Diss. ETH No. 14520

Control and Estimation of Hybrid Systems with Mathematical Optimization

A dissertation submitted to the

Swiss Federal Institute of Technology (ETH)

Zurich

for the degree of

Dr. sc. techn.

presented by

Domenico Mignone

Dipl. El.-Ing. ETH

born on March 23, 1972

from Italy

accepted on the recommendation of

Prof. Dr. Manfred Morari, examiner

Dr. Jürg Tödtli, co-examiner

January, 2002

Preface

to boldly go
where no man has gone before

This work has been developed during my stay at the Automatic Control Laboratory of ETH Zürich, Switzerland. I had the unique opportunity to work with several outstanding scientists, from whom I learned to structure new thoughts, to search the answers where they are least expected and to be perseverant in pursuing a goal.

First of all I express my gratefulness to Prof. Manfred Morari, who gave me the opportunity to do a PhD thesis. I admire the illuminating inspiration of his ideas, his enthusiasm and his ability to motivate people. A very special thanks goes to Alberto Bemporad, who initiated the research on the topic, this thesis is about. His stimulating intuitions lay the foundations to several parts of this work. I'm very indebted to Giancarlo Ferrari Trecate, for the long and fruitful collaboration I had with him, which is also reflecting itself in many parts of this work. Many thanks go to Jürg Tödtli who accepted to be co-examiner and provided essential suggestions to improve this text. I express my big gratitude to the whole hybrid systems group at the Automatic Control Laboratory. It was a pleasure and an honour to work with Fabio Danilo Torrisi, Francesco Cuzzola, Francesco Borrelli, Tobias Geyer and Mato Baotic.

I sincerely thank the proof readers of my thesis for their patience, their valuable comments and suggestions. They are Janusz Milek, Francesco Cuzzola, Fabio Danilo Torrisi, Jan Ulrich, Giancarlo Ferrari Trecate and Franta Kraus.

Thanks also to all colleagues of the Automatic Control Laboratory for the nice atmosphere they contributed to create, for the intriguing discussions and for the exciting lunch breaks. In particular, I thank Eleonora Bonanomi, Cornelius Dorn, Jan Ulrich, Oliver Kaiser, Christian Frei, Konrad Stadler, Ron Pearson, Vaclav Cechticky, Mayuresh Kothare, Markus Kottmann, Marco Sanvido and Federica Piras. I would like to thank the administrative staff of the Automatic Control Laboratory including Esther Hagenow, Martine D'Emma, Danielle Couson, Alice Vyskocil and Myriam Lämmel.

The collaboration with researchers from other laboratories at ETH Zürich was instructive and fruitful. Especially I thank Samarjit Chakraborty, Philipp Kutter and Lukas Finschi. I express my acknowledgment to all people from institutions outside ETH, with whom I had the chance to discuss about my work, in particular Jacques Chapuis, Kazuro Tsuda, Eric Kerrigan, Tal Pasternak, Laurent El Ghaoui, Carsten Scherer, Nick Sahinidis and the ABB corporate research center team in Dättwil, Switzerland, with Eduardo Gallestey and Dario Castagnoli, as well as ABB power automation in Baden, Switzerland with Andrew Paice, Babak Mataji and Marc Antoine.

A big acknowledgment goes to all students, I worked with: Dario Castagnoli, Paolo Letizia, Matteo Spedicato, Marina Vasić, Natascia Monachino, Massimo Mannelli. The experimental results in this work have been made possible with the valuable help of Hanspeter Mössner, Ulrich Wenk, Martin Vogt and Daniel Wegmann.

My thoughts are also with Hannes Wichser, who left us too early. May his soul find a peaceful rest.

I would like to thank all my friends and relatives for the support, for the confidence and for the affection during the last years. Innanzitutto vorrei ringranziare Mami e Papi per avermi dato la possibilità di portare a termine questa impresa. Per quanto mi potrò allontanare e cercare una mia vita propria, voi sarete sempre parte di me, e ne sono fiero. A huge thanks goes to Gunesh and Nina for turning on a light that was off for so much time, and for the helping hand in many occasions. Another huge thanks goes to Naty (... the 3 PM mosquito ☺...) for the marvellous discussions, the innumerable dinners and for the really great time I had with you. Thanks to Tobi, Ele, Max, Roberto, Nicole and Thiban, who were beside me during this journey. All of you made my life richer and happier in these years.

Thanks also to the CHAutomatOS SOLA teams, to the Kondi instructors Markus Burri, Claudia Marth, Edgar Sieber, Mischa Senn, to the Mensa teams for the many "grossi Portione", to Cinzia, Damaris, Sabina and to my cousins Francesco and Miranda.

And last, I thank God for giving me the strength and the health to master the obstacles and to go down the way He planned for me.

Domenico Mignone

Abstract

Many practical systems consist of both continuous valued and discrete valued components. The term *hybrid systems* has been used to describe models, where both continuous and discrete aspects are considered within the same framework. This thesis presents theoretical results and experiments about the Mixed Logic Dynamical systems modelling framework for hybrid systems, introduced by Bemporad and Morari (1999a). The framework allows to model systems comprising interacting propositional logic, continuous dynamics and constraints. Besides their broad modelling capabilities, Mixed Logic Dynamical systems allow to systematically formulate and solve various analysis and synthesis problems using mathematical optimization techniques.

The contributions of this thesis are:

- We present systematic methods for the efficient and automatized derivation of models. Classical analytical approaches for the modelling of system components are compared to a geometrical approach. The proposed methods reduce the number of variables in the model.

- Two practical systems are modelled in the Mixed Logic Dynamical systems framework: a three tank laboratory system and the outflow units of a hydroelectric power plant. Both systems illustrate the typical modeling capabilities of Mixed Logic Dynamical systems.

- We report experiments with a Model Predictive Control scheme for Mixed Logic Dynamical systems. The controller is applied to the three tank laboratory model, a multi-input multi-output hybrid plant.

- We introduce a fault tolerant control scheme for hybrid systems comprising two stages:

  First, we present an approach for fault detection and state estimation of Mixed Logic Dynamical systems. It casts the fault detection problem as an estimation problem to be solved as a mixed integer continuous optimization.

  Second, we introduce strategies for the controller reconfiguration, to be used if the plant exhibits hardware redundancies. If a fault occurs, we show that the decision about the choice of alternative system components can be formulated within the same modeling framework.

  The fault detection and the reconfiguration algorithms are illustrated both with simulations and experiments.

- The mathematical problems arising for Mixed Logic Dynamical systems are NP-hard. However, the experience of this work shows that the worst case of computational burden is seldom necessary to achieve a solution. We present a heuristic rule, which speeds up the computations on average. It is a tree exploring strategy in the branch and bound algorithm, which exploits the structure of the optimization problems, as they arise for control and estimation of Mixed Logic Dynamical systems. We give a quantification of its complexity.

- The model class of Mixed Logic Dynamical systems is equivalent to the class of Piecewise Affine systems. In this thesis we present sufficient conditions for stability of Piecewise Affine systems. These methods rely on linear matrix inequalities in order to find piecewise quadratic and more general Lyapunov functions.

Zusammenfassung

Viele praktische Systeme bestehen sowohl aus wertekontinuierlichen, als auch aus wertediskreten Komponenten. Der Begriff *hybride Systeme* wurde eingeführt, um solche Systeme zu beschreiben, bei denen sowohl diskrete, als auch kontinuierliche Aspekte innerhalb desselben Ansatzes berücksichtigt werden. In dieser Dissertation werden theoretische und experimentelle Resultate für hybride Systeme vorgestellt, die mit dem Ansatz der "Mixed Logic Dynamical"-Systeme modelliert wurden. Dieser Ansatz wurde von Bemporad and Morari (1999a) eingeführt und erlaubt die Modellierung von logischen Komponenten, kontinuierlicher Dynamik und Einschränkungen. Mit dem "Mixed Logic Dynamical"-Systeme Ansatz können vielfältige Systemklassen modelliert werden. Zudem können verschiedene Analyse- und Synthese-Probleme formuliert und mittels mathematischer Optimierung gelöst werden.

Die Beiträge dieser Arbeit sind:

- Diese Arbeit präsentiert systematische Methoden zur effizienten und automatisierten Herleitung von Modellen. Klassische Verfahren werden mit einem geometrischen Verfahren verglichen. Die Methoden reduzieren die Anzahl der Systemvariablen.
- Es wird die Modellierung von zwei praktischen Anlagen mit dem Formalismus der "Mixed Logic Dynamical"-Systeme durchgeführt. Es handelt sich dabei um ein Drei-Tank Labormodell und um die Abfluss-Organe eines Laufwasserkraftwerkes. Beide Systeme illustrieren typische Modellierungseigenschaften dieses Ansatzes.

- Ein Ansatz zur Modellprädiktiven Regelung von hybriden Systemen wird in dieser Arbeit experimentell validiert. Dazu wird das Labormodell eines Drei-Tank Systems verwendet, ein hybrides Mehrgrössensystem.

- Ein Verfahren zur fehlertoleranten Regelung von hybriden Systemen wird eingeführt. Es besteht aus zwei Stufen:
  Einerseits wird ein Ansatz für die Fehlererkennung und Zustandsschätzung von "Mixed Logic Dynamical" Systeme vorgestellt. Die Fehlererkennung wird als Schätzungsproblem formuliert, das mittels gemischt ganzzahlig kontinuierlicher Optimierung gelöst werden kann.
  Andererseits beschreibt diese Arbeit verschiedene Strategien zur System-Rekonfiguration bei Anlagen mit redundanten Komponenten. Bei Auftreten von Fehlern, wird die Entscheidung über die Wahl alternativer Komponenten als Optimierung innerhalb desselben Ansatzes gelöst.
  Die Fehlererkennungs- und die Rekonfigurationsalgorithmen werden sowohl mit Simulationen, als auch experimentell validiert.

- Die mathematischen Probleme, die beim Ansatz der "Mixed Logic Dynamical"-Systeme entstehen, sind NP-hart. Die Erfahrung in dieser Arbeit zeigt jedoch, dass der Rechenaufwand häufig einen Umfang aufweist, der weit unter der maximalen Anzahl Rechenschritte liegt. Es wird eine heuristische Regel vorgestellt, welche die Rechenzeit im Durchschnitt verkürzt. Es handelt sich um eine Such-Strategie in einem "Branch and Bound" Algorithmus, welche die Problemstruktur ausnützt, wie sie bei Optimierungen für "Mixed Logic Dynamical"-Systeme auftritt.

- Die Modellklasse der "Mixed Logic Dynamical"-Systeme ist äquivalent zur Klasse der stückweise affinen Systeme. In dieser Arbeit werden hinreichende Bedingungen für die Stabilität von stückweise affinen Systeme vorgestellt. Diese Methoden beruhen auf lineare Matrixungleichungen, welche für die Bestimmung von stückweise quadratischen oder allgemeinere Lyapunov Funktionen benützt werden.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation for Hybrid Systems

Models in Engineering

One of the main goals of engineering is the application of scientific laws and natural resources for the benefit of mankind. To achieve this goal, the different disciplines of engineering are concerned with the development of processes and structures, the combination of available components, and the forecast of the system behaviour under certain influences. A common notion in virtually all engineering fields is the concept of a *model*. A model is an abstract, simplified representation of the real world, which has a sufficient degree of complexity to allow the description of the behaviour of interest. At the same time a model should exclude all features that are not relevant for the current investigations and might form an obstacle to an efficient solution of the specified tasks.

Each branch of science and engineering has developed its own formalisms for models that allow to tackle the problems of interest in the corresponding fields. In systems and control theory we are concerned with the manipulation of system variables to achieve goals like

stabilization, disturbance rejection or robustness. The key idea is the usage of a feedback structure, where the evolution of some monitored variables determines the current or future actions on the system. The concept of a model of a system is therefore traditionally associated with differential or difference equations describing those evolutions. Typically a model is either derived by application of physical laws governing the dynamics of the system, or by estimation from experimental data sequences collected for modelling purposes. Therefore, most of the control theory and tools have been developed for such systems, in particular for systems, whose evolution is described by smooth linear or nonlinear state transition functions, either in continuous time

$$\dot{x}(t) = f(x(t), u(t), t) \tag{1.1}$$

$$x(t) \in \mathbb{R}^n \qquad u(t) \in \mathbb{R}^m \qquad t \in \mathbb{R}_{\geq t_0} \qquad f : \mathbb{R}^{n+m+1} \to \mathbb{R}^n$$

or in discrete time

$$x(k+1) = g(x(k), u(k), k) \tag{1.2}$$

$$x(k) \in \mathbb{R}^n \qquad u(k) \in \mathbb{R}^m \qquad k \in \mathbb{N}_{\geq k_0} \qquad g : \mathbb{R}^{n+m+1} \to \mathbb{R}^n$$

Most practical systems can be studied with tools devised for continuous systems. However, in many applications the system to be analyzed, supervised or controlled comprises both parts described by discrete valued variables and parts described by continuous valued variables. The discrete valued parts of a system include all those components that can take on only a finite number of states. Their range is therefore not a continuum as for the states x and the inputs u in (1.1) and (1.2). Practical examples of such components include for instance on/off switches or valves, gears or speed selectors, evolutions dependent on if-then-else rules, and many more.

Several activities in computer science and digital circuit design dealt with modelling formalisms, where the system variables take on values in a finite set. Microprocessor based circuits and computer codes for control or supervision of technical systems have a quantized state space. Techniques devised for the analysis of computer codes in embedded systems

lead to formalisms for discrete event systems. The most common discrete counterparts to (1.1) and (1.2) are finite state machines or Petri-Nets.

While control engineers often neglected a systematic treatment of the discrete valued components, focusing on the continuous valued world, computer scientists did the opposite by working with models that were often inadequately describing the continuous valued parts. Indeed, an ad-hoc treatment of one or the other part might give satisfactory results in some cases, but a priori there is no good reason to do so, besides a hitherto lack of good modelling frameworks combining both paradigms. In the past, control of these systems was based on heuristic rules inferred from practical plant operation. Analysis and synthesis of these systems relied on common sense and practical knowledge of the plant. An extensive simulation phase was performed to make plausible the correctness of the algorithms.

The importance of systematically combining the continuous valued and the discrete valued components of a system has been recently recognized. Levis et al. (1987) point out the lack of good dynamically oriented models that take into account both types of variables and allow the application of control and analysis techniques. A case of a dramatic failure of a system partly due to the insufficient consideration of these interactions is mentioned in (van der Schaft and Schumacher, 1999): the failure of the European space mission Ariane-5 was attributed to a software error in the inertial reference system. The code had been ported from the Ariane-4 predecessor environment to Ariane-5, however the interactions of the computer code with the continuous dynamics of the new launcher were insufficiently considered (Lions, 1996).

Hybrid Systems

Recently, several research activities started dealing with *hybrid systems*, i.e. the class of systems that explicitly takes into account continuous valued and discrete valued variables as well as their interaction in one common framework, see Figure 1.1. We will intention-

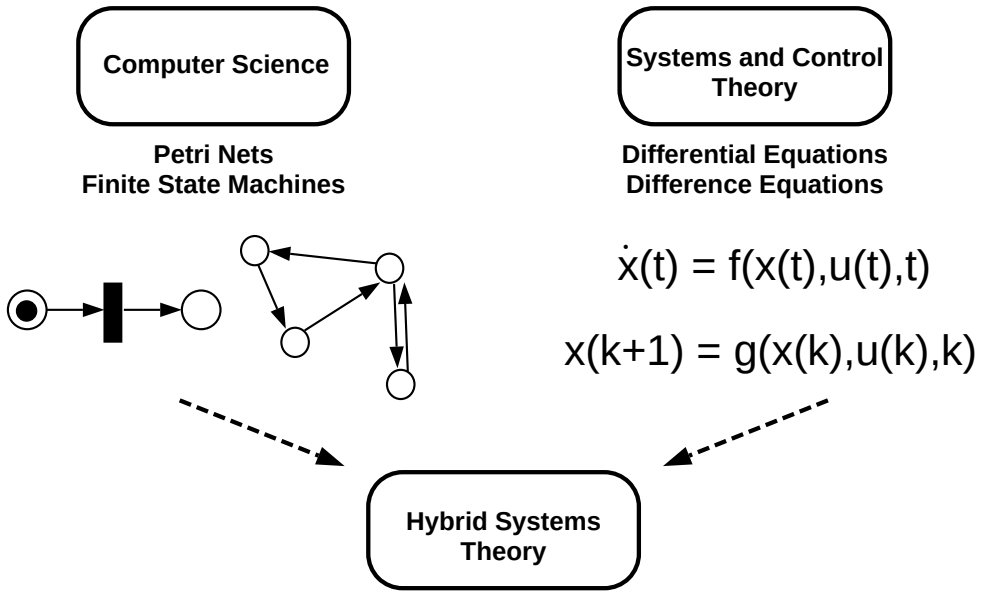


Figure 1.1: Hybrid systems

ally avoid to give a more precise definition of a hybrid system, because in view of the rich variety of system classes that have been proposed, a more formal definition would be too restrictive. The theory of hybrid systems aims at joining the contributions from continuous system theory with discrete event systems theory. The field of hybrid systems offers theoretical challenges to be tackled, and it is expected to have a significant impact on practical applications. The rise of interest in this class of systems can be tracked e.g. in the increasing number of sessions dedicated to hybrid systems at periodic international control conferences, like the American Control Conference (ACC) or the Conference on Decision and Control (CDC). Moreover, several meetings and workshops strictly dedicated to hybrid systems have taken place recently, some of them have already taken on a fixed repetition period, like the international workshop on "Hybrid Systems: Computation and Control (HSCC)", see (Henzinger and Sastry, 1998; Vaandrager and van Schuppen, 1999; Lynch and Krogh, 2000; Di Benedetto and Sangiovanni-Vincentelli, 2001), or the biennial conference "Automation des Processus Mixed (ADPM)" (Engell et al., 2000). Several international control journals have published special issues on hybrid systems, like Automatica

(Morse et al., 1999), System and Control Letters (Evans and Savkin, 1999), the IEEE Transactions on Automatic Control (Antsaklis and Nerode, 1998). Other special issues of control journals have appeared in (Krebs and Schnieder, 2000; Krebs and Schnieder, 2001). A research program founded by the European Science Foundation has dealt with verification aspects of hybrid systems and has involved several research groups from European Universities (*VHS*). Some of the first monographs in this field include: (van der Schaft and Schumacher, 1999) and (Matveev and Savkin, 2000).

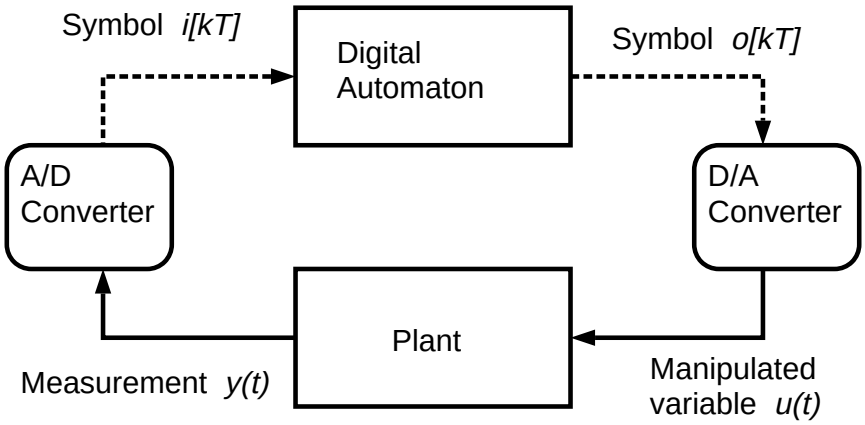


Figure 1.2: A hybrid system structure

Hybrid systems manifest themselves in a large variety of application and forms. One often encountered structure consists of hierarchical systems comprising dynamical components at the lower level, governed by upper level logical or discrete components, as represented schematically in Figure 1.2. However, in some applications such a precise distinction between different hierarchic levels is not possible, especially when dynamical and logical facts are dramatically interdependent. In fact, hybrid behaviour of a system can arise not only from the interaction with external devices, but also due to internal causes, like failures or operating mode changes (Labinaz et al., 1997). For these systems the first challenge is to decide upon a suitable modelling framework. Two basic requirements for a model of a hybrid system are:

1. The model must be descriptive enough to capture with sufficient precision the behaviour of the system. In particular the interdependence between the parts described by logic and the parts described by continuous dynamics must be reflected in the modelling framework.

2. The model must allow to formulate problems arising in the system design and its operation, like e.g. controller synthesis, state estimation, or fault detection. These problems must be solvable with reasonable effort, typically using numerical tools.

Some of the modelling methods proposed in the literature are reviewed in Section 1.2. The rapid progress of computer and information technology represents a further driving force for dealing with hybrid systems. In fact, only recently the hardware tools became available that allowed tackling complex computational problems, as they arise dealing with hybrid systems. Note however that several questions are NP-hard (Blondel and Tsitsiklis, 1999), therefore the ultimate bottleneck for a comprehensive treatment in arbitrary dimensions will most likely be the available computing power. In spite of this, there exist several applications, where the recent state-of-the-art hardware and software tools, support a solution in reasonable time. Moreover, we recall that the computational time grows exponentially with the problem dimension in the *worst case*. Practical problems, however, usually have a lower complexity than predicted by the worst case.

An interesting thought can be found in the editorial to (Antsaklis and Nerode, 1998). We are now speaking about hybrid systems, stressing the heterogeneous nature of the system components. However, in the future when the hybrid systems theory will be mature, we will have a motivation to consider the system as a whole entity. As soon as the tools will become powerful enough to handle all questions of interest, there will be no need anymore to point out that a system is “hybrid”, it will just be a system!

1.2 Modelling Approaches

1.2.1 Model Classes for Hybrid Systems

Hybrid systems have attracted the interest of control and computer science communities from different perspectives. Many modelling approaches have been proposed that put more emphasis either on the continuous valued or on the discrete valued components. The analysis and synthesis methodologies for these models strongly depend on the modelling framework adopted. Among the different approaches (Labinaz et al., 1997; Grossmann et al., 1993) we mention:

Hybrid Automata: A hybrid automaton is a finite state machine, where a continuous dynamics is associated to each discrete state. More precisely (see e.g. (van der Schaft and Schumacher, 1999) and references therein) a hybrid automaton consists of seven components:

- A set L of *discrete* states or *locations*
- A set X of *continuous states*
- A set E of *edges* or *transitions*, where every edge is a 5-tuple:

  $$(l, a, Guard_{ll'}, Jump_{ll'}, l')$$

  The transition from the discrete state $l \in L$ to the discrete state $l' \in L$ is enabled if $x \in Guard_{ll'} \subset X$. During the transition the state $x \in X$ jumps to $x' \in X$ according to the relation given in $Jump_{ll'} \subset X \times X$. $a \in A$ labels the edge.
- A set A of symbols labelling the edges
- A set W of the continuous external variables
- A mapping Inv from L to the subsets of X. Whenever the system is at location l, the continuous state has to satisfy $x \in Inv(l)$. All sets $Inv(l)$ are called *location invariants*.

- A mapping *Act* assigning to each discrete state l a differential-algebraic equation system F_l. The solution to these DAE are called the *activities* of the system.

A *switch* is a change of the discrete location of a hybrid automaton. A *jump* is a resetting of the continuous state of a hybrid automaton when a switch takes place. The time instant, when the switch takes place is called *event time*. The switch and the jump are two phenomena associated with each event. The graph of a hybrid automaton is depicted in Figure 1.3 (source: (van der Schaft and Schumacher, 1999)).

Petri Nets: A Petri Net is a model of a discrete event system, where transitions can occur asynchronously, i.e. out of a fixed time schedule. Petri Nets allow to model behaviours comprising concurrency, synchronization and resource sharing. This framework can be *fluidified* or *continuized*, meaning that it can be modified to include continuous dynamics. An overview of their characteristics and possible extensions is given by David and Alla (1994).

Generalized Hybrid Dynamical Systems: Branicky (1995) collects the most important hybrid phenomena encountered in practice and presents a review of several models for hybrid systems, introduced in the literature. Branicky et al. (1998) show, how a modelling framework can be defined that encompasses a number of model classes in a unified set-up.

Linear Complementarity Systems: This framework was originally used to model mechanical systems with inequality constraints but it can be extended to a general class of hybrid systems. The question about existence and uniqueness of solutions can be formulated within this framework. We refer to (van der Schaft and Schumacher, 1998) for more details.

Piecewise Affine Systems: The evolution of a piecewise affine system is governed by a different affine model according to the present value of the state or the input. These

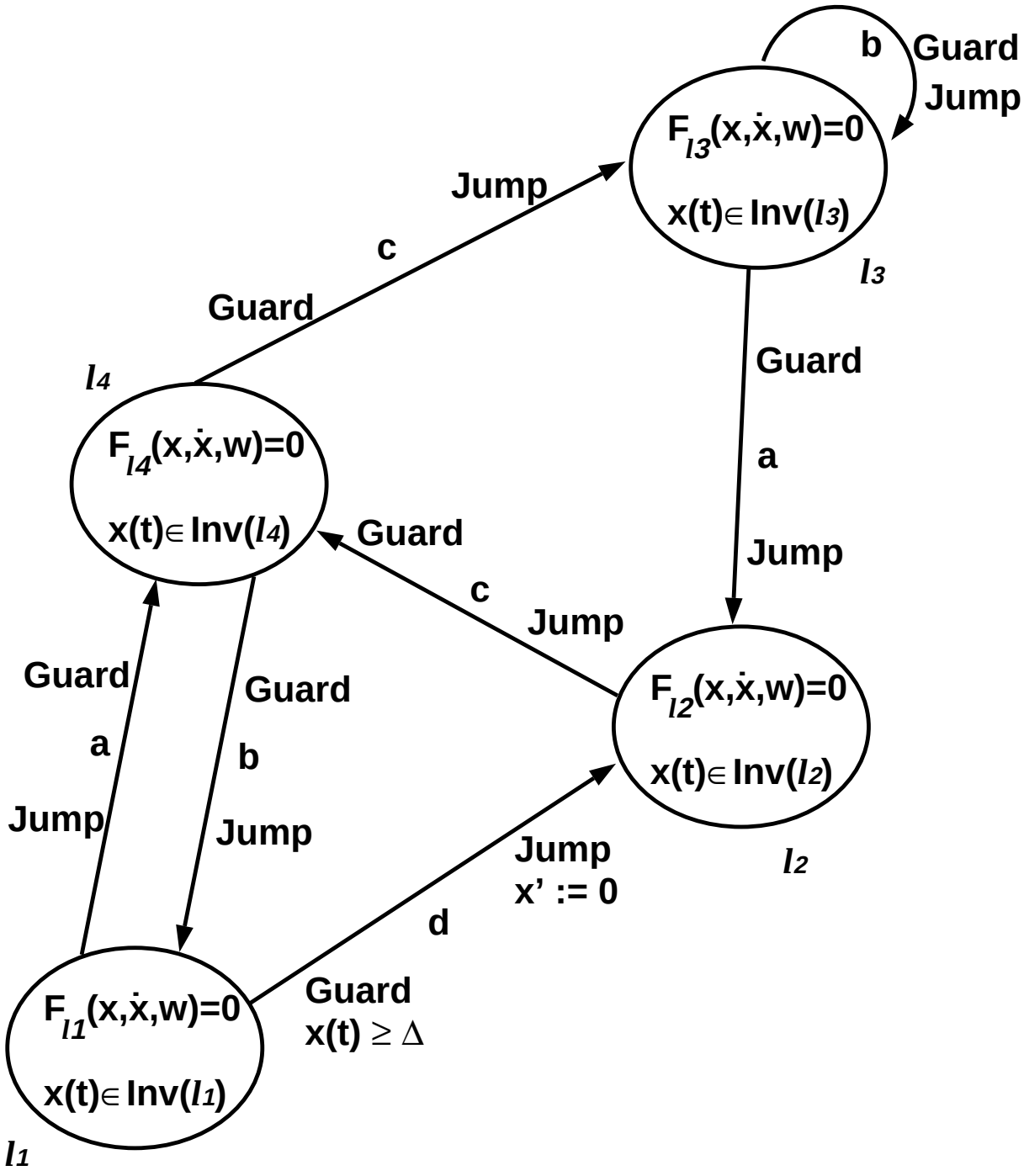


Figure 1.3: Hybrid automaton: The locations are labelled by ℓ_i $(i = 1 \ldots 4)$, the edges are labelled by the letters a, b, c, d, the jumps and guards are explicitly given only for the edge $(\ell_1, d, Guard_{\ell_1\ell_2}, Jump_{\ell_1\ell_2}, \ell_2)$, in each location the invariants and activities are specified.

systems are introduced in Section 2.6 and are analyzed in Chapter 6. We refer to (Sontag, 1981) for theoretical results about state and output feedback, observers, and inverses for piecewise affine systems.

