\boldsymbol{A}_2\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } x_2 \in \mathcal{D}_3 \\
\{\boldsymbol{A}_2, \boldsymbol{A}_3\}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } x_2 \in \mathcal{D}_4 \\
\boldsymbol{A}_3\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } x_2 \in \mathcal{D}_5
\end{cases} \tag{7.49}$$

where $\{\boldsymbol{A}_1, \boldsymbol{A}_2\}$ represents a system matrix $\boldsymbol{A}$ as a convex combination of the vertices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$.

To perform the controller synthesis, the Algorithm 7.4 has been implemented using the MPT Toolbox [64] and YALMIP [67]. The control objective was to drive the system states towards the terminal set $\Omega = \{\boldsymbol{x} \mid 2.652 \leq x_2 \leq 2.852\}$ in the minimal possible number of time steps, while minimizing the objective function $J = |u_k| + |10(\boldsymbol{x}_{k+1} - 2.751)|$ at each step k. This operating range corresponds to the unstable mode of the reactor. The implementation of Algorithm 7.4 resulted in PWA feedback law of the form (7.25) defined over 483 regions in the two-dimensional $\boldsymbol{z}$ space, which are depicted in Figure 7.18. The proposed Algorithm 7.4 was compared with a standard parallel distributed compensation (PDC) approach of [111]. Input constraints have been incorporated into PDC design using and PDC controller was computed via YALMIP interface. For the initial condition $\boldsymbol{x}_0 = (0.6,\ 9)^T$, Figure 7.19 shows the closed-loop evolution of system state x_2 for both controllers (TOC refers to time optimal control). While PDC controller reacts on the initial condition conservatively, time optimal controller is significantly faster and utilizing the on-line measurements of μ_i's it drives system states towards the chosen terminal set in the minimum number of steps. The corresponding values of the control actions are shown in Figure 7.20.

Conclusions

The multiparametric approach for computing explicit solution to time optimal MPC problem for the class of TS models has been presented. This novel way of controlling TS models comprises of two steps. Firstly, TS model with fuzzy logical rules is over-approximated by

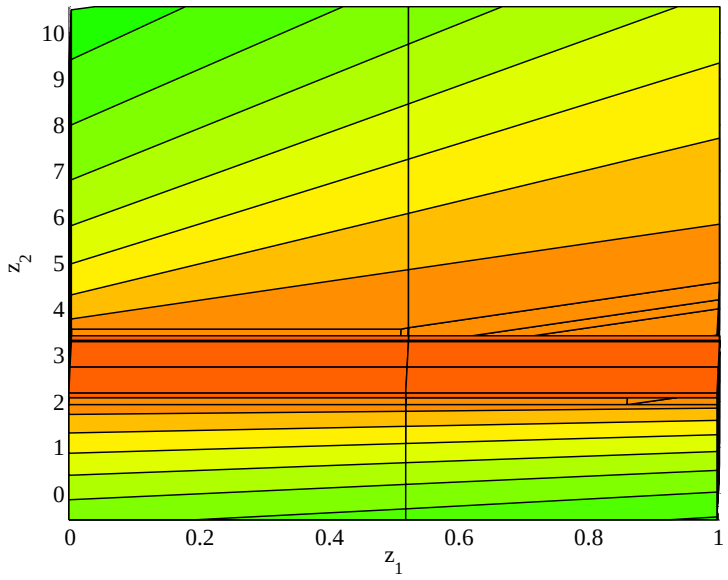


Figure 7.18: Regions of the time-optimal control law $u(\boldsymbol{z})$. Regions depicted by the same color denote regions, from where it takes the same number of steps to reach the initial terminal set. The more red the color is, the lower is the number of required steps.

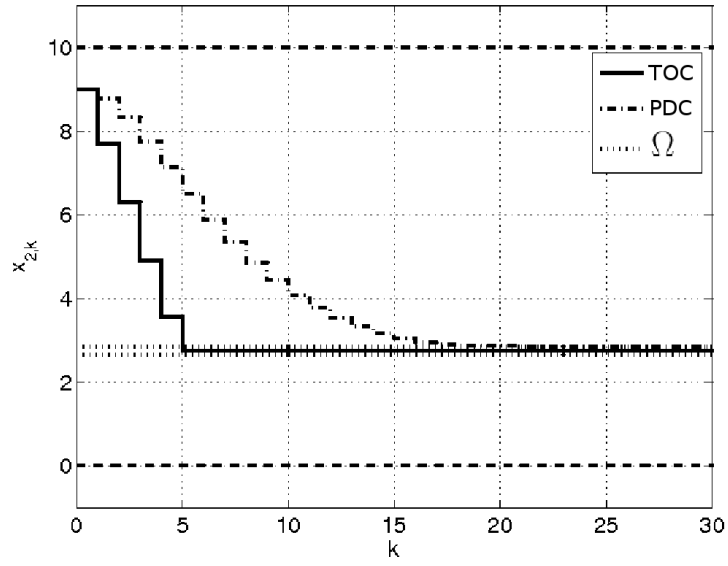


Figure 7.19: Closed-loop evolutions of the dimensionless temperature.

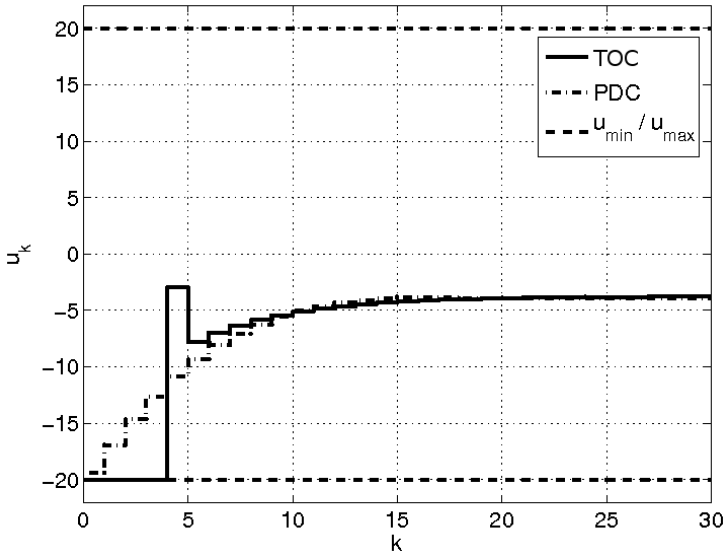


Figure 7.20: Values of the control actions.

PWA model with uncertain dynamical matrix. Secondly, the theory of adaptive time optimal control for PWA systems presented in Section 7.2.2 is applied which results in the explicit MPC controller characterized in the given parameter. The controller obtained in this way guarantees convergence toward desired terminal set in the minimal number of steps while satisfying the state and input constraints. The approach has been investigated in CSTR case study where the results show the desired performance.

7.4 Polynomial Approximation of MPC

Although the explicit solution to MPC possesses very appealing properties from the implementation point of view, the sole implementation task might not be as trivial as is seems. Especially, when it comes to fast sampling rates (i.e. order of microseconds) and low computational resources one has to take a closer look on the complexity of the implementation scheme. In particular, there are two aspects to be considered:

1. Memory consumption.

2. Evaluation time for the control law (floating point operations - FLOPS).

The time needed to evaluate the lookup table limits the minimal admissible sampling time of the control system. The simplest, but also the least efficient way of processing is a sequential search, in which all elements of the table are inspected sequentially for the location of the actual measurement of the plant's state. It can be shown that the time complexity and memory requirements of a sequential search procedure depend linearly on the number of controller regions. However, there are applications where the processing

power needed to perform a sequential search can be prohibitive; e.g. controlling dc-to-dc converters [8, 43] where sampling rates are in the range of microseconds and controller partitions comprising several thousands of regions. In order to decrease the processing and memory requirements, one may use the binary search tree approach of [106], which allows to search the lookup tables in time logarithmic in the number of regions of the table. The approach has been also presented in Section 5.4.
However, the number of operations needed to traverse the binary tree, as well as the memory requirement to store the tree itself, can still be prohibitive either because of a high number of controller regions, or due to the lack of processing power. Therefore, in this section it is proposed to approximate the optimal control law by a single polynomial, which can be evaluated more efficiently compared to binary search trees, while maintaining stability and performance loss guarantees. Specifically, it will be illustrated that the approximation-based scheme can be evaluated in a constant number of CPU[1] operations, regardless of the complexity of the parametric solution. Similarly, the memory requirement is only a constant function of the degree of the approximation polynomial and does not depend on the complexity of the lookup table or underlying partition.

7.4.1 Stability Analysis

Stability analysis for PWA systems has been addressed in many publications, (see for instance [33] for references) where geometric properties of the explicit solutions to CFTOC problem (5.25) are exploited to certify stability of the closed loop. The common denominator of these approaches is to attain a set invariance and to get rid of unstable parts. One of the algorithms, which generates the maximum invariant set for PWA systems, has been proposed by [88]. The interest here is to introduce the approach by [32] which uses Lyapunov stability theory to find sets of stabilizing controllers. Given the explicit solution to CFTOC problem (5.25), the knowledge of (5.26) can be employed for construction of Lyapunov function [32]. Consequently, the notion of *stability tubes* needs to be introduced.

Definition 7.2 (Stability tube [32]) *Let $V(\boldsymbol{x}_k)$ be a Lyapunov function for the closed-loop system (5.18b) under stabilizing control (5.25) and constraints (5.18c), (5.18d), (5.18e) for some small number $0 < \beta \ll 1$. Then the set*

$$\mathcal{S}(V,\beta) := \left\{ \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} \in \mathbb{R}^{n+m} \,\middle|\, V(\boldsymbol{x}_{k+1}) - V(\boldsymbol{x}_k) \leq -\beta ||\boldsymbol{x}_k||_\infty \right\} \tag{7.50}$$

is called stability tube.

Simply speaking, the stability tube is a set in the $\boldsymbol{x}$-$\boldsymbol{u}$ space where the Lyapunov function $V(\boldsymbol{x}_k)$ decreases with a factor $\beta||\boldsymbol{x}_k||_\infty$, hence, it denotes the set of stabilizing controllers for given explicit solution. Important is, that for PWA systems, the stability tubes can

[1] central processing unit

be computed using geometric operations with polyheral regions. Since the whole operations are performed with polyhedrals, resulting stability tubes are given as a collection of polytopes

$$\mathcal{S}(V,\beta) = \left\{ \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} \in \mathbb{R}^{n+m} \,\middle|\, \boldsymbol{H}_j^{\mathcal{S}} \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{u}_k \end{pmatrix} \leq \boldsymbol{l}_j^{\mathcal{S}},\ j = 1,\ldots,n_S \right\}. \tag{7.51}$$

By constructing the stability tubes, the control law is replaced with a set and it depends on the selection of parameter β. As β is close to zero, these sets are large and as β approaches one, the stability tubes totally shrink to the control law. This parameter can be used as tuning parameter for bounding the closed loop performance.
Simply speaking, stability tubes create implicit boundaries for possible perturbations of the control law. As long as these perturbations remain in this set, the closed loop will be stable. Thus, the main idea of the polynomial approximation is to find a polynomial control law, which is as close as possible to optimal solution (5.25), while respecting the boundaries of the stability tubes. For more details about the stability tubes, the reader is referred to original contribution [32].

7.4.2 Polynomial Approximation

The objective of this section is to find a polynomial control law which maintains similar properties as optimal solution (5.25), but is very cheap to implement (i.e. requires less memory and evaluation time). The approximated polynomial takes the following form

$$\boldsymbol{\mu}(\boldsymbol{x}_k)_r = \boldsymbol{a}_1 \boldsymbol{x}_k + \ldots + \boldsymbol{a}_r \boldsymbol{x}_k^r \tag{7.52}$$

where r denotes the degree and $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_r$ are the coefficients of the polynomial $\boldsymbol{\mu}(\boldsymbol{x}_k)_r$. This form of the polynomial is specifically of interest because it does not contain cross products (e.g. $x_1 x_2$) and the evaluation of this type of polynomial is very fast. Furthermore, the polynomial (7.52) does not contain affine term otherwise the closed loop system would be unstable around origin.
To ensure stabilizing properties of the polynomial control law (7.52), it must hold

$$\begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{\mu}(\boldsymbol{x}_k)_r \end{pmatrix} \in \mathcal{S}(V,\beta) \quad \forall \boldsymbol{x}_k \in \mathcal{P} \tag{7.53}$$

for guaranteeing desired decrease β for given Lyapunov function V. Expression $\mathcal{P} = \bigcup_i \mathcal{P}_{k,i}$ corresponds to feasible domain of explicit solution to CFTOC problem (5.25). Secondly, to maintain some performance of the polynomial, it is required that the polynomial remains as close as possible to the optimal solution. These conditions can be met by formulating the optimization problem as follows:

$$\min_{\boldsymbol{a}_0,\ldots,\boldsymbol{a}_r} \sum_j \|\boldsymbol{\mu}(\boldsymbol{x}_k)_r - \boldsymbol{u}(\boldsymbol{x}_k)\|_2 \tag{7.54a}$$

$$\text{s.t.} \quad \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{\mu}(\boldsymbol{x}_k)_r \end{pmatrix} \in \mathcal{S}(V,\beta) \tag{7.54b}$$

where the difference between the approximated polynomial (7.52) and optimal PWA function (5.25) is minimized such that both states and inputs lie in the interior of stability tubes. As the stability tubes (7.51) are given as collection of polytopes, the condition (7.54b) can be put into matrix form, i.e.

$$\boldsymbol{H}_j^{\mathcal{S}} \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{\mu}(\boldsymbol{x}_k)_r \end{pmatrix} \leq \boldsymbol{l}_j^{\mathcal{S}} \tag{7.55}$$

for $j = 1, \ldots, n_{\mathcal{S}}$. Problem (7.54) is a non-convex because of the union of polytopes (7.51), but can be solved efficiently by transformation to SDP problem (2.4). To do this, following lemmas are needed:

Lemma 7.2 ([95]) *The system of polynomial inequalities*

$$h(\boldsymbol{x}) \geq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \quad \boldsymbol{g}_i(\boldsymbol{x}) \geq 0, \tag{7.56}$$

with $h(\boldsymbol{x}) = \sum_\alpha \boldsymbol{a}_\alpha \boldsymbol{x}^\alpha$ *and* $g_i(\boldsymbol{x}) = \sum_\alpha \boldsymbol{b}_{\alpha,i} \boldsymbol{x}^\alpha$ *is satisfied for all* $\boldsymbol{x}$ *from the set* $g_i(\boldsymbol{x}) \geq 0$ *if there exist non-negative polynomials* $s_i(\boldsymbol{x}) = \sum_\alpha \boldsymbol{c}_{\alpha,i} \boldsymbol{x}^\alpha$ *such that*

$$h(\boldsymbol{x}) - \sum_i s_i(\boldsymbol{x}) g_i(\boldsymbol{x}) \quad \geq \quad 0, \tag{7.57a}$$

$$s_i(\boldsymbol{x}) \quad \geq \quad 0. \tag{7.57b}$$

Lemma 7.3 ([82]) *A real valued polynomial* $P(\boldsymbol{x})$ *is non-negative for all* $\boldsymbol{x}$ *if there exists a sum of squares (SOS) decomposition of the form*

$$P(\boldsymbol{x}) = \sum_i p_i^2(\boldsymbol{x}), \quad p_i(\boldsymbol{x}) \in \mathbb{R}. \tag{7.58}$$

The Lemmas 7.2 and 7.3 allows reformulation of the approximated problem (7.54) into SOS problem, for which efficient solvers exist [67, 96, 104]. By Lemma 7.2 equation (7.54b) (cf. (7.53)) can be expressed as

$$h(\boldsymbol{x}_k) = \boldsymbol{l}_j^{\mathcal{S}} - \boldsymbol{H}_j^{\mathcal{S}} \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{\mu}(\boldsymbol{x}_k)_r \end{pmatrix}, \tag{7.59a}$$

$$g_i(\boldsymbol{x}_k) = [\boldsymbol{l}_i^{\mathcal{P}}]_c - [\boldsymbol{H}_i^{\mathcal{P}}]_c \boldsymbol{x}_k, \tag{7.59b}$$

where $[\cdot]_c$ is to be interpreted row-wise. Consequently, Lemma 7.3 is used to form (7.58) by merging (7.59) to

$$P(\boldsymbol{x}_k) = \boldsymbol{l}_c - \boldsymbol{H}_c \begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{\mu}(\boldsymbol{x}_k)_r \end{pmatrix} - \sum_i s_i(\boldsymbol{x}_k)(\boldsymbol{l}_{i,c} - \boldsymbol{H}_{i,c} \boldsymbol{x}_k). \tag{7.60}$$

Hence, the approximated problem (7.54) can be expressed as SOS problem, i.e.

$$\min_{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_r} \sum_j \| \boldsymbol{\mu}(\boldsymbol{x}_k)_r - \boldsymbol{u}(\boldsymbol{x}_k) \|_2 \tag{7.61a}$$

$$\text{s.t.} \quad P(\boldsymbol{x}_k) \text{ in (7.60) is sum of squares} \tag{7.61b}$$

$$s_i(\boldsymbol{x}_k) \text{ in (7.57b) are sum of squares} \tag{7.61c}$$

and solved using e.g. YALMIP [67]. Procedure for computing the approximated polynomial stems from the assumption, that there exist feasible solution to SOS problem (7.61) because conditions in (7.61) are merely sufficient. The approximated polynomial can still exist, even if the problem (7.61) is infeasible. The main result of this section is stated by the following theorem.

Theorem 7.5 (Polynomial approximation) *There exists a state feedback $\boldsymbol{\mu}(\boldsymbol{x})_r$ of the form (7.52) which stabilizes PWA system (5.18b) if the polynomials $s_i(\boldsymbol{x}_k)$ in (7.57b) and polynomial $P(\boldsymbol{x}_k)$ in (7.60) given by (7.58) are sum-of-squares. Moreover, the coefficients $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r$, of the polynomial $\boldsymbol{\mu}(\boldsymbol{x}_k)_r$ in (7.52) can be found by solving (7.61) as a semi-definite program.*

PROOF: By definition of stability tubes 7.2, any control law $\boldsymbol{\mu}(\boldsymbol{x}_k)_r$ with $(\boldsymbol{x}_k^T, \boldsymbol{\mu}(\boldsymbol{x}_k)_r^T)^T \in \mathcal{S}(V, \beta)$, stabilizes system (5.18b) while satisfying the system constraints (5.18c), (5.18d), (5.18e). The polynomial $\boldsymbol{\mu}(\boldsymbol{x}_k)_r$ fulfills this condition for all admissible states $\boldsymbol{x} \in \bigcup_i \mathcal{P}_i$ if and only if (7.51) is fulfilled. According to Lemma 7.2 and (7.59), the satisfaction of (7.60) is implied by the existence of polynomials $s_i(\boldsymbol{x}_k) \geq 0$ and the non-negativity of (7.57a). By substituting (7.59a) and (7.59b) into (7.57a) equation (7.60) is obtained. It follows from Lemma 7.3 that (7.60) will be globally non-negative if there exists a set of coefficients $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r$ of the polynomial $\boldsymbol{\mu}(\boldsymbol{x}_k)_r$ defined by (7.52) such that (7.60) is a sum-of-squares and, simultaneously, there exists a SOS decomposition of the polynomials $s_i(\boldsymbol{x}_k)$. Finally, results from [81] shows that the SOS decomposition can be found using semi-definite programming. □

7.4.3 Complexity Analysis

This section compares the complexity of the polynomial approximation scheme with the binary search tree approach presented in Section 5.4.

Memory Requirement

In the case of the binary search tree, one has to allocate at least $D(n+3) + N_u m(n+1)$ elements, where m and n denote the dimension of system inputs and states, respectively, D is the depth of the tree and N_u stands for the number of unique control laws. In [106], $D = 1.7 \log_2 n_{\mathcal{P}}$ is given as a realistic estimate, where $n_{\mathcal{P}}$ is the total number of controller regions and $N_u \approx \frac{1}{4} n_{\mathcal{P}}$. The total memory requirement for the binary search tree is therefore $1.7(n+3)\log_2 n_{\mathcal{P}} + \frac{1}{4} n_{\mathcal{P}} m(n+1)$ elements. It is clear that the number of controller regions $n_{\mathcal{P}}$ is a decisive factor which determines the maximal storage requirements of the binary search approach.
For memory requirements required by the polynomial (7.52), one has to store r coefficients $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r$ of size $m \times n$, thus it gives rnm elements to be allocated. It is worth noting that the memory storage for this approach does not depend on the number of controller regions. As can be seen from Figure 7.21, the memory requirements of the polynomial controller is significantly lower compared to the one of a binary search tree approach.

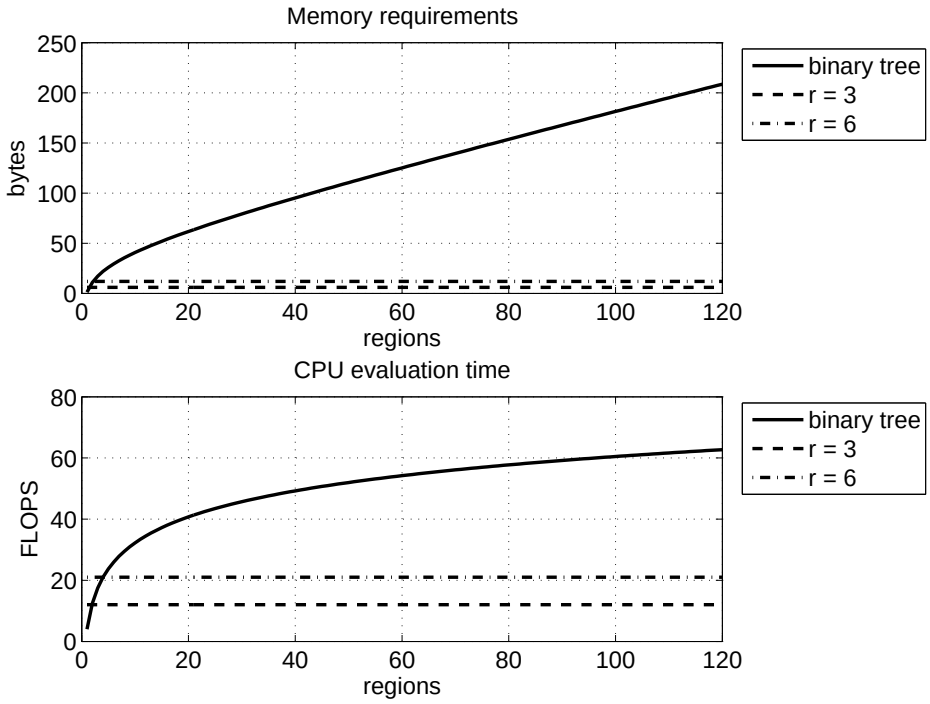


Figure 7.21: Memory and processing requirements needed to store and evaluate the binary search tree and the polynomial controller (7.52).

Evaluation

The number of CPU instructions which are needed to evaluate a binary search tree for a particular state measurement is a function of the total number of controller regions $n_{\mathcal{P}}$. Specifically, one has to perform $1.7(2n+1)\log_2 n_{\mathcal{P}} + 2nm$ mathematical operations (multiplications, additions and comparisons) to traverse the tree and calculate the corresponding control action.
In the case of polynomial controllers of the form (7.52), the number of operations that need to be performed in order to evaluate the polynomial at the given measured state is, again, independent of the number of regions $n_{\mathcal{P}}$. Specifically, if the *Horner's* scheme [39] is used for the evaluation, one needs at most $\frac{1}{2}mn(3r+5)$ operations to obtain the control input. The CPU requirements for both approaches are illustrated in Figure 7.21 for the case of $m = 1$ and $n = 2$.

7.4.4 Example

Consider the example from [15]

$$\boldsymbol{x}_{k+1} = \frac{4}{5}\begin{pmatrix}\cos\alpha(\boldsymbol{x}_k) & -\sin\alpha(\boldsymbol{x}_k)\\ \sin\alpha(\boldsymbol{x}_k) & \cos\alpha(\boldsymbol{x}_k)\end{pmatrix}\boldsymbol{x}_k + \begin{pmatrix}1\\0\end{pmatrix}u_k \tag{7.62a}$$

$$\alpha(\boldsymbol{x}_k) = \begin{cases}\frac{\pi}{3} & \text{if} \quad x_{1,k} \geq 0,\\ -\frac{\pi}{3} & \text{if} \quad x_{1,k} < 0,\end{cases} \tag{7.62b}$$

which is subject to constraints

$$\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^2 \mid \boldsymbol{x} \in [-10,\ 10] \times [-10,\ 10], \quad \mathcal{U} = \{u \in \mathbb{R} \mid u \in [-4,\ 4]\}. \tag{7.63}$$

CFTOC Problem (5.18) was solved with the parameters $\boldsymbol{Q} = \boldsymbol{I}$, $R = 1$, $N = 3$, $\boldsymbol{P} = \boldsymbol{0}$, $p = \infty$ using the MPT Toolbox [64] yielding PWA control law defined over a partition comprising of 26 regions. Subsequently the stability tubes $\mathcal{S}$ have been calculated with the choice of the performance tuning parameter $\beta = 1{\cdot}10^{-5}$. The approximation problem (7.61) was then successfully solved for different degrees of the approximation polynomial (7.52) using YALMIP [67]. Figure 7.22 depicts the approximation polynomial with degree 6 (red) along with the optimal feedback law (blue) in 3D.

Conclusions

The section presents the polynomial approximation approach to explicit MPC which aims at reducing the complexity of explicit solutions in MPC. The proposed alternative solution takes form of the polynomial (7.52) which has similar properties as the exact optimal solution but it is cheap to implement. Concretely, the most significant is the decrease in memory requirements which is evident in Fig. 7.21. The scheme uses the concept of stability tubes to find sets of possibly all stabilizing controllers. Subsequently, the controller

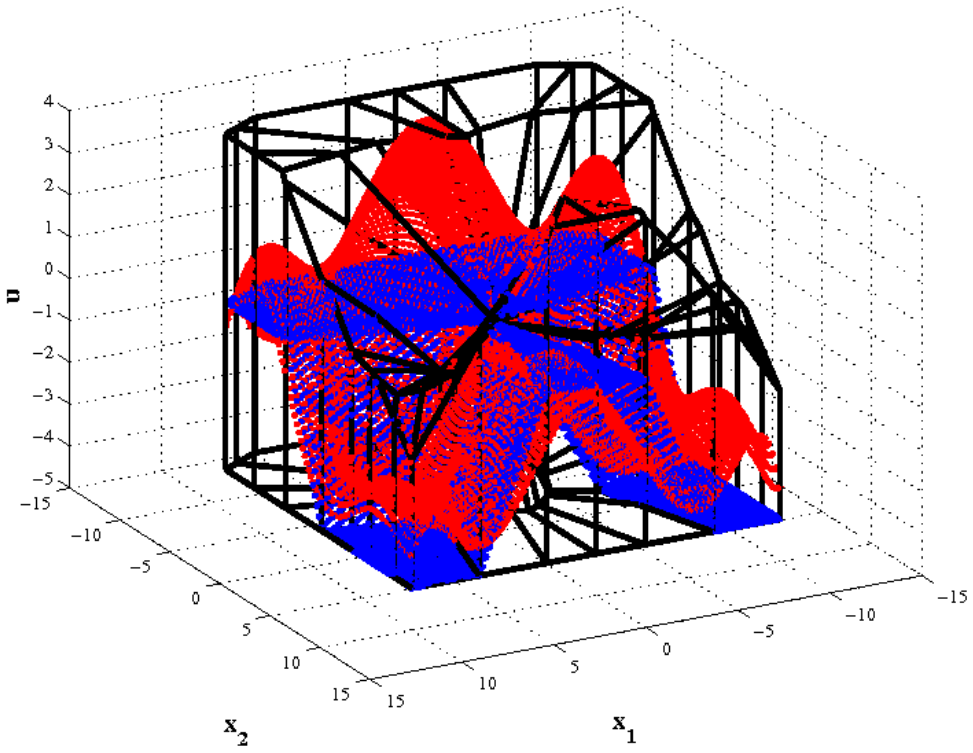


Figure 7.22: 3D representation of the closed-form PWA control law (blue) and corresponding polynomial approximation (red) with degree 6.

is selected from this set which has the least deviation from the optimal one by solving the approximation problem (7.61). The experimental validation of the scheme will be presented in Part III.