Mixed Logic Dynamical Systems: This is the modelling framework, this thesis is mainly dedicated to. We defer to Section 1.2.4 for an introduction to the framework, Chapters 2 and 3 for details about modelling, and Chapter 4 for algorithms about analysis and synthesis for these systems[1].

Often the proposed approaches are adaptations of common modelling tools in the respective field, for instance a hybrid automaton can be seen as a finite state machine, for which continuous valued components have been added to.

These modelling frameworks differ from each other both for their capability to capture different system behaviours and for their suitability to solve analysis and synthesis problems. While a particular system is more accurately modelled with one method, the solution to some practical problems might be easier to find, if a model in another framework is available. For instance, there exist stability criteria for piecewise affine systems (Johansson and Rantzer, 1998), while conditions on existence and uniqueness of trajectories can be established for linear complementarity systems (van der Schaft and Schumacher, 1998).

The list of modelling approaches above should not be understood as a disjoint classification of models. It can rather be shown that under certain circumstances the system classes are equivalent, in the sense that they represent the same behaviour within different frameworks (Bemporad and Morari, 1999a; Bemporad, Ferrari-Trecate and Morari, 2000; Heemels et al., 2001). In principle this allows to translate analysis and synthesis techniques for one class to another.

[1] In some references, the term "Mixed Logical Dynamical Systems" is used. Of course, both names refer to the same class of systems.

1.2.2 Examples of Hybrid Systems

There is a wide range of systems that can be modelled in a hybrid systems framework. Some classes are listed next:

Multiple model systems: These are systems whose overall evolution is governed by different submodels, either by partitioning the state space into cells (e.g. piecewise linear systems) or by changing system parameters according to a given signal (e.g. switched systems or systems with operating mode changes)

Systems with switchings components: Systems in this category include switching elements like relays, dead-zones or hysteresis. A practical example is the heating and cooling system in houses. A thermostat provides the current information as one of the symbols { cold, normal, hot }. The furnace, the air conditioner or the blower take on these symbols as inputs and form, along with the heat flow characteristics of the rooms, a continuous valued system. Their combination can be considered as a hybrid system.

Adaptive systems: The hybrid nature of these systems derives from switching rules, provided e.g. by finite state machines governing the adaptation law.

Systems with modelled failures: In case of sudden or abrupt faults[2], the occurrence of a failure in a system can be modelled as a switching signal. The fault-prone system can then be considered as a hybrid system.

Systems involving synchronization signals: Such systems arise e.g. in communication networks.

[2]This is the opposite case of slowly occurring or *incipient* faults.

1.2.3 Practical Applications

Driven by the recent developments of modelling, analysis and synthesis techniques, the number of applications for practical hybrid systems grows steadily. In this work we focus on two practical systems: a three tank system and a hydroelectric power plant that are described in detail in Chapter 3. Several other examples were considered in the literature. Here we list just a few:

- Air Traffic Management Systems: The problem of modelling and controlling air traffic collision avoidance maneuvers can be considered with techniques for hybrid systems, as studied e.g. in (Tomlin, 1998).
- Automated Highway Systems: The functionality of these systems ranges from the provision of driver advisory information to the complete control of the lateral and longitudinal motion. The goal of managing compositions of vehicles driving at short distance from each other can only be reached, if the safety requirements are fulfilled, especially for maneuvers like emergency decelerations (Lygeros and Lynch, 1998).
- Power Generation Units: Spedicato (2001) and Letizia (2001) consider a model of a combined cycle power plant with one steam and one gas turbine. The optimal operation of these units over a daily or weekly horizon, taking into account startup times, wear and economic considerations is a problem that can be considered in its complete detail within a hybrid systems framework.

1.2.4 Mixed Logic Dynamical Systems

The seminal paper by Bemporad and Morari (1999a) first introduced the modelling framework of *Mixed Logic Dynamical* (MLD) systems. It laid the foundations for a growing research area at the Automatic Control Laboratory of ETH Zürich and several other academic institutions. While in the first years of this research the focus was on theoretical

aspects, now the efforts are dedicated both to theory and applications with industrial partners, corroborating the relevance of the methods.

The evolution of an MLD model is governed by linear dynamic equations subject to linear mixed integer inequalities, i.e. inequalities involving both *continuous* and *binary* variables. Binary variables represent the discrete-valued components and they are introduced according to logical inference techniques used in operations research. The key idea is to transform propositional logic statements into linear inequalities involving integer and continuous variables.

MLD systems generalize a wide set of models, among which there are constrained linear systems, finite state machines interacting with dynamic systems, some classes of discrete event systems, piecewise linear systems, systems with discrete inputs and more. Several aspects when dealing with MLD systems can be formulated as mixed integer optimization problems. For feedback control purposes, a predictive control scheme can be set up, which is able to stabilize MLD systems on desired reference trajectories while fulfilling operating constraints, and possibly take into account qualitative knowledge in the form of heuristic rules (Bemporad and Morari, 1999a). The dual problem of state estimation can be set up as an optimization problem over a sequence of past measurements (Ferrari-Trecate, Mignone and Morari, 2000; Bemporad, Mignone and Morari, 1999b). Using model equivalences and mathematical optimization methods the online computations for model predictive controllers can be moved offline by synthesizing piecewise linear state feedback control laws (Bemporad, Borrelli and Morari, 2000b).

A key issue when working with hybrid systems is *verification*. The aim of verification is to check, whether there exists an initial condition and an input sequence, such that a system can be driven to a certain state space region. Typically such a region represents a set of dangerous or unsuitable operating conditions. This is closely related to the reachability question. Verification allows to systematically specify safety requirements and to reduce system simulation in the test phase. For systems in MLD form this problem can be solved

through algorithms based on linear and mixed integer linear programs (Bemporad and Morari, 1999b; Bemporad, Torrisi and Morari, 2000).

The list of problems that can be recast as a mixed integer optimization with help of MLD models is constantly growing, including recently observability and reachability analysis, reconfiguration, scheduling, system identification or selective constraint prioritization in model predictive control.

1.3 Organization and Contributions of this Thesis

This thesis presents theoretical results and experiments on the Mixed Logic Dynamical (MLD) systems modelling framework for hybrid systems.

In Chapter 2 we summarize the general MLD form, as it was introduced in (Bemporad and Morari, 1999a). We present systematic methods for the efficient and automatized derivation of MLD models, focusing on the translation of propositional logic into inequalities. Classical analytical approaches for the modelling of system components are compared to a new geometrical approach. The proposed methods reduce the total number of variables in the model. Moreover, we discuss some typical properties of MLD systems, like well posedness and steady states.

In Chapter 3 we present the MLD model of two practical systems: a three tank benchmark system for fault detection and reconfigurable control and the outflow units of a hydroelectric power plant. The goal of this chapter is to show in detail the modeling capabilities of MLD systems.

The models of Chapter 3 are used as examples for the algorithms presented in Chapter 4. In this chapter we outline the formulation of several problems in control and supervision for MLD systems. In Section 4.1 we report experiments of the Model Predictive Control

scheme of Bemporad and Morari (1999a) on the three tank laboratory model, a multi-input multi-output hybrid plant. We consider the minimum time control problem and show, how to integrate heuristic rules within the systematic methods for MLD systems. We introduce a fault tolerant control scheme for hybrid systems, which is composed of two stages: In Section 4.4 we present a new approach for fault detection and state estimation. The method relies on the modelling framework of MLD systems. It casts the fault detection problem as an estimation problem to be solved as a mixed integer continuous optimization. In Section 4.5 we introduce strategies for the controller reconfiguration, to be used if the plant exhibits hardware redundancies. If a fault occurs, we show that the decision problem about the choice of alternative system components can be formulated within the MLD framework. The fault detection and the reconfiguration algorithms are illustrated both with simulations and experiments.

The techniques presented in Chapter 4 rely on mathematical optimization methods that are reviewed in Chapter 5. These problems are in general difficult to solve as the problem size increases, since they are NP-hard. However, the experience of this work shows that the worst case of computational burden is seldom necessary to achieve a solution. We present a heuristic rule, which speeds up the computations on average. It is a tree exploring strategy in the branch and bound algorithm. It exploits the structure of the optimization problems, as they arise for analysis and synthesis of Mixed Logic Dynamical systems. We give a quantification of its complexity.

The model class of Mixed Logic Dynamical systems is equivalent to several other classes of systems, in particular to the class of Piecewise Affine systems. In this thesis we present sufficient conditions for stability of Piecewise Affine systems. These methods rely on linear matrix inequalities in order to find piecewise quadratic and more general Lyapunov functions.

Some conclusions and an outlook are given in Chapter 7. The appendix collects further documentations to various topics referenced within the text. In particular, for better

understanding of this text, we draw the reader's attention to the list of abbreviations in Appendix D and the index at the end of this work.

This thesis is based on the post-diploma thesis (Mignone, 1999) and the following publications:

- (Bemporad, Mignone and Morari, 1999b) for a fault detection scheme of MLD systems.
- (Bemporad, Mignone and Morari, 1999a) presenting a computational strategy for improving the speed in solving the optimization problems, as they arise for MLD systems.
- (Mignone, Bemporad and Morari, 1999) showing a geometrical interpretation of the variable space of an MLD system and suggesting an efficient modelling technique for components of MLD systems.
- (Morari et al., 1999a), (Morari et al., 1999b), (Bemporad, Ferrari-Trecate, Mignone, Morari and Torrisi, 1999) summarize Bemporad and Morari's (1999a) publication and show simulations of controllers and estimators of MLD systems.
- (Ferrari-Trecate, Mignone and Morari, 2000), (Ferrari-Trecate, Mignone and Morari, 2001) present a state estimation technique with convergence guarantees for piecewise affine systems.
- (Kerrigan, Bemporad, Mignone, Morari and Maciejowski, 2000) considers systematic constraint prioritization within MLD systems.
- (Ferrari-Trecate, Mignone, Castagnoli and Morari, 2000) shows the modelling phase for the model of a hydroelectric power plant.
- (Mignone, Ferrari-Trecate and Morari, 2000a), (Ferrari-Trecate, Cuzzola, Mignone and Morari, 2001) consider the stability analysis for piecewise affine systems.

- (Tsuda, Mignone, Ferrari-Trecate and Morari, 2001) suggests several reconfiguration strategies for MLD systems.

The following technical reports contain further material about selected topics considered in this thesis:

- (Bemporad and Mignone, 2000) describes a solver for mixed integer quadratic programs for Matlab, which has been developed during this thesis.
- (Mignone, 2001) collects several tables supporting the translation of logic propositions into inequalities.
- (Torrisi, Mignone and Morari, 2001) presents a comparison between heuristic and systematic methods for controller design for hybrid systems that achieve the control goal in minimum time. The techniques are tested on a laboratory model.
- (Torrisi, Bemporad and Mignone, 2000) considers some theoretical modelling aspects and describes a software tool for automated model generation.
- (Mignone and Monachino, 2001) is a manual for the operation of a laboratory experimental set-up used to validate the methods presented in this work.
- (Tsuda, Mignone, Ferrari-Trecate and Morari, 2000) contains details and simulations about the application of reconfiguration strategies on the three tank system.

Chapter 2

The Mixed Logic Dynamical Systems Framework

In this chapter we introduce the modelling framework for hybrid systems used in this thesis.

2.1 Introduction

Several systems include logic components like e.g. relays, switches or finite state machines interacting with continuous valued components. Logic components usually cannot be described by continuous equations like (1.1) or (1.2). Their functionality is better represented by propositional logic (Williams, 1993; Mendelson, 1964), than by difference and differential equations. In some cases the inclusion of logic rules in a model occurs without an interaction to a control mechanism. This situation happens, for example, in nonlinear systems, for which piecewise linear approximations are used to model the system evolution. There, the logic rules express an internal mode change. Even though this type of switching may not be physically present in the plant, it is introduced in the model, and any analysis or synthesis technique is therefore required to take it into account. In addition to a quanti-

tative system description, there might be some available qualitative information about the system, for instance in terms of heuristic knowledge that is gathered from the expertise of operators. In classical design methods it is often unclear, how this knowledge can be systematically taken into consideration, beyond the suggestion of tuning guidelines derived from it. For an accurate system description it is useful to have the possibility to include this knowledge into the model. However, it should be suitably combined to the sharp mathematical description stemming from first principle modelling or system identification.

Recently it was shown (Bemporad and Morari, 1999a; Tyler and Morari, 1999) that expressing logical propositions in the form of linear constraints on binary variables represents the key to a powerful modelling framework for hybrid dynamical systems, the *Mixed Logic Dynamical* (MLD) form. It allows to describe several classes of systems, like piecewise linear systems, systems with mixed discrete/continuous inputs and states, continuous valued systems interacting with discrete valued components, and many others more. The framework permits to include and prioritize constraints, and incorporate heuristic rules in the description of the model. Therefore it offers the possibility to cover most of the modelling capabilities required for hybrid systems. In this chapter, by presenting the main properties of MLD systems, we will show that the apparently contradicting and broad requirements previously stated, can be subsumed into the common framework of MLD systems.

In Section 2.2 we outline the connection between logic propositions and linear constraints, taken from the theory of propositional calculus. Section 2.3 describes the general form of MLD systems. Section 2.4 is dedicated to the efficient modeling of MLD system, focusing on the inclusion of logic propositions in MLD models. Some considerations about well posedness are given in Section 2.5. MLD systems are equivalent to other system classes. We summarize these equivalences in Section 2.6. We conclude this chapter briefly describing a tool for computer aided modeling and some properties of MLD systems in Sections 2.7 and 2.8.

2.2 Propositional Calculus and Linear Integer Programming

2.2.1 Logic Propositions

In operations research the field of logical inference has received some attention (Chandru and Hooker, 1999). By following standard notation (Williams, 1977; Cavalier et al., 1990; Williams, 1993), we adopt capital letters X_i to represent statements, e.g. "$x \geq 0$" or "Temperature is hot". X_i is commonly referred to as a *literal* or *Boolean variable* , and has a truth value of either "T" (true) or "F" (false). Boolean algebra enables statements to be combined in compound statements by means of operators, often termed *connectives*. These are:

$\wedge$ logical conjunction, logical "and"
$\vee$ logical disjunction, logical "or"
$\bar{\cdot}$ logical negation, logical "not"
$\rightarrow$ logical implication
$\leftrightarrow$ logical equivalence, if and only if
$\oplus$ logical "exclusive or"

A comprehensive treatment of Boolean calculus can be found in digital circuit design texts, e.g. (Christiansen, 1997; Hayes, 1993). For a rigorous exposition see e.g. (Mendelson, 1964).

Connectives satisfy several properties (Christiansen, 1997) that can be used to transform compound statements into equivalent statements involving different connectives, and simplify complex statements. It is known that all connectives can be defined in terms of a subset of them, which is said to be a complete set of connectives.

Example 2.1:

The set of connectives $\{\vee, \bar{\cdot}\}$ is a complete set of connectives. Other connectives can be written in terms of these by applying logic equivalences, for instance

$$X_1 \rightarrow X_2 \quad \text{is the same as} \quad \bar{X}_1 \vee X_2 \tag{2.1a}$$

$$X_1 \rightarrow X_2 \quad \text{is the same as} \quad \bar{X}_2 \rightarrow \bar{X}_1 \tag{2.1b}$$

$$X_1 \leftrightarrow X_2 \quad \text{is the same as} \quad (X_1 \rightarrow X_2) \ \wedge \ (X_2 \rightarrow X_1) \tag{2.1c}$$

$$X_1 \wedge X_2 \quad \text{is the same as} \quad \overline{\bar{X}_1 \vee \bar{X}_2} \tag{2.1d}$$

□

Correspondingly one can associate with a literal X_i a *binary variable* $\delta_i \in \{0, 1\}$, which has a value of either 1 if X_i =T, or 0 if X_i =F.

Once a set of logic propositions has been formulated for a system, it is possible to check the truth of properties formulated in terms of the involved literals. To this purpose integer programming has been advocated as an efficient inference engine to perform such an automated deduction (Cavalier et al., 1990). A propositional logic problem, where a statement X_1 must be proven to be true given a set of (compound) statements involving literals X_2, ... , X_n, can in fact be solved by means of a linear integer program (Schrijver, 1986), by suitably translating the original compound statements into linear inequalities involving binary variables δ_i. These inequalities are termed *integer linear inequalities*. In fact, the propositions and linear constraints reported in Table 2.1 can easily be seen to be equivalent. Similar ideas originally came up in the early 1960s dealing with the synthesis of linear switching circuits (Minnick, 1961; Stram, 1961). Threshold logic was introduced to synthesize arbitrary combinational circuits as an alternative to realizations with AND/OR/NOT gates. Alternative methods and formulations for transforming propositional logic problems into equivalent integer programs exist, and no method is uniformly better than the others. Cavalier et al. (1990) conclude that the choice of an efficient modelling approach is depen-

relation	logic proposition	mixed integer (in)equalities
AND ($\wedge$) $X_1 \wedge X_2$	$[\delta_1 = 1] \wedge [\delta_2 = 1]$	$\delta_1 = 1$ $\delta_2 = 1$
$X_3 \leftrightarrow (X_1 \wedge X_2)$	$[\delta_3 = 1] \leftrightarrow$ $[\delta_1 = 1] \wedge [\delta_2 = 1]$	$-\delta_1 + \delta_3 \leq 0$ $-\delta_2 + \delta_3 \leq 0$ $\delta_1 + \delta_2 - \delta_3 \leq 1$
OR ($\vee$) $X_1 \vee X_2$	$[\delta_1 = 1] \vee [\delta_2 = 1]$	$\delta_1 + \delta_2 \geq 1$
$X_3 \leftrightarrow (X_1 \vee X_2)$	$[\delta_3 = 1] \leftrightarrow$ $[\delta_1 = 1] \vee [\delta_2 = 1]$	$\delta_1 - \delta_3 \leq 0$ $\delta_2 - \delta_3 \leq 0$ $-\delta_1 - \delta_2 + \delta_3 \leq 0$
NOT ($\bar{\cdot}$) $\bar{X}_1$	$\overline{[\delta_1 = 1]}$	$\delta_1 = 0$
XOR ($\oplus$) $X_1 \oplus X_2$	$[\delta_1 = 1] \oplus [\delta_2 = 1]$	$\delta_1 + \delta_2 = 1$
$X_3 \leftrightarrow (X_1 \oplus X_2)$	$[\delta_3 = 1] \leftrightarrow$ $[\delta_1 = 1] \oplus [\delta_2 = 1]$	$-\delta_1 - \delta_2 + \delta_3 \leq 0$ $-\delta_1 + \delta_2 - \delta_3 \leq 0$ $\delta_1 - \delta_2 - \delta_3 \leq 0$ $\delta_1 + \delta_2 + \delta_3 \leq 2$
IMPLY ($\rightarrow$) $X_1 \rightarrow X_2$	$[\delta_1 = 1] \rightarrow [\delta_2 = 1]$	$\delta_1 - \delta_2 \leq 0$
IFF ($\leftrightarrow$) $X_1 \leftrightarrow X_2$	$[\delta_1 = 1] \leftrightarrow [\delta_2 = 1]$	$\delta_1 - \delta_2 = 0$

Table 2.1: Conversion of basic logic relations into integer inequalities

dent on the form of the logical statements. It is clear that a preprocessing for reduction of the propositional logic problem might produce large benefits for the numerical solution of the resulting mixed integer program. A similar problem, i.e. the problem of finding minimal forms, is well known in the digital network design realm, where the need arises to minimize the number of gates and connections, and therefore the implementation cost. A variety of methods exist to perform such a task. The reader is referred to (Hayes, 1993, Chap. 5) for a detailed exposition.

2.2.2 Mixed Logic-Continuous Propositions

We borrow the logical inference technique to model logical parts of processes and heuristic knowledge about plant operation as integer linear inequalities. As we are interested in systems which have both discrete logic and continuous dynamics, we wish to establish a link between the two worlds. In particular, we need to establish how to build statements from operating events concerning continuous dynamics. The key idea is to use *mixed integer linear inequalities* , i.e. linear inequalities involving both continuous variables $x \in \mathbb{R}^n$ and logical variables $\delta \in \{0,1\}^m$. In (Raman and Grossmann, 1991; Raman and Grossmann, 1992) the idea of introducing qualitative knowledge in process synthesis using propositional logic has been presented. An early application to scheduling and resource allocation is described in (Glover, 1975). In Table 2.2 a list of equivalences can be found that allow to perform the translation into a set of inequalities to be used as constraints in mathematical optimization problems. An extensive list of such transformations is given in (Mignone, 2001). Note that relations involving the form $[\delta = 0]$ instead of $[\delta = 1]$ in Tables 2.1 and 2.2 can be obtained by substituting $(1 - \delta)$ for δ in the corresponding inequalities.

Example 2.2:

As an illustration of Table 2.2 consider for instance the statement

$$X \triangleq [f(x) \leq 0] \tag{2.2}$$

relation	logic proposition	mixed integer inequalities
IMPLY ($\rightarrow$) $[f(x) \leq 0] \rightarrow X$	$[f(x) \leq 0] \rightarrow [\delta = 1]$	$f(x) \geq \epsilon + (m - \epsilon)\delta$
$X \rightarrow [f(x) \leq 0]$	$[\delta = 1] \rightarrow [f(x) \leq 0]$	$f(x) \leq M - M\delta$
IFF ($\leftrightarrow$) $[f(x) \leq 0] \leftrightarrow X$	$[f(x) \leq 0] \leftrightarrow [\delta = 1]$	$f(x) \leq M - M\delta$ $f(x) \geq \epsilon + (m - \epsilon)\delta$
Product IF X THEN $z = f(x)$ ELSE $z = 0$	$z = \delta \cdot f(x)$	$z \leq M\ \delta$ $-z \leq -m\ \delta$ $z \leq f(x) - m(1 - \delta)$ $-z \leq -f(x) + M(1 - \delta)$

Table 2.2: Conversion of basic mixed logic continuous relations into mixed integer inequalities. M, m in $\mathbb{R}$ are upper and lower bounds to the linear function $f(x)$ for $x \in \mathcal{X}$, $\epsilon > 0$ is a small tolerance, see Section 2.5

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is linear, assume that $x \in \mathcal{X}$, where $\mathcal{X}$ is a given bounded set, and define

$$M \triangleq \max_{x \in \mathcal{X}} f(x) \tag{2.3}$$

$$m \triangleq \min_{x \in \mathcal{X}} f(x) \tag{2.4}$$

Theoretically, an over[under]-estimate of M $[m]$ suffices for our purpose. However, more realistic estimates provide computational benefits (Williams, 1993, p. 171). By associating the binary variable δ with the literal X, one can transform (2.2) into the following mixed integer inequalities

$$f(x) \leq M - M\delta \tag{2.5}$$

$$f(x) \geq \epsilon + (m - \epsilon)\delta \tag{2.6}$$

where ϵ is a small tolerance, typically the machine precision (see Chapter 2.5), beyond which the constraint is regarded as violated. To check the equivalence

of the logical statement to the given set of inequalities, one can verify that (2.5) and (2.6) reduce to the four constraints in column "resulting implication" of Table 2.3, provided that the corresponding additional assumptions in column "fixed quantity" are made.

fixed quantity	inequality (2.5)	inequality (2.6)	resulting implication
$\delta = 0$	$f(x) \leq M$ (inactive)	$f(x) \geq \epsilon$	$f(x) \geq \epsilon$
$\delta = 1$	$f(x) \leq 0$	$f(x) \geq m$ (inactive)	$f(x) \leq 0$
$f(x) \leq 0$	$\delta \in \{0,1\}$ (inactive)	$\delta = 1$	$\delta = 1$
$f(x) \geq \epsilon$	$\delta = 0$	$\delta \in \{0,1\}$ (inactive)	$\delta = 0$

Table 2.3: Implications of (2.5) and (2.6)

Note that according to the "fixed quantity" in Table 2.3, one inequality of (2.5) and (2.6) becomes inactive in the sense that it is always fulfilled[1], assuming (2.3), (2.4) and $\delta \in \{0,1\}$. □

From a semantic point of view, note that the range of δ in the right column of Table 2.2 is understood as a subset of the real numbers. Even though we are often speaking about δ being a logic or a discrete variable, when it comes down to the translation to inequalities, the inherent assumption is

$$\delta \in \{0,1\} \subset \mathbb{R} \tag{2.7}$$

In this text we will sometimes abuse of the notation concerning the Boolean variable X and its associated binary variable δ, by using the correspondences as follows:

$$\begin{aligned} \text{false} &\quad \Leftrightarrow \quad F \quad \Leftrightarrow \quad 0 \\ \text{true} &\quad \Leftrightarrow \quad T \quad \Leftrightarrow \quad 1 \end{aligned}$$

[1] a tautology

Sometimes this translation requires the introduction of *auxiliary variables* (Williams, 1993), for instance according to the last rule in Table 2.2 a product between binary and continuous quantities requires the introduction of a real variable z. In Section 2.4 we will return to these concepts.

2.3 Structure of Mixed Logic Dynamical Systems

Mixed Logic Dynamical models for hybrid systems rely on the combination of three ideas and observations that have been presented isolatedly in the literature. The ideas are:

1. Representation of logical propositions as linear inequalities involving binary variables.

2. Representation of propositions coupling logic relations to continuous variables as linear inequalities involving continuous and binary variables.

3. Inclusion of binary variables and linear constraints in a description for continuous valued systems.

The first two points have been considered in Section 2.2. The last step is merging all inequalities that are built using the methods in Section 2.2 into a set of constraints. To allow the formulation of computationally tractable problems for system analysis and synthesis, a linear discrete time model is chosen for the state evolution. The logic components influence the continuous dynamics, therefore it is necessary to introduce binary variables in a continuous valued system description. One of the first works using this idea, is reported in (Witsenhausen, 1966), where a linear model is suggested, allowing to have both continuous and discrete valued states and inputs.

The general form of an MLD system is given by the following expressions:

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k) \tag{2.8a}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k) \tag{2.8b}$$

$$E_2 \delta(k) + E_3 z(k) \leq E_1 u(k) + E_4 x(k) + E_5 \tag{2.8c}$$

Equation (2.8a) is the state update function, (2.8b) is the output function, and the set of inequalities in (2.8c) collects constraints on the system, as well as translations from logic proposition, as in Tables 2.1 and 2.2. The inequality (2.8c) has to be understood componentwise. The meaning of the variables is the following:

- x are the continuous and binary states:

$$x = \begin{bmatrix} x_c \\ x_\ell \end{bmatrix}, \; x_c \in \mathbb{R}^{n_c}, \; x_\ell \in \{0,1\}^{n_\ell}$$

- y are the continuous and binary outputs:

$$y = \begin{bmatrix} y_c \\ y_\ell \end{bmatrix}, \; y_c \in \mathbb{R}^{p_c}, \; y_\ell \in \{0,1\}^{p_\ell}$$

- u are the continuous and binary inputs:

$$u = \begin{bmatrix} u_c \\ u_\ell \end{bmatrix}, \; u_c \in \mathbb{R}^{m_c}, \; u_\ell \in \{0,1\}^{m_\ell}$$

- $\delta \in \{0,1\}^{r_\ell}$ are the auxiliary binary variables
- $z \in \mathbb{R}^{r_c}$ are the auxiliary continuous variables

Note that by removing (2.8c) and by setting δ and z to zero, Equations (2.8a) and (2.8b) reduce to an unconstrained linear discrete time system in state space. The variables δ and z are introduced when translating logic propositions into linear inequalities. All constraints are summarized in the inequality (2.8c). The description (2.8) only appears to be linear

since the variables δ, x_ℓ, y_ℓ and u_ℓ are constrained to be binary. To illustrate the modelling procedure in MLD form, we defer to Chapter 3, as well as e.g. to (Castagnoli, 2000; Mignone, 1999). We point out that the variables δ, x_ℓ, u_ℓ and y_ℓ are implicitly assumed to be binary, and this is not explicitly mentioned in each MLD model.

The system (2.8) is assumed to be *completely well posed* (Bemporad and Morari, 1999a). This means that given a state $x(k)$ and an input $u(k)$ the inequalities (2.8c) have a unique solution for $\delta(k)$ and $z(k)$. In other words, at each time instant, the maps $(x(k), u(k)) \mapsto \delta(k)$ and $(x(k), u(k)) \mapsto z(k)$ are single-valued. As a consequence, $x(k+1)$ is uniquely defined, allowing to find a unique trajectory of the state $x(.)$ given an initial state $x(0)$ and an input sequence $u(0) \dots u(N)$. Well posedness is a condition for well defined simulation of MLD systems:

Algorithm 2.1 *Simulation of MLD systems*

1. *Set $k = k_0$ and choose $x(k)$.*
2. *Given $x(k)$ and $u(k)$, obtain $\delta(k)$ and $z(k)$ from a feasibility test over (2.8c).*
3. *Determine $x(k+1)$ from (2.8a).*
4. *Set $k = k+1$ and go to 2.*

It is possible to determine $\delta(k)$ and $z(k)$ in point 2 with a *mixed integer feasibility test* (MIFT). This problem, along with related ones are considered in Chapter 5. The assumption on well posedness is usually not restrictive in practice. Even though it is easy to define MLD systems that are not well posed, these cases are often only of academic interest, and they arise if the system is not modelled precisely, see e.g. Section 2.5. For simulation of large systems, more efficient algorithms than Algorithm 2.1 can be devised (Torrisi, 2002).

The matrices $A, B_1, \dots, B_3, C, D_1, \dots, D_3, E_1, \dots, E_5$ are real and have appropriate dimensions. While the matrices A, B_1, B_3 in (2.8a) are set as general matrices, some entries

have to be excluded, if binary states are present in x. More precisely, the logic components x_ℓ of the state cannot depend directly on continuous variables x_c, u_c, z through (2.8a). The corresponding entries in A, B_1, B_3 have to be zero for a valid MLD model (Torrisi, 2002; Ferrari-Trecate, Cuzzola and Morari, 2002). For instance, the state update matrix A has the form

$$A = \begin{bmatrix} A_{cc} & A_{cd} \\ 0_{n_\ell \times n_c} & A_{dd} \end{bmatrix}$$

2.4 Efficient Modelling of Mixed Logic Dynamical Systems

The translation of logic propositions or mixed logic continuous statements into inequalities in Section 2.2 is a step that critically determines the size of the resulting MLD model in terms of the number of variables. As it will be clear in Chapter 5, we are envisioning to find models that have a small number of binary variables. This allows to improve the speed in solving the mathematical optimization problems, required for analysis, synthesis and supervision of hybrid systems. In this section we propose methods to efficiently find the translation of logic proposition into inequalities by decreasing the number of auxiliary variables needed. The methods are applicable when the propositional logic statements are more complex than in Table 2.1, involving arbitrary combinations of connectives.

2.4.1 Logic Propositions

General Remarks

Assume that X_i are Boolean variables. A *Boolean formula* F

$$F(X_1, \ldots, X_n) \tag{2.9}$$

is an arbitrary combination of literals X_i with the connectives listed on page 21, i.e. the unary operator $\bar{.}$ (NOT), the binary logic operators $\vee, \oplus, \wedge, \leftarrow, \rightarrow, \leftrightarrow$ and brackets (.). A Boolean formula F is a *conjunctive normal form* (CNF) or *product of sums*, if it is written in the following form:

$$F = \bigwedge_{i=1}^{n} \psi_i \tag{2.10}$$

where

$$\psi_i = \bigvee_{j=1}^{m} X_{ij} \tag{2.11}$$

The Boolean formulas ψ_i are called *terms of the product* or *maxterms*, and X_{ij} are the *terms of the sum*. It is well known that any Boolean formula can be converted into CNF, by using laws of distribution, negation, and De Morgan. A CNF is minimal, if it has the minimum number of maxterms and each maxterm has the minimum number of terms of the sum. This definition of minimality is motivated from the logic circuit design field, where each logic operator of a CNF is implemented using a gate or having an additional input in multi-input gates. Every Boolean formula can be rewritten as a minimal CNF (Kohavi, 1978).

A Boolean formula f is called *Boolean function* if, after possibly renaming the literals, it is used to define a literal X_n as a function of $X_1, \ldots, X_{n-1}$ as

$$X_n \leftrightarrow f(X_1, X_2, \ldots, X_{n-1}) \tag{2.12}$$

which can be equivalently written as $X_n = f(X_1, X_2, \ldots, X_{n-1})$, where f is an arbitrary Boolean formula. More generally, we can define relations among Boolean variables $X_1, \ldots, X_n$ through a *Boolean relation*

$$F(X_1, \ldots, X_n) = \mathsf{T} \tag{2.13}$$

where F is a Boolean formula. Note that each Boolean function is also a Boolean relation, but the opposite is not true. The *satisfiability problem* for a Boolean relation is the problem

to determine whether there exists an assignment of $\{F, T\}$ to its literals such that the corresponding Boolean formula evaluates to *true*. A satisfying truth assignment for a Boolean formula is called a *valid point*[2]. The idea of solving this logic inference problem with optimization methods relies on tools from the field of integer programming (Chandru and Hooker, 1999; Nemhauser and Wolsey, 1988; Schrijver, 1986). Note that

- Each Boolean function has exactly 2^{n-1} valid points.
- A Boolean formula with exactly 2^{n-1} valid points need not be a Boolean function.

By associating binary variables $\delta_i \in \{0, 1\}$ with each Boolean variable $X_i \in \{F, T\}$, a valid point for a Boolean formula can be equivalently described as an element of the set $\{0, 1\}^n \subset [0, 1]^n$. Therefore the set of all valid points of a Boolean formula can be represented as set of vertices of the hypercube $[0, 1]^n$, i.e. as the vertices of a polytope in $[0, 1]^n$. We call P_{CH} the polytope spanned by all the valid points of a Boolean formula F:

$$P_{CH}(F) \triangleq \{x \in [0, 1]^n : x \in \text{conv}(\text{all valid points of } F)\} \tag{2.14}$$

Here, $conv(Y)$ denotes the convex hull of the set of points Y.

A polytope has two standard representations: the *V-representation* that specifies the polytope through its vertices and the *H-representation* that specifies the polytope as the intersection of half-spaces given by linear inequalities (Ziegler, 1995). A vertex, all components of which are integers, is called an *integral vertex*.

The following methods can be seen as a generalization of the rules of Table 2.1. The goal is to find equivalent linear inequalities for arbitrary Boolean relations (2.13). We propose three methods: the substitution method, the analytical method, and the geometrical method. The latter method is the main contribution of this section.

[2]According to the nomenclature of Chandru and Hooker (1999) a valid point is also called a *model* for a Boolean formula. However, we will not use this name to avoid confusions with system models.

Substitution Method

Given a Boolean relation (2.13), this method consists of recognizing elementary terms listed in Table 2.1 and substituting them with additional Boolean variables. The idea is easily applicable since it is based on a recursive application of the basic rules of Table 2.1. However, the introduction of additional variables is a severe disadvantage, and it is unnecessary, as we show with the next methods.

Analytical Method, or CNF method

The analytical method consists of first converting (2.12) or (2.13) into CNF, which can be performed automatically by using one of several techniques available, see e.g. (Miller, 1965; Sasao, 1999). Let the CNF have the form

$$\bigwedge_{j=1}^{m} \left(\bigvee_{i \in P_j} X_i \ \vee \ \bigvee_{i \in N_j} \overline{X_i} \right) \tag{2.15}$$

where P_j denotes the set of indices of the literals that appear in positive form in the $j-$th maxterm, and N_j denotes the set of indices of the literals that appear in negated form in the $j-$th maxterm. The CNF indicates that all maxterms must be true in order for the Boolean formula to be true. By setting up a set of constraints in an optimization problem we require as well that the feasible points satisfy all inequalities simultaneously. For each maxterms we can therefore deduce one integer inequality. Note further that a maxterm is true, if and only if at least one of its terms of the sum is true. With these observations, the set of integer linear inequalities corresponding to (2.15) is

$$\begin{cases} 1 \leq \sum_{i \in P_1} \delta_i + \sum_{i \in N_1} (1 - \delta_i) \\ \vdots \\ 1 \leq \sum_{i \in P_m} \delta_i + \sum_{i \in N_m} (1 - \delta_i) \end{cases} \tag{2.16}$$

Inequalities (2.16) can be included as constraints in an MLD model in (2.8c). With these inequalities we can define the set $P_{CNF}(F)$ for any Boolean formula F as:

$$P_{CNF}(F) \triangleq \{x \in [0,1]^n : (2.16) \text{ is satisfied with } x = [\delta_1, \ldots, \delta_n] \in [0,1]^n\} \tag{2.17}$$

$P_{CNF}(F)$ is the polytope in the unit hypercube of dimension n that includes all integer points $x \in \{0,1\}^n$ for which $F(x) = T$. It is the set of points defined by (2.16) relaxing the integrality constraints on δ_i from $\delta_i \in \{0,1\}$ to $\delta_i \in [0,1]$.

No auxiliary variables need to be introduced with this technique. Another advantage is its simplicity, once the CNF of the desired Boolean formula is known. One disadvantage is the effort in computing symbolically the CNF from an arbitrary Boolean formula. This may require techniques like Karnaugh maps, see e.g. (Kohavi, 1978), if the Boolean formula is given implicitly by its valid points. Another drawback concerns the set $P_{CNF}(F)$. As it will be outlined next, $P_{CNF}(F)$ may be a set, which is larger than actually required in order to define the integer feasible points. Indeed, $P_{CNF}(F)$ may have non-integral vertices.

Geometrical Method, or Truth Table Method

The third method is based on a geometric interpretation. We show that this method overcomes the drawback concerning $P_{CNF}(F)$ of the analytical method described previously. It allows an automatic translation of truth tables representing Boolean formulas into linear integer inequalities.

A *truth table* for a Boolean formula $F(X_1, \ldots, X_n)$ is the complete list of valid points for $F(.)$, i.e. all vectors $\underline{X} = (X_1, \ldots, X_n) \in \{F, T\}^n$ for which $F(\underline{X}) = T$. For any Boolean formula, the truth table can be set up e.g. by enumeration. The rows of the truth table form the complete list of valid points, i.e. all those points in $\{0,1\}^n$ that render $F(\underline{X})$ true, or satisfy $X_n = f(X_1, \ldots, X_{n-1})$.

When we model systems in MLD form we often encounter the following problems:

- **P1** Impose that the Boolean formula $F(X_1, \ldots, X_n)$ is true, i.e. add the constraint given by the Boolean relation $F(X_1, \ldots, X_n) = T$
- **P2** Define the Boolean function $X_n = f(X_1, \ldots, X_{n-1})$

Both f and F are assumed to be defined with the basic connectives on page 21. The problem P1 specifies a constraint or a characteristic of the system.

Example 2.3:
An example for P1 is the statement that two binary inputs u_1 and u_2 are not allowed to take on the value "1" at the same time, if a third input u_3 is zero. The corresponding Boolean formula F_1 reads

$$F_1(u_1, u_2, u_3) = \bar{u}_3 \rightarrow \overline{(u_1 \wedge u_2)}$$

□

While P1 has the character of a constraint, P2 defines an auxiliary quantity for further use in the MLD model and has the character of an algebraic equation.

Example 2.4:
A typical case where P2 appears is the definition of a state transition in a finite automaton. $X_1, \ldots, X_{n-1}$ code the present state and inputs and X_n is the subsequent state. Then, the function f represents the state transition function. □

As it is immediate to transform Boolean relations into truth tables, without loss of generality we assume that F, f are defined via a truth table. Note that the truth table in P2 has always 2^{n-1} rows and n columns. On the other hand the truth table of P1 can have any number of rows[3] from 1 to $2^n - 1$.

[3]The truth table with no rows is not meaningful since it represents an infeasible system and the truth table with 2^n rows does not exclude any combinations of variables $X_1, \ldots, X_n$ and is therefore tautologic.

Example 2.5:

To further illustrate the difference between P1 and P2, consider the example of the Boolean formula $X_1 \wedge X_2$ and the Boolean function $X_3 = X_1 \wedge X_2$. The truth tables are given in Table 2.4.

X_1	X_2	$\Rightarrow$	δ_1	δ_2
T	T		1	1

X_1	X_2	X_3	$\Rightarrow$	δ_1	δ_2	δ_3
T	T	T		1	1	1
T	F	F		1	0	0
F	T	F		0	1	0
F	F	F		0	0	0

Table 2.4: Truth tables for the Boolean formula $X_1 \wedge X_2$ (left) and the Boolean function $X_3 = X_1 \wedge X_2$

Note that in the first case there is only one valid point and the truth table consists therefore only of one row. □

Consider the unit hypercube $\mathcal{H} \triangleq [0,1]^n$, and let H denote the set of its vertices. Let $\text{conv}(S)$ be the convex hull of a set $S \subseteq H$, and $C_H(S)$ the complementary set in H of S, i.e.

$$\begin{aligned} C_H(S) \bigcup S &= H \\ C_H(S) \bigcap S &= \emptyset \end{aligned}$$

Lemma 2.1 says that the subsets S of H can be "wrapped" inside a polytope which does not contain any vector of the complementary set $C_H(S)$.

Lemma 2.1 *Let $S \subseteq H$. Then $\text{conv}(S) \bigcap C_H(S) = \emptyset$.*

Proof. All vectors $h \in H$ are extreme points of $\mathcal{H}$. Therefore, they cannot be written as nontrivial convex linear combinations of other vectors in $\mathcal{H}$. In

particular, the vectors $h_c \in C_H(S)$ cannot be written as convex combinations of vectors $h_s \in S$. This proves that $h_c \notin \text{conv}(S)$, $\forall h_c \in C_H(S)$. □

Consider now the truth table $\mathcal{T}$ expressing the valid points associated with problems P1 or P2. Let m be the number of rows in $\mathcal{T}$, and let $T = \{R_1, \dots, R_m\}$ the set of the rows R_i of $\mathcal{T}$, considered as vectors in $\mathbb{R}^n$, i.e. the collection of all valid points. Each component of a row R_i is either 0 or 1, and therefore $R_i \in H$, i.e. R_i is a vertex of the hypercube $\mathcal{H}$.

Theorem 2.1 *Let $\mathcal{T}$ be a truth table with rows T, associated to a Boolean formula F on literals $X_1, \dots, X_n$. Then $F(X_1, \dots, X_n)$ is true if and only if the associated vector $\delta \triangleq [\delta_1, \dots, \delta_n] \in \{0,1\}^n$ satisfies the inequalities*

$$A\delta \leq B$$

where A and B define the polytope

$$P_{CH} \triangleq \{\delta \in [0,1]^n : \ A\delta \leq B\} = \text{conv}(T) \tag{2.18}$$

Moreover, each integer translation $\bar{P} = \{\delta \in [0,1]^n : \ \bar{A}\delta \leq \bar{B}\}$ of F is such that $P_{CH} \subseteq \bar{P}$, i.e. P_{CH} is minimal.

Proof. If $\delta \in \{0,1\}^n$ is a combination of binary variables associated to a valid point of F, then δ is a row R_i of the truth table, i.e. $\delta \in T$, and therefore $\delta \in \text{conv}(T)$.

On the other hand, let $\delta \in \text{conv}(T)$. If δ were a *false* combination, i.e. $\delta \in C_H(T)$, Lemma 2.1 would be violated. Moreover, if $\bar{P}$ is an integer representation of F, then $R_i \in \bar{P}$, $\forall i = 1, \dots, m$. Since $\bar{P}$ is convex, $P_{CH} = \text{conv}(\{R_1, \dots, R_m\}) \subseteq \bar{P}$. □

The proposed approach does not introduce any additional Boolean variable. This feature, and the minimality of the relaxed set, are particularly appealing when the model is used

in an optimization problem. While there is no need to do symbolical manipulations for finding the conjunctive normal form, this method suffers from the heavy computations required to find the convex hull of the valid points and the large storage requirements to store the truth table. Some software tools for the computation of the convex hull of points are listed in the next paragraph.

Convex Hull Computation

Several packages exist for transforming a polyhedron P from the form

$$P = \{x : \ x = \sum_{i=1}^{m} \lambda_i x_i + \sum_{i=1}^{p} \mu_i r_i\}$$

where x_i, r_i are the extreme points and extreme directions of P respectively, and

$$0 \leq \lambda_i \leq 1, \quad \sum_{i=1}^{m} \lambda_i = 1, \quad \mu_i \geq 0$$

to the form

$$P = \{x : \ Ax \leq B\}$$

For a detailed survey of these packages, the reader is referred to `http://www.geom.umn.edu/software/cglist/ch.html`. A small selection is listed next:

- `qhull` by Brad Barber, David Dobkin and Hannu Huhdanpaa, The Geometry Center (Barber et al., 1996)
- `chD` by Ioannis Emiris, U.C. Berkeley
- `Hull` by Ken Clarkson, Bell Labs
- `Porta` by Thomas Christof, Heidelburg University and Andreas Loebel, Konrad-Zuse-Zentrum fur Informatik (ZIB).
- `cdd` by Komei Fukuda, ETH Zurich, Switzerland and University of Tsukuba, Japan

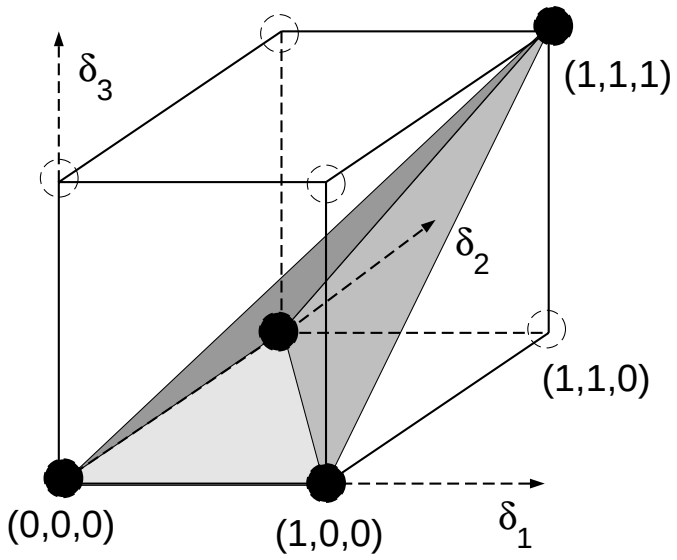


Figure 2.1: Convex hull of the rows of the truth table of $X_3 = X_1 \wedge X_2$.

- `lrs (rs, qrs)` by David Avis, McGill University (`ftp://mutt.cs.mcgill.ca/pub/C`) (Avis, 1997).

Example 2.6:
Consider the Boolean function $X_3 = X_1 \wedge X_2$, whose truth table is given in Table 2.4. The rows of the truth table are represented as points in $\mathbb{R}^3$ in Fig. 2.1. Their convex hull, computed by using the package `lrs`, coincides with the linear inequalities mentioned in the second rule of Table 2.1. □

Comparison of the methods

There are cases where the substitution method gives a smaller number of inequalities than the other methods (Cavalier et al., 1990). However, the complexity of introducing additional binary variables should be avoided. The other two methods do not require the introduction of any additional variable. However, they require pre-processing, either

for building up the truth table or for deriving the conjunctive normal form. Especially, building up the complete truth table and deriving the convex hull of its rows can be a computationally expensive task, even though efficient software is available for this purpose. Note, however, that it is a task that can be done offline. In some cases the truth table might contain only a small number of rows, because some combinations of the independent variables can be excluded a priori.

Using the truth table method or the transformation to CNF for finding the inequalities representing a Boolean formula do not have to give the same results. From Theorem 2.1 it holds that $P_{CH} \subseteq P_{CNF}$. In other words, all integral vertices of P_{CNF} are valid points of the corresponding Boolean formula, however P_{CNF} can have nonintegral vertices. There exist Boolean formulas, for which $P_{CH} \neq P_{CNF}$, even if P_{CNF} is derived from a minimal CNF.

Example 2.7:
Consider the Boolean formula described by:

$$(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) \tag{2.19}$$

Since (2.19) is in CNF, we can use (2.16) to get the corresponding inequalities as

$$P_{CNF} = \{x \in [0,1]^3 : \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\} \tag{2.20}$$

The truth table of (2.19) contains the rows $[1,1,0], [1,0,1], [0,1,1], [1,1,1]$. The Boolean formula (2.19) is characterized by the fact that at least two Boolean variables are true. It can therefore be shown that

$$P_{CH} = \{x \in [0,1]^3 : x_1 + x_2 + x_3 \geq 2\} \tag{2.21}$$

While in P_{CH} all vertices are integral by construction, P_{CNF} includes one vertex at $(0.5, 0.5, 0.5)$ that is not integral. This example illustrates that the inequal-

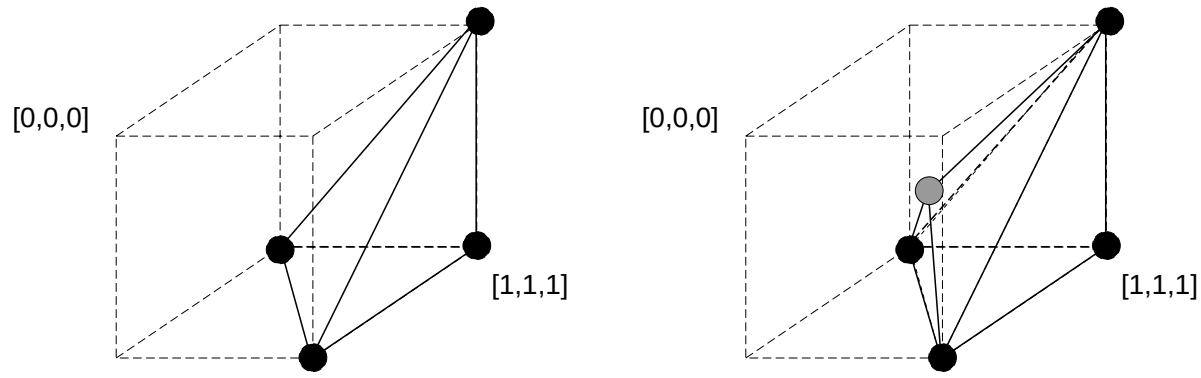


Figure 2.2: Left: P_{CH} of the rows of the truth table of (2.19), Right: P_{CNF} of (2.19), the light vertex is not integral

ities stemming from the CNF can describe a nonminimal representation of the relaxation of a logic proposition, because of the presence of nonintegral vertices.

By direct enumeration, it can be shown that besides (2.19) there is only one[4] Boolean formula involving three Boolean variables, for which $P_{CH} \neq P_{CNF}$, namely

$$(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3) \tag{2.22}$$

For all other Boolean Formulas in $\{0,1\}^3$ it holds that $P_{CH} = P_{CNF}$. □

Clausal Inequalities

A set of inequalities $Ax \geq b$ is *clausal*, if each element of A is 0, 1 or -1, and each b_i satisfies

$$b_i = 1 - \#\{\alpha : \alpha = A_{ij} = -1,\ j = 1 \dots n\} \tag{2.23}$$

where A_{ij} is the j-th element of row i in A and # denotes the cardinality of a set (Chandru and Hooker, 1999). The set of inequalities (2.16) associated with a CNF is clausal by construction.

[4] not considering permutation of variables

Theorem 2.2 *Let F be a Boolean formula. If $P_{CH}(F)$ is given by clausal inequalities, then there exists a CNF, for which $P_{CH}(F) = P_{CNF}(F)$.*

Proof. If the H-representation of $P_{CH}(F)$ has only clausal inequalities, then each one of them can be mapped to a maxterm of a CNF. □

For dimensions up to 3, it holds that if f is a Boolean function, then P_{CNF} has only integral vertices[5], i.e. $P_{CH} = P_{CNF}$. This can be seen by enumeration. However, this rule does not extend to higher dimensions. As a counterexample, in Appendix A we report a Boolean function $f : \{0,1\}^4 \rightarrow \{0,1\}$ for which P_{CH} has a nonclausal inequality.

We apply these methods to an example in Section 3.2.6.

2.4.2 Mixed Logic-Continuous Propositions

In this section the results of Section 2.4.1 are extended to mixed continuous-logic propositions. We consider the same three approaches that were outlined for Boolean formulas and show their application to propositions involving both continuous and binary variables.

A *Generalized Boolean Formula* (GBF) is the extension of a Boolean formula obtained by allowing linear inequalities $\tilde{X}_i = [f_i(x) \leq 0]$ as additional terms in a Boolean formula. A *Generalized Conjunctive Normal Form* (GCNF) is defined accordingly, as the expression in Equation (2.24)

$$\bigwedge_{j=1}^{p} \left(\bigvee_{i=1}^{m_j} [f_{ij}(x) \leq 0] \vee \bigvee_{k=1}^{n_j} [\delta_{kj} = 1] \right) \tag{2.24}$$

Every GBF admits a GCNF equivalent representation. We show that every GCNF can be translated into mixed integer inequalities, provided that a sufficient number of auxiliary variables is introduced. We develop a systematic way for translating complex logical

[5]Note that the P_{CNF} with a nonintegral vertex (2.20) is defined by a Boolean formula, but not a Boolean function.

propositions into mixed integer inequalities. An assignment $(x, \delta) \in \mathbb{R}^{n_1} \times \{0,1\}^{n_2}$, that renders (2.24) true, is called a *valid point*.