Part III

APPLICATIONS

Chapter 8

Servo Engine

Servo engine represents one of the typical elements frequently used in the industry. Basically, it comprises of an inductance motor which is used to transform the input electrical energy into a into a mechanical load e.g. for flow control. Due to the presence of mechanical elements, the servo engine is affected by friction forces and exhibits so called *deadzone* effect. Deadzone effect can be viewed as an uncontrollable mode of a plant, where there's no response on plant outputs to inputs applied within the deadzone limits. If the effect of the deadzone is not directly considered in the control design, it may cause unwanted performance loss and may lead to chattering control around the deadzone limits. Generally these properties are neglected in the control design and treated rather afterwards, as an implementation issue [113]. If the controller is synthesized in this way, control actions actually applied to a plant may differ from the calculated ones and this ideal assumption may lead to seriously degraded performance or even instability [30, 108].
The initial research around the deadzone was propagated by [91] and continued by [102, 103]. The solution in these cases relies on adaptive control with construction of so called deadzone-inverse. Similar idea is employed in other control approaches, involving artificial neural network [60], fuzzy logic [26] or model predictive control (MPC) [22, 30, 113]. It will be shown in the sequel that the deadzone can naturally be modeled using PWA models and that such models are suitable for design of control policies which take the deadzone behavior into account. Consequently, the time optimal control problem will be formulated and solved using multiparametric methods. Furthermore, the explicit solution will be applied in real time, and experimental results are provided. The results in this chapter have been published in [52].

8.1 Physical Setup

The laboratory servo engine is a mechanical device which represents a valve opening mechanism. It consist of a rotational flywheel attached to an electric engine. The manipulated variable is a voltage ranging from $-15\,\mathrm{V}$ to $+15\,\mathrm{V}$ generated by an actuator, which drives the rotation of the flywheel counterclockwise (when negative voltage is applied) or in the opposite direction (positive voltage). The angular velocity of the flywheel is the output sig-

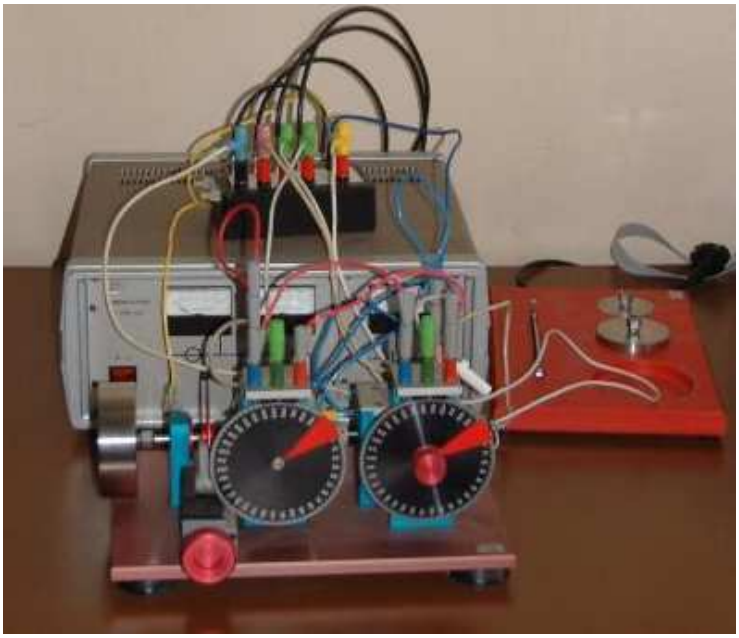


Figure 8.1: Front view of the laboratory servo engine.

nal which can be measured. The flywheel is connected to a circular pointer which indicates the position of the valve. The position of the wheel is the secondary output which ranges from the position "fully open" (0 degrees) to "fully closed" (360 degrees). A front view of the servo engine is illustrated in Fig. 8.1. Notice the two circular pointers with red triangles in the foreground of the two discs. The left one is the position indicator and the right one represents the setpoint which can be manipulated manually. Under the left pointer there is a magnetic brake whose force can be adjusted by turning the red colored switch. The brake is considered to be an external disturbance. The laboratory servo engine is connected to a personal computer via three devices: an actuator, a transducer and the connector CP1102. The devices actually convert the operating range of the dSPACE input-output card ($-10\,\mathrm{V}$, $+10\,\mathrm{V}$) to the actuator voltage range ($-15\,\mathrm{V}$, $+15\,\mathrm{V}$). MATLAB's Real Time Workshop (RTW) serves as a tool for implementing the control policy.

8.2 Hybrid Model and Experimental Validation

This section reviews the modeling issues of the deadzone. In the first part it is investigated how the deadzone is present in the mathematical model and how this phenomenon is traditionally neglected in control design. Secondly, a hybrid model based on experimental identification is presented and its validation with real plant is provided.
Construction of the plant model including deadzone has been studied by [30, 102] and the authors compose the controlled model by interconnection of a static nonlinearity block (representing deadzone) and a linear model, as shown in Fig. 8.2. Signals u, y represent

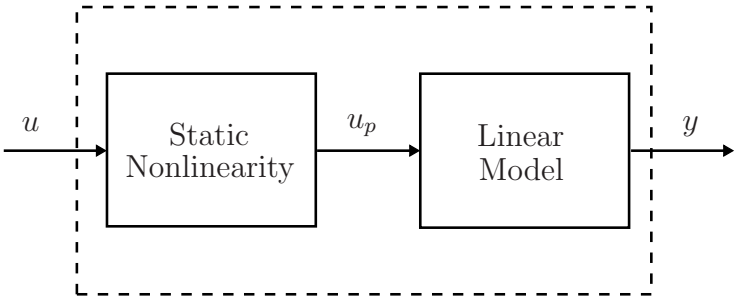


Figure 8.2: Standard model of a plant with deadzone.

plant's inputs, outputs, respectively, and u_p is the control input actually transmitted to the plant. The output u_p from a deadzone block is given by

$$u_p = \begin{cases} m_d(u-c) & \text{if } u > c \\ 0 & \text{if } -c \leq u \leq c \\ m_d(u+c) & \text{if } u < -c \end{cases} \tag{8.1}$$

where, m_d denotes the slope and c the break points of the deadzone, respectively. The deadzone characteristics (8.1) is illustrated in Fig. 8.3. Control desing and analysis of the

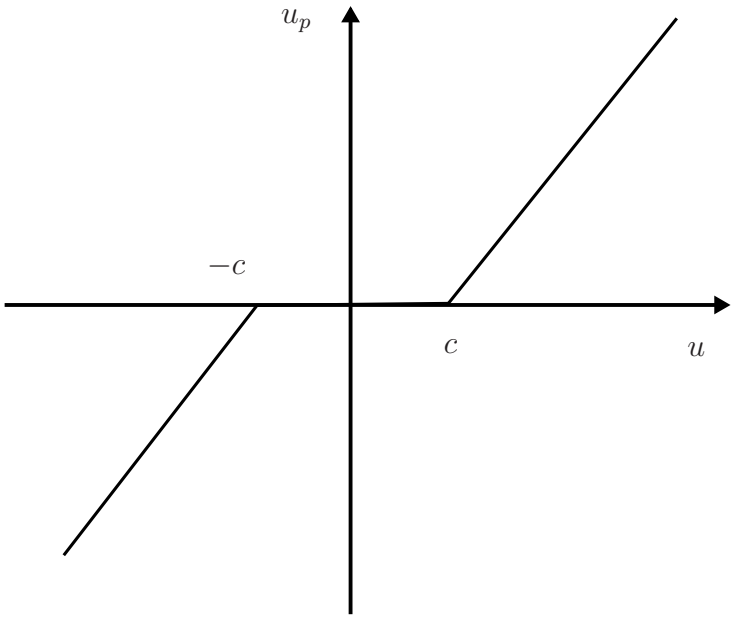


Figure 8.3: Theoretical model of a deadzone.

closed-loop behavior is difficult with the use of deadzone model (8.1), because for a real-time implementation it requires compensation via deadzone inverse approach. In order to avoid efforts involved in deadzone inverse approach, [73] proposed to model the plant

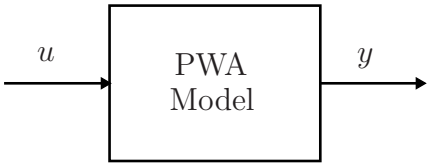


Figure 8.4: PWA model naturally incorporates the deadzone nonlinearity.

with deadzone via PWA approach where the related analysis turns to be a relatively simple task. The underlying idea is to incorporate the deadzone characteristics directly into linear model by considering the switching conditions in (8.1) as internal modes of PWA model. Such model now inherently relates the plant inputs u directly to outputs y (Fig. 8.4) and its immediate advantage is that control theory for hybrid systems can be applied for synthesis and analysis.
In the sequel a measurement of the deadzone is provided and based on the experimental data, an appropriate PWA model of the servo engine is constructed.

8.2.1 Deadzone Measurement

Experimental measurement proved the presence of the deadzone with symmetrical limits $c = \pm 1.3\,\mathrm{V}$. Interestingly, the measurement confirmed the specificity of the deadzone. In particular, this effect is different for the acceleration phase and for the braking phase, as shown in Fig. 8.5. Values in Fig. 8.5 were collected measuring the velocity in the steady state with respect to applied input. This behavior can be explained by friction forces which influence the rotation. In the acceleration phase, the deadzone is caused by the initial load to be conquered, and in the braking phase, by the friction. What is important to notice, is that the boundaries of the deadzone are varying. Moreover, if the flywheel starts up with the turned brake, deadzone limits are enlarged to $c = \pm 1.5\,\mathrm{V}$. Thus, to consider the whole possible widths of the deadzone in the feedback control, the worst case realization is adopted, hence $c = \pm 1.5\,\mathrm{V}$. Fig. 8.5 also indicates the hybrid nature of the servo engine, in the sense of differentiating between two different modes: accelerating and braking mode. It is then worth to describe such behavior using a hybrid model.

8.2.2 PWA Model

The motivation to use PWA system to model the deadzone effect is in a natural separation principle between two operating modes. In the first mode the plant operates normally, and in the second mode the deadzone is active. In this approach, as suggested by [73, 74], these modes are distinguished by polyhedral partitions of the state space, defined as a set of linear inequalities and in the dynamics is linear each mode. However, in this case the model of the deadzone is not obvious, due to its specific type. To be able to exactly capture its characteristics, one would require to partition the state space with respect to information about the measured velocity at the current time v_k, at the previous time v_{k-1}, and with

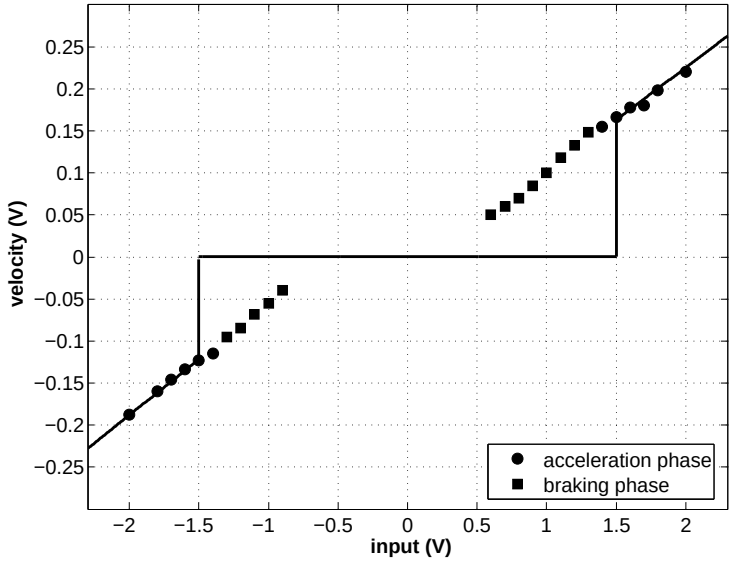


Figure 8.5: Experimental measurement of the deadzone clearly indicates two phases and varying boundaries.

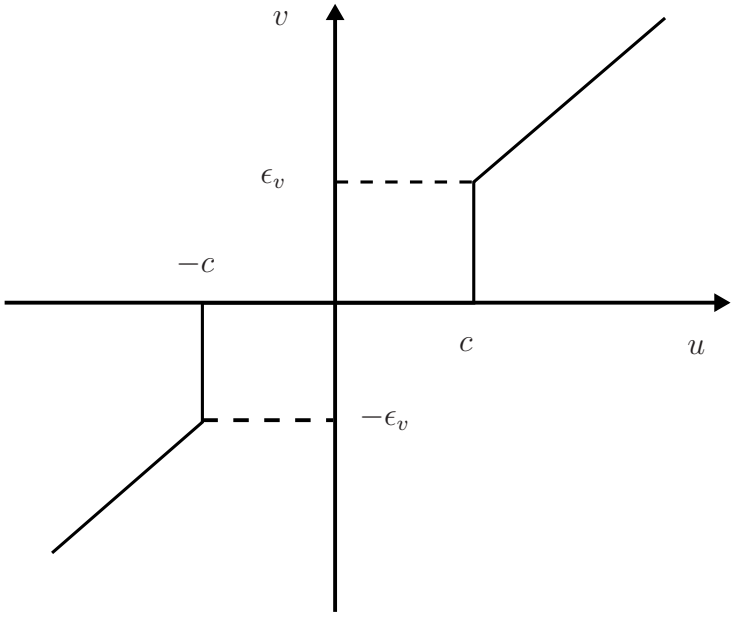


Figure 8.6: The deadzone is modeled as the PWA system with help of speed tolerance ϵ_v.

respect to current input u_k. This would lead to PWA model defined in the extended state space, for which the parametric MPC approach would seem computationally expensive. These problems can be avoided by introducing a tolerance on the velocity ϵ_v, which will specify the boundary of the deadzone. In fact, a zone of steady state will be defined, where the deadzone forbids any movement and thus this region is inherently invariant. Objective of the control design then becomes to drive the engine to this zone, because here the position should be equal to its reference. By this way the sensitivity of the controller can be adjusted, since the position is coupled by integral from the velocity. Hence, the deadzone can be straightforwardly modeled as PWA system with three modes:

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } |v_k| \geq \epsilon_v & \text{(braking phase)} \\ \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } |v_k| < \epsilon_v \text{ and } |u_k| \geq c & \text{(acceleration phase)} \\ \boldsymbol{A}\boldsymbol{x}_k & \text{if } |v_k| < \epsilon_v \text{ and } |u_k| < c & \text{(steady state)} \end{cases} \tag{8.2}$$

Here c and ϵ_v denote the breakpoints of the deadzone. The state vector $\boldsymbol{x}_k = (v_k,\ y_k)^T$ is composed of the measured angular velocity of the flywheel and its position, respectively. The numerical values of the matrices describing the state-space dynamics $\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k$ were obtained in [49] using experimental identification. The response of the velocity has been identified as a first-order system

$$T_v\dot{v}(t) + v(t) = Ku(t), \tag{8.3}$$

with the time constant $T_v = 6.7057\,\mathrm{s}$ and the gain $K = 0.1271$. Position is added to the model (8.3) as an additional state which is an integral of the velocity and values of $\boldsymbol{A}$ and $\boldsymbol{B}$ are then calculated by discretizing the model with sampling time $T_s = 0.7\,\mathrm{s}$. Graphically is the model of the deadzone illustrated in Fig. 8.6 and formally it is captured by the PWA model (8.2). Note that the tolerance ϵ_v does not have to be necessarily equal the value from Fig. 8.5, since it represents steady state value, but it can be used as a tuning factor to adjusts the boundaries of steady state (invariant) zone, i.e. to specify the sensitivity of the controller. The value of $\epsilon_v = 0.01\,\mathrm{V}$ was chosen.