Motivation

In the sequel, we assume that upper and lower bounds are known on real affine functions $f_i(x) = a_i'x - b_i$, $f_i : \mathbb{R}^n \mapsto \mathbb{R}$. Consider the logic disjunction

$$\bigvee_{i=1}^{m} [f_i(x) \leq 0] \tag{2.25}$$

which in general defines a nonconvex set in $\mathbb{R}^n$, e.g. the set of Figure 2.3, corresponding to the disjunction

$$[x_1 \leq 0] \vee [x_2 \leq 0] \tag{2.26}$$

where $m_1 \leq x_1 \leq M_1$, $m_2 \leq x_2 \leq M_2$. The set in Figure 2.3 is not convex, and therefore it

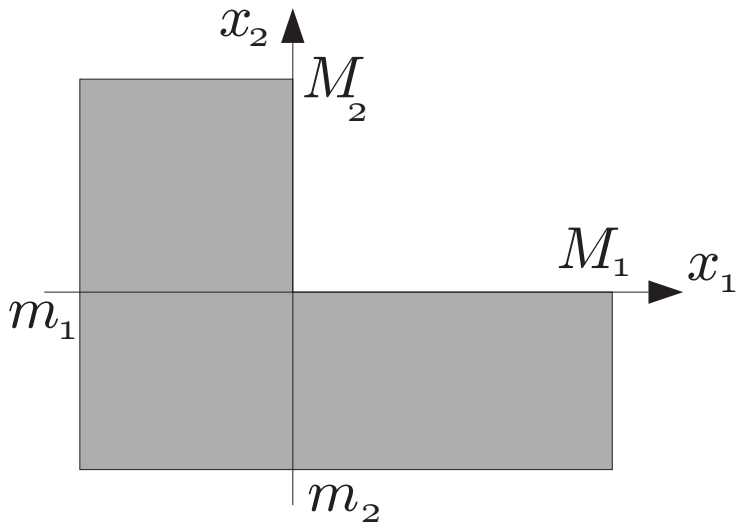


Figure 2.3: Feasible set defined by (2.26)

cannot be written as a system of linear constraints $\{x \in \mathbb{R}^n : Cx \leq d\}$. Indeed, the smallest convex set containing the feasible region of (2.26) contains points in the first quadrant

excluded by (2.26). However, by introducing additional binary variables $\delta \in \{0,1\}$ it becomes possible to represent it as a set of mixed integer constraints. Using Tables 2.1 and 2.2, we can introduce $\delta_1, \delta_2 \in \{0,1\}$ as

$$\delta_1 \triangleq [x_1 \leq 0] \tag{2.27}$$

$$\delta_2 \triangleq [x_2 \leq 0] \tag{2.28}$$

The statement in Equation (2.26) becomes

$$\delta_1 \vee \delta_2 \tag{2.29}$$

Each one of the three propositions (2.27), (2.28) and (2.29) can be translated into inequalities using Tables 2.1 and 2.2. This shows that in the space $(x_1, x_2, \delta_1, \delta_2)$, i.e. in a subset of $\mathbb{R}^4$, the feasible set of (2.26) can be represented as $\{x \in \mathbb{R}^2 \times \{0,1\}^2 : Cx \leq d\}$. This implies that after the relaxation of the integrality constraints to $0 \leq \delta_i \leq 1$, the feasible set of (2.26) becomes a convex polyhedral set in $\{x \in \mathbb{R}^4 : Cx \leq d\}$. Increasing the dimension of the space by binary variables permits a convex representation of a set that in the original coordinates of interest is not convex.

Substitution Method

Similar to the case of logical relations, the substitution method consists of iteratively applying the rules of Tables 2.1 and 2.2 to substitute elementary logical propositions with additional binary auxiliary variables. With Equations (2.27), (2.28) and (2.29) we illustrated the method. Contrary to the substitution for purely logical function in Section 2.4.1, we have shown with the logical proposition in (2.26) that the introduction of auxiliary variables may be necessary for mixed logic continuous translations. The substitution method may however introduce more variables than required, as will be shown next.

The Generalized Conjunctive Normal Form

Here we provide a general procedure for translating GBFs into a set of mixed integer linear inequalities. Theorem 2.3 is the main contribution of this Section. The proof of Theorem 2.4 contains some constructive hints for this translation.

Theorem 2.3 *The GCNF with affine real-valued functions* $f_{ij}(x)$

$$\bigwedge_{j=1}^{p}\left(\bigvee_{i=1}^{m_j}[f_{ij}(x)\leq 0]\vee\bigvee_{k=1}^{n_j}[\delta_{kj}=1]\right) \tag{2.30}$$

can be translated into a set of mixed integer inequalities adding $\sum_{j=1}^{p}\lceil\log_2(m'_j+1)\rceil$ *binary variables, where* $m'_j\leq m_j$ *is the number of non-redundant constraints in the set* $\{x : f_{ij}(x)>0, i=1,\dots,m_j\}$

Before proving Theorem 2.3 we need some preliminaries. A constraint $[f_j(x)\leq 0]$ is *redundant* for the proposition (2.25), if

$$\left\{x:\bigvee_{i=1}^{m}[f_i(x)\leq 0]\right\}=\left\{x:\bigvee_{i=1,i\neq j}^{m}[f_i(x)\leq 0]\right\} \tag{2.31}$$

Theorem 2.4 *The logical proposition with* m *affine functions* $f_i(x)$

$$\bigvee_{i=1}^{m}[f_i(x)\leq 0] \tag{2.32}$$

can be translated into a set of mixed integer inequalities by adding $\ell=\lceil\log_2(m')\rceil$ *binary variables, where* $m'\leq m$ *is the number of non-redundant constraints*

Proof. The proof is constructive and involves the following steps:

1. Define the closure of the infeasible set B as[6]

$$B=\{x:\bigwedge_{i=1}^{m}[f_i(x)\geq 0]\}. \tag{2.33}$$

Since all f_i are affine, B is convex.

[6]Note that we include the edge points in B.

2. Eliminate the redundant constraints. This can be done by solving for each $i = 1 \dots m$ a linear program of the form

$$\begin{array}{lll} J_c(i) & = \min_x & f_i(x) \\ & \text{s.t.} & \bigwedge_{j=1,\ j\neq i}^{m} [f_j(x) \geq 0]. \end{array} \tag{2.34}$$

If $J_c(i) \geq 0$, the i-th constraint is redundant. This gives the number of non-redundant constraints m'. Note that the disjunction (2.32) can be written equivalently only using the non-redundant constraints. This step aims at having a minimal representation of (2.32). Moreover, m' is the number of facets of the polytope B.

3. According to Bemporad, Morari, Dua and Pistikopoulos (1999, Theorem 3), the set $X \backslash B$ can be partitioned into m' disjoint convex sets C_j. Each set can be described by the inequalities

$$C_j = \{x : A_j x \leq b_j\}$$

where by appropriate numbering, we have $A_j \in \mathbb{R}^{j \times s}$ and $b_j \in \mathbb{R}^j \quad (j = 1 \dots m')$.

4. Temporarily introduce m' binary variables δ_j with the property that

$$[\delta_j = 1] \leftrightarrow [x \in C_j].$$

According to the rules in Section 2.2, the latter relations can be translated into mixed integer inequalities as:

$$A_j x - b_j \leq M_j(1 - \delta_j) \quad (j = 1 \dots m') \tag{2.35}$$

$$\sum_{i=1}^{m'} \delta_i = 1. \tag{2.36}$$

5. Code the regions C_i as follows: Introduce $\ell = \lceil \log_2(m') \rceil$ binary variables $\mu_1 \dots \mu_\ell$, such that

$$[\delta_j = 1] \ \leftrightarrow \ [j \ = \ 1 + \sum_{i=0}^{\ell-1} 2^i \mu_{i+1}]. \tag{2.37}$$

Use them in (2.35) to obtain:

$$\begin{aligned} A_1 x - b_1 &\leq M_1(\ell - (1-\mu_1) - \ldots - (1-\mu_\ell)) \\ A_2 x - b_2 &\leq M_2(\ell - \mu_1 - \ldots - (1-\mu_\ell)) \\ \vdots &\quad \vdots \\ A_{m'} x - b_{m'} &\leq M_{m'}(\ell - \mu_1 - \ldots - \mu_\ell) \end{aligned}$$

If $m' < 2^\ell$, add constraints to exclude the combinations of $(\mu_1 \ldots \mu_\ell)$ that are not used to code any region. For the representation with δ_j, (2.36) expresses the fact that exactly one δ_j is one for each data point x. This is not required when working with $(\mu_1 \ldots \mu_\ell)$, since each code $(\mu_1 \ldots \mu_\ell)$ either identifies a region, or it is excluded.

□

The number of inequalities resulting from the algorithm described above is given by

- (i) the bounds on either x or each function $f_i(x)$
- (ii) $\sum_{i=1}^{m'} i = \frac{m'(m'+1)}{2}$ constraints defining $\mu_1, \ldots, \mu_\ell$, and
- (iii) the constraints excluding the unused combinations of $\mu_1, \ldots, \mu_\ell$

Note that neglecting the redundancy removal procedure in step 2 increases the number of binary auxiliary variables to $\lceil \log_2(m) \rceil$, but does not change the algorithm itself.

Corollary 2.1 *The logical proposition with m affine functions $f_i(x)$*

$$\bigvee_{i=1}^{m} [f_i(x) \leq 0] \ \vee \ \bigvee_{k=1}^{n} [\delta_k = 1] \tag{2.38}$$

can be translated into a set of mixed integer inequalities adding $\ell' = \lceil \log_2(m'+1) \rceil$ binary variables, where $m' \leq m$ is the number of non-redundant constraints in the disjunction $\bigvee_{i=1}^{m} [f_i(x) \leq 0]$.

Proof. Using the constructive arguments of Theorem 2.4, we can model the terms involving $\bigvee_{i=1}^{m}[f_i(x) \leq 0]$. Contrary to Theorem 2.4, we have to assign also a combination of $(\mu'_1 \dots \mu'_\ell)$ to the infeasible set B, because (2.38) can be true even if $x \in B$. For computational simplicity we assign the code $[0, \dots, 0]$ to B. Therefore the number of regions to code with $(\mu'_1 \dots \mu'_\ell)$ is increased by one. The disjunction (2.38) can then be imposed by the inequality

$$\sum_{i=1}^{\ell'} \mu_i + \sum_{k=1}^{m} \delta_k \geq 1 \tag{2.39}$$

□

Finally, the last step consists of translating general GBFs to inequalities over continuous and binary variables. This can be done by considering the inequalities $[f_i(x) \leq 0]$ as symbolic Boolean variables and manipulating the overall function according to Section 2.4.1:

1. Replace each inequality $[f_i(x) \leq 0]$ over continuous variables with a Boolean variable y_i.
2. Find the CNF of the expression using the methods described in Section 2.4.1.
3. Replace back the Boolean variables y_i with $[f_i(x) \leq 0]$.
4. Apply the theorem above

Note that the effectiveness of the approach presented here can be increased, if the scheme is given the possibility to recognize common parts in the logic relations, like the multiple occurrence of an inequality $[f_{ij}(x) \leq 0]$ in (2.30).

Proof. (of Theorem 2.3) Using Corollary 2.1, each term in the conjunction (2.30) can be translated into inequalities. The overall conjunction can then be modelled by contemporarily imposing all constraints obtained. □

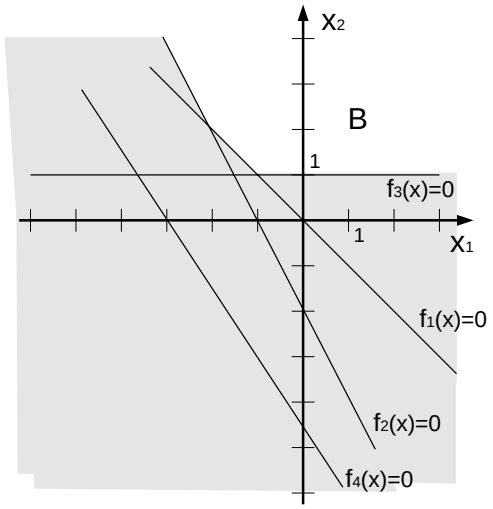


Figure 2.4: Feasible points represented by (2.40)

Example 2.8:

Consider the following GBF

$$\bigvee_{i=1}^{4}[f_i(x) \leq 0] \tag{2.40}$$

where $x = [x_1, x_2]^T \in [-10, 10]^2 \subset \mathbb{R}^2$, f_i: $\mathbb{R}^2 \to \mathbb{R}$ and

$$f_1(x) = x_1 + x_2 \tag{2.41}$$

$$f_2(x) = 2x_1 + x_2 + 2 \tag{2.42}$$

$$f_3(x) = x_2 - 1 \tag{2.43}$$

$$f_4(x) = 1.5x_1 + x_2 + 4.5 \tag{2.44}$$

The set of feasible points to represent is the nonconvex, shaded area in Figure 2.4. We apply the procedure described in the proof of Theorem 2.4 to find the inequalities describing (2.40), following the same steps and terminology introduced there.

1. The set B is given by $B = \{x : \bigwedge_{i=1}^{4}[f_i(x) \geq 0]\}$ and it is shown as the unshaded area in Figure 2.4.

2. Solving (2.34) yields the results in Table 2.5. Therefore $i = 4$ denotes a

i	1	2	3	4
$J_c(i)$	-0.5	-7.33	-11	3.5

Table 2.5: Solution of (2.34) for the example (2.40)

redundant constraint and $m' = 3$.

3. The sets C_j are:

$$C_1 = \{x \in \mathbb{R}^2 : \begin{bmatrix} 1 & 1 \end{bmatrix} x \leq 0\} \tag{2.45}$$

$$C_2 = \{x \in \mathbb{R}^2 : \begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix} x \leq \begin{bmatrix} 0 \\ -2 \end{bmatrix}\} \tag{2.46}$$

$$C_3 = \{x \in \mathbb{R}^2 : \begin{bmatrix} -1 & -1 \\ -2 & -1 \\ 0 & 1 \end{bmatrix} x \leq \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}\} \tag{2.47}$$

4. Using the bounds on x, the expressions (2.35) become:

$$\begin{bmatrix} 1 & 1 \end{bmatrix} x \leq 20(1 - \delta_1) \tag{2.48}$$

$$\begin{bmatrix} -1 & -1 \\ 2 & 1 \end{bmatrix} x - \begin{bmatrix} 0 \\ -2 \end{bmatrix} \leq \begin{bmatrix} 20 \\ 32 \end{bmatrix} (1 - \delta_2) \tag{2.49}$$

$$\begin{bmatrix} -1 & -1 \\ -2 & -1 \\ 0 & 1 \end{bmatrix} x - \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \leq \begin{bmatrix} 20 \\ 28 \\ 9 \end{bmatrix} (1 - \delta_3) \tag{2.50}$$

5. Here it holds that $\ell = 2$ and we code the regions as:

$$[\delta_1 = 1] \Leftrightarrow [\mu_1 \ \mu_2] = [0 \ 0] \tag{2.51}$$

$$[\delta_2 = 1] \Leftrightarrow [\mu_1 \ \mu_2] = [1 \ 0] \tag{2.52}$$

$$[\delta_3 = 1] \Leftrightarrow [\mu_1 \ \mu_2] = [0 \ 1] \tag{2.53}$$

Finally, defining $X = [x_1 \ \ x_2 \ \ \mu_1 \ \ \mu_2]^T$ and using the expressions above,

we obtain the desired inequalities as

$$\begin{bmatrix} 1 & 1 & -20 & -20 \\ -1 & -1 & 20 & -20 \\ 2 & 1 & 30 & -30 \\ -1 & -1 & -20 & 20 \\ -2 & -1 & -30 & 30 \\ 0 & 1 & -10 & 10 \end{bmatrix} X \leq \begin{bmatrix} 0 \\ 20 \\ 30 \\ 20 \\ 30 \\ 10 \end{bmatrix} \tag{2.54}$$

□

Geometrical Method

We have motivated that a geometrical interpretation of introducing additional binary variables is an enlargement the dimension of the space considered. In the higher dimensional space, the specified region is a convex set. The projection onto the space of original variables is still nonconvex. While this is an appealing interpretation, it is not immediate to generalize it to a procedure for finding the inequalities, beyond low dimensional spaces. Indeed, it is not clear a priori, how to enlarge the space with additional binary variables. For the GBF (2.26) we sketch in Figure 2.5 the effect of applying the idea of the following algorithm:

1. Introduce $n_{aux} = 1$ auxiliary binary variable.
2. Split the nonconvex set into disjoint convex sets, the union of which gives in the original set.
3. Assign to each subset so obtained a different combination of the auxiliary binary variables.
4. Compute in the higher dimensional space the convex hull of the vertices.
5. If the convex hull contains points excluded by the logic proposition,

- increase the number n_{aux} of auxiliary binary variables in 1. or
- choose a different splitting in 2. or
- choose a different code in 3.

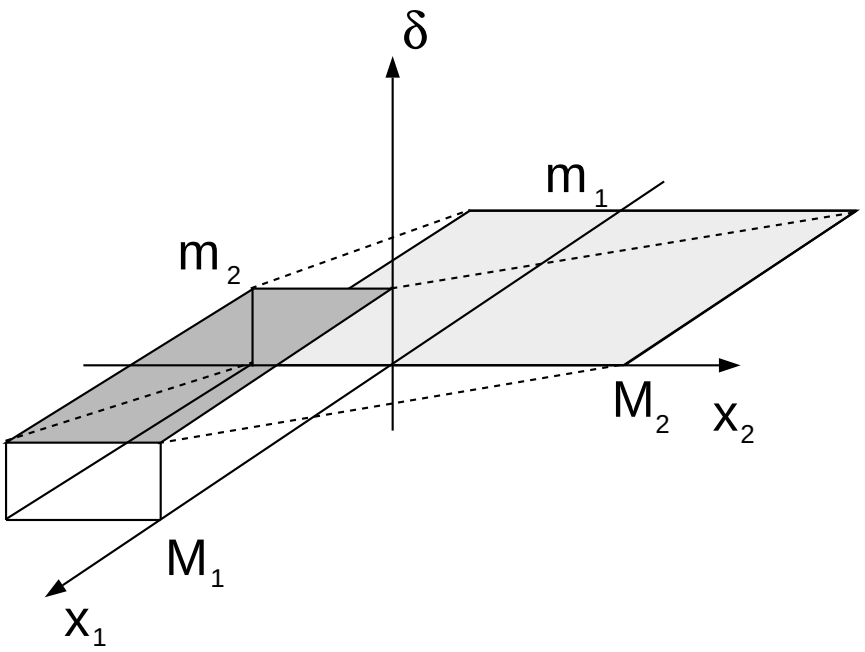


Figure 2.5: Feasible set defined by (2.26) in a three dimensional space

In general, we do not recommend to rely on these geometrical interpretations for translating GBFs into inequalities, due to the unsystematic actions to be taken in point 5.

2.4.3 Summary

In Sections 2.4.1 and 2.4.2 we have motivated a fundamental difference between the translation of purely logical relations and mixed logic continuous relations into inequalities: The former can be translated without adding additional variables, while the latter can require the introduction of Boolean variables. By construction we have shown the following statement.

Theorem 2.5 *Let $\mathcal{P}$ be a logical proposition involving n_c continuous and n_b binary variables. Let $V \subset X_V = \mathbb{R}^{n_c} \times \{0,1\}^{n_b}$ be the set of valid points of $\mathcal{P}$ and assume that V is compact set.*
Then, by introducing n_a additional binary variables, V can always be delimited by inequalities in a space $\tilde{X} = \mathbb{R}^{n_c} \times \{0,1\}^{n_a+n_b}$ of dimension $n_a + n_b + n_c$ such that in the higher dimensional space the convex hull of valid points does not contain any invalid points.

Moreover, if $n_c = 0$ then $n_a = 0$

Proof. Theorem 2.3 shows by construction that adding at most

$$n_a = \sum_{j=1}^{p} \lceil \log_2(m'_j + 1) \rceil$$

binary variables, allows to find the desired inequalities. The statement for purely logical functions $n_c = 0$ has been shown either in Theorem 2.1 or in (2.16). □

2.5 The Tolerance ϵ of Mixed Logic Dynamical Systems and Well Posedness

For the translation of some logical propositions into inequalities, it is required to introduce small tolerances $\epsilon > 0$, as can be seen in Table 2.2. The main justification for these variables is the ability to express the inequalities as non-strict, rather than as strict inequalities. This allows their inclusion as constraints in optimization problems.

This section describes a couple of consequences of this approach for the constraints so obtained and it shows, how well posedness depends on the choice of ϵ.

Example 2.9:

Consider the for instance the proposition:

$$[\delta = 1] \leftrightarrow [f(x) \leq 0] \tag{2.55}$$

and its translation

$$f(x) \leq M - M\delta \tag{2.56}$$

$$f(x) \geq \epsilon + (m - \epsilon)\delta \tag{2.57}$$

It is easy to verify that inequalities (2.56) and (2.57) actually define the relations:

$$[\delta = 1] \leftrightarrow [m \leq f(x) \leq 0] \tag{2.58}$$

$$[\delta = 0] \leftrightarrow [\epsilon \leq f(x) \leq M] \tag{2.59}$$

If $f(x)$ takes on values in the range

$$0 < f(x) < \epsilon \ , \tag{2.60}$$

the variable δ remains undefined and the inequalities (2.56) and (2.57) are infeasible for any pairs $(\delta, f(x)) \in \{0, 1\} \times (0, \epsilon)$. Indeed, if e.g. $f(x) = \frac{\epsilon}{2}$, then (2.56) and (2.57) become

$$\frac{\epsilon}{2} \leq M - M\delta \tag{2.61}$$

$$-\frac{\epsilon}{2} \geq (m - \epsilon)\delta \tag{2.62}$$

Here, (2.61) is feasible for $\delta = 0$ only, whereas (2.62) can only be satisfied with $\delta = 1$. Note that if (2.60) is guaranteed, never to occur, no feasibility problems are expected. In this case ϵ can be chosen as the representable machine precision. □

With a geometrical interpretation we can visualize, whether there are alternatives for representing (2.55), apart from (2.56) and (2.57). We plot the set of valid points in the

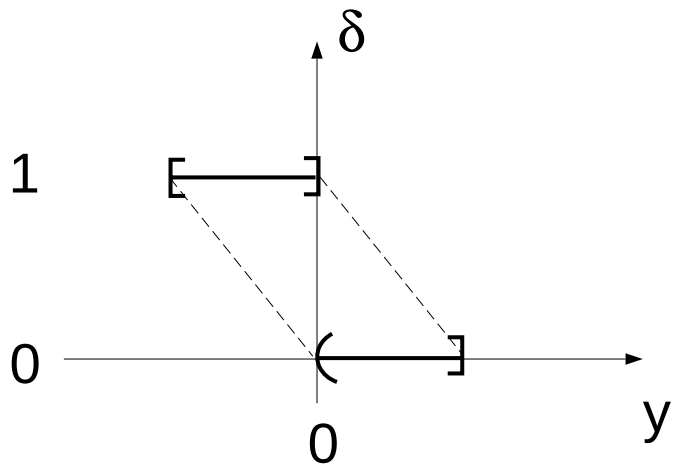


Figure 2.6: Set of valid points of (2.55) in the (δ, y) plane represented as the bold lines

(δ, y) plane, where $y = f(x)$, as shown in Figure 2.6. From Figure 2.6, we see that the point $(\delta, y) = (0, 0)$ is element of the convex hull of valid points, even though it is invalid. Note that extending the space with additional binary variables does not help in this case, since the set of valid points is an open set. This does not meet the assumption of compactness in Theorem 2.5. For any extension of the domain space (δ, y) with auxiliary variables, the set of valid points is a nonconvex set.

One way to represent this relation as inequalities is to sacrifice well posedness. The solution (2.56) and (2.57) modifies the set of valid points, such that there is a gap, in which δ is undefined, as shown in Figure 2.7. This means that strictly speaking the system is not completely well posed, since for $y \in (0, \epsilon)$ no δ is defined. If we are willing to sacrifice well posedness, we can propose an alternative translation, as shown in Figure 2.8. Here we do not have ranges of the continuous variable y, where the binary variable δ is undefined, but we have both possible values of δ, if $y = 0$. The inequalities in this case are obtained by setting $\epsilon \leq 0$ in (2.56) and (2.57).

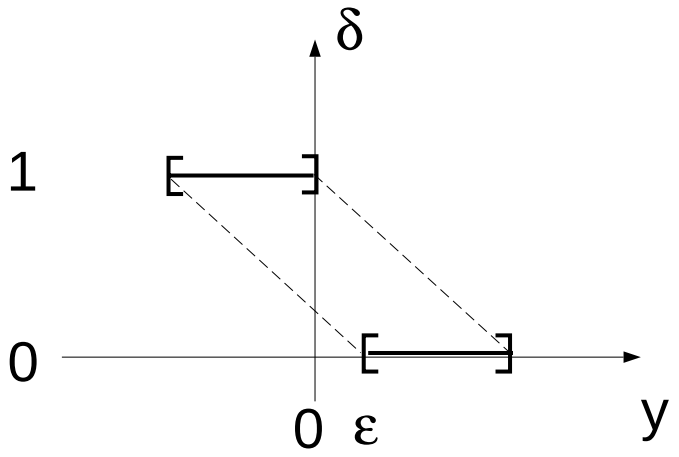


Figure 2.7: Set of valid points of (2.55) in the (δ, y) plane using the ϵ approach. The valid points are represented as bold lines.

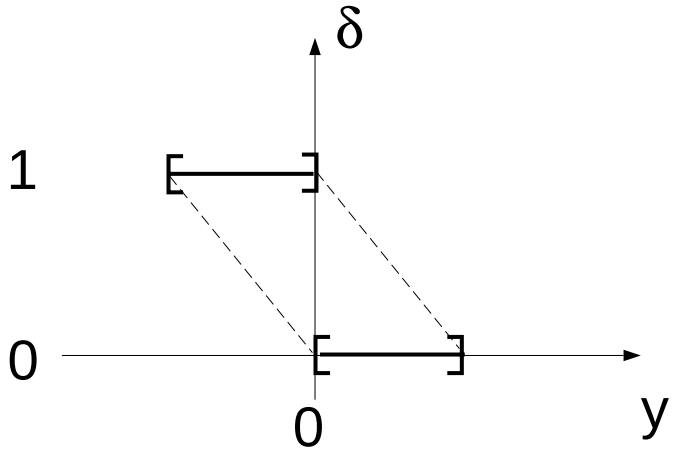


Figure 2.8: Set of valid points of (2.55) in the (δ, y) plane without well posedness. The valid points are represented as bold lines.

2.6 Modelling Capabilities of Mixed Logic Dynamical Systems

In this section we mention a few general considerations about the modeling capabilities of MLD systems. In Chapter 3 we model two practical systems within the MLD framework.

2.6.1 Piecewise Affine Systems

One class of hybrid systems, having a close connection to MLD systems, is the class of *piecewise affine* (PWA) systems, considered in Chapter 6. A PWA system is defined as follows. Assume that the state space is partitioned into disjoint cells. In each state space cell the dynamics is governed by a different affine dynamical system. PWA systems are therefore described by the state-space equations:

$$\begin{aligned} x_{k+1} &= A_i x_k + B_i u_k + a_i \\ y_k &= C_i x_k + D_i u_k + c_i \ , \text{ for } \left[\begin{smallmatrix} x_k \\ u_k \end{smallmatrix}\right] \in \mathcal{X}_i \\ (x,u) &\in \mathbb{X} \end{aligned} \tag{2.63}$$

where the state+input set $\mathbb{X} \subset \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell} \times \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ is a polyhedron containing the origin, $\{\mathcal{X}_i\}_{i=1}^s$ is a polyhedral partition[7] of $\mathbb{X}$ and a_i, c_i are constant vectors of suitable dimension. We refer to each $\mathcal{X}_i$ as a *cell*.

In (Bemporad and Morari, 1999a) it is shown, how a general PWA system can be written as an MLD system. We point out that *one* single MLD model can represent the dynamics of a PWA system with an *arbitrary* number s of linear subsystems. Logic propositions modelled as linear inequalities, as shown in this chapter, are determining which dynamics is currently active. The converse to this result is stated in the next proposition.

Theorem 2.6 ((Bemporad, Ferrari-Trecate and Morari, 2000)) *Consider generic tra-*

[7]Each set $\mathcal{X}_i$ is a (not necessarily closed) convex polyhedron s.t. $\mathcal{X}_i \bigcap \mathcal{X}_j = \emptyset$, $\forall i \neq j$, $\bigcup_{i=1}^s \mathcal{X}_i = \mathbb{X}$.

jectories x_k, u_k, y_k of the MLD system (2.8). Then, there exist a polyhedral partition $\{\mathcal{X}_i\}_{i=1}^s$ of the state+input space $\mathbb{X}$ and 5-tuples $(A_i, B_i, C_i, D_i, a_i, c_i)$, $i = 1, \ldots, s$, such that x_k, u_k, y_k satisfy (2.63).

Therefore, PWA systems are equivalent to MLD systems, in the sense that they can describe the same model behaviour. Theorem 2.6 was proven in (Bemporad, Ferrari-Trecate and Morari, 2000) by using a constructive argument that allows the explicit computation of the sets $\mathcal{X}_i$ and the matrices $(A_i, B_i, C_i, D_i, a_i, c_i)$ defining the PWA system.