Furthermore, notice that the PWA description (8.2) is defined over a non-convex domain. In order to convert the model into a form where each mode is valid over a convex region, the non-convex domains $|v_k| \geq \epsilon_v$ and $|u_k| \geq c$ each have to be split in two parts. The resulting PWA model is then given by following 5 modes:

$$\boldsymbol{x}_{k+1} = \begin{cases} \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } v_k \geq \epsilon_v \\ \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } v_k \leq \epsilon_v \\ \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } -\epsilon_v \leq v_k \leq \epsilon_v \quad \wedge\ u_k \geq c \\ \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}u_k & \text{if } -\epsilon_v \leq v_k \leq \epsilon_v \quad \wedge\ u_k \leq -c \\ \boldsymbol{A}\boldsymbol{x}_k & \text{if } -\epsilon_v \leq v_k \leq \epsilon_v \quad \wedge\ -c \leq u_k \leq c. \end{cases} \tag{8.4}$$

Furthermore, the manipulated voltage is constrained by hard limits of the input-output card in the interval $-10\,\mathrm{V} \leq u_k \leq 10\,\mathrm{V}$. Similarly, state constraints restrain the variables

to operate within intervals $-0.5\,\mathrm{V} \leq v_k \leq 0.5\,\mathrm{V}$ for velocity and $0\,\mathrm{V} \leq y_k \leq 1.6\,\mathrm{V}$ for position. Expressing the constraints in the form of (9.9) and combining with modes of the deadzone (8.4) the PWA model takes a compact form of (4.12).

8.2.3 Experimental Validation

A test scenario was used to validate PWA model (8.4) versus measured data. By imposing a known control profile, output from the model was compared with the data collected from the device. As can be seen from Fig. 8.7, the model approximates the real behavior of the plant with sufficient accuracy. Successful modelling of the deadzone is clearly seen from the first 6 seconds of the comparison. Since voltage inputs ranging from $-1\,\mathrm{V}$ to $+1\,\mathrm{V}$ have been applied during that period, the deadzone forbids the flywheel to start moving. This is correctly captured by the model. Once the input voltage exceeds deadzone threshold, the model correctly switches to a different operating mode where the input starts to influence the angular velocity of the flywheel. Even though some visible discrepancies are obvious in Fig. 8.7 no further improvement of PWA model is needed, since the deadzone effect is captured. Basically, the plant/model mismatch will be treated using MPC feedback control policy, as outlined in the next section.

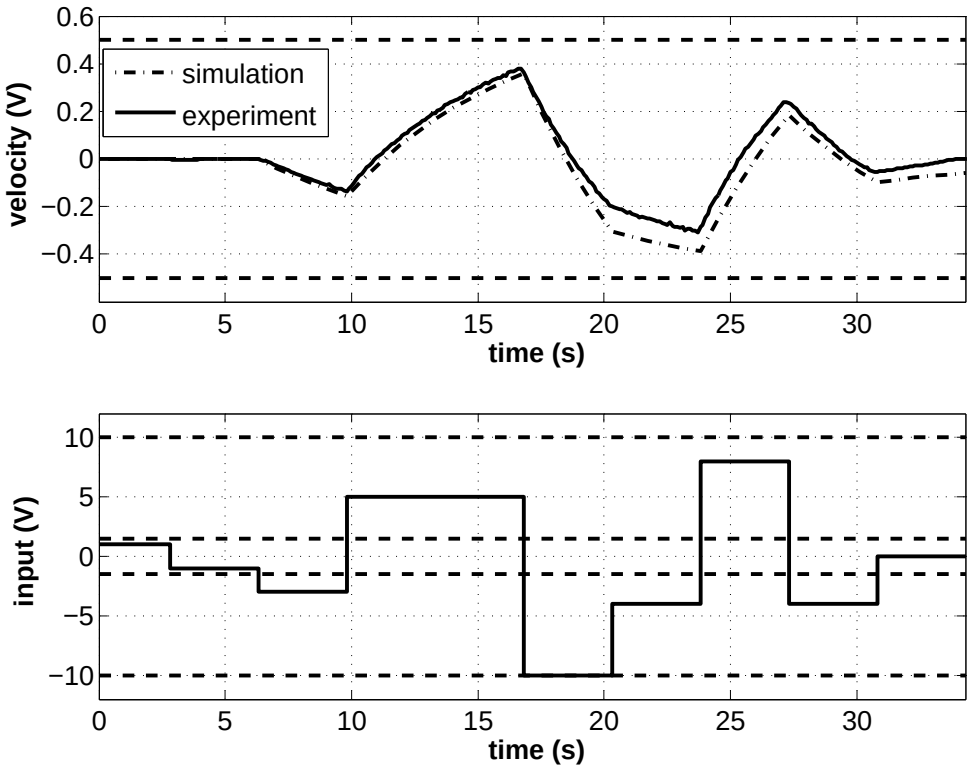


Figure 8.7: Comparison between the model behavior and measured signals.

8.3 Real-Time Implementation

In this section the time optimal tracking of a varying reference approach, presented in Section 7.1, is implemented in real-time on the servo engine. The objective is to minimize the number of steps needed to reach the time-varying reference signal while avoiding the control to be chattering around deadzone. Furthermore, state and input constraints must be satisfied for all time.
Since PWA model already contains one invariant mode (steady state), this mode can be used as the stabilizing terminal set with specified tolerances. Despite this, one would like to use PWA model to design the stabilizing target set instead. Notice, that every time, as the reference changes, the initial conditions become non-zero, and before the states reach the target set, the servo engine will be always in the braking phase. Thus, the terminal set based can be designed on the local model in the braking phase, or on the overall PWA model (8.4) according to the scheme presented in section 7.1.3. The terminal set was calculated for $N = 2$, which means that the reference will be reached in 2 time steps. Secondly, by applying the algorithm of [88] the invariant set Ω_{inv} was obtained.
Starting from this set the time optimal controller was calculated using Algorithm 7.1 in 11 iterations as a look-up table consisting of 622 regions in the three dimensional state space. Using MPT toolbox the controller was subsequently exported to C code and compiled with the help of RTW to dSPACE platform. The closed loop control using time optimal MPC approach was measured and the data are shown in Fig. 8.8. The deadzone effect is effectively compensated with help of the PWA model and whenever a setpoint change is made, controller generates inputs greater than the deadzone limits. This is obvious around 140s when even despite small change in reference, the tracking goal is attained. Moreover, closed loop behavior is not chattering around the deadzone limits. One can notice that input and state constraints are respected for the whole time which is especially important for valves, which operate under hard constraint.

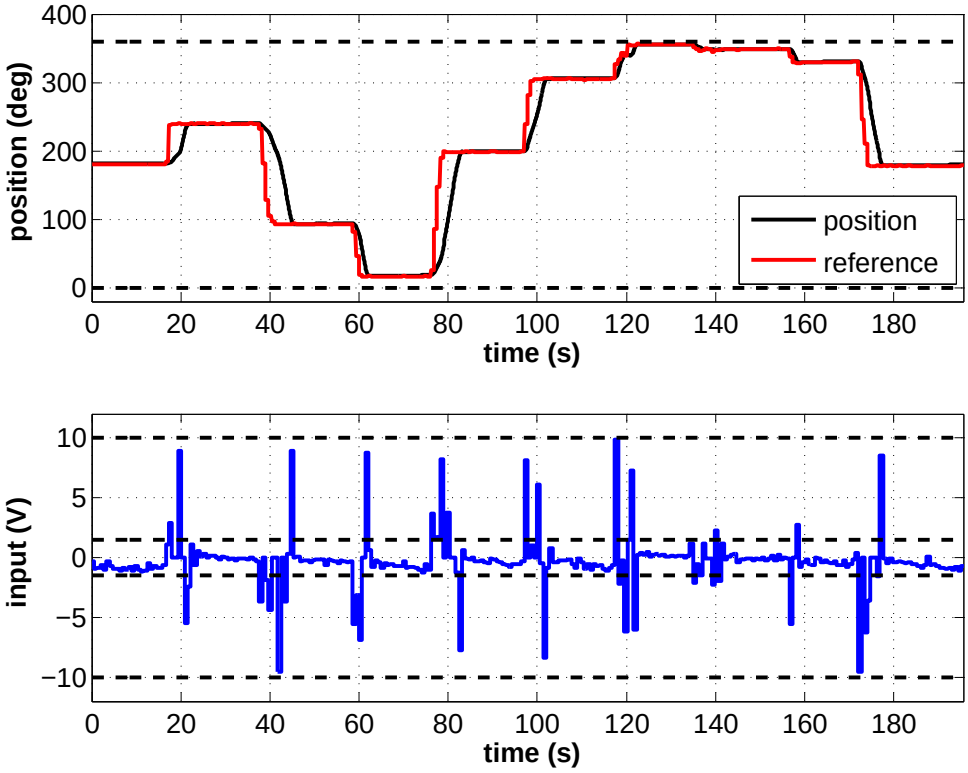


Figure 8.8: Position tracking using the time optimal approach.

Chapter 9

Thermo-Optical Device

This chapter presents a real-time control of a thermo-optical device using the polynomial approximation scheme presented in Section 7.4. The scheme is applied to obtain an alternative solution to the explicit MPC controller. Such an approximate controller enjoys the key benefits of MPC schemes, namely it provides all-time constraint satisfaction and closed-loop stability guarantees. The main advantage of the proposed approximation scheme is that it can be implemented in real time using very limited computational resources. The results in this chapter have been published in [53].

9.1 Device Description

The uDAQ28/LT thermal-optical system is an experimental device aimed primarily for education purposes [54]. The device allows real time measurement and control of temperature and light intensity. It can be connected to a personal computer via an universal serial bus without requiring an input-output card (Fig. 9.1). Data acquisition and real-time control of the uDAQ28/LT device is carried out in the Matlab/Simulink environment which allows very easy manipulation with the device.
The plant represents a dynamical system which combines slow and fast dynamics. The slow process is characterized by a heat transfer and the fast process corresponds to light emission. Both processes are caused by an embedded light bulb which is controlled by an input voltage signal. In general, the plant is characterized by five inputs and eight outputs whereas only three controlled inputs and three measured outputs are of interest. A precise description of these signals is given in Tab. 9.1.
The construction of the device suggests two main control loops. The primal loop regulates the light bulb intensity by manipulating the input voltage (or input voltage to LED[1] diode). The second loop maintains the inner temperature in safety limits by manipulating the revolutions of a cooling fan. Presence of physical constraints on manipulated and controlled variables makes the control task challenging and the device has often been used for benchmark of constrained PID control approaches [55].

[1] Light Emitting Diode

Figure 9.1: Front view on a thermo-optical device uDAQ28/LT.

Table 9.1: Description of measured and controlled signals.

Signal Name	Range
Input voltage to light bulb	0-5 V
Input voltage to cooling fan	0-5 V
Input voltage to LED	0-5 V
Inner temperature	0-100 deg C
Light intensity	not given
Revolutions of the cooling fan	0-6000 rpm

9.2 Identification and PWA Modelling

In the sequel, only the optical channel of the light-bulb is considered. This decision is motivated by the fact that this channel is represented by a fast dynamics, which makes real-time implementation of a control system a challenging task. Due to very fast responses of the light channel, the sampling rate was selected the lowest admissible by Windows, i.e. $T_s = 0.05\,\mathrm{s}$. As the optical channel is sampled, it immediately suggests identification of input-output relations in discrete time.
Input-output relations of the optical channel have been identified with the help of IDTOOL Toolbox [34] as a second order discrete transfer function

$$G(z^{-1}) = \frac{bz^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{9.1}$$

where b, a_1, a_2 are constant parameters and z^{-1} is a discrete time delay operator [72]. IDTOOL toolbox contains the recursive least squares method of [62] which provides very good estimates of the unknown parameters. However, as transfer function is valid only locally, the identification was performed over four operating points and the results are summarized in Tab. 9.2. For the use in explicit MPC scheme, the input-output representation (9.1) is

Table 9.2: Identification data over four operation points.

	input	output	b	a_1	a_2
(1)	1.3	6.84	2.03	-1.07	0.46
(2)	2.5	19.46	3.56	-0.97	0.43
(3)	3.5	32.09	4.51	-0.91	0.41
(4)	4.5	45.86	5.39	-0.87	0.40

transformed to a discrete state-space model. It is achieved by introducing state variables with discrete time instant k, i.e. $v_{1,k} = y_{k-1}$, $v_{2,k} = y_{k-2}$ and the state space model reads

$$v_{1,k+1} = -a_1 v_{1,k} - a_2 v_{2,k} + b w_k \tag{9.2a}$$
$$v_{2,k+1} = v_{1,k} \tag{9.2b}$$
$$y_k = v_{2,k}. \tag{9.2c}$$

In (9.2) w_k represents the input voltage applied directly to the plant and y_k is the measured output. Voltage input is constrained

$$w_k \in [0, 5]\,\mathrm{V} \tag{9.3}$$

and the measured output lies inside the interval

$$y_k \in [0, 55] \tag{9.4}$$

of light intensity units (are not given in the reference manual). The overall input-output behavior of the optical channel can be recovered by aggregation of the local linear models

(9.2) which forms PWA model. Here, the operating area is first split into regions and local linear models are assigned to each such region. The overall behavior of PWA model is then driven by switching between the locally valid models using logical IF-THEN rules. To perform partitioning of the operating area according to linearization points in Tab. 9.2, a Voronoi diagram [2] is constructed, which directly returns partitions of the state space as a sequence of convex polytopes. This operation was executed using one of the routines included in MPT toolbox [64] and it returned following regions:

$$\mathcal{D}_1 = \{v_k \in \mathbb{R}^2 \mid 0 \leq v_{2,k} < 13.15\} \tag{9.5a}$$
$$\mathcal{D}_2 = \{v_k \in \mathbb{R}^2 \mid 13.15 \leq v_{2,k} < 25.77\} \tag{9.5b}$$
$$\mathcal{D}_3 = \{v_k \in \mathbb{R}^2 \mid 25.77 \leq v_{2,k} < 38.97\} \tag{9.5c}$$
$$\mathcal{D}_4 = \{v_k \in \mathbb{R}^2 \mid 38.97 \leq v_{2,k} < 55\} \tag{9.5d}$$

To each of the regions (9.5), a corresponding local linear dynamics (9.2) is assigned, and it forms overall PWA model.