2.6.2 Equivalences of Classes

In (Heemels et al., 2001) the equivalence between system classes is taken a step further. It is shown that under some technical conditions, five classes of models are equivalent to each other. These are:

- MLD (Mixed Logic Dynamical) Systems
- LC (Linear Complementarity) Systems
- ELC (Extended Linear Complementarity) Systems
- PWA (Piecewise Affine) Systems
- MMPS (Max-Min-Plus-Scaling) Systems

LC systems in discrete time are given by the equations:

$$x(k+1) = Ax(k) + B_1u(k) + B_2w(k) \tag{2.64a}$$

$$y(k) = Cx(k) + D_1u(k) + D_2w(k) \tag{2.64b}$$

$$v(k) = E_1x(k) + E_2u(k) + E_3w(k) + g_4 \tag{2.64c}$$

$$0 \leq v(k) \perp w(k) \geq 0 \tag{2.64d}$$

Equation (2.64d) expresses the fact that each component of $v(k)$ and $w(k)$ is nonnegative and the scalar product of $v(k)$ and $w(k)$ is zero.

ELC systems are given in discrete time:

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 d(k) \tag{2.65a}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 d(k) \tag{2.65b}$$

$$E_1 x(k) + E_2 u(k) + E_3 d(k) \leq g_4 \tag{2.65c}$$

$$\sum_{i=1}^{p} \prod_{j \in \phi_i} (g_4 - E_1 x(k) - E_2 u(k) - E_3 d(k))_j = 0 \tag{2.65d}$$

An MMPS system is given by:

$$x(k+1) = M_x(x(k), u(k), d(k)) \tag{2.66a}$$

$$y(k) = M_y(x(k), u(k), d(k)) \tag{2.66b}$$

$$M_c(x(k), u(k), d(k)) \leq c \tag{2.66c}$$

where M_x, M_y, M_c are MMPS expressions. An MMPS expression M of the variables x_i is defined by the rule

$$M \triangleq x_i \mid \alpha \mid \max(M_k, M_i) \mid \min(M_k, M_i) \mid M_k + M_i \mid \beta M_k \tag{2.67}$$

where M_i are MMPS expressions.

In this work, we will limit ourselves to the equivalence of MLD and PWA systems. It is however useful to keep in mind the opportunities given by the equivalent representation of a system in different modelling frameworks.

2.6.3 Example: Finite State Machines in MLD and PWA Form

As an illustration for the equivalence of MLD systems and PWA systems, we describe the model structure for a finite state machine in both frameworks. Assume that a synchronous

finite automaton $\mathcal{A}$ is given by the five-tuple

$$\mathcal{A} = \{X, U, f, x_0, t\} \tag{2.68}$$

where the symbols have the following meaning:

X The finite state space of the automaton

U The finite input space of the automaton

f The state transition function

$$f: \qquad X \times U \longrightarrow X \tag{2.69}$$

$$(x(k), u(k)) \longmapsto x(k+1) \tag{2.70}$$

x_0 The initial state

t The clock determining the state transitions

This system can be modelled in the MLD form, provided that the state transitions occur synchronously at a fixed rate given by the sampling period. The first step is to define a coding of the state values in X and input values in U as subsets of $\{0,1\}^{n_x}$ and $\{0,1\}^{n_u}$, respectively. Note that the state update given by the function f is determined with logical propositions. It is therefore required to formulate the state update as an update of a binary variable in the MLD model, rather than a direct update given by equation (2.8a) in the MLD form. We define for each binary state

$$\delta(k) = x(k+1) \tag{2.71}$$

and we formulate the state update as a logical proposition involving $(u(k), x(k), \delta(k))$. As we have seen in the previous sections, these propositions can be translated into inequalities that are added as constraints to the MLD model. Summarizing, the MLD model of a finite

state machine has the following structure:

$$x(k+1) = \delta(k) \tag{2.72a}$$

$$y(k) = x(k) \tag{2.72b}$$

$$E_2\delta(k) \leq E_1u(k) + E_4x(k) + E_5 \tag{2.72c}$$

The procedure to derive the piecewise affine form of a finite state machine is outlined next. First define the state and input set

$$\Omega = X \times U \tag{2.73}$$

The state and input space partition $\cup_{i=1}^{s}\mathcal{X}_i = \Omega$ of the PWA system is defined such that each element of Ω is in exactly one region $\mathcal{X}_i$. The PWA system has zero matrices A_i and B_i. The state update is given by the drift term a_i in the state update formula of (2.63), which is uniquely determined by the current state and current input. The PWA model has therefore the structure

$$x(k+1) = a_i \tag{2.74a}$$

$$y(k) = x(k) \tag{2.74b}$$

$$\text{for } \left[\begin{smallmatrix} x_k \\ u_k \end{smallmatrix}\right] \in \mathcal{X}_i \subset \Omega$$

The information about the state update transitions is coded in the partition of the state and input space.

2.7 HYSDEL – The Hybrid Systems Description Language

The derivation of MLD models "by hand" is a tedious task and involves the repeated application of the rules presented in the previous sections. This procedure is further complicated if one aims at finding a good model in terms of a low number of variables or inequalities

involved. To simplify the modelling for the user, a compiler was developed for the automated translation of a high level modelling description into the MLD form. The aim of the compiler is to generate the matrices A, B_i, C, D_i and E_i in (2.8). These matrices are generated in Matlab syntax (Mat).

The problem specification language to the compiler is HYSDEL (HYbrid System DEscription Language). A preliminary work about HYSDEL is reported in (Anlauff et al., 1999). Torrisi et al. (2000) describe the recent status of the language. In Appendix B we included an example of a HYSDEL listing. At present, HYSDEL translates logical propositions into inequalities by the CNF method.

HYSDEL is tailored for modeling MLD systems, but it is by no means the only modelling language for hybrid systems. Other languages and tools are described e.g. in (Benveniste et al., 1993; Fabian, 1999). The connection of HYSDEL to other existing modelling and simulation tools for hybrid systems is under investigation (Torrisi, 2002). The tool LPL is described in (Huerlimann, 2001). It allows the modelling of systems containing logic components and translates the model into the MPS format (Murtagh, 1981), a widespread formalism for the specification of mathematical optimization problems.

2.8 Steady States for Mixed Logic Dynamical Systems

MLD systems are capable to model rich system behaviours. They can exhibit several properties that are typical for nonlinear systems, like multiple isolated equilibria or limit cycles. For instance, Kennedy (1993) analyzes an electrical circuit described by piecewise linear equations, that exhibits chaotic behaviour. In this section selected properties concerning steady states are mentioned that have been encountered in the case studies considered in Chapter 3. Properties like deadlocks or livelocks, typically encountered in discrete event

systems, can be found in MLD models as well, however the detection of these phenomena and their avoidance will not be considered in this work.

A steady state value x_f for an MLD system can be determined by solving the following mixed integer program (see Chapter 5):

$$\min_{x_f,u_f,\delta_f,z_f} ||y_f - r||_{\rho_y} + ||x_f||_{\rho_4} + ||u_f||_{\rho_1} + ||z_f||_{\rho_3} + ||\delta_f||_{\rho_2} \tag{2.75}$$

s.t.

$$x_f = Ax_f + B_1u_f + B_2\delta_f + B_3z_f \tag{2.76}$$

$$y_f = Cx_f + D_1u_f + D_2\delta_f + D_3z_f \tag{2.77}$$

$$E_2\delta_f + E_3z_f \leq E_1u_f + E_4x_f + E_5 \tag{2.78}$$

Here $||.||.$ is an arbitrary norm, the vector r is the constant reference, $\rho_i = \rho_i^T$ are nonnegative definite weighting matrices. In general, the complete set of all steady states is defined by the feasible points satisfying (2.76),(2.78). However, since the solution x_f, u_f, δ_f, z_f is typically used for control purposes (see Section 4.1.2), we are interested in a solution of (2.75)-(2.78), where $\rho_y \gg \rho_j$, $(j = 1 \dots 4)$, i.e. a steady state yielding an output close to the reference.

Due to the rich behaviour of general MLD systems, the result $(x_f, u_f, \delta_f, z_f)$ of (2.75) - (2.78) should be further analyzed prior to its usage in a control scheme. Some difficulties that can arise are listed next.

Unreachability of Steady States The steady state obtained with (2.75) - (2.78) can be unreachable. We call a state x_e *unreachable*, if there exists an initial state x_0, for which no feasible input sequence $[u(0), u(1), \dots, u(N)]$ of length N $(N \geq 1)$ exists, such that $x(N) = x_e$ for all N. We refer to Section 3.3.7 for an example of such a system.

Multiplicity of Steady States Even though all weights ρ_i in (2.75) are chosen different from zero, the steady state obtained with (2.75) - (2.78) can be nonunique because of the

presence of binary variables. A simple example for this phenomenon is a piecewise affine system, where the steady state belonging to the affine dynamics of region X_i lies in X_i for all i. Systems can be found, the steady states of which give the same value of (2.75) for all steady states of the subsystems.

Limit Cycles Consider the problem of stabilizing an MLD system to a constant reference r. When searching for a state giving an output y of an MLD system, which stays close to r, (2.75) - (2.78) might give a unique and reachable solution. However, in some cases it turns out that the equilibrium so obtained is far away from the reference. The cumulated error over a finite horizon of length M

$$\sum_{k=1}^{M} ||y_f(k) - r||$$

can be smaller, if y tracks a cycle of states instead of being controlled to a constant value. Such a cycle is generated by the periodical switching of discrete system components. In other words, the system goes through a cycle of states to stay close to the reference, instead of converging to a constant state. The three tank model discussed in Section 3.2 shows this behaviour for certain reference values, as we show in Section 3.2.8.

For systems exhibiting limit cycles, instead of finding one single steady state, we can search for entire state sequences. The optimization is then given by:

$$\min_{\underline{x},\underline{u},\underline{\delta},\underline{z}} ||\underline{y}_f - \underline{r}|| + ||\underline{x}_f||_{\underline{\rho_4}} + ||\underline{u}_f||_{\underline{\rho_1}} + ||\underline{z}_f||_{\underline{\rho_3}} + ||\underline{\delta}_f||_{\underline{\rho_2}} \tag{2.79}$$

s.t.

$$\begin{aligned}
x_f(1) &= Ax_f(0) + B_1u_f(0) + B_2\delta_f(0) + B_3z_f(0) \\
x_f(2) &= Ax_f(1) + B_1u_f(1) + B_2\delta_f(1) + B_3z_f(1) \\
&\vdots \\
x_f(N) &= Ax_f(N-1) + B_1u_f(N-1) + B_2\delta_f(N-1) + B_3z_f(N-1) \\
x_f(0) &= Ax_f(N) + B_1u_f(N) + B_2\delta_f(N) + B_3z_f(N) \\
E_2\delta_f(0) + E_3z_f(0) &\leq E_1u_f(0) + E_4x_f(0) + E_5 \\
&\vdots \\
E_2\delta_f(N) + E_3z_f(N) &\leq E_1u_f(N) + E_4x_f(N) + E_5
\end{aligned}$$

The variables $\underline{x}_f, \underline{u}_f, \underline{\delta}_f, \underline{z}_f$ in (2.79) are sequences of the corresponding variables x_f, u_f, δ_f, z_f of length N, i.e.

$$\underline{x}_f = \begin{bmatrix} x_f(0) \\ \vdots \\ x_f(N) \end{bmatrix} \quad \underline{u}_f = \begin{bmatrix} u_f(0) \\ \vdots \\ u_f(N) \end{bmatrix} \quad \underline{\delta}_f = \begin{bmatrix} \delta_f(0) \\ \vdots \\ \delta_f(N) \end{bmatrix} \quad \underline{z}_f = \begin{bmatrix} z_f(0) \\ \vdots \\ z_f(N) \end{bmatrix} \quad \underline{r} = \begin{bmatrix} r \\ \vdots \\ r \end{bmatrix} \tag{2.80}$$

$\underline{r}$ is the constant reference vector. In general the length N of the sequence is not known a priori and has to be found iteratively. If a short cycle is required, we solve the optimization

$$\min_{N\in\mathbb{N}}\{\min_{y_f(0),\ldots,y_f(N)} ||\underline{y}_f - \underline{r}||_{\tilde{\rho}} + \gamma(N)\} \tag{2.81}$$

where $\gamma(.) : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is a strictly monotonically increasing, nonnegative function penalizing the use of long cycle lengths N.

Chapter 3

Mixed Logic Dynamical Models of Practical Systems

In this chapter we present the MLD models of two practical systems, a three tank laboratory model and a hydroelectric power plant.

3.1 Introduction

We illustrate the modeling capabilities of MLD models with two practical systems, a three tank laboratory model and a hydroelectric power plant. The control and supervision algorithms that are considered in Chapter 4 have been applied to these systems. In the first case study we show how to model a system for fault detection and isolation purposes.

In general, the following model classes can be written as an MLD system:

- Linear hybrid systems;
- Sequential logical systems (finite state machines, automata);

- Nonlinear dynamic systems, where the nonlinearity can be expressed through combinational logic;
- Systems with discrete inputs and outputs;
- Systems with qualitative inputs and outputs;
- Some classes of discrete event systems;
- Linear time-invariant systems, possibly subject to constraints
- Systems interacting with logic automata

We defer to (Bemporad and Morari, 1999a) for a detailed discussion of these model types.

To model a system in MLD form means to find the numerical expressions for the matrices A, B_1, B_2, B_3, C, D_1, D_2, D_3, E_1, E_2, E_3, E_4, E_5 in (2.8). While some derivations are nowadays automatically performed and supported by the MLD modelling software tool HYSDEL, mentioned in Section 2.7 , several steps shown in the two case studies are typical for the modelling procedure in MLD form and need to be performed in order to obtain a syntax that can be used with HYSDEL.

Several authors have modelled other practical systems in MLD form. In (Bemporad and Morari, 1999a) a gas supply system to an electric power plant is modelled in the MLD framework. Three types of gases are generated as a by-product from a steel-works and have to be distributed to five different boilers in order to generate electric power. Bemporad and Morari (1999b) consider an active car suspension system, controlled by a pneumatic system. In (Borrelli et al., 2001) a traction control scheme for road vehicles is considered. The goal is to improve safety of the car by preventing the wheel from slipping. Bemporad et al. (2001) consider a batch evaporator system, for which safety requirements are verified. Spedicato (2001) and Letizia (2001) derive a model of a combined cycle power plant with one steam turbine and one gas turbine. The goal is a maximization of profit for the

operator of the turbines over a daily, weekly, or longer horizon, taking into account the electric power demand, the startup times, the wear and operating costs.

3.2 Case Study: Three Tank System

3.2.1 The System Description

The three tank system depicted in Figure 3.1 has been adopted recently as a benchmark problem for fault detection algorithms and reconfigurable control (Lunze, 1998; Heiming and Lunze, 1999; Steffen and Lunze, 2001; Lunze et al., 2001). The system has been elaborated within the research program "Control of Complex Systems" (COSY), which is sponsored by the European Science Foundation (COSY),(Åström et al., 2001). In this chapter we consider the modelling in MLD form. This system is used to illustrate the analysis and synthesis techniques in Chapter 4.

The system consists of three liquid tanks that can be filled with two identical, independent pumps acting on the outer tanks 1 and 2. The pumps deliver the liquid flows Q_1 and Q_2 and they can be continuously manipulated from a flow of 0 to a maximum flow $Q_{\max}$. The tanks are interconnected to each other through upper and lower pipes. The flow through these pipes can be interrupted with switching valves V_1, V_2, V_{13}, V_{23} that can assume either the completely open or the completely closed position. The liquid levels h_1, h_2, h_3 in each tank can be measured with continuous valued level sensors[1]. The nominal outflow from the system is located at the middle tank, i.e. V_{L3} is open. The outflows Q_{L1} and Q_{L2} through valves V_{L1} and V_{L2} are zero in nominal behaviour and are used to model failures

[1]Note that this is a slight modification to the original benchmark problem, where an additional qualitative measurements is assumed. These sensors report that the liquid level is within a certain range of values, i.e. "low", "medium", or "high" (Heiming and Lunze, 1999). Medium level is chosen as the range, which fulfills the nominal requirements.

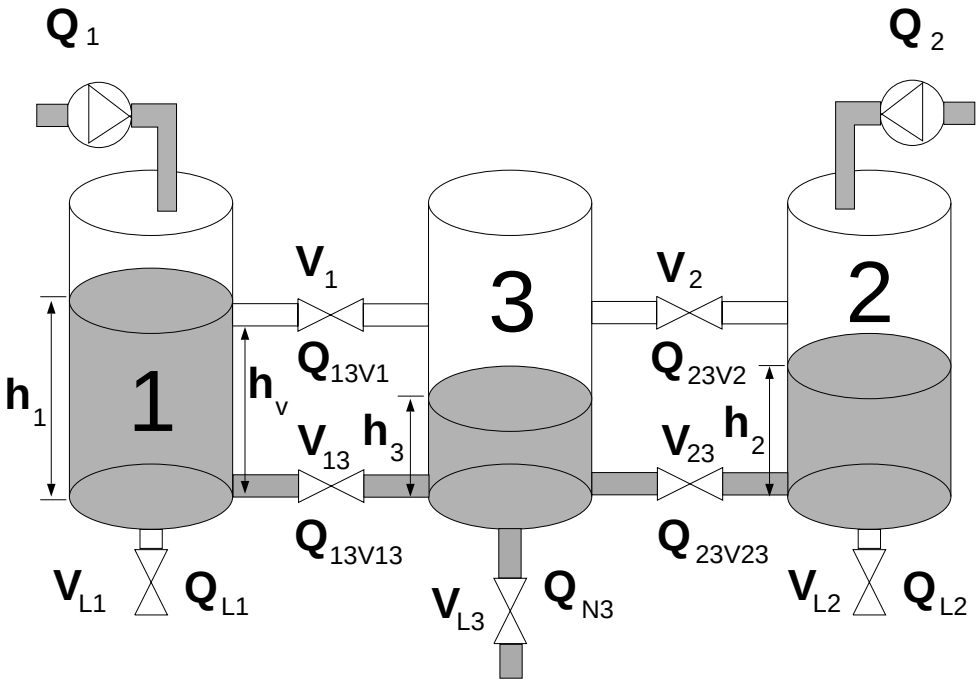


Figure 3.1: COSY three tank benchmark system

of the system. The system represents a processing unit, the goal of which is to provide a constant flow Q_{N3} through the middle tank to a consumer.

3.2.2 System Equations

From the conservation of mass in the tanks we obtain the differential equations

$$\dot{h}_1 = \frac{1}{A}(Q_1 - Q_{13V1} - Q_{13V13} - Q_{L1}) \tag{3.1}$$

$$\dot{h}_2 = \frac{1}{A}(Q_2 - Q_{23V2} - Q_{23V23} - Q_{L2}) \tag{3.2}$$

$$\dot{h}_3 = \frac{1}{A}(Q_{13V1} + Q_{13V13} + Q_{23V2} + Q_{23V23} - Q_{N3}) \tag{3.3}$$

where the Q's denote flows and A is the cross-sectional area of each of the tanks (Lunze, 1998). Details are given in Tables 3.1 and 3.2.

With a slight abuse of notation we will use the term V_h to denote both the valve h and its

Symbol	Meaning
h_i	water level in tank i $(i = 1, 2, 3)$
Q_i	inflow through pump i $(i = 1, 2)$
Q_{ijVh}	flow between tank i and tank j through valve V_h
Q_{Li}	outflow due to leak in tank i $(i = 1, 2)$
Q_{N3}	outflow from tank 3
V_h	status of valve h (0=closed, 1=open)

Table 3.1: Variables of the three tank system

Symbol	Value (MKS)	Meaning
A	.0154	tank section
a_z	1	flow correction term
S_h	$2 \cdot 10^{-5}$	cross-section of valve V_h
g	9.81	gravity constant
h_v	0.3	height of valves V_1, V_2
$h_{\max}$	0.62	maximum water level in each tank
$Q_{i\max}$	10^{-4}	maximum inflow through pump i $(i = 1, 2)$
T_s	5	sampling time

Table 3.2: Model parameters of the three tank system, as defined for the benchmark system

status. In the latter case V_h has to interpreted as a binary signal. Assuming that the flow obeys Torricelli's law, the flow through a lower valve V_{i3} is given by:

$$Q_{i3Vi3} = V_{i3} a_z S_{i3} \operatorname{sign}(h_i - h_3) \sqrt{|2g(h_i - h_3)|} \quad (i = 1, 2) \tag{3.4}$$

The flows through the valves V_{L1}, V_{L2} and V_{N3} are given in a similar way:

$$Q_{L1} = V_{L1} a_z S_{L1} \sqrt{2gh_1} \tag{3.5}$$

$$Q_{L2} = V_{L2} a_z S_{L2} \sqrt{2gh_2} \tag{3.6}$$

$$Q_{N3} = V_{N3} a_z S_{N3} \sqrt{2gh_3} \tag{3.7}$$

The flows through the upper valves V_i $(i = 1, 2)$ are:

$$Q_{i3Vi} = V_i a_z S_i \operatorname{sign}(\max\{h_v, h_i\} - \max\{h_v, h_3\}) \sqrt{|2g(\max\{h_v, h_i\} - \max\{h_v, h_3\})|} \tag{3.8}$$

3.2.3 Derivation of the MLD Model of the Three Tank System

The three tank system can be modelled in MLD form. To this purpose, the following five main steps are required:

1. Linearization of nonlinear relations. For this system, this concerns the flow through the pipes given by Equations (3.4)-(3.8).

2. Introduction of binary level indicator variables, denoting whether the liquid level has reached the height of the upper pipes.

3. Introduction of the states of the valves as binary inputs V_i.

4. Elimination of products of binary and continuous variables using the techniques shown in Chapter 2.

5. Formulation of the system in discrete time.

These steps are now outlined for the three tank system. In order to express the physical model (3.1)–(3.8) in the MLD form (2.8), we approximate the nonlinearity in (3.4) with a straight line, as depicted in Figure 3.2 (right), to obtain for $i = 1, 2$

$$Q_{i3Vi3} \approx k_{i3} V_{i3} (h_i - h_3) \tag{3.9}$$

$$k_{i3} \triangleq a_z S_{i3} \sqrt{\frac{2g}{h_{\max}}} \tag{3.10}$$

Note that more accurate approximations of the square root could be used, choosing piece-

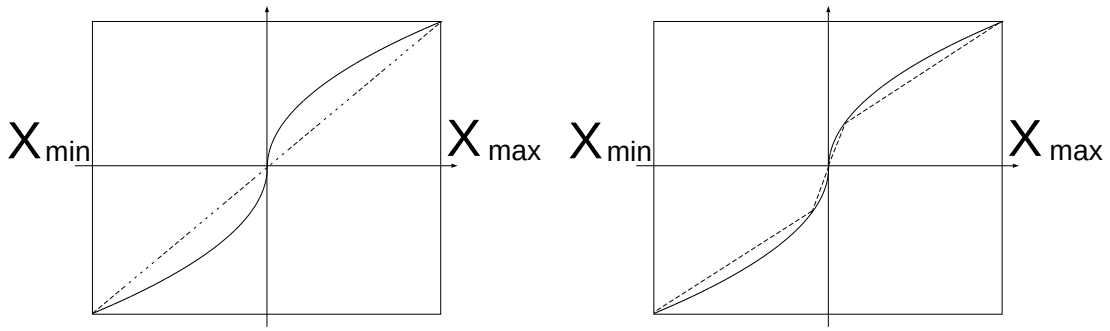


Figure 3.2: Approximation of $\text{sign}(x)\sqrt{|x|}$ with the straight line $k_1 x$, $|x| \leq x_{\max}$ (left) and with piecewise linear functions (right)

wise linear functions, like e.g. in Figure 3.2 (left). The higher accuracy comes however at the expense of a larger number of binary variables. Using the straight line approximation

of Figure 3.2, the other flows are linearized similarly:

$$
\begin{aligned}
Q_{L1} &\approx k_{L1} V_{L1} h_1 && (3.11)\\
k_{L1} &\triangleq a_z S_{L1} \sqrt{\frac{2g}{h_{\max}}} && (3.12)\\
Q_{L2} &\approx k_{L2} V_{L2} h_2 && (3.13)\\
k_{L2} &\triangleq a_z S_{L2} \sqrt{\frac{2g}{h_{\max}}} && (3.14)\\
Q_{N3} &\approx k_{N3} V_{N3} h_3 && (3.15)\\
k_{N3} &\triangleq a_z S_{N3} \sqrt{\frac{2g}{h_{\max}}} && (3.16)\\
Q_{i3Vi} &\approx k_i V_i (\max(h_v, h_i) - \max(h_v, h_3)) \quad (i = 1, 2) && (3.17)\\
k_i &\triangleq a_z S_i \sqrt{\frac{2g}{h_{\max} - h_v}} \quad (i = 1, 2) && (3.18)
\end{aligned}
$$

According to Table 2.2 we have that:

$$
[f(x) \leq 0] \leftrightarrow [\delta = 1] \quad \text{is true iff} \quad \begin{cases} f(x) \leq M(1 - \delta) \\ f(x) \geq \epsilon + (m - \epsilon)\delta \end{cases} \tag{3.19}
$$

where the symbols have the meaning introduced in Chapter 2. Moreover, we use the equivalence:

$$
z = \delta f(x) \quad \text{is true iff} \quad \begin{cases} z \leq M\delta \\ z \geq m\delta \\ z \leq f(x) - m(1 - \delta) \\ z \geq f(x) - M(1 - \delta) \end{cases} \tag{3.20}
$$

(3.19) and (3.20) are used to obtain the MLD form. By introducing the auxiliary variables

$$
z_{i3} \triangleq V_{i3}(h_i - h_3) \quad (i = 1, 2) \tag{3.21}
$$

and using (3.20), Equation (3.9) can be expressed through mixed integer linear inequalities. In order to take into account the flows through the upper valves V_1, V_2, we define for $i = 1, 2, 3$ the auxiliary binary variables

$$
[\delta_{0i}(t) = 1] \;\leftrightarrow\; [h_i(t) \geq h_v] \tag{3.22}
$$

and continuous variables

$$z_{0i} \triangleq \max\{h_v, h_i\} - h_v = \delta_{0i}(h_i - h_v). \tag{3.23}$$

Then, for $i = 1, 2$ we define

$$z_i \triangleq V_i(z_{0i} - z_{03}) \tag{3.24}$$

and we obtain

$$Q_{i3Vi} \approx k_i z_i \tag{3.25}$$

$$k_i \triangleq a_z S_i \sqrt{\frac{2g}{h_{\max} - h_v}} \tag{3.26}$$

Similarly, one has

$$Q_{L1} \approx k_{L1} z_{L1} \tag{3.27}$$

$$Q_{L2} \approx k_{L2} z_{L2} \tag{3.28}$$

$$Q_{N3} \approx k_{N3} z_{N3} \tag{3.29}$$

where k_{L1}, k_{L2}, k_{N3} depend on S_{L1}, S_{L2}, S_{N3} respectively and are defined as in (3.10), and

$$z_{L1} \triangleq V_{L1} h_1 \tag{3.30}$$

$$z_{L2} \triangleq V_{L2} h_2 \tag{3.31}$$

$$z_{N3} \triangleq V_{N3} h_3 \tag{3.32}$$

In addition, h_i, Q_j must fulfill the operating constraints

$$0 \leq h_i \leq h_{\max}, \quad (i = 1, 2, 3) \tag{3.33}$$

$$0 \leq Q_j \leq Q_{\max}, \quad (j = 1, 2) \tag{3.34}$$

Finally, the differential equations (3.1)-(3.3) are discretized by replacing $\dot{h}_i(t)$ with the forward difference $\frac{h_i(t+1) - h_i(t)}{T_s}$, where T_s is the sampling time. Defining

$$x \triangleq [h_1 \; h_2 \; h_3]^T$$

$$u \triangleq [Q_1 \; Q_2 \; V_{13} \; V_{23} \; V_1 \; V_2 \; V_{L1} \; V_{L2} \; V_{N3} \;]^T$$

$$\delta \triangleq [\delta_{01} \; \delta_{02} \; \delta_{03} \;]^T$$

$$z \triangleq [z_{13} \; z_{23} \; z_{01} \; z_{02} \; z_{03} \; z_1 \; z_2 \; z_{L1} \; z_{L2} \; z_{N3}]^T$$

one obtains the form (2.8) as:

$$h_1(k+1) = h_1(k) + \frac{T_s}{A}\Big(Q_1(k) - k_1 z_1(k) - k_{13} z_{13}(k) - k_{L1} z_{L1}(k)\Big) \tag{3.35a}$$

$$h_2(k+1) = h_2(k) + \frac{T_s}{A}\Big(Q_2(k) - k_2 z_2(k) - k_{23} z_{23}(k) - k_{L2} z_{L2}(k)\Big) \tag{3.35b}$$

$$h_3(k+1) = h_3(k) + \frac{T_s}{A}\Big(k_1 z_1(k) + k_2 z_2(k) + k_{13} z_{13}(k) + k_{23} z_{23}(k) - k_{N3} z_{N3}(k)\Big) \tag{3.35c}$$

$$y(k) = x(k) \tag{3.35d}$$

$$E_2\delta(k) + E_3 z(k) \leq E_1 u(k) + E_4 x(k) + E_5 \tag{3.35e}$$

The matrices E_i $(i = 1, \ldots, 5)$ in (3.35e) collect all constraints of the three tank system (3.33), (3.34) and all inequalities stemming from the propositions (3.21), (3.23), (3.24), (3.30), (3.32). Here they are omitted, however the complete set of constraints can be found in (Mignone, 1999). In (Lunze et al., 2001) the three tank system is modelled in a nonlinear multi-model set-up, using a case distinction over $2^3 = 8$ cases to determine the currently valid dynamics[2].

3.2.4 Nominal Operation

The nominal operating condition (Heiming and Lunze, 1999) is described by constant set-point values for levels h_1 and h_3, as

$$\begin{aligned} h_{1ref} &= 0.5 \\ h_{3ref} &= 0.1 \end{aligned}$$

In closed loop operation, level h_1 is controlled by a PI controller manipulating the flow Q_1. To control level h_3, a switching controller manipulating valve V_1 is used, having the description:

$$V_1 = \begin{cases} 1 & \text{if } h_3 < 0.09 \\ 0 & \text{if } h_3 > 0.11 \end{cases} \tag{3.36}$$

[2]The eight regions of the state space are obtained by choosing each level h_i $(i = 1, 2, 3)$ to be higher or lower than the height h_v of the upper valves

The valves $V_{13}, V_2, V_{23}, V_{L1}, V_{L2}$ are closed and the flow Q_2 is zero. Tank 2 is therefore not used in nominal operation. Its purpose is outlined in the next section.

3.2.5 The Problem Definition of the Benchmark System

The three tank benchmark system has been introduced as a test system for fault detection algorithms and reconfigurable control. Three fault scenarios are considered within the benchmark system:

- Tank 1 has a leak
- The actuator V_1 is blocked closed
- The actuator V_1 is blocked open

The first goal is to detect, whether such a failure has occurred by measuring the liquid levels in the tanks, and to determine, which failure actually affects the system. This problem is a *Fault Detection and Isolation* (FDI) task.

The second goal is to propose remedies to keep the outflow from tank 3 as constant as possible. In order to achieve latter goal, it might be required to use the redundant actuators present in the system, i.e. valves V_{13}, V_2, V_{23} and pump Q_2. This procedure will be called *Reconfiguration* in the sequel. Tank 2 is therefore a redundant storage tank, the usage of which is allowed, if failures occur in the system.

3.2.6 Including Faults in the MLD Model

The MLD form can be used to model systems with faults in a concise way. Consider a system, where the occurrence of f faults can be modelled with unmeasured binary distur-

bances. We assume that the dynamics of the system in the presence of each fault is known. Therefore we extend the MLD model (2.8) by including three unmeasured variables:

- Fault variable $\phi(t) \in \{0,1\}^f$
- Input disturbance $\xi(t) \in \mathbb{R}^n$
- Output disturbance $\zeta(t) \in \mathbb{R}^p$

ξ and ζ are introduced, because in Section 4.4.2 we will consider the fault detection problem as an estimation problem. We define the *Mixed Logic Dynamical Faulty* (MLDF) form:

$$x(t+1) = Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) + B_6\phi(t) + \xi(t) \tag{3.37a}$$

$$y(t) = Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) + D_6\phi(t) + \zeta(t) \tag{3.37b}$$

$$E_2\delta(t) + E_3z(t) \leq E_1u(t) + E_4x(t) + E_5 + E_6\phi(t) \tag{3.37c}$$

We introduce the following notation for the faults of the three tank system mentioned in Section 3.2.5:

symbol	type of fault
ϕ_1	Tank 1 has a leak Q_{L1}
ϕ_2	Valve V_1 is stuck closed
ϕ_3	Valve V_1 is stuck open

Table 3.3: Faults of the three tank system

Including Faults in the Three Tank Model

In Section 4.4.2 a model based FDI algorithm is used to perform fault detection and isolation of the three tank system. The assumption is the availability of a model (3.37)

that is capable of describing both the faulty and the faultless system behaviour. Fault ϕ_1 has already been considered in the modelling of Section 3.2.3 as binary input $u_7 = V_{L1}$. To model ϕ_2 and ϕ_3, we can resort to the techniques of Section 2.4.1, since the effect of valve V_1 blocking can be described by logic propositions. The control signal u_5 to valve V_1 is filtered with a processing unit that introduces the potential faults, see Figure 3.3. The

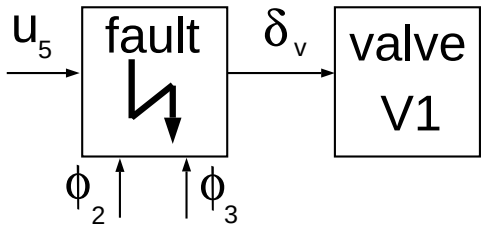


Figure 3.3: The faults ϕ_2 and ϕ_3 can override the binary control signal u_5 to valve V_1

actual actuator signal to valve V_1 is a new auxiliary variable δ_v, defined by:

$$[\delta_v = 1] \quad \leftrightarrow \quad ([\phi_2 = 0] \wedge [u_5 = 1]) \vee [\phi_3 = 1] \tag{3.38}$$

$$[\delta_v = 0] \quad \leftrightarrow \quad ([\phi_3 = 0] \wedge [u_5 = 0]) \vee [\phi_2 = 1] \tag{3.39}$$

$$\overline{(\phi_2 \wedge \phi_3)} \tag{3.40}$$

In case of faultless operation, δ_v is identical to the output u_5 of the switching controller. Equation (3.38) determines when V_1 is open, Equation (3.39) determines, when V_1 is closed, and Equation (3.40) excludes the occurrence of both faults contemporarily. To translate these relations into linear inequalities we can use the three approaches of Section 2.4.1.

Substitution Method Two auxiliary boolean variables δ_x and δ_y are introduced to take account of the inner terms of the logical propositions in the expressions (3.38) and (3.39):

$$[\delta_x = 1] \quad \leftrightarrow \quad [\phi_2 = 0] \wedge [u_5 = 1] \tag{3.41}$$

$$[\delta_y = 1] \quad \leftrightarrow \quad [\phi_3 = 0] \wedge [u_5 = 0] \tag{3.42}$$

Equations (3.38) and (3.39) can then be written as:

$$[\delta_v = 1] \quad \leftrightarrow \quad ([\delta_x = 1] \vee [\phi_3 = 1]) \tag{3.43}$$

$$[\delta_v = 0] \quad \leftrightarrow \quad ([\delta_y = 1] \vee [\phi_2 = 1]) \tag{3.44}$$

Using Table 2.1 we get the following inequalities representing the desired Boolean formula. Inequality (3.45) stems from (3.40), inequalities (3.46)-(3.48) come from (3.41), inequalities (3.49)-(3.51) from (3.42), inequalities (3.52)-(3.54) from Equation (3.43), and inequalities (3.55)-(3.57) from Equation (3.44):

$$\begin{aligned}
\phi_2 + \phi_3 &\leq 1 && (3.45)\\
\delta_x + \phi_2 &\leq 1 && (3.46)\\
\delta_x - u_5 &\leq 0 && (3.47)\\
-\delta_x - \phi_2 + u_5 &\leq 0 && (3.48)\\
\delta_y + \phi_3 &\leq 1 && (3.49)\\
\delta_y + u_5 &\leq 1 && (3.50)\\
-\delta_y - \phi_3 - u_5 &\leq -1 && (3.51)\\
\delta_x - \delta_v &\leq 0 && (3.52)\\
\phi_3 - \delta_v &\leq 0 && (3.53)\\
-\delta_x - \phi_3 + \delta_v &\leq 0 && (3.54)\\
\delta_y + \delta_v &\leq 1 && (3.55)\\
\phi_2 + \delta_v &\leq 1 && (3.56)\\
-\delta_y - \phi_2 - \delta_v &\leq -1 && (3.57)
\end{aligned}$$

Conjunctive Normal Form Equations (3.38) - (3.40) can be transformed into CNF. The steps to do the transformation are to move negations inwards to the literals and to

repeatedly use de Morgan's law (Clocksin and Mellish, 1981). The resulting CNF is:

$$\begin{aligned}(\overline{\phi_2} \vee \overline{\phi_3}) \quad &\wedge \quad (\delta_v \vee \phi_2 \vee \overline{u_5}) \wedge \\ (\overline{\delta_v} \vee \overline{\phi_2}) \quad &\wedge \quad (\delta_v \vee \overline{\phi_3}) \ \wedge \ (\overline{\delta_v} \vee \phi_3 \vee u_5)\end{aligned} \tag{3.58}$$

Using (2.16), we obtain the following set of inequalities:

$$-\phi_2 - \phi_3 \geq -1 \tag{3.59}$$

$$\delta_v + \phi_2 - u_5 \geq 0 \tag{3.60}$$

$$-\delta_v - \phi_2 \geq -1 \tag{3.61}$$

$$\delta_v - \phi_3 \geq 0 \tag{3.62}$$

$$-\delta_v + \phi_3 + u_5 \geq 0 \tag{3.63}$$

Truth Table Method The truth table describing the propositions (3.38)-(3.40) is given in Table 3.4, along with the inequalities defining P_{CH} as presented in Section 2.4.1. All

u_5	ϕ_2	ϕ_3	δ_v
0	0	0	0
0	0	1	1
0	1	0	0
1	0	0	1
1	0	1	1
1	1	0	0

$$\Rightarrow \begin{cases} -u_5 & & -\phi_3 & +\delta_v & \leq 0 \\ u_5 & +\phi_2 & -\phi_3 & +\delta_v & \leq 2 \\ & & \phi_3 & -\delta_v & \leq 0 \\ u_5 & -\phi_2 & & -\delta_v & \leq 0 \\ & \phi_2 & +\phi_3 & & \leq 1 \end{cases}$$

Table 3.4: Truth table for the relations between the control signal to V_1 and the faults and the corresponding linear inequalities

combinations of $[u_5, \phi_2, \phi_3, \delta_v] \in \{0,1\}^4$ not appearing as a row in Table 3.4 are not valid and are excluded by one or more inequalities in Table 3.4. On the other hand, each row of the truth table satisfies each inequality.

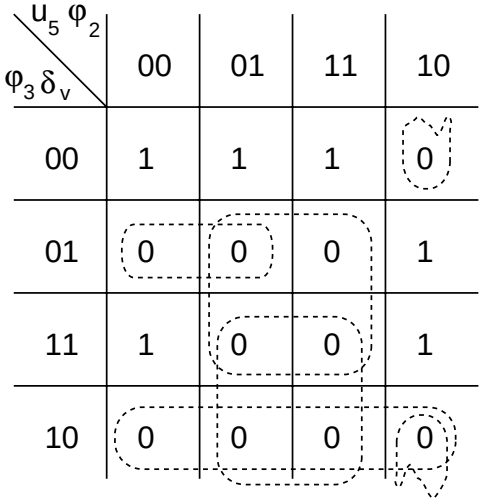


Figure 3.4: Karnaugh map

Karnaugh Maps and Comparison of the Methods To verify the derivation of the CNF, we can use a Karnaugh map (Turner, 1968; Hayes, 1993; Grogg, 1991). To each combination of $[u_5, \phi_2, \phi_3, \delta_v]$ in Table 3.4 we assign the truth value 1, and to each combination not occurring in the table[3] we assign the truth value 0. The Karnaugh map is shown in Figure 3.4. The terms of the CNF in Equation (3.58) correspond exactly to the marked blocks.

Model Size Table 3.5 summarizes the parameters of the MLD model of the three tank system after the inclusion of faults.

3.2.7 System Description in HYSDEL

Modelling the three tank system as in Section 3.2.2 and 3.2.3 enables the user to use HYSDEL (Torrisi et al., 2000) as a tool for automated translation of an MLD model into Matlab syntax. The HYSDEL description of the three tank system is reported in Appendix B.

[3] i.e. to each invalid combination

Number of auxiliary binary variables δ	4
Number of auxiliary continuous variables z	9
Number of states x	3
Number of inputs u	7
Number of constraints	59

Table 3.5: Variables of the three tank system

3.2.8 Implementation and Experiments

The three tank system has been built as a laboratory experiment at the Automatic Control Laboratory of ETH Zürich. A picture of the system can be seen in Figure 3.5. Details about the components, the usage of the system, the software and applications can be found in (Mignone and Monachino, 2001). All experiments of this thesis have been collected from the set-up shown in Figure 3.5. The nominal behaviour of the system is represented in Figure 3.6, where we show a simulation of the MLD system, and in Figure 3.7, where the measured data of an experiment on the laboratory set-up can be seen. The sampling time in all plots is 10 seconds. The differences between simulation and experiment, especially for the time constant of trajectory h_3, are mainly due to the nonideal characteristics of the connecting valves. In (Mignone and Monachino, 2001) we report several measurement series used to identify the valve parameters.

Differences to the Benchmark System Compared to the benchmark system (Lunze, 1998), the model at the Automatic Control Laboratory of ETH Zürich exhibits mainly quantitative differences. The parameters of the experimental set-up differ from the ones listed in Table 3.2:

- The cross-sectional areas of the valves are not all of the same size.
- The connecting valves V_1, V_2, V_{13}, V_{23} have a 4-5 times smaller cross sectional area

Figure 3.5: Three tank system: Laboratory set-up

than the benchmark system. Moreover, the flow through the valves is direction dependent.

- The outflows are located at each tank and can be manually, continuously opened from closed position to the opening position defined in the benchmark system.

- The pumps convey a maximum flow that is about twice as large as the maximum flow of the benchmark system. The pump action is artificially limited by the software.

These differences along with a smaller hysteresis width reflect themselves in smaller oscillations of the levels h_1 and h_3 in Figure 3.6.

In Figure 3.8 we show the manipulated variables during the simulation in Figure 3.6. Note that the setpoint of $[0.5, 0, 0.1]$ generates a switching trajectory for valve V_1. Indeed, this system exhibits a limit cycle as described in Section 2.8. The cycling behaviour is

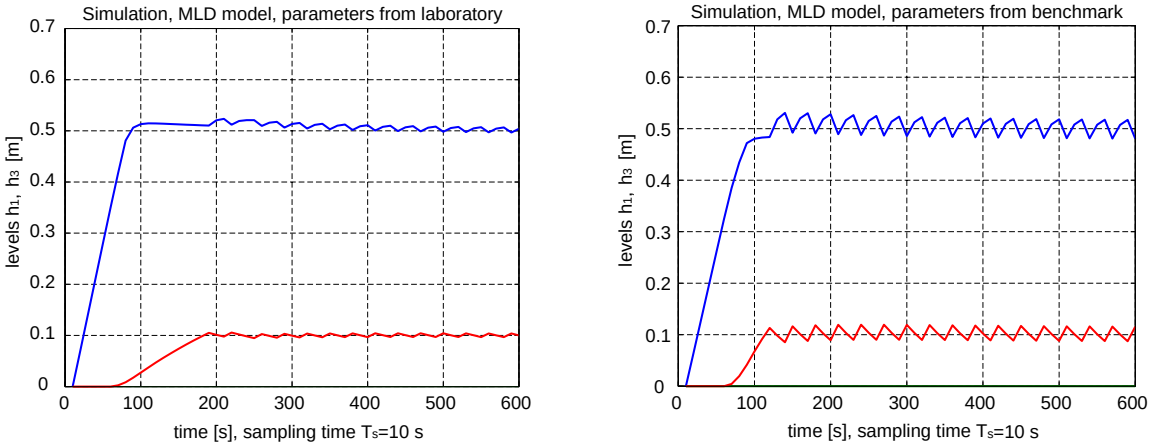


Figure 3.6: Nominal behaviour of the three tank system: Simulation of the MLD model using the estimated parameters of the laboratory set-up (left) and using the parameter defined in the COSY benchmark system (right). The reference at 0.5 m corresponds to h_1.

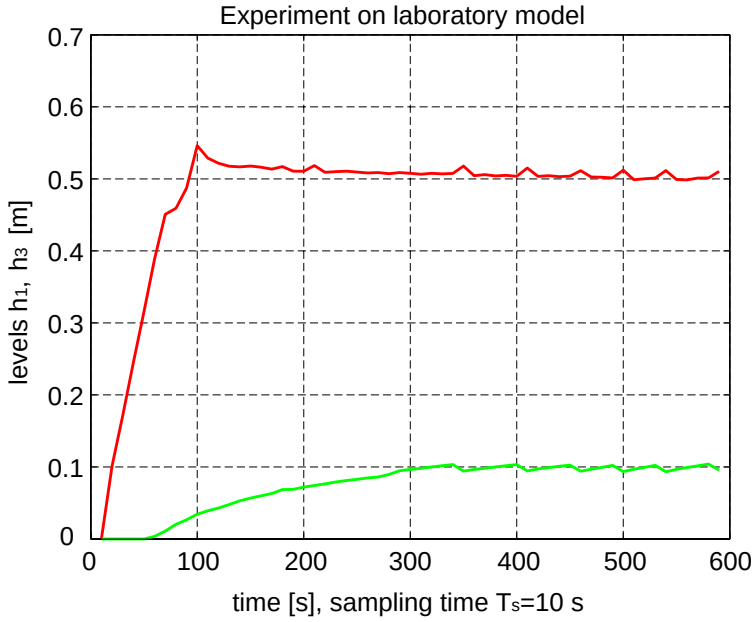


Figure 3.7: Nominal behaviour of the three tank system: Measurement on the laboratory set-up. The reference at 0.5 m corresponds to h_1.

a property observed only for certain reference values, like the one chosen above. Other reference values, like $[0.4, 0, 0.1]$ do not result in any cycles.

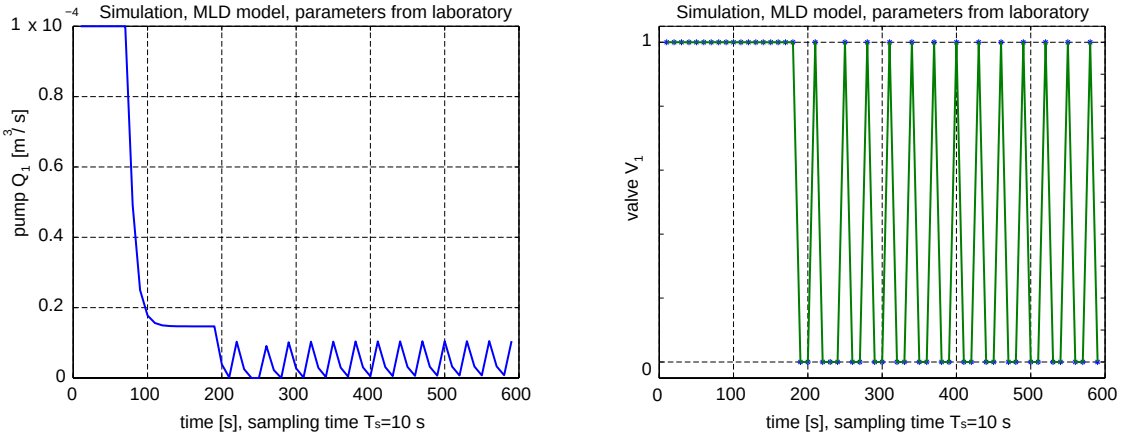


Figure 3.8: Pump action of Q_1 (left) and valve V_1 (right) during the simulation of the MLD model of the three tank system using the estimated parameters of the laboratory set-up

3.3 Case Study: Hydroelectric Power Plant

This section is a modified version of the paper (Ferrari-Trecate, Mignone, Castagnoli and Morari, 2000) and has been written based on the Master's thesis by Castagnoli (2000), from which most of the results have been taken. We present the model of a hydroelectric power plant in the framework of MLD systems. Each outflow unit exhibits a hybrid behaviour since flaps, gates and turbines are controlled with logical inputs and the outflow dynamics depend on the logical state of the unit. We show how to derive detailed models of the units considering not only standard operating conditions, but also emergencies and startup and shut-down procedures.

Our goal is to provide a practical modelling example of a system in MLD form, rather than solve the complex control problem connected to it. To illustrate the controller synthesis, we reduce the model to a simplified form in Section 3.3.7. The simplified model is used in the next chapter for controller synthesis, even though we do not claim it to be representative of the actual plant.

3.3.1 Introduction

The outflow control for hydroelectric power plants is a multiobjective, multivariable, constrained control problem that is difficult to handle with conventional control techniques in its full generality. The final goal of maximal power generation can be achieved by properly distributing the outflow through the available outflow units. In (Chapuis, 1998) the control problem has been solved with a fuzzy controller. Several aspects further complicate the controller synthesis problem, like environmental aspects (protection of river banks, respect for the fauna, possibility of sailing), or operational constraints (contracts with other plants along the river)

In this section we consider the hydroelectric power plant described in (Chapuis, 1998;

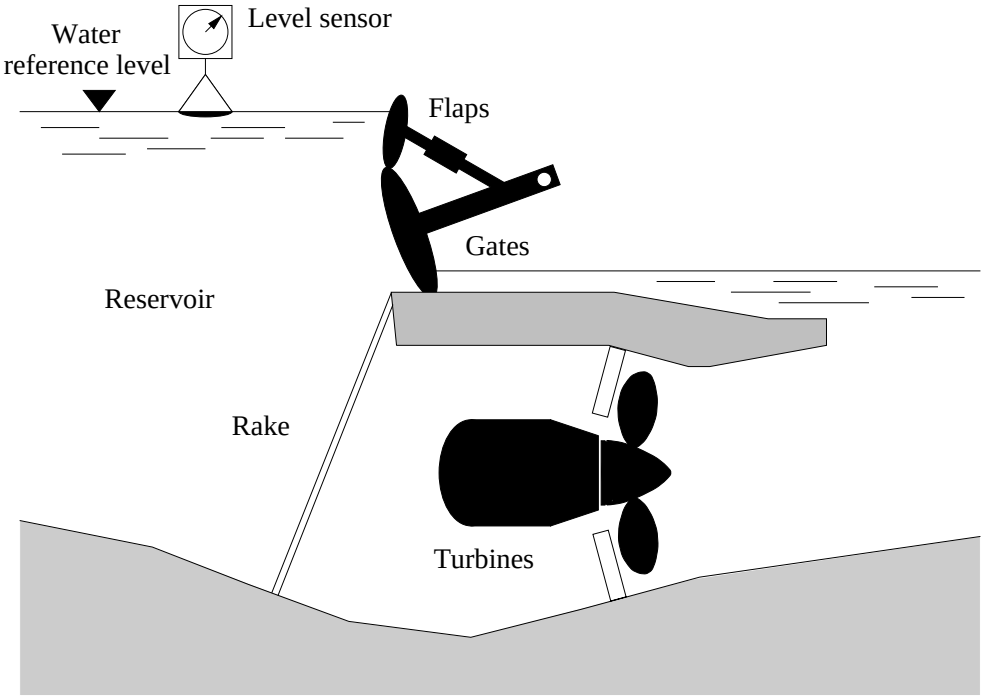


Figure 3.9: Configuration of the river power plant

Chapuis and Kraus, 1999) that includes three types of water outflow units: Four gates, four flaps and two turbines. The weirs (i.e. the gates and the flaps) form a barrage across the river (see Figure 3.9, source: (Chapuis, 1998)) and the total outflow of the dammed river is determined both by the water level in the reservoir and by the opening of the units. The manipulated variables are the openings of each outflow element and the measured variables are the total outflow and the power generated.

There are three main reasons why the outflow units are suitably modelled as hybrid systems:

1. The actuator action is given by a discrete input to the stepper motors of the components, namely the commands to open, close, or leave unchanged the opening of the outflow element.

2. The internal description of every element is given by a finite state machine that

associates different opening dynamics to different logical states.

3. The plant operation is strongly influenced by qualitative decision rules about the use of one type of outflow unit rather than another one.

The preferences mentioned in point 3 above involve for instance rules, like: if the desired outflow increases, the controller should first try to increase the flow through the turbines, rather than the weirs, in order to maximize the produced power. A less trivial constraint is that the procedure of opening and closing an outflow element cannot be arbitrarily short. If an outflow element has started to move, it should be kept on moving for a given minimal time. Moreover, the minimal opening of an outflow unit is given. If the opening falls below this limit, the outflow element must be closed completely. Such a behaviour avoids damages due to obstacles, which might be trapped in the plant.

In this work we model the outflow units as hybrid systems in the MLD form. For MLD systems the control synthesis problem can be formulated and solved in a systematic way using a Model Predictive Control scheme, see Section 4.1. Besides of being capable to handle the hybrid characteristics of the outflow elements, the MLD form allows to prioritize the use of some outflow elements or outflow units in certain operating regimes, as it is outlined in Section 4.6. Other reasons for using MLD models are the possibility to include heuristic rules in the model via propositional logic statements and to include hard and/or soft constraints, the possibility to solve fault detection and state estimation problems within a receding horizon estimation scheme.