The output from PWA model has been compared to the real measured output from the plant and the result is depicted in Fig. 9.2. For the given scenario PWA model follows correctly the plant's output, thus the accuracy of the model is verified. It can be noticed that at the beginning there is larger mismatch between the plant and the model. It is caused by physical properties of a filament in bulb which requires certain time to incandesce from a cold startup. As this phase is over, PWA model correctly captures the optical channel of the plant and it can be employed for MPC design.

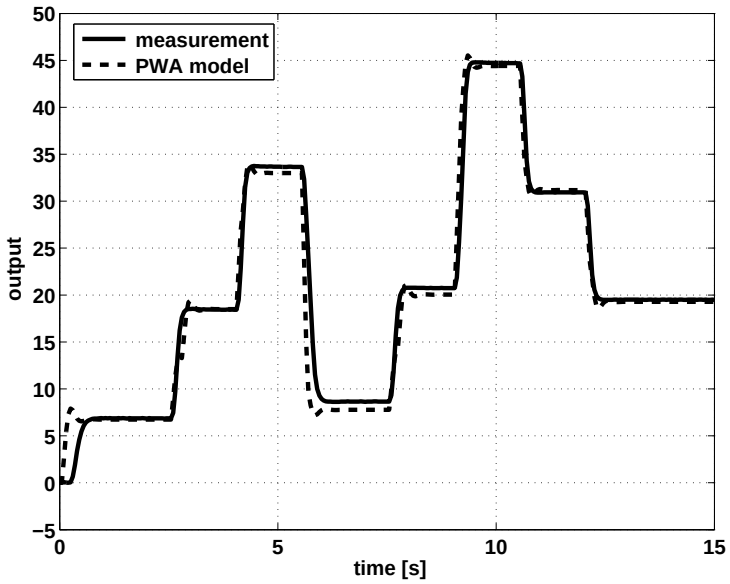


Figure 9.2: Verification of PWA model.

Table 9.3: Matrices of the normalized model (9.8).

$\boldsymbol{A}_1$	$\boldsymbol{B}_1$	$\boldsymbol{c}_1$
$\boldsymbol{A}_2$	$\boldsymbol{B}_2$	$\boldsymbol{c}_2$
$\boldsymbol{A}_3$	$\boldsymbol{B}_3$	$\boldsymbol{c}_3$
$\boldsymbol{A}_4$	$\boldsymbol{B}_4$	$\boldsymbol{c}_4$

=

1.072	-0.464	0.277	-1.492
1	0	0	0
0.969	-0.431	0.485	-0.642
1	0	0	0
0.913	-0.410	0.616	0
1	0	0	0
0.868	-0.402	0.735	0.471
1	0	0	0

9.3 Control Design

9.3.1 Prediction Model

In order to prevent numerical issues when employing the PWA model for MPC synthesis, it is advised to perform coordinate transformation and normalization. This can be achieved by introducing normalized variables x_1, x_2 and u as follows:

$$x_{1,k+1} = \frac{v_{1,k} - v_{1,\text{ref}}}{\bar{v}_1}, \tag{9.6a}$$

$$x_{2,k+1} = \frac{v_{2,k} - v_{2,\text{ref}}}{\bar{v}_2}, \tag{9.6b}$$

$$u_k = \frac{w_k - w_{\text{ref}}}{\bar{w}}. \tag{9.6c}$$

The suffix "ref" represent the desired steady state value, i.e.

$$v_{1,\text{ref}} = 32.09, \quad v_{2,\text{ref}} = 32.09, \quad w_{\text{ref}} = 3.5 \tag{9.7}$$

which is basically the linearization point of the third dynamics (see Tab. 9.2) and $\bar{v}_1 = 3.67$, $\bar{v}_2 = 3.67$, $\bar{v} = 0.5$ are constants. Applying the normalization, the transformed PWA model yields

$$\boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, u_k) = \boldsymbol{A}_i \boldsymbol{x}_k + \boldsymbol{B}_i u_k + \boldsymbol{c}_i \tag{9.8}$$

where $i = 1, 2, 3, 4$ and state update matrices are given in Tab. 9.3. The state space model (9.8) is associated with the following regions

$$\mathcal{D}_1 = \{\boldsymbol{x}_k \in \mathbb{R}^2 \mid -8.75 \leq x_{2,k} \leq -5.16\} \tag{9.9a}$$

$$\mathcal{D}_2 = \{\boldsymbol{x}_k \in \mathbb{R}^2 \mid -5.16 \leq x_{2,k} \leq -1.72\} \tag{9.9b}$$

$$\mathcal{D}_3 = \{\boldsymbol{x}_k \in \mathbb{R}^2 \mid -1.72 \leq x_{2,k} \leq 1.88\} \tag{9.9c}$$

$$\mathcal{D}_4 = \{\boldsymbol{x}_k \in \mathbb{R}^2 \mid 1.88 \leq x_{2,k} \leq 6.25\} \tag{9.9d}$$

Besides the dynamics as in (9.8), the following constraints are assumed to be imposed on the behavior of the prediction model:

$$\mathcal{X} = \{\boldsymbol{x}_k \in \mathbb{R}^2 \mid -8.75 \leq x_{1,k} \leq 6.25, \ -8.75 \leq x_{2,k} \leq 6.25\} \tag{9.10a}$$

$$\mathcal{U} = \{u_k \in \mathbb{R} \mid -7 \leq u_k \leq 3\}. \tag{9.10b}$$

State constraints $\mathcal{X}$ are derived from the operating range of light intensity (9.4) and input constraints $\mathcal{U}$ represent the saturation limits (9.3).

9.3.2 Control Problem

The aim of the control strategy is to find an optimal sequence of control inputs such that all system states are driven to a desired equilibrium. The equilibrium is given by the linearization point for the third PWA dynamics (9.8) and in the transformed coordinates (9.6) it is exactly the origin, i.e. $x_{1,k} = 0$, $x_{2,k} = 0$, $u_k = 0$. Mathematically, the problem can be formulated as to find a sequence of future control moves $\boldsymbol{U} = (u_0, u_1, \ldots, u_{N-1})^T$ up to horizon $N \in \mathbb{N}^+$ which steer the system states/input to the origin while satisfying constraints (9.10). More precisely,

$$\min_{\boldsymbol{U}} \sum_{k=0}^{\infty} \|\boldsymbol{Q}\boldsymbol{x}_k\|_1 + \|Ru_k\|_1 \tag{9.11a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{PWA}}(\boldsymbol{x}_k, u_k) \tag{9.11b}$$

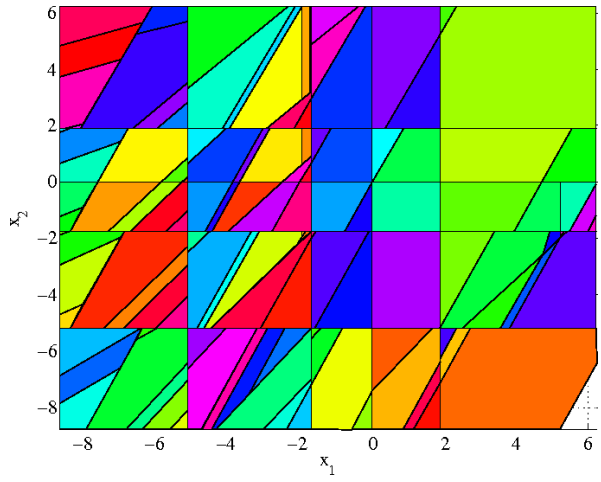
$$\boldsymbol{x}_k \in \mathcal{X} \tag{9.11c}$$

$$u_k \in \mathcal{U} \tag{9.11d}$$

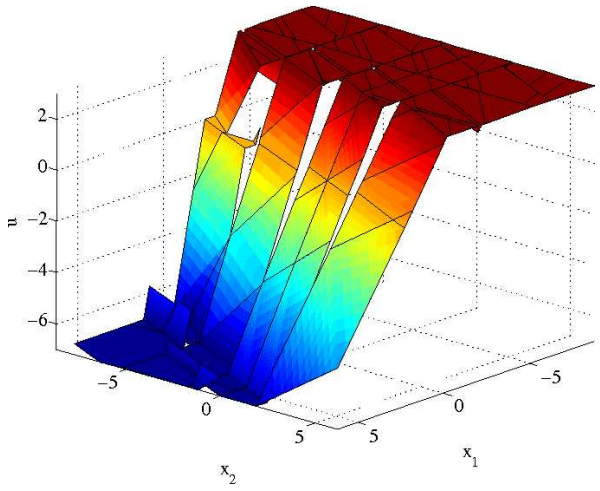
where $\boldsymbol{x}_k = (x_{1,k}, x_{2,k})^T$ represents the state vector, the function $\boldsymbol{f}_{\text{PWA}}(\cdot)$ describes the PWA model defined in (9.8) and the sets $\mathcal{X}$, $\mathcal{U}$ are the constraints on input and state variables given by (9.10). Due to the presence of switching rules in PWA model (9.8), the overall optimization problem (9.11) is cast using additional binary variables as mpMILP. The problem is consequently solved using MPT toolbox [64].

9.3.3 Explicit Solution

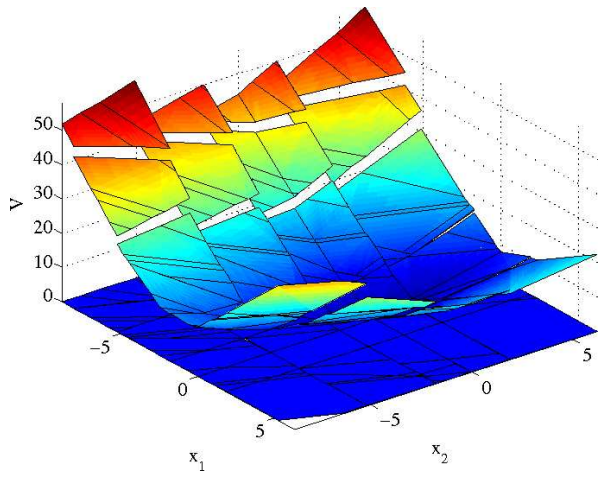
Solving problem (9.11) in a multiparametric fashion a closed form solution u_k as PWA function which maps $\boldsymbol{x}_k$ onto $\mathcal{U}$. In particular, as was shown by [23], it is given by PWA function $u = \boldsymbol{F}_i\boldsymbol{x} + \boldsymbol{G}_i$ if $\boldsymbol{x} \in \mathcal{P}_i$ for $i = 1, \ldots, n_{\mathcal{P}}$. Here, $\mathcal{P}_i = \{\boldsymbol{x} \mid \boldsymbol{H}_i\boldsymbol{x} \leq \boldsymbol{l}_i\}$ are polytopic regions of the state-space. Similarly, a closed-form expression for the optimal cost function (9.11a) is again PWA function of the state, i.e. $V = \boldsymbol{\Phi}_i\boldsymbol{x} + \Gamma_i$ if $\boldsymbol{x} \in \mathcal{P}_i$.
The problem (9.11) has been solved with parameters $\boldsymbol{Q} = \boldsymbol{I}$, $R = 0.5$. The infinite choice of prediction horizon guarantees that the obtained MPC feedback law will provide closed-loop stability [5]. The resulting PWA control law builds a look-up table divided into 118 regions, defined in variables x_1, x_2, and these regions are plotted in Fig. 9.3(a). Over each one of these regions a local feedback law is defined as illustrates Fig. 9.3(b). Similarly, the cost function is shown in Fig. 9.3(c). Note that in the case of multiparametric MILP solutions, the resulting PWA control law can be discontinuous (Fig. 9.3(b)) and defined over a nonconvex set. This is a consequence of using binary variables to encode the IF-THEN rules which describe behavior of the PWA prediction model.
To implement the resulting look-up table in the on-line experiment, one has to store and evaluate the data. While storing part is limited by the available memory, the evaluation



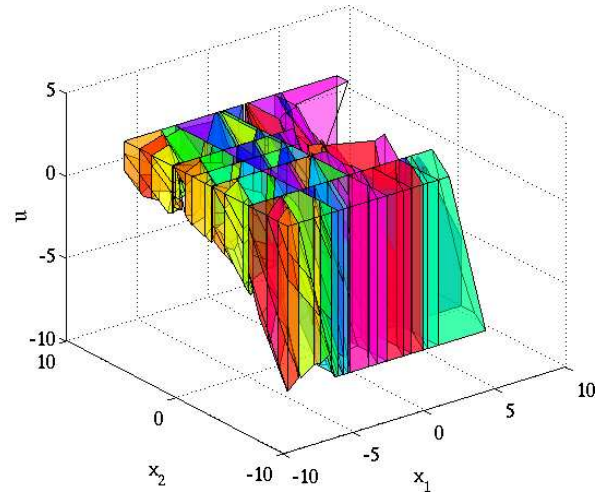
(a) Regions of the look-up table.



(b) Local control laws over each region.



(c) Value function.



(d) Stability tubes.

Figure 9.3: Explicit solution to Problem (9.11) consists of PWA map defined over 118 regions.

task is limited by the sampling time. The complexity of both tasks depend on the number of regions $n_{\mathcal{P}}$. Even with the use of binary search tree algorithm, where the evaluation time is logarithmic in $n_{\mathcal{P}}$ [106], the scheme can still be prohibitive for implementation. Motivated by this fact, the goal is to apply the approximation scheme presented in section 7.4 where the whole look-up table is replaced by one polynomial, which is very cheap to implement. To do so, one has to find the set of all perturbations of the control law under which the closed loop renders stability. This will be explained in the next section.