3.3.2 The Model of the Flaps

The openings of the turbines and the weirs control the total outflow of the hydroelectric power plant. An outer loop with a simple PI regulator is present, whose output is the setpoint value for the outflow controller (Lahlou, 1994). The level in the water reservoir,

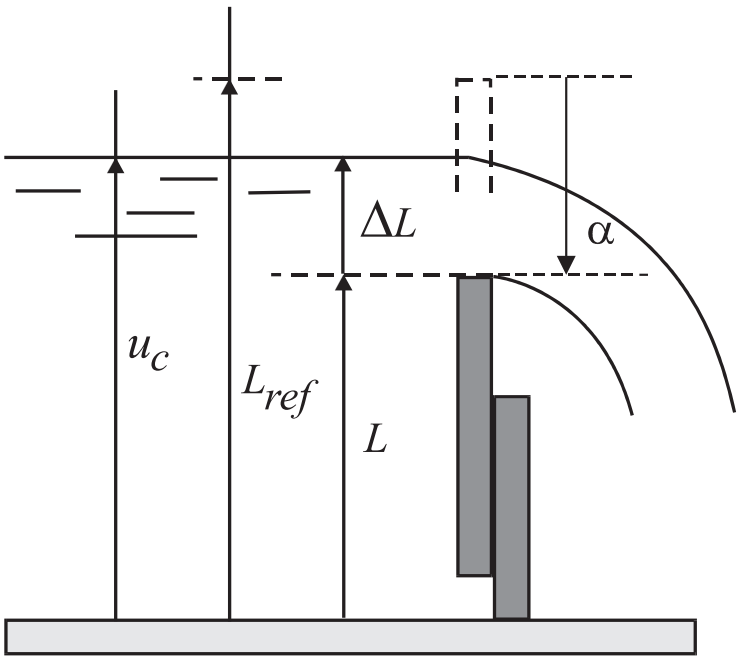


Figure 3.10: Model of the flap: Parameters characterizing the outflow of a flap

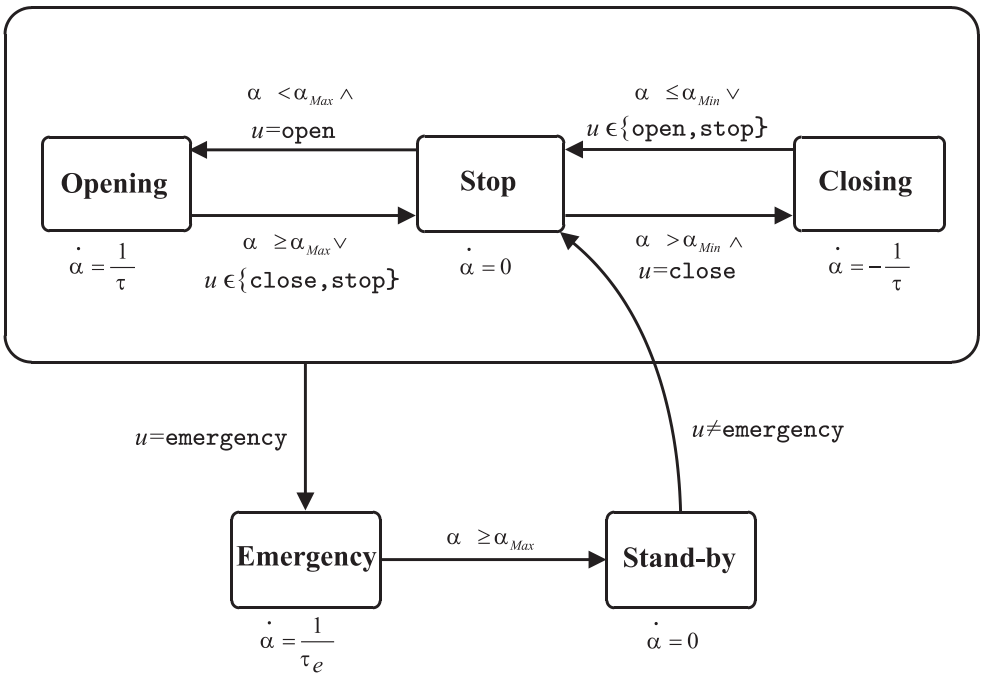


Figure 3.11: Model of the flap: Automaton of the stepper motor

denoted by u_c in Figure 3.10, is therefore a continuous input to the system formed of flaps and gates. The opening α $[m]$ of a single weir is regulated by a stepper motor whose behaviour in normal and emergency conditions is described by the automaton in Figure 3.11. An emergency usually occurs when there are problems due to the connected electric network: In this case the turbines must be closed as fast as possible and the weirs must be opened quickly to maintain the global outflow constant. For sake of simplicity, in Figure 3.11 we omitted the obvious transitions from one state to itself.

The stepper motor is driven by a discrete input $u_\ell(t) \in \{\texttt{open, close, stop, emergency}\}$ that controls the transitions between the five logical states **Opening, Closing, Stop, Emergency** and, **Stand-by**. In order to derive an MLD representation of the stepper motor, we have to code the logical states and the inputs into vectors of binary variables. In principle, it is possible to associate with each different state/input a new binary variable, but this procedure would increase the computational burden for the simulation (and the control) of the model. Therefore, we use the minimal number of logical variables, i.e. two logic input variables $u_{\ell,1}$, $u_{\ell,2}$ and three logic state variables $x_{\ell,1}$, $x_{\ell,2}$, $x_{\ell,3}$ that code the discrete inputs and states of the automaton as in Tables 3.6 and 3.7.

	`stop`	`open`	`close`	`emergency`
$u_{\ell,1}$	0	1	0	1
$u_{\ell,2}$	0	0	1	1

Table 3.6: Coding of the logical inputs of the stepper motor

	Stop	**Opening**	**Closing**	**Emergency**	**Stand-by**
$x_{\ell,1}$	0	1	0	1	1
$x_{\ell,2}$	0	0	1	1	1
$x_{\ell,3}$	$\star$	$\star$	$\star$	0	1

Table 3.7: Coding of the logical states of the stepper motor. The symbol $\star$ denotes indifferently 0 or 1.

In Figure 3.11 we note that the admissible transitions depend on the value of α that is

constrained between the minimum and maximum opening ($\alpha_{min} = 0\,[m]$ and $\alpha_{max} = 2\,[m]$). For instance, the transition from **stop** to **opening** is not allowed if the weir is completely opened (i.e. $\alpha = \alpha_{max}$) even if the command `open` occurs. To take into account these conditions, we introduce two logical variables $\delta_{max}(t)$ and $\delta_{min}(t)$ defined by

$$[\delta_{max}(t) = 1] \quad \leftrightarrow \quad \alpha(t) \geq \alpha_{max} \tag{3.64}$$

$$[\delta_{min}(t) = 1] \quad \leftrightarrow \quad \alpha(t) \leq \alpha_{min} \tag{3.65}$$

To model the evolution of the logical state $x_\ell(t)$, it is convenient to introduce three logical variables $\delta_{\ell,1}(t)$, $\delta_{\ell,2}(t)$, $\delta_{\ell,3}(t)$ defined as

$$\delta_{\ell,i}(t) \triangleq x_{\ell,i}(t+1) \qquad i \in \{1,2,3\} \tag{3.66}$$

and

$$\delta_\ell \triangleq \begin{bmatrix} \delta_{\ell,1} \\ \delta_{\ell,2} \\ \delta_{\ell,3} \end{bmatrix} \tag{3.67}$$

This is similar to the definition of a finite state machine in Section 2.6.3. With the notations introduced so far, every state transition of the automaton depicted in Figure 3.11 can be described as a logical implication. For instance, the transition from the state **Opening** at time t to **Stop** at time $t+1$ is modelled as

$$\left(x_\ell(t) = \begin{bmatrix} 1 \\ 0 \\ \star \end{bmatrix} \wedge \left(u_\ell(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \vee u_\ell(t) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \vee \delta_{max} = 1 \right) \wedge \overline{\left(u_\ell(t) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)} \right) \rightarrow \left(\delta_\ell = \begin{bmatrix} 0 \\ 0 \\ \star \end{bmatrix} \right) \tag{3.68}$$

In order to find the linear inequalities representing (3.68) we can use the methods shown in Section 2.4.1. For this work we have chosen the truth table method.

The dynamics of the opening α depends on the logical state of the stepper motor

$$\dot{\alpha} = \begin{cases} \frac{1}{\tau} & \text{if } x_\ell = \textbf{Opening} \\ -\frac{1}{\tau} & \text{if } x_\ell = \textbf{Closing} \\ 0 & \text{if } x_\ell = \textbf{Stop} \text{ or } \textbf{Stand-by} \\ \frac{1}{\tau_e} & \text{if } x_\ell = \textbf{Emergency}. \end{cases} \tag{3.69}$$

where for a flap $\tau = 120$ $[s]$ and $\tau_e = 60$ $[s]$. Since all the dynamics are simple integrators, we can discretize them with a sampling time $T_C = 10$ $[s]$ and write

$$\alpha(t+1) = \alpha(t) + \delta_o(t)\frac{T_C}{\tau} - \delta_c(t)\frac{T_C}{\tau} + \delta_e(t)\frac{T_C}{\tau_e} \tag{3.70}$$

provided that the new logic variables δ_o, δ_c, δ_e are set equal to one when the values of the logical states, the logical inputs and the saturation indicators δ_{max} and δ_{min} allow opening or closing the weir. For instance, the value of δ_e is assigned by the proposition

$$\left(\left(x_\ell(t) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \vee \; u_\ell(t) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) \wedge \overline{(\delta_{max}(t) = 1)} \right) \rightarrow \delta_e(t) = 1. \tag{3.71}$$

The last task is to compute the outflow that is the output of the system. For this purpose we describe the outflow of a flap. The case of a gate is similar. The outflow law is given by (Krivchenko, 1994):

$$y(t) = \begin{cases} \frac{2}{3} b C_D \sqrt{2g} \Delta L(t)^{\frac{3}{2}} & \Delta L(t) > 0 \\ 0 & \Delta L(t) \leq 0 \end{cases} \tag{3.72}$$

where $b = 14.5$ $[m]$ is the length of the flap, $\Delta L = u_c - L_{ref} + \alpha$ is the difference between the level of the river and the opening of the flap, see Figure 3.10, $L_{ref} = 7.5$ $[m]$ and $C_D = 0.6$ is the discharge coefficient. From (3.72) it is apparent that the outflow is a nonlinear function of ΔL. In order to embed (3.72) in the MLD form (2.8) we have to

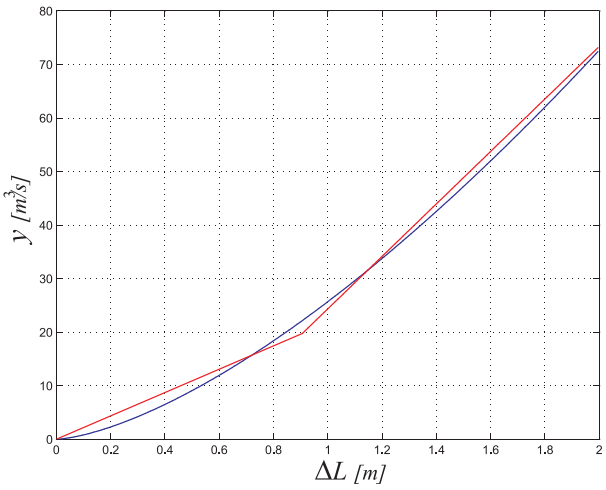


Figure 3.12: Approximation of the outflow profile.

approximate it either in a linear or piecewise affine way. By using a linear function it was impossible to reduce the maximum error below 10.63% in the range of interest for ΔL. To reduce the error, we approximate (3.72) with the piecewise affine function depicted in Figure 3.12, thus obtaining a maximum error of 3.31%. Then, the approximated outflow is described by the equations

$$y(t) = \begin{cases} 0 & \Delta L(t) \leq 0 \\ m_1 \Delta L(t) & 0 \leq \Delta L(t) \leq p \\ m_2 \Delta L(t) + p(m_1 - m_2) & \Delta L(t) \geq p \end{cases} \tag{3.73}$$

where $p = 0.91$, $m_1 = 21.81$, and $m_2 = 48.54$. The piecewise linear function (3.73) can be incorporated into the MLD equations in the following way. First, we introduce the logical variables

$$[\delta_{lin1}(t) = 1] \quad \leftrightarrow \quad [\Delta L(t) \geq 0] \tag{3.74}$$

$$[\delta_{lin2}(t) = 1] \quad \leftrightarrow \quad [\Delta L(t) \leq p] \tag{3.75}$$

$$[\delta_{lin3}(t) = 1] \quad \leftrightarrow \quad [\Delta L(t) \geq p] \tag{3.76}$$

and write the outflow as

$$\begin{aligned} y(t) &= 0 \cdot (1-\delta_{lin1})\ \delta_{lin2}\ (1-\delta_{lin3}) + m_1\Delta L(t) \cdot\ \delta_{lin1}\ \delta_{lin2}\ (1-\delta_{lin3}) \\ &\quad +(m_2\Delta L(t) + p(m_1-m_2)) \cdot \delta_{lin1}\ (1-\delta_{lin2})\ \delta_{lin3}. \end{aligned} \tag{3.77}$$

Then, we introduce the binary variable

$$\delta_{lin}(t) = \delta_{lin1}(t)\delta_{lin2}(t)$$

and the auxiliary real variables

$$\begin{aligned} z_{lin1}(t) &= \delta_{lin}(t)u_c(t) \\ z_{lin2}(t) &= \delta_{lin}(t)\alpha(t) \\ z_{lin3}(t) &= \delta_{lin3}(t)u_c(t) \\ z_{lin4}(t) &= \delta_{lin3}(t)\alpha(t) \end{aligned}$$

Finally, we obtain the expression

$$\begin{aligned} y(t) &= m_1 z_{lin1}(t)\ + m_1 z_{lin2}(t)\ + m_2 z_{lin3}(t)\ + m_2 z_{lin4}(t)\ - L_{ref} m_1 \delta_{lin}(t) \\ &\quad +(p(m_1-m_2)\ -\ m_2 L_{ref})\ \delta_{lin3}(t) \end{aligned} \tag{3.78}$$

where we have made use of the identities

$$\delta_{lin}\delta_{lin3} = 0 \tag{3.79}$$

$$\delta_{lin1}\delta_{lin3} = \delta_{lin3} \tag{3.80}$$

To summarize the results obtained so far, the overall MLD model, for a single flap is described by the following variables

$$\begin{aligned} x &= \begin{bmatrix} \alpha & x_{\ell,1} & x_{\ell,2} & x_{\ell,3} \end{bmatrix}^T \\ u &= \begin{bmatrix} u_c & u_{\ell,1} & u_{\ell,2} \end{bmatrix}^T \\ z &= \begin{bmatrix} z_{lin1} & z_{lin2} & z_{lin3} & z_{lin4} \end{bmatrix}^T \\ \delta &= \begin{bmatrix} \delta_{\ell,1} & \delta_{\ell,2} & \delta_{\ell,3} & \delta_{max} & \delta_{min} & \delta_o & \delta_c & \delta_e & \delta_{lin1} & \delta_{lin2} & \delta_{lin3} & \delta_{lin} \end{bmatrix}^T \end{aligned}$$

input u	u_c	water level in the reservoir $[m]$
	$u_{\ell,1}, u_{\ell,2}$	code of the input signals to the stepper motor
state x	$\alpha(t)$	opening of the flap $[m]$
	$x_{\ell,1}(t), x_{\ell,2}(t), x_{\ell,3}(t)$	code of the stepper motor state
output y	$y(t)$	outflow $[\frac{m^3}{s}]$

Table 3.8: Interpretations of the variables in model of the flaps

The inputs, states and outputs have the interpretation listed in Table 3.8 and the matrices are given in (3.81)-(3.88).

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.81}$$

$$B_1 = O_{4\times 3} \tag{3.82}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{T_C}{\tau} & -\frac{T_C}{\tau} & \frac{T_C}{\tau_e} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.83}$$

$$B_3 = O_{4\times 4} \tag{3.84}$$

$$C = O_{1\times 4} \tag{3.85}$$

$$D_1 = O_{1\times 3} \tag{3.86}$$

$$D_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (p(m_1-m_2)-m_2L_{ref}) & -L_{ref}m_1 \end{bmatrix} \tag{3.87}$$

$$D_3 = \begin{bmatrix} m_1 & m_1 & m_2 & m_2 \end{bmatrix} \tag{3.88}$$

The symbol $O_{a\times b}$ denotes the zero matrix of dimensions $a \times b$. The 125 inequalities stemming from the representation of the δ and z variables are collected in the matrices E_i, $i = 1, \ldots, 5$ of (2.8c) and are not reported here due to the lack of space.

3.3.3 Simulation of the Flap Model

The MLD system derived in the previous section is completely well-posed and can be simulated as outlined in Algorithm 2.1 of Section 2.3. Although the model has a moderately large size, the simulation can be performed very efficiently. For instance, the simulation presented in Figure 3.13 was computed in 11.23s on a Sun Sparc Ultra60 workstation running Matlab 5.3 and using the code in (Bemporad and Mignone, 2000).

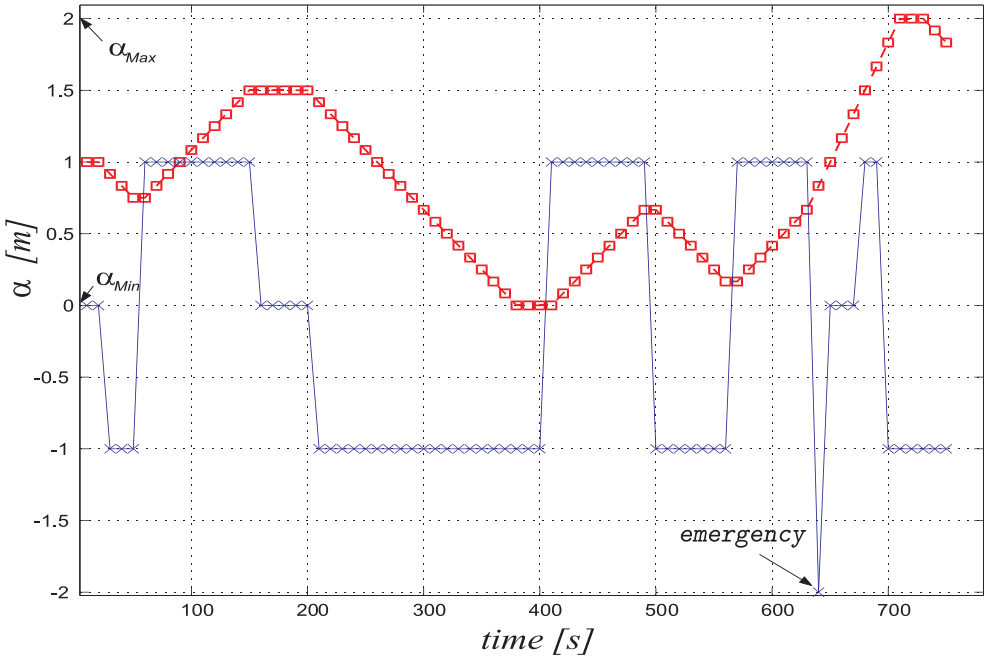


Figure 3.13: Simulation of the flap opening: Squares: opening $\alpha(t+1)$, Crosses: logical input $u_\ell(t)$. We associate the value 1 with `open`, the value 0 with `stop`, the value -1 with `close` and the value -2 with `emergency`

The simulation in Figures 3.13 and 3.14 was obtained by keeping the water level in the reservoir constant ($u_c(t) = 7$ $[m]$), and setting the initial conditions as $\alpha(0) = 1$ $[m]$ and $x_\ell(0) =$ **Stop**. The corresponding outflow profile is depicted in Figure 3.14. We excited the system in such a way to show its behavior in many operating conditions. For instance, when the `emergency` signal occurs , the flap correctly opens up to α_{max} at the fast speed

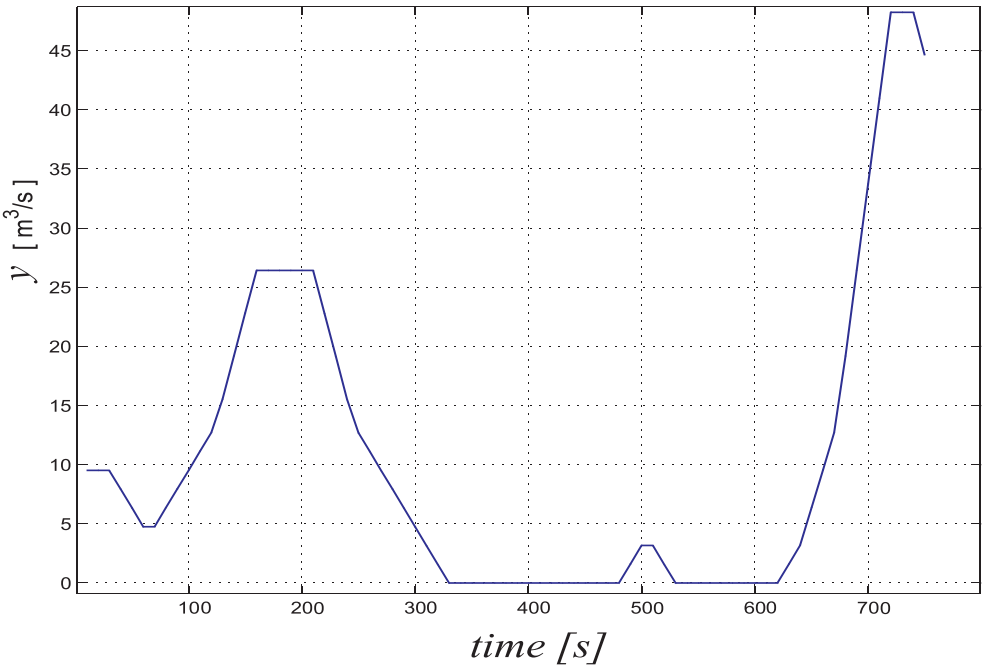


Figure 3.14: Simulation of the flap: Outflow profile.

$\frac{T_c}{\tau_e}$ ignoring the input applied at the next time samples.

3.3.4 Models of Gates and Turbines

By using the methodology outlined in Section 3.3.2, we can derive the MLD description of the gates and the turbines. Details are available in (Castagnoli, 2000). Here we only summarize the main issues in rather broad lines in order to illustrate the modelling capabilities of MLD systems.

The model of the gates differs from the flap model only in the approximation of the outflow law, which is

$$y(t) = \begin{cases} b_G C_G \sqrt{2g}\alpha(t)\sqrt{\frac{2gu_c^2(t)}{u_c(t)+C_G\alpha(t)}} & \alpha(t) > 0 \\ 0 & \alpha(t) \leq 0 \end{cases} \tag{3.89}$$

and depends in a nonlinear way both on $\alpha(t)$ and $u_c(t)$.

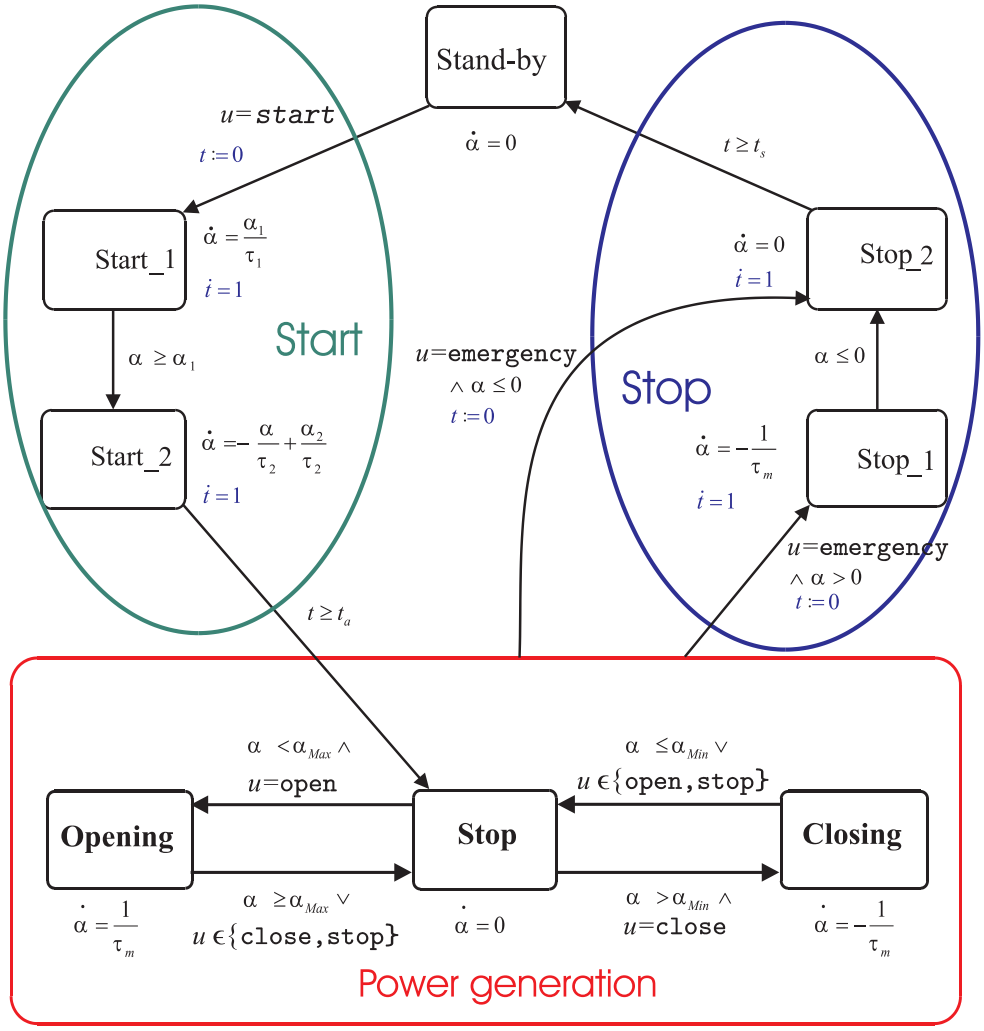


Figure 3.15: Automaton of the turbine.

Concerning the turbines, the automaton of which is depicted in Figure 3.15, one has to take into account the start and stop procedures that involve the presence of timers. In the MLD framework, clocks can be modelled by introducing additional states and they can be initialized by properly modelling the resetting logic (Castagnoli, 2000).

The schematic representation of the turbine is given in Figure 3.16. The inputs are the difference of water levels (the head) ΔY $[m]$ before and after the power plant (see Figure 3.17) and the discrete control input u_ℓ to the stepper motor. The states are the opening

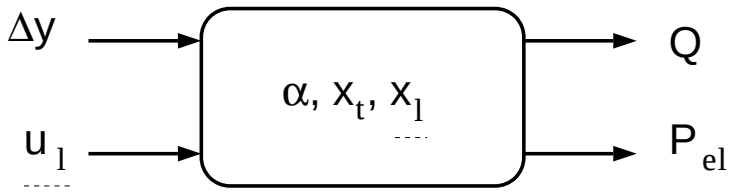


Figure 3.16: Variables of the turbine: binary variables are underlined with a dashed line

α [%] of the water guide vane, the timer x_t $[s]$ and the discrete operating mode x_ℓ. The outputs are the outflow Q $[\frac{m^3}{s}]$ and the electric power produced P_{el} $[kW]$. The dynamics

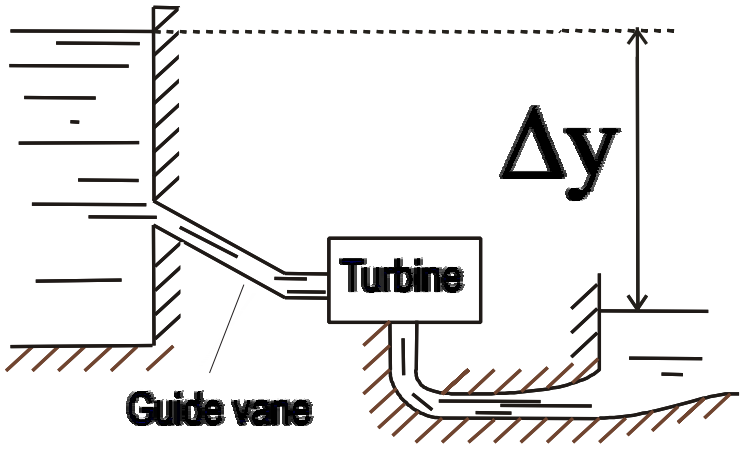


Figure 3.17: Turbine location

of the guide vane opening can be described as in (Castagnoli, 2000):

$$\alpha(t+1) = \alpha\,(t) + \delta_\alpha \frac{T_s}{\tau_m} - \delta_\beta \frac{T_s}{\tau_m} + \delta_\gamma \frac{T_s}{T_{start1}} - \delta_\xi \frac{T_s}{T_{start2}} \tag{3.90}$$

where $\delta_\alpha, \delta_\beta, \delta_\gamma, \delta_\xi$ denote, whether the motor is in the mode opening, closing / stop_1, start_1 or start_2 respectively[4]. If all δ's are zero, the turbine is in stand-by, stop or stop_2 mode. The fact that the turbine is exactly in one mode at each time instant is enforced with the constraint

$$\delta_\alpha + \delta_\beta + \delta_\gamma + \delta_\xi \leq 1 \tag{3.91}$$

The outflow has been determined in (Castagnoli, 2000) by linearizing the nonlinear relation $Q = f(\Delta Y, \alpha)$ over 6 regions in the $\Delta y, \alpha-$plane. The linearized outflow in (3.92)

$$Q\,(\alpha, \Delta Y) = \begin{cases} L_1\alpha + M_1\Delta Y + N_1 & \text{for } \begin{bmatrix} \alpha \\ \Delta Y \end{bmatrix} \in \mathcal{X}_1 \\ \vdots & \\ L_6\alpha + M_6\Delta Y + N_6 & \text{for } \begin{bmatrix} \alpha \\ \Delta Y \end{bmatrix} \in \mathcal{X}_6 \end{cases} \tag{3.92}$$

achieves an average error of the outflow of 1.6% over the feasible range of ΔY and α. The regions $\mathcal{X}_j$ are defined by the partition shown in Figure 3.18 and L_i, M_i and N_i are the coefficients of the linearized outflow.