9.3.4 Polynomial Approximation

Using the approximation scheme (Section 7.4) the goal is to find a polynomial control law of the form

$$\mu(\boldsymbol{x})_3 = (a_{11},\, a_{12}) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (a_{21},\, a_{22}) \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} + (a_{31},\, a_{32}) \begin{pmatrix} x_1^3 \\ x_2^3 \end{pmatrix}$$

which, when applied as a state feedback, guarantees closed-loop stability and constraint satisfaction. The choice for the polynomial (9.12) is driven by needs to have the low memory consumption and fast evaluation as explained in Section 7.4.2.
Theorem 7.5 provides a sufficient condition for existence of such a polynomial feedback law in the sense that if $(\boldsymbol{x}^T, \mu(\boldsymbol{x})_3)^T \in \mathcal{S}(V, \beta),\ \forall \boldsymbol{x} \in \bigcup_i \mathcal{P}_i$, then $\mu(\boldsymbol{x})_3$ will provide closed-loop stability and constraint satisfaction. Therefore the search for suitable polynomial coefficients of (9.12) can be cast as the following optimization problem:

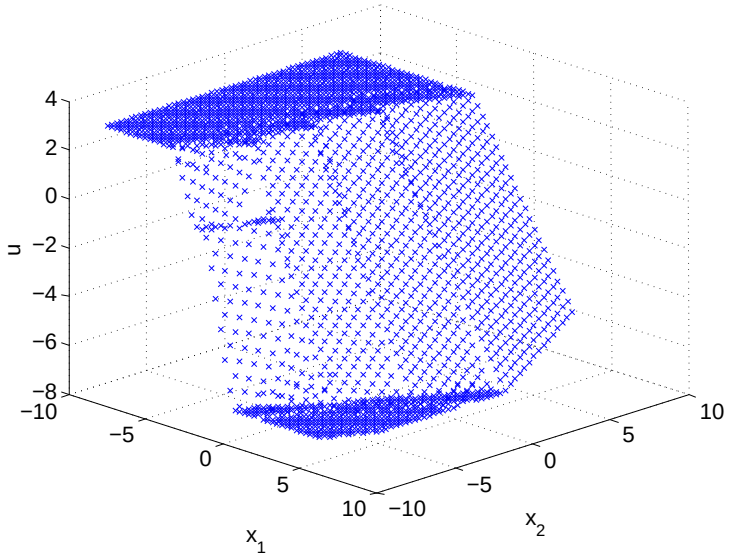
$$\min_{a_{11},\ldots,a_{32}} \sum_j \|(u(\boldsymbol{x}) - \mu(\boldsymbol{x}))\|_2 \tag{9.12a}$$

$$\text{s.t.} \quad \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{\mu}(\boldsymbol{x})_3 \end{pmatrix} \in \mathcal{S}(V, \beta). \tag{9.12b}$$

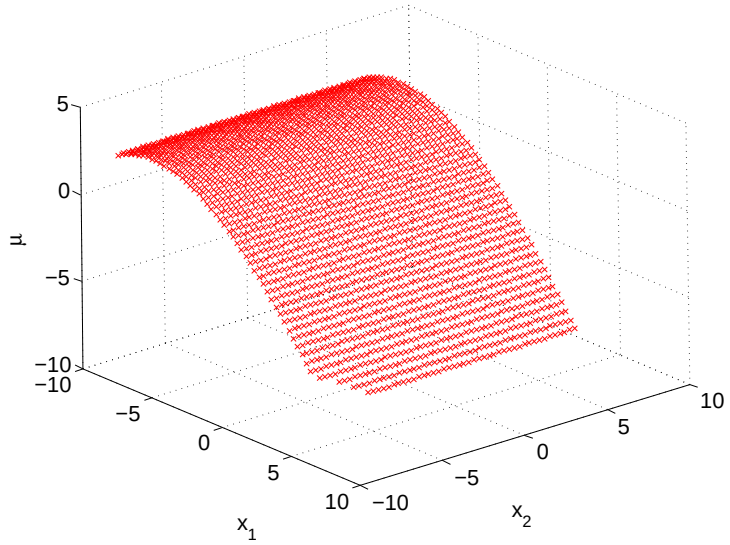
From all possible choices of $\mu(\boldsymbol{x})_3$ which satisfy (9.12b), cost function (9.12a) is used to select the coefficients which provide best approximation of the optimal feedback law $u(\boldsymbol{x})$. As was shown in section 7.4, optimization problem (9.12) can be formulated as a SOS problem, which can be solved using off-the-shelf tools.
The approximation scheme has been applied to obtain polynomial control law of type (9.12) with the help of YALMIP [67]. Computed coefficients are given in Tab. 9.4. It should be noted that the polynomial control law (9.12) has been searched for different degrees $r = \{2, 3, 4, 5\}$ and due to implementation simplicity, the polynomial with degree $r = 3$ has been selected. Graphical representation of the computed polynomial of order 3 is shown in Fig. 9.4(b). To visibly see the differences comparing to optimal controller (shown in Fig. 9.4(a)), a cross-section through $x_2 = 0$ is provided in Fig. 9.5. Illustration of the approximation scheme is shown in Fig. 9.5 which represents a cross-section in stability tubes along the coordinate $x_2 = 0$. The polyhedral sets in Fig. 9.5 demonstrate the space of the stability tubes where there exists a stabilizing control law according to Theorem 7.5. Inside this space the approximated polynomial (9.12) has been fitted and it is shown in Fig. 9.5 with a dashed line while the optimal control law is depicted with solid line.

Table 9.4: Coefficients of the approximated polynomial (9.12).

a_{11}, a_{12}	-0.8718, -0.0007
a_{21}, a_{22}	-0.0519, 0.0004
a_{31}, a_{32}	0.0019, 0.0001



(a) Optimal control law.



(b) Polynomial with order 3.

Figure 9.4: Optimal control law and polynomial approximation.

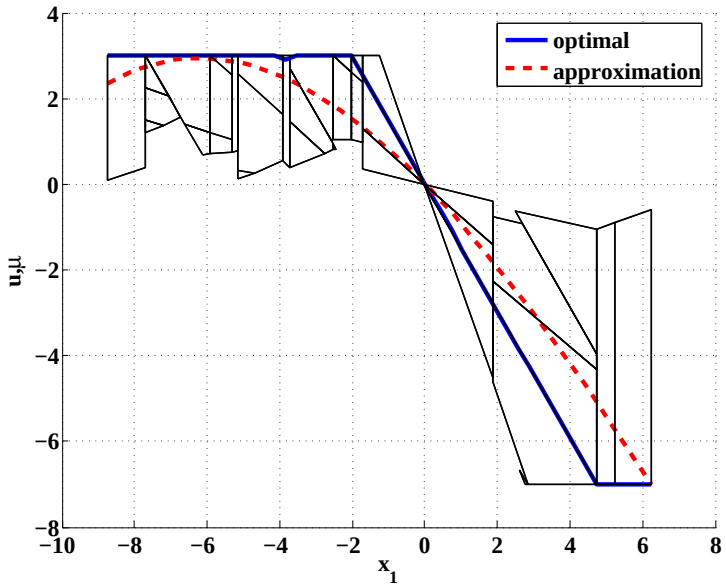


Figure 9.5: Cross-section of the control laws through $x_2 = 0$.

9.4 Real-Time Implementation

In this section computational requirements are evaluated for the optimal and approximated controller. Both controllers are applied in the real-time experiment and measured performance is discussed.

9.4.1 Computational Demands

Implementation of the optimal controller in the on-line experiment is limited by the sampling time $T_s = 0.05\,\mathrm{s}$. If the look-up table, obtained previously and consisting of 118 regions, is stored and evaluated using the binary search tree algorithm [105], the number of FLOPS which are required to evaluate such a controller for a given initial condition is at most 41. The memory requirements are 2832 bytes for the control law and 1536 bytes for the search tree which gives a total of 4368 bytes.

In the polynomial approximation scheme, the number of FLOPS depend on the degree of approximated polynomial and on the polynomial degree. By considering the polynomial (9.12) with degree of three, the upper bound for evaluation FLOPS is 14, less than a half of the runtime for the binary search tree. More prominent, however, is the drop in memory consumption. As state above, the explicit MPC solution with 118 regions requires 4368 bytes of memory storage, while to store the polynomial feedback law (9.12), mere 24 bytes of memory are required (6 polynomial coefficients, each of them consuming 4 bytes when represented as floating point numbers).

9.4.2 Experimental Data

The optimal explicit MPC controller as well as the polynomial feedback strategy have been implemented in real time and obtained results are shown in Figs. 9.7(a), 9.7(b) and 9.6. The plots represent the transition from the initial condition $\boldsymbol{x}_0 = (-8.7, -8.7)^T$ to the origin. Input signal generated by the optimal controller immediately jumps to the upper limit and then gently approaches the origin. In the polynomial controller this effect is different, the controller is slightly slower, but the same stabilizing effect is achieved. State and input profiles converge to desired steady state, hence the control objective was met with both approaches. It is interesting to note that a polynomial controller acts better (in the sense of the selected performance criterion (9.11a)) than the optimal one. In particular, (9.11a) evaluates to 146.34 when the optimal MPC controller is used as a feedback, compared to value of (9.11a) amounting to 142.96 for the case where the polynomial controller was used. This small difference can be attributed to the fact that the optimal controller is more sensitive to changes of the states. Nevertheless, the difference is small enough to say that both controllers share roughly the same performance while the approximated controller is significantly cheaper than the optimal one.

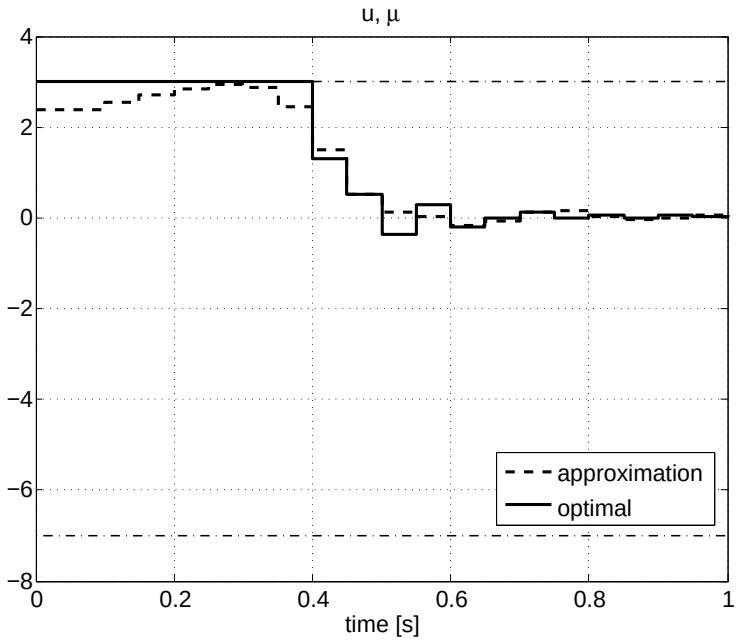
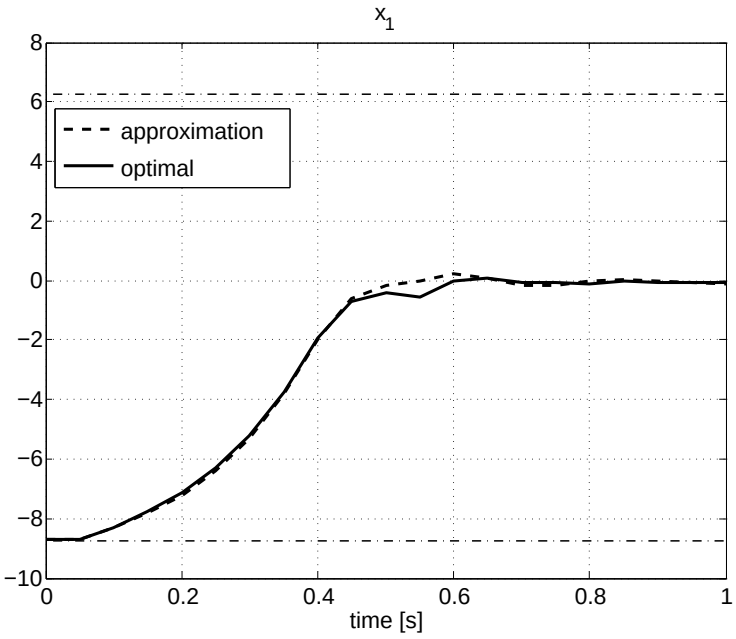
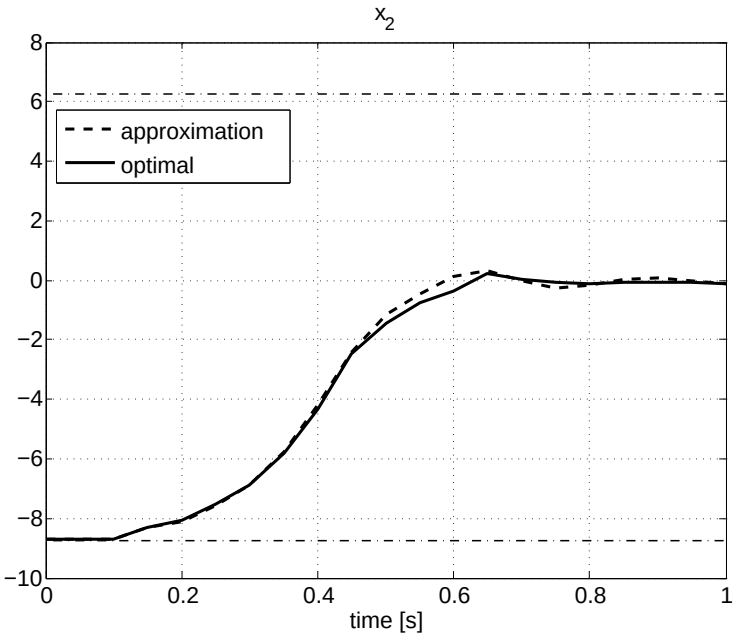


Figure 9.6: Input profiles for optimal and polynomial controller.



(a) Profiles of the state variable x_1.



(b) Profiles of the state variable x_2.

Figure 9.7: State profiles for optimal and polynomial controller.

Chapter 10

Conclusions

The thesis deals with the explicit MPC for processes on both theoretical as well as practical sides of the problematics. The essentials for understanding the concepts of MPC are explained in the Part I that also reviews the current state of the art in the field. Formulation of multiparametric problems are derived and methods for solving constrained control problems are presented. Part II is devoted to modeling and control of hybrid systems which builds the main part of the thesis. Some of the proposed approaches have been tested in the real-time experiments and the results are given in Part III. The concrete contributions of the thesis are summarized in the sequel.

The first main contribution is the software implementation for modeling of hybrid systems. The developed software HYSDEL 3.0 offers flexible modeling language which also supports the use of symbolic variables. The tool is thus suitable for advanced modeling and composition of systems in a user-friendly way. It allows automated generation of well-conditioned models from simple logical statements which has a strong application for the use in MPC, as well as for further analysis.

The second main result is the explicit solution for time optimal tracking of a varying reference for PWA systems. The approach proposes systematic algorithms for the design and the implementation phase which have been experimentally verified in the laboratory. The explicit MPC controller guarantees constraint satisfaction and ensures that the desired setpoint will be reached in the minimal number of steps. Obtained experimental results comply with theoretical background and show the suitability of the approach for practical problems.

The third main result, which has been experimentally tested, is the polynomial approximation approach to explicit MPC. This scheme offers suboptimal, but very cheap solutions for a real-time implementation. The approach is especially suitable for cases, when there is only limiting computational power available and the sampling time is very small, e.g. for control of DC-DC converters in power electronics.

The fourth main contribution of the thesis is the explicit solution to time optimal control for uncertain PWA systems and Takagi-Sugeno systems. The importance of the result is that uncertainties in the process model can be directly considered in the design phase which allows the explicit controller to be resistant against small model-plant mismatch

while respecting all design constraint. This result is significant for Takagi-Sugeno fuzzy systems where the multiparametric approach is applied for the first time. The time optimal control approach has been tested on various examples which shows the applicability in the real-time experiments.

Bibliography

[1] A. Antoniou and W.-S. Lu. *Practical Optimization: Algorithms and Engineeging Applications*. Springer Science+Business Media, LLC, New York, 2007.

[2] F. Aurenhammer. Voronoi diagramns – survey of a fundamental geometric data structure. *ACM Computing Surverys*, 3(23):345–405, 1991.

[3] M. Baotic. *Optimal Control of Piecewise Affine Systems – a Multi-parametric Approach*. PhD thesis, ETH Zurich, Switzerland, March 2005.

[4] M. Baotić, F. J. Christophersen, and M. Morari. A new algorithm for constrained finite time optimal control of hybrid systems with a linear peformance index. In *Proc. of the European Control Conference*, Cambridge, U.K., September 2003.

[5] M. Baotić, F. J. Christophersen, and M. Morari. Constrained optimal control of hybrid systems with a linear performance index. *IEEE Transactions on Automatic Control*, 51(12):1903–1919, December 2006.

[6] M. Barič, S. V. Raković, T. Besselman, and M. Morari. Max-min optimal control of constrained discrete-time systems. In *Proc. of the 17th IFAC World Congress*, pages 8803–8808, Seoul, South Korea, July 2008.

[7] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming, Theory and Algorithms*. Wiley-Interscience, 3rd edition, 2006.

[8] A. G. Beccuti, G. Papafotiou, and M. Morari. Optimal control of the buck dc-dc converter operating in both the continuous and discontinuous conduction regimes. In *Proc. of the Conf. on Decision & Control*, pages 6205–6210, San Diego, California, USA, December 2006.

[9] R. Bellman. *Dynamic Programming*. Princeton University Press, New York, 1957.

[10] A. Bemporad. Efficient conversion of mixed logical dynamics systems into an equivalent piecewise affine form. *IEEE Trans. on Aut. Control*, 49(5):832–838, 2004.

[11] A. Bemporad, F. Borrelli, and M. Morari. The explicit solution of constrained LP-based receding horizon control. In *Proc. of the 39th IEEE Conf. on Decision and Control*, pages 632–637, 2000.

[12] A. Bemporad, F. Borrelli, and M. Morari. The explicit solution of constrained LP-based receding horizon control. In *Proc. of the 39th IEEE Conf. on Decision and Control*, pages 1810–1815, 2000.

[13] A. Bemporad, F. Borrelli, and M. Morari. Optimal controllers for hybrid systems: Stability and piecewise linear explicit form. In *Proc. of the 39th IEEE Conference of Decision and Control*, pages 1810–1815, Sydney, Australia, 2000.

[14] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming – the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1975–1985, December 2002.

[15] A. Bemporad and M. Morari. Control of Systems Integrating Logic, Dynamics, and Constraints. *Automatica*, 35(3):407–427, march 1999.

[16] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[17] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 2004.

[18] T. Besselmann, J. Löfberg, and M. Morari. Explicit model predictive control for systems with linear parameter-varying state transition matrix. In *Proc. of the 17th IFAC World Congress*, pages 13163–13168, Seoul, South Korea, July 2008.

[19] W. L. Bialkowski. Dreams vs. reality: a view from both sides of the gap. In *Proc. of Control Systems Conference*, pages 283–294, Whistler, Canada, 1992.

[20] R. R. Bitmead, M. Gevers, and V. Wertz. *Adaptive optimal control – The thinking man's GPC*. Prentice-Hall, Englewook Cliffs, NJ, 1990.

[21] F. Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions. *IEEE Tran. on Aut. Control*, 39(2):428–433, Feb. 1994.

[22] V. Bobál, M. Kubalčík, P. Chalupa, and P. Dostál. Adaptive predictive control of nonlinear system with constraint of manipulated variable. In K. M. Hangos, editor, *Proc. of the IASTED International Conference Modelling, Identification and Control*, pages 349–354, Innsbruck, Austria, 2009.

[23] F. Borrelli. Constrained Optimal Control of Linear and Hybrid Systems. In *Lecture Notes in Control and Information Sciences*, volume 290. Springer, 2003.

[24] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[25] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer Verlag, 1st. edition, 1999.

[26] J. Campos and F.L. Lewis. Deadzone compensation in discrete time using adaptive fuzzy logic. In *Proc. of the 37th IEEE Conf. on Decision and Control*, pages 2920–2926, Tampa, FL, USA, December 1998.

[27] Y.-Y. Cao and P. M. Frank. Analysis and synthesis of nonlinear time-delay systems via fuzzy control approach. *IEEE Transactions on Fuzzy Systems*, 8(2):200–211, April 2000.

[28] H. Chen and F. Allgöwer. A quasi-infinity horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[29] W.-H. Chen, J. O'Reilly, and D. J. Ballance. On the terminal region of model predictive control for non-linear systems with input/state contraints. *International Journal of Adaptive Control and Signal Processing*, 17:195–207, 2003.

[30] C.-M. Chow and D. W. Clarke. Actuator nonlinearities in predictive control. In *Advances in Model-Based Predition Control, Conf. Proc.*, volume II, pages 79–91, Oxford, UK, Sept. 1993. Dep. of Eng. Sc., University of Oxford.

[31] F. J. Christophersen. *Optimal Control and Analysis for Constrained Piecewise Affine Systems*. PhD thesis, ETH Zurich, Switzerland, August 2006.

[32] F. J. Christophersen. Optimal Control of Constrained Piecewise Affine Systems. In *Lecture Notes in Control and Information Sciences*, volume 359. Springer-Verlag, 2007.

[33] F. J. Christophersen, M. Baotić, and M. Morari. Stability Analysis of Hybrid Systems with a Linear Performance Index. In *Proc. of the Conf. on Decision & Control*, pages 4589–4594, Atlantis, Paradise Island, Bahamas, December 2004. http://control.ee.ethz.ch/index.cgi?page=publications&action=details&id=1836.

[34] Ľ. Čirka, M. Fikar, and P. Petruš. IDTOOL 4.0 - A Dynamical System Identification Toolbox for MATLAB/Simulink. In *14th Annual Conference Proceedings: Technical Computing Prague 2006*, pages 29–29. The MathWorks, Inc. & HUMUSOFT s.r.o. & Institute of Chemical Technology in Prague, October 2006.

[35] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control – Parts I-II. *Automatica*, 23(2), 1987.

[36] C. R. Cutler and B. L. Ramaker. Dynamic matrix control – a computer control algorithm. In *Proceedings, Joint American Control Conference*, San Francisco, California, USA, 1980.

[37] E. de Klerk, C. Roos, and T. Terlaky. Nonlinear optimization, 2004. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.2146.

[38] S. L. de O. Kothare and M. Morari. Contractive Model Predictive Control for Constrained Nonlinear Systems. *IEEE Transactions on Automatic Control*, 45(6):1053–1071, June 2000.

[39] J. Eve. The Evaluation of Polynomials. *Numerische Mathematik*, 6:17–21, 1964.

[40] G. Feng. A Survey on Analysis and Design of Model-Based Fuzzy Control Systems. *IEEE Transactions on Fuzzy Systems*, 14(5):676–697, Oct. 2006.

[41] G. Feng. A Survey on Analysis and Design of Model-Based Fuzzy Control Systems. *IEEE Transactions on Fuzzy Systems*, 14(5):676–697, Oct. 2006.

[42] C. E. Garcia and M. Morari. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323, 1982.

[43] T. Geyer, G. Papafotiou, and M. Morari. Model Predictive Control in Power Electronics: A Hybrid Systems Approach. In *Proc. of the Conf. on Decision & Control*, Seville, Spain, December 2005.

[44] E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, 36(9):1008–1020, Sept. 1991.

[45] P. Grieder. *Efficient Computation of Feedback Controllers for Constrained Systems*. PhD thesis, ETH Zurich, Switzerland, October 2004.

[46] P. Grieder, M. Kvasnica, M. Baotic, and M. Morari. Stabilizing low complexity feedback control of constrained piecewise affine systems. *Automatica*, 41, issue 10:1683–1694, Oct. 2005.

[47] B. Grünbaum. *Convex Polytopes*. Springer-Verlag, 2nd edition, 2000.

[48] W. P. M. H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37:1085–1091, 2001.

[49] M. Herceg and M. Fikar. Constrained Predictive Control of Laboratory Servoengine. In *Proc. of 15th Int. Conference on Process Control 2005*, Štrbské Pleso, Slovakia, June 7-10 2005.

[50] M. Herceg, M. Kvasnica, and M. Fikar. Stabilizing Predictive Control of Fuzzy Systems Described by Takagi-Sugeno Models. In J. Mikleš, M. Fikar, and M. Kvasnica, editors, *Proceedings of the 16th Int. Conf. on Process Control*, page 223f.pdf. Slovak University of Technology in Bratislava, 2007.

[51] M. Herceg, M. Kvasnica, and M. Fikar. Transformation of Fuzzy Takagi-Sugeno Models into Piecewise Affine Models. In M. Kryszkiewicz, J.F. Peters, H. Rybinski, and A. Skowron, editors, *Rough Sets and Intelligent Systems Paradigms, Proceedings*, volume 4585 of *Lecture Notes in Computer Science*, pages 211–220, Warsaw, Poland, June 28-30 2007. Springer.

[52] M. Herceg, M. Kvasnica, and M. Fikar. Minimum-time predictive control of a servo engine with deadzone. *Control Engineering Practice*, 17(11):1349–1357, 2009.

[53] M. Herceg, M. Kvasnica, M. Fikar, and Ľ. Čirka. Real-time control of a thermo-optical device using polynomial approximation of mpc scheme. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 17th International Conference on Process Control '09*, pages 332–340, Štrbské Pleso, Slovakia, June 9– 12, 2009 2009. Slovak University of Technology in Bratislava. 082.pdf.

[54] M. Huba, P. Kurčík, and M. Kamenský. *Thermo-optical device uDAQ28/LT*. STU Bratislava, Illkovičova 3, Bratislava, 2006. In Slovak.

[55] M. Huba and D. Vrančič. Constrained control of the plant with two different modes. In J. Mikleš, M. Fikar, and M. Kvasnica, editors, *16th Int. Conf. Process Control*, pages paper Le–Tu–5, 205p.pdf, 2007.

[56] M. Johansson, A. Rantzer, and K.-E. Årzén. Piecewise Quadratic Stability of Fuzzy Systems. *IEEE Transactions on Fuzzy Systems*, 7(6):713–722, December 1999.

[57] C. N. Jones and M. Morari. Multiparametric Linear Complementarity Problems. In *IEEE Conference on Decision and Control*, pages 5687–5692, San Diego, USA, December 2006.

[58] A. Karas, B. Rohaľ-Ilkiv, and C. Belavý. *Praktické aspekty prediktívneho riadenia (Practical aspects of model predictive control)*. Slovak University of Technology Press, Slovenská e-akadémia n.o., Bratislava, Slovakia, 2007. In Slovak.

[59] S. S. Keerthi and E. G. Gilbert. Optimal, infinite horizon feedback laws for a general class of constrained discrete time systems: Stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57:265–293, 1988.

[60] T. Knohl and H. Unbehauen. Adaptive position control of electrohydraulic servo systems using ANN. *Mechatronics*, 10:127–143, 2000.

[61] I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4(4):317–367, 1998.

[62] R. Kulhavý and M. Kárný. Tracking of slowly varying parameters by directional forgetting. In *Proceedings of the 9th IFAC World Congres*, Budapest, Hungary, 1984.

[63] M. Kvasnica. *Efficient software tools for control and analysis of hybrid systems*. PhD thesis, ETH Zurich, Switzerland, February 2008.

[64] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-Parametric Toolbox (MPT). In *Hybrid Systems: Computation and Control*, pages 448–462, March 2004. http://control.ee.ethz.ch/~mpt.

[65] M. Lazar. *Model predictive control of hybrid systems: Stability and robustness*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.

[66] J. T. Linderoth and T. K. Ralphs. *Noncommercial software for mixed-integer linear programming*, pages 253–303. CRC Press Operations Research Series, 2005. http://www.lehigh.edu/~jtl3/papers/MILP04.pdf.

[67] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. http://control.ee.ethz.ch/~joloef/yalmip.php.

[68] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[69] D. Q. Mayne and H. Michalska. Receding Horizon Control of Nonlinear Systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, July 1990.

[70] D. Q. Mayne, J. B. Rawlings C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[71] H. Michalska and D. Q. Mayne. Robust Receding Horizon Control of Constrained Nonlinear Systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, Nov. 1993.

[72] J. Mikleš and M. Fikar. *Process Modelling, Identification, and Control*. Springer Verlag, Berlin Heidelberg, 2007.

[73] B. E. A. Milani. Ultimate boundedness sets for continuous-time linear systems with deadzone feedback controls. In *Proc. of the 44th IEEE Conf. on Decision and Control and European Control Conf.*, pages 6853–6858, Seville, Spain, December 2005.

[74] B. E. A. Milani and A. D. Coelho. Ultimate boundedness sets for discrete-time linear systems with deadzone feedback controls. In *Proc. of the 40th IEEE Conf. on Decision and Control*, pages 2163–2164, Orlando, FL, USA, December 2001.

[75] S. Mollov, R. Babuška, J. Abonyi, and H.B. Verbruggen. Effective Optimization for Fuzzy Model Predictive Control. *IEEE Transactions on Fuzzy Systems*, 12(5):661–675, Oct. 2004.

[76] S. Mollov, T. van den Boom, F. Cuesta, A. Ollero, and R. Babuška. Robust Stability Constraints for Fuzzy Model Predictive Control. *IEEE Transactions on Fuzzy Systems*, 10(1):50–64, Feb. 2002.

[77] M. Morari and J. H. Lee. Model predictive control: Past, present and future. In *6th International Symposium on Process Systems Engineering and 30th European Symposium on Computer Aided Process Engineering ESCAPE-7*, Trondheim, Norway, May 1997.

[78] V. Nevistic and J. A. Primbs. Constrained nonlinear optimal control: a converse HJB approach. Technical report, ETH Zürich, 1996. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.2763.

[79] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, 1999.

[80] P. Parillo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, Pasadena, California, USA, May 2000.

[81] P. A. Parrilo. Sums of squares of polynomials and their applications. In *ISSAC '04: Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 1–1, New York, NY, USA, 2004. ACM Press.

[82] P. A. Parrilo and S. Lall. Semidefinite programming relaxations and algebraic optimization in Control. *European Journal of Control*, 9(2-3), 2003.

[83] E. Pistikopoulos, M. Georgiadis, and V. Dua, editors. *Multi-Parametric Model-Based Control*, volume 2 of *Process Systems Engineering*. WILEY-VCH Verlag, 2007.

[84] E. Pistikopoulos, M. Georgiadis, and V. Dua, editors. *Multi-Parametric Programming*, volume 1 of *Process Systems Engineering*. WILEY-VCH Verlag, 2007.

[85] D. M. Prett and C. E. Garcia. *Fundamental Process Control*. Butterworths, Boston, 1988.

[86] J. A. Primbs, V. Nevistic, and J. Doyle. Nonlinear optimal control: A control lyapunov function and receding horizon perspective. *Asian Journal of Control*, 1(1):14–24, March 1999.

[87] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering Practice*, 11:733–764, 2003.

[88] S. V. Raković, 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 *Proceeding of the 43rd IEEE Conference on Decision and Control*, pages 1418–1423, Atlantis, Paradise Island, Bahamas, December 2004.

[89] S. V. Raković, E. C. Kerrigan, D. Q. Mayne, and K. I. Kouramas. Optimized robust control invariance for linear discrete–time systems: Theoretical foundations. *Automatica*, 43(5):831–841, 2007.

[90] J. R. Rawlings and K. R. Muske. The Stability of Constrained Receding Horizon Control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, Oct. 1993.

[91] D. Recker, P.V. Kokotović, D. Rhode, and J. Winkelman. Adaptive nonlinear control of systems containing a dead-zone. In *Proc. of the 30th IEEE Conf. on Decision and Control*, pages 2111–2115, 1991.

[92] A. Richards and J. How. Mixed-integer programming for control. In *American Control Conference*, pages 2676–2683, 2005.

[93] P. O. M. Scokaert, D. Q. Mayne, and J. B. Rawlings. Suboptimal Model Predictive Control (Feasibility Implies Stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, March 1999.

[94] E. D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, April 1981.

[95] G. Stengle. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry . *Math. Ann.*, 207:87–97, 1974.

[96] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, pages 625–653, Oct. 1999.

[97] R. Suard, J. Löfberg, P. Grieder, M. Kvasnica, and M. Morari. Efficient Computation of Controller Partitions in Multi-Parametric Programming. In *IEEE Conference on Decision and Control*, Bahamas, December 2004.

[98] T. Takagi and M. Sugeno. Fuzzy identications of fuzzy systems and its applications to modelling and control. *IEEE Trans. Systems Man and Cybernetics*, 15:116–132, 1985.

[99] K. Tanaka and M. Sano. Trajectory stabilization of a model car via fuzzy control. *Fuzzy Sets and Systems*, 70:155–170, 1995.

[100] K. Tanaka and H. O. Wang. *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*, chapter 14. John Wiley & Sons, Inc., 2001.

[101] K. Tanaka and H. O. Wang. T-S Fuzzy Model as Universal Approximator. *Fuzzy Control Systems Design and Analysis*, pages 277–289, 2002.

[102] G. Tao and P.V. Kokotović. Adaptive control of plants with unknown dead-zones. *IEEE Transactions on Automatic Control*, 39(1):59–68, Jan. 1994.

[103] G. Tao and P.V. Kokotović. Discrete-time adaptive control of systems with unknown deadzones. *International Journal of Control*, 61(1):1–17, 1995.

[104] K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 – a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11/12:545–581, 1999.

[105] P. Tøndel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489–497, 2003.

[106] P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of Piecewise Affine Control via Binary Search Tree. *Automatica*, 39(5):945–950, May 2003.

[107] F. D. Torrisi and A. Bemporad. HYSDEL – A Tool for Generating Computational Hybrid Models for Analysis and Synthesis Problems. *IEEE Transactions on Control Systems Technology*, 12(2):235–249, march 2004.

[108] T. T. C. Tsang and D. W. Clarke. Generalised predictive control with input constraints. *IEE Proc. Pt. D*, 6(135):451–460, 1988.

[109] http://www.elsevier.com/wps/find/journaldescription.cws_home/270/description#description.

[110] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 2nd edition, 2001. http://www.princeton.edu/~rvdb/LPbook/onlinebook.pdf.

[111] H. O. Wang, K. Tanaka, and M. F. Griffin. An Approach to Fuzzy Control of Nonlinear Systems: Stability and Design Issues. *IEEE Transactions on Fuzzy Systems*, 4(1), Feb. 1996.

[112] W.-J. Wang, Y.-J. Chen, and C.-H. Sun. Relaxed Stabilization Criteria for Discrete-Time T-S Fuzzy Control Systems Based on a Switching Fuzzy Model and Piecewise Lyapunov Function. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 37(3):551–559, June 2007.

[113] H. Zabiri and Y. Samyudia. A hybrid formulation and design of model predictive control for systems under actuator saturation and backlash. *Journal of Process Control*, 16:693–709, 2006.

[114] L. A. Zadeh. Fuzzy sets. *Inform. Control*, 8:338–353, 1965.

[115] M. Zeilinger, C. N. Jones, and M. Morari. Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization. In *Proc. of the 47th IEEE Conf. on Decision and Control*, pages 4718–4723, Cancun, Mexico, December 2008.

Publication List

Chapter or pages in book

1. **Herceg**, **M**., Kvasnica, M., and Fikar, M.: Parametric Approach to Nonlinear Model Predictive Control, In *Nonlinear Model Predictive Control*, Editor(s): Magni, L. and Raimondo, D. M. and Allgöwer, F., Springer Berlin / Heidelberg, pp. 381–389, 2009.

2. **Herceg**, **M**., Kvasnica, and M. Fikar, M.: Transformation of Fuzzy Takagi-Sugeno Models into Piecewise Affine Models. Editor(s): M. Kryszkiewicz, J. F. Peters, H. Rybinski, A. Skowron, In *Lecture Notes in Artificial Intelligence*, vol. 4585, *Proceedings of the International Conference on Rough Sets and Intelligent Systems Paradigms*, Springer, Warsaw, Poland, pp. 211–220, 2007.

3. **Herceg**, **M**. Fikar, M.: Constrained Predictive Control of a Laboratory Servoengine, In *Selected Topics in Modelling and Control*, Editor(s): Mikleš, J., Veselý, V., Slovak University of Technology Press, vol. 5, pp. 115–120, 2007.

Article in CC journal

1. **Herceg**, **M**., Kvasnica, M., Fikar, M.: Minimum-time predictive control of a servo engine with deadzone, *Control Engineering Practice*, 17(11):1349–1357, 2009.

Article in journal

1. **Herceg**, **M**., Kvasnica, M., Fikar, M., Čirka, Ľ.: Real-time Control of a Thermo-Optical Device Using Polynomial Approximation of MPC Scheme. *AT&P Journal Plus*, no. 2, pp. 36–42, 2009.

2. **Herceg**, **M**., Kvasnica, M., Čirka, Ľ., and Fikar, M.: Real-Time Predictive Control of a Servo Engine. *AT&P Journal*, no. Plus2, pp. 124–130, 2007.

3. **Herceg**, **M**., Kvasnica, M., and Fikar, M.: Stabilizing Predictive Control of Fuzzy Systems Described by Takagi-Sugeno Models. *AT&P Journal*, no. Plus2, pp. 105–111, 2007.

Article in conference proceedings

1. Kvasnica, M., **Herceg**, **M**., Čirka, Ľ, Fikar, M.: Robust Adaptive Minimum-Time Control of Piecewise Affine Systems, *48th IEEE Conference on Decision and Control*, Shanghai, China, 2009. Accepted.

2. Kvasnica, M., **Herceg**, **M**., Čirka, Ľ., and Fikar, M.: Time-Optimal Control of Takagi-Sugeno Fuzzy Systems. In *Proceedings of the 10th European Control Conference*, Budapest, Hungary, 2009.

3. **Herceg**, **M**., Kvasnica, M., Fikar, M., and Čirka, Ľ.: Real-Time Control of a Thermo-Optical Device Using Polynomial Approximation of MPC Scheme. Editor(s): Fikar, M., Kvasnica, M., In *Proceedings of the 17th International Conference on Process Control '09*, Slovak University of Technology in Bratislava, Štrbské Pleso, Slovakia, pp. 332–340,2009.

4. Kvasnica, M., **Herceg**, **M**., Čirka, Ľ., and Fikar, M.: Time Optimal Control of Fuzzy Systems: a Parametric Programming Approach. In *Proceedings of the 28th IASTED Conference on Modelling, Identification and Control*, pp. 640-805.pdf, 2009.

5. **Herceg**, **M**., Kvasnica, M., and Fikar, M.: Stabilization of an Inverted Pendulum via Fuzzy Explicit Predictive Control. In *Proceedings of the 8th International Scientific - Technical Conference Process Control 2008*, University of Pardubice, Kouty nad Desnou, Czech Republic, pp. C021_b-1–C021_b-5, 2008.

6. **Herceg**, **M**., Mikleš, J., Fikar, M., Kvasnica, M., and Čirka, Ľ.: Real-time 2DoF Control of a Quadruple Tank System with Integral Action. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control*, Seoul, Korea, pp. 8666–8671, 2008.

7. Kvasnica, M., Christophersen, F. J., **Herceg**, **M**., and Fikar, M.: Polynomial Approximation of Closed-form MPC for Piecewise Affine Systems. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control*, Seoul, Korea, pp. 3877–3882, 2008.

8. **Herceg**, **M**., Kvasnica, M., and Fikar, M.: Parametric Approach to Nonlinear Model Predictive Control. Editor(s): L. Magni, D. Raimondo, F. Allgöwer, In *International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control*, Pavia, Italy, pp. PI1-1–PI1-8, 2008.

9. Čirka, Ľ., Fikar, M., Kvasnica, M., and **Herceg**, **M**.: Experimental Identification – an Interactive Online Course. In *Proceedings of the 17th World Congress of the International Federation of Automatic Control*, Seoul, Korea, pp. 9812–9816, 2008.

10. Fikar, M.,Čirka, Ľ., **Herceg**, **M**., and Podmajerský, M.: E-learning in Course Operating Systems. Editor(s): M. Huba, In *Proceedings of the 9th International Conference Virtual University 2008*, E-academia Slovaca, fid000091.pdf, 2008.

11. Kvasnica, M., **Herceg**, **M**., Čirka, Ľ, and Fikar, M.: Adaptive Model Predictive Control of Piecewise Affine Systems. Editor(s): L. Magni, D. Raimondo, F. Allgöwer, In *International Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control*, Pavia, Italy, pp. PIV3-1–PIV3-8, 2008.

12. Mariethoz, S., **Herceg**, **M**., and Kvasnica, M.: Model Predictive Control of buck DC-DC converter with nonlinear inductor. In *Proceedings of the Eleventh IEEE Workshop on Control and Modeling for Power Electronics*, Zurich, Switzerland, pp. 1–8, 2008.

13. Kvasnica, M., **Herceg**, **M**., Čirka, Ľ., and Fikar, M.: Model Predictive Control of a CSTR: a Hybrid Modelling Approach. In *Proceedings of the 8th International Scientific - Technical Conference Process Control 2008*, University of Pardubice, Kouty nad Desnou, Czech Republic, pp. C021_a-1–C021_a-9, 2008.

14. **Herceg**, **M**., Kvasnica, M., Čirka, Ľ, and Fikar, M.: Real-time Predictive Control of a Servo Engine. Editor(s): J. Mikleš, M. Fikar, M. Kvasnica, In *Proceedings of the 16th International Conference Process Control '07*, Slovak University of Technology in Bratislava, pp. 222f.pdf, 2007.

15. **Herceg**, **M**., Kvasnica, M., and Fikar, M.: Stabilizing Predictive Control of Fuzzy Systems Described by Takagi-Sugeno Models. Editor(s): J. Mikleš, M. Fikar, M. Kvasnica, In *Proceedings of the 16th International Conference Process Control '07*, Slovak University of Technology in Bratislava, pp. 223f.pdf, 2007.

16. Čirka, Ľ., Fikar, M., Kvasnica, M., and **Herceg**, **M**.: New Features in Course on Experimental Identification. Editor(s): Huba, M., In *Proceedings of 8th International Conference Virtual University*, Bratislava, pp. 182–187, 2007.

17. Čirka, Ľ., Bakošová, M., Fikar, M., and **Herceg**, **M**.: Dynamic Simulations of Chemical Processes via the MATLAB Web Server. In *Proceedings of the 15th Annual Conference Technical Computing Prague 2007*, ČVUT Praha, pp. 34, 2007.

18. **Herceg**, **M**., Raff, T., Findeisen, R., and Allgöwer, F.: Nonlinear Model Predictive Control of a Turbocharged Diesel Engine. In *Proceedings of the 2006 IEEE International Conference on Control Applications*, Munich, Germany, pp. 2766–2771, 2006.

19. **Herceg**, M. and Fikar, M.: Constrained Predictive Control of a Laboratory Servoengine. Editor(s): M. Vitek, In *Proceedings of the international Interdisciplinary Student Competition and Conference*. Honeywell EMI conference and competition 2005, VUT Brno, Brno, Czech Republic, pp. 93–97, 2005.

Article in collection

1. Kvasnica, M., **Herceg**, **M**., and Fikar, M.: Polynomial Approximation of Polytopic Regions. In *Proceedings IAM 2007 - Workshop on Informatics, Automation and*

Mathematics, Editor(s): Fikar, M., Kolesárová, A., Bakošová, M., STU Press, pp. 59–63, 2007.

Diploma thesis

1. **Herceg**, **M**.: Nonlinear Model Predictive Control of a Diesel Engine with Exhaust Gas Recirculation and Variable Geometry Turbocharger. Diploma thesis IST-0021, University of Stuttgart, 2006.

Bachelor's thesis

1. **Herceg**, **M**.: Control of a Laboratory Fan Heater (in Slovak). Bachelor's thesis, OIRP FCHPT STU, Bratislava, 2004.

Curriculum Vitae

Martin Herceg

born on November 22, 1982 in Šaľa, Slovakia

09/2006–09/2009	Doctorate studies, Slovak University of Technology in Bratislava, Slovakia
11/2007–08/2008	Visiting researcher, Automatic Control Laboratory, ETH Zürich, Switzerland
11/2005–06/2006	Visiting student, Institute for Systems Theorie and Automatic Control, University of Stuttgart, Germany
09/2001–06/2006	Graduate studies, Slovak University of Technology in Bratislava, Slovakia
09/1993–06/2001	Grammar School, Šaľa, Slovakia