For a turbine, one must compute the produced power (that depends on the outflow, among other variables) as additional output. Usually, this nonlinear relation between the opening α of the turbine, the head Δy and the outflow Q is provided by the manufacturer of the turbine in the form of a lookup table. Therefore one has to approximate the lookup table with a piecewise linear function keeping in mind that the finer the approximation is chosen, the more complex the overall model becomes in terms of binary variables.

[4]We use the same terminology as in Figure 3.15 for the turbine modes.

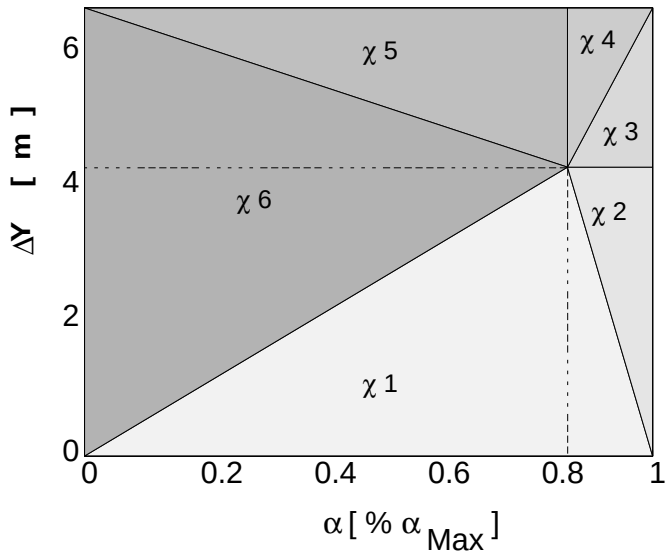


Figure 3.18: Regions $\mathcal{X}_j$ on the $\Delta y, \alpha$-plane, for which the outflow is linearized

3.3.5 Dimensions of the Power Plant Components

In Table 3.9 we report the size of the models of gates, flaps and turbines derived in (Castagnoli, 2000).

3.3.6 Overall Plant

For the successful application of the controller there are a list of constraints and specifications to be taken into account. For instance:

- if the outflow should increase, open turbines rather than flaps
- if the outflow should increase, open flaps rather than gates
- if the outflow should decrease, close flaps rather than turbines

	Turbine	Flap	Gate
continuous states	2	1	1
binary states	3	3	3
continuous inputs	1	1	1
binary inputs	3	2	2
auxiliary binary variables	19	12	9
auxiliary continuous variables	7	4	2
number of constraints	287	125	106

Table 3.9: Dimension of MLD models of the outflow elements

For a complete description of such specifications we defer to (Chapuis, 1998). These specifications can be added as soft or as hard constraints in the model of the power plant and they can be systematically considered in the choice of the control moves, as it is shown in Section 4.6.

The control algorithm is applied to the overall plant, which is obtained by aggregating the individual outflow units. For nominal operation however, the complexity of the components modelled in (Castagnoli, 2000) is unnecessarily high, see Table 3.9. For instance, there is no need to model the emergency signals, since emergencies are very unlikely to be handled by the MPC (see Section 4.1.2) scheme in an automated way. The use of simplified models allows to improve the speed of the numerical computations involved for the controller synthesis. The complexity of such problems depends mainly on the number of binary variables in the model. To improve the efficiency of the control algorithm, we have built a simplified model of lower complexity in terms of the total number of binary variables, reducing the description of the automata governing the outflow components.

3.3.7 Simplified Power Plant

To derive a simplified model of the power plant, we model each of the three components (turbines, gates and flaps) as a "universal" outflow unit. No qualitative distinction between the three types is made. A simplified, linear, discrete-time model of an outflow unit is given by:

$$\alpha_{o\ell}(t+1) = \alpha_{o\ell}(t) + \mu_o \Delta Q^+_{o\ell}(t) - \mu_o \Delta Q^-_{o\ell}(t) \tag{3.93}$$

subject to saturations on $\alpha_{o\ell}$

$$\alpha_{o\,\min} \leq \alpha_{o\ell}(t) \leq \alpha_{o\,\max} \tag{3.94}$$

The variables are explained in Table 3.10.

o	index denoting one of the outflow units, i.e. $o \in \{g, f, t\}$ where: g: gates; f: flaps; t: turbines
$\alpha_{o\ell}$	opening of outflow element ℓ
μ_o	coefficient relating the opening of the outflow element to the control signal of the stepper motor
$\Delta Q^+_{o\ell}$	signal to stepper motor of element ℓ (opening) $\Delta Q^+_{o\ell} \in \{0, 1\}$
$\Delta Q^-_{o\ell}$	signal to stepper motor of element ℓ (closing) $\Delta Q^-_{o\ell} \in \{0, 1\}$

Table 3.10: Variables in the model of a simplified outflow unit

The modelling of saturated variables can be done in several ways for MLD systems. The simplest way is to extend the linear model in Equation (3.93) with constraints on the state, as in Equation (3.94). A more elaborated way is the introduction of 2 binary indicator variables that allow to identify the 3 regions where the outflow element can be: lower saturation, linear operation and upper saturation. The advantage of introducing these binary variables are:

- The additional binary variables allow to impose easily constraints on the use of outflow elements and units. The availability of an outflow unit to perform a closing or an opening operation is given in terms of these variables
- The prioritizations of one type of unit rather than another one can be expressed in terms of these variables.
- The simulation of the system is always feasible, also for input sequences that drive the system into saturations. For MLD systems the simulation is performed solving a mixed integer continuous feasibility test at each time step, as outlined in Algorithm 2.1.

The additional binary variables have been introduced in the following way:

$$\delta_1 = 1 \quad \Leftrightarrow \quad \text{outflow element is in upper saturation} \tag{3.95}$$

$$\delta_2 = 1 \quad \Leftrightarrow \quad \text{outflow element is in linear operation} \tag{3.96}$$

Assuming $\alpha_{o\,\min} = 0$, the MLD model of an outflow element is given by

$$\alpha_{o\ell}(t+1) = z(t) \tag{3.97}$$

$$Q(t) = \nu_o x(t) \tag{3.98}$$

subject to

$$\begin{bmatrix} \alpha_{o\,\max} & \epsilon_o \\ -\mu_o & 0 \\ \mu_o & \mu_o \\ 1 & 1 \\ -\alpha_{o\,\max} & -\alpha_{o\,\max} + \epsilon_o \end{bmatrix} \underline{\delta}(t) + \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} z(t) \leq \begin{bmatrix} 0 & 0 \\ -\mu_o & \mu_o \\ \mu_o & -\mu_o \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \underline{u}(t) + \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ \mu_o \\ 1 \\ 0 \end{bmatrix} \tag{3.99}$$

The output Q is the outflow through the element, ϵ_o is a small tolerance. In Equation (3.98) we assume a linear relation between the outflow and the opening of the element, with a proportionality coefficient ν_o $[\frac{m^3}{s}]$. The dimensions of the MLD model of one single outflow element are summarized in Table 3.11. To obtain the size of an MLD model for

variable type	symbol	number of variables
binary inputs u	$[\Delta Q^+_{o\ell}, \Delta Q^-_{o\ell}]$	$n_u = 2$
binary auxiliary variables δ	$[\delta_1, \delta_2]$	$n_\delta = 2$
real auxiliary variables z	z	$n_z = 1$
states x	$\alpha_{o\ell}$	$n_x = 1$
inequalities		$n_e = 5$

Table 3.11: Dimensions of the MLD model for one single outflow element

the overall plant, the numbers n_u, n_δ, n_z, n_x in Table 3.11 scale linearly with the number of elements required in the plant. The coupling between the outflow elements is specified with additional constraint. For instance, we have to impose that at most one outflow element is moved at each time step. Moreover, if we take into account the priorities on the operation of opening and closing the outflow elements, summarized in Table 3.12, the total number of inequalities would become 177.

	turbines	flaps	gates
$\Delta Q \geq 0$	high	medium	low
$\Delta Q < 0$	low	medium	high

Table 3.12: Priorities for the use of outflow units

Comparison with the Detailed Models of the Outflow Units

Compared to the detailed model of the outflow unit, the simplified model does not take into account several features:

- The emergency mode of each outflow unit is neglected.
- The procedure of opening and closing an outflow element cannot be arbitrarily short.

If an outflow element has been started moving, it should be kept on moving for a given minimal time.

- The minimal opening of an outflow unit is given. If the opening falls below this limit, the outflow element must be closed completely. Such a behaviour avoids damages due to obstacles, which might be trapped in the plant.

Simulation of the Power Plant

In Figure 3.19 we simulated the outflow units of a power plant according to the model described in this section. We have chosen a system with 2 gates, 2 flaps, and 1 turbine. The plot shows the opening of the outflow elements that are driven by an arbitrary control sequence. The only additional constraint in this simulation is the requirement that at most one element can be moved at each sampling time. For the trajectories of the openings in Figure 3.19, we obtain a total outflow shown in Figure 3.20.

Steady States

When setting up the model predictive control scheme, the required reference values in Equation (4.1) are obtained by solving (2.75) - (2.78) for x_f, u_f, δ_f, z_f. The latter optimization problem can lead to an unreachable steady state, see Section 2.8. The resulting state x_f is a valid state for the MLD system, from which a system trajectory can be simulated. However, for any initial state, there exists no input sequence leading to the steady state.

This situation occurs for the system under consideration. The result of the optimization (2.75) - (2.78) leads to a steady state where one gate is open, while the turbines and flaps are closed. Indeed, the model contains explicit constraints about opening turbines, flaps and gates in this order, and closing them in reverse order, see Table 3.12. Note that the

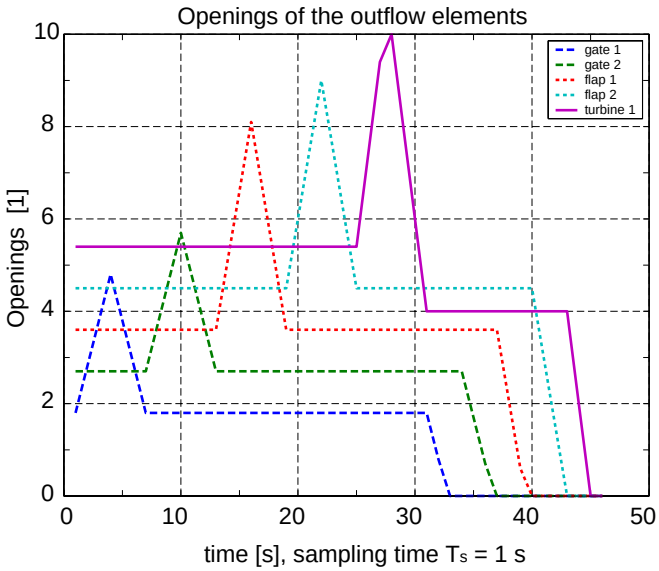


Figure 3.19: Opening of the outflow elements on the simplified power plant excited by some arbitrarily chosen input sequence; the parameters are $\alpha_{gmax} = 20$, $\alpha_{fmax} = 15$, $\alpha_{tmax} = 10$, $\mu_g = 1$, $\mu_f = 1.5$, $\mu_t = 2$, $\nu_g = 5$, $\nu_f = 2$, $\nu_t = 1$ (normalized per unit representation)

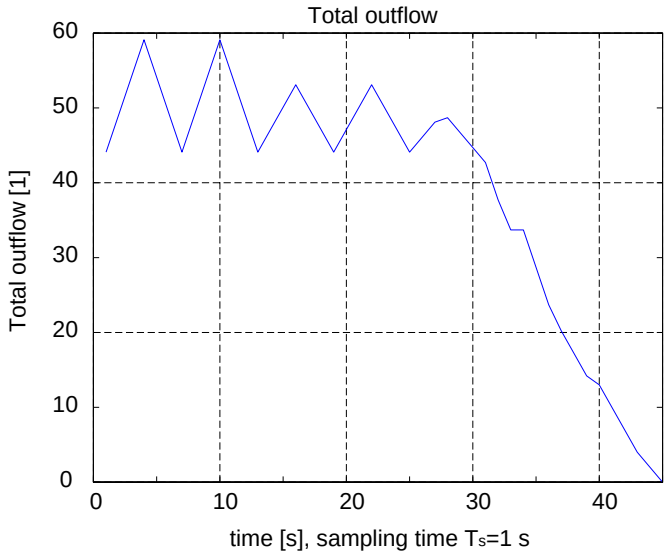


Figure 3.20: Total outflow with the openings shown in Figure 3.19 (normalized per unit representation)

system inputs are binary and modelled as Δu. They denote a step up or down in opening or closing the outflow elements. This guarantees that starting the model from a sensible initial condition will keep the system on a trajectory of valid states.

The components which are responsible for the unreachable states in the simplified power plant are isolated in the following MLD model. Here we show, how this problem can arise. Consider the system:

$$x_1(k+1) = x_1(k) + \mu_1 u_1^+(k) - \mu_1 u_1^-(k) \tag{3.100}$$

$$x_2(k+1) = x_2(k) + \mu_2 u_2^+(k) - \mu_2 u_2^-(k) \tag{3.101}$$

$$y(k) = c_1 x_1(k) + c_2 x_2(k) \tag{3.102}$$

where c_1, c_2, μ_1, μ_2 are positive coefficients; $x_1 \geq 0$, $x_2 \geq 0$ are continuous states, u_1^+, u_1^-, u_2^+, u_2^- are binary inputs. Let the system be subject to the following constraints:

$$[\delta_1 = 0] \Leftrightarrow [x_1 \geq x_{max}] \tag{3.103}$$

$$[\delta_1 = 1] \Rightarrow [u_2^+ = 0] \tag{3.104}$$

$$\overline{u_1^+(k) \wedge u_1^-(k)} \tag{3.105}$$

$$\overline{u_2^+(k) \wedge u_2^-(k)} \tag{3.106}$$

where $c_1 > 0$; $c_2 > 0$; $\mu_1 > 0$; $\mu_2 > 0$. We assume the initial state $x(k=0) = 0$. The inputs to this system are binary and they can drive the system in a quantized way, either a step up or down or they can leave the state unchanged. The logical proposition (3.104) expresses the fact that state x_2 can be manipulated in positive direction, only if the state x_1 has reached a given threshold. This is a form of hard prioritization of state x_1 with respect to state x_2. Expressions (3.105) and (3.106) exclude that the states can be increased and decreased at the same time. Defining $u = [u_1^+,\ u_1^-,\ u_2^+,\ u_2^-]$ and $x = [x_1,\ x_2]$, the MLD form of this system is:

$$x(k+1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} \mu_1 & -\mu_1 & 0 & 0 \\ 0 & 0 & \mu_2 & -\mu_2 \end{bmatrix} u(k) \tag{3.107}$$

$$y(k) = \begin{bmatrix} c_1 & c_2 \end{bmatrix} x(k) \tag{3.108}$$

$$\begin{bmatrix} -m_1 \\ -M_1-\epsilon \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta(k) \leq \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} u(k) + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} -x_{max}-m_1 \\ x_{max}-\epsilon \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad (3.109)$$

where

$$M_1 = \max(x_1 - x_{max}) \quad (3.110)$$

$$m_1 = \min(x_1 - x_{max}) \quad (3.111)$$

For this system we compute a steady state, assuming that the reference values for y lies in the range

$$r \in [0, c_1 x_{max} - \epsilon]$$

Indeed, the maximal output for which $\delta_1 = 0$ is $c_1 x_{max} - \epsilon$. This means that all reference values up to $c_1 x_{max} - \epsilon$ can be reached having $x_{2f} = 0$. Because of (3.105) and (3.106), in steady state the inputs to the system must all be zero. The steady state for δ is zero, since we assume that $x_{1e} < x_{max}$. Therefore, with the choice of

$$x_e = \begin{bmatrix} \frac{r}{c_1} \\ 0 \end{bmatrix} \quad (3.112)$$

we can reach the reference value exactly. Note that (3.112) is the unique reachable steady state for the system under the assumption of $r < c_1 x_{max}$.

The solution of (2.75) - (2.78), however, can lead to an unreachable state. With the considerations above, we find that the cost function (2.75) of the optimization problem reduces to the minimization

$$\min_x \rho\left(||x||\right) \quad (3.113)$$

This is due to the facts that the reference value can be reached exactly, the system does not have any auxiliary real variable z and the steady state values for u and δ are zero. Therefore all terms in (2.75) are zero, except for $||x||$. Without taking into account the reachability properties, the optimization (3.113) chooses the state with the smallest norm. This results in the steady state

$$\tilde{x}_f = \begin{bmatrix} \frac{c_1 r}{c_1^2+c_2^2} \\ \frac{c_2 r}{c_1^2+c_2^2} \end{bmatrix} \tag{3.114}$$

The condition $r < c_1 x_{max}$ implies that $\frac{c_1 r}{c_1^2+c_2^2} < x_{max}$. In the resulting steady state both state components are nonzero, and $\tilde{x}_{1f} < x_{max}$. Therefore $\tilde{x}_f$ cannot be reached from the initial state $x(0) = 0$.

Chapter 4

Control and Supervision of Mixed Logic Dynamical Systems

In this chapter we consider some analysis and synthesis aspects of MLD models by presenting strategies and algorithms, on how to tackle control and supervision problems. The effectiveness of the methods is demonstrated with simulations and practical experiments on a laboratory model.

4.1 Model Predictive Control of Mixed Logic Dynamical Systems

4.1.1 Introduction

The traditional approach to design embedded controllers is based on the development of rules that rely on heuristics or on the plant knowledge by the operators. Such rules are often based on "common sense" and take into account specific plant scenarios. However

the analysis of the overall controlled plant for all the possible scenarios is usually a hard task, which is typically solved in industry by performing a large number of simulations or runs of the controller.

Formal verification certifies that a hybrid system satisfies certain requirements. This amounts to solve the following reachability problem: For a given set of initial conditions and disturbances, guarantee that all possible trajectories never enter a set of unsafe states, or possibly provide a counterexample. Clearly, verification only assesses if an existing controller performs in a safe way or not, but does not provide any direct hint on how the controller should be modified. Moreover, it is well known that formal verification of hybrid systems is an undecidable problem (Alur et al., 1993). In spite of this complexity, several tools for formal verification of hybrid systems have been proposed in the literature, see (Bemporad, Torrisi and Morari, 2000; Torrisi, 2002) and references therein.

Recent research efforts produced techniques that allow a direct synthesis of controllers that are in some sense optimal and take into account physical and operational constraints. For MLD systems the main methods available are:

- Model predictive control of MLD systems using online mixed integer continuous optimizations (Bemporad and Morari, 1999a). We review this method in Section 4.1.2.
- Piecewise linear state feedback for the equivalent PWA systems obtained using linear matrix inequality optimizations (Mignone et al., 2000a). This technique is presented in Chapter 6.
- Explicit form of Model Predictive Controllers obtained by multiparametric optimization techniques (Bemporad, Borrelli and Morari, 2000a). The main idea of this approach is to move the computations offline that are required to synthesize the optimization based controller. It can be shown that the control law obtained with an online model predictive control scheme can be reproduced with a piecewise linear state feedback. We defer to (Borrelli, 2002) for details.

A comparison of these methods is beyond the scope of this work. We will focus on the first two approaches.

4.1.2 Description of the Method

Model predictive control (MPC) is a form of feedback control, where the current value of the manipulated variables is determined online as the solution of an optimal control problem over a horizon of given length. The behaviour of the system over the horizon is predicted with a model and the current state of the plant is the initial state for this prediction. While a possibly large set of control moves is computed, only the first one is implemented on the plant. When an updated information about the plant state is available at the next sampling instant, the optimization is repeated over a shifted horizon. The ability to systematically include constraints and the possibility to handle plants with multiple inputs and outputs have made MPC an attractive technique in process industry (Lee and Cooley, 1997; Mayne, 1997; Qin and Badgewell, 1997; Mayne et al., 2000; Garcia et al., 1989). We defer to the recent monographs (Camacho and Bordons, 1999; Maciejowski, 2002) for an introduction to classical model predictive control.

In the present context, due to the presence of integer variables, the optimization procedure is a *Mixed Integer Program*. We consider this type of mathematical optimization problem in Chapter 5. A first attempt to use on-line mixed integer programming to control dynamic systems subject to logical conditions has appeared in (Tyler and Morari, 1995). Other works aiming at combining MPC to hybrid control have appeared in (Slupphaug and Foss, 1997; Slupphaug et al., 1997).

An application of the concepts of predictive control to MLD systems has been proposed in (Bemporad and Morari, 1999a). Assuming a quadratic cost function and given x_0, for an

MLD system the optimization at time $t = 0$ has the following form:

$$\min_{u_0^{T-1},\delta_0^{T-1},z_0^{T-1}} J(u_0^{T-1}, x_0) = \sum_{t=0}^{T-1} \|u(t) - u_f\|_{Q_1}^2 + \|\delta(t, x_0, u_0^t) - \delta_f\|_{Q_2}^2 + \|z(t, x_0, u_0^t) - z_f\|_{Q_3}^2 + \|x(t, x_0, u_0^{t-1}) - x_f\|_{Q_4}^2 + \|y(t, x_0, u_0^{t-1}) - y_f\|_{Q_5}^2 \quad (4.1)$$

subject to

$$x(t+1, x_0, u_0^t) = Ax(t, x_0, u_0^t) + B_1 u(t) + B_2 \delta(t, x_0, u_0^t) + B_3 z(t, x_0, u_0^t) \ (t{=}0 \ldots T{-}1) \quad (4.2)$$

$$x(T, x_0, u_0^{T-1}) = x_f \quad (4.3)$$

$$E_2 \delta(t, x_0, u_0^t) + E_3 z(t, x_0, u_0^t) \leq E_1 u(t) + E_4 x(t, x_0, u_0^t) + E_5 \quad (t = 0 \ldots T) \quad (4.4)$$

where $\|x\|_Q^2 = x^T Q x$, $Q_i = Q_i^T \geq 0$, $i = 1, \ldots, 5$, are given weight matrices, x_f, u_f, δ_f, z_f, y_f are given offset vectors satisfying (2.8b)–(2.8c), for which we recall the possible difficulties described in Section 2.8. $u_0^{T-1} = [u(0|0), u(1|0), \ldots u(T-1|0)] \in \mathbb{R}^{n_u \times T}$ is the sequence of manipulated variables over a horizon of T steps computed at time $t = 0$, u_0^{k-1} is the sequence u_0^{T-1} truncated to the first k values. $x(t+1, x_0, u_0^t)$ is the state at time $t+1$ computed from the initial state x_0 with the input sequence u_0^t. The other quantities are defined accordingly. The weighting matrices on the auxiliary variables δ and z allow to formulate the optimization problem with a positive definite cost function. For a practical application however, the weights Q_1, Q_4, Q_5 are more important, since they determine the trade-off between the control action effort and the accuracy of the reference tracking.

The resulting mathematical optimization problem is a mixed integer quadratic program (MIQP), see Chapter 5. In (4.1) a quadratic cost function is used to penalize the deviations between the references and the model predictions. In (Bemporad, Borrelli and Morari, 2000a) the $1/\infty$-norm is chosen, where the ∞-norm is taken over the vector components and the 1-norm over the horizon. The resulting mathematical problem is a Mixed Integer Linear Program (MILP), see Chapter 5.

4.1.3 Application of Model Predictive Control to the Three Tank System: Experiment

We have applied the control scheme (4.1)-(4.4) to the three tank system described in Section 3.2. In this section we report an application to the real experimental set-up of Figure 3.5. In the experiment shown in Figure 4.1 we assume a time varying trajectory for the levels h_1, h_2 and a constant reference for h_3 in the middle tank. This scenario simulates, e.g., a varying supply to the outer tanks, despite of which the outflow from the middle tank is required to be constant. This experiment has been performed with the solution of MIQPs online, using the solver Xpress-MP (Dash Associates, 1999). The main parameters are listed in Table 4.1. In order to show the capability of handling a multivariable hybrid control problem, we have used both pumps and all four switching valves as actuators. Figure 4.2 shows the pump action during the experiment. The maximum pump action at

control horizon	3 steps
prediction horizon	3 steps
sampling time	10 sec
Q_4	diag(1,1,1)
Q_5	diag(1000, 3000, 10000)
Q_1, Q_2, Q_3	diag(0.01, 0.01, 0.01, ...)

Table 4.1: Parameters of the control experiment in Figure 4.1; diag(x,y,...) denotes a diagonal matrix with the entries x, y, ... in this order.

the beginning of the experiment is due to the fact that the initial states all lie below their reference. Note that the experiment involves a significant switching of the binary inputs and the level indicator variables δ. The inputs to the switching valves required to keep track of the references is shown in Figure 4.3.

In Figure 4.1 the state trajectories show a systematic delay compared to their reference.

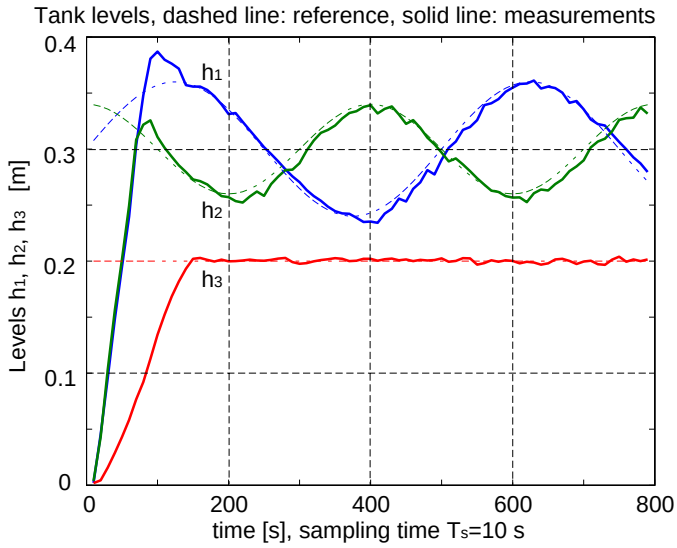


Figure 4.1: Control experiment on the three tank system: The dashed lines represent the reference trajectory for the three levels, the solid lines are the measured levels

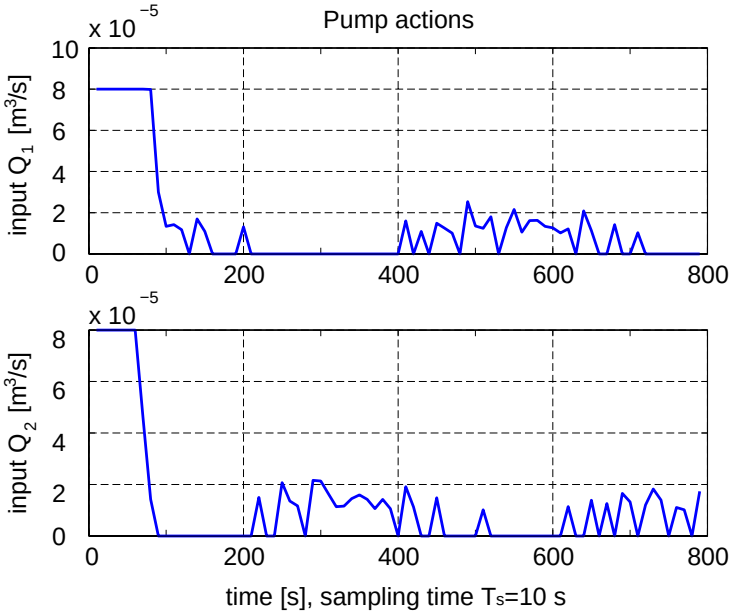


Figure 4.2: Control experiment on the three tank system: Pump action during the experiment. The maximum flow is limited to $0.8 \cdot 10^{-4} \frac{m^3}{s}$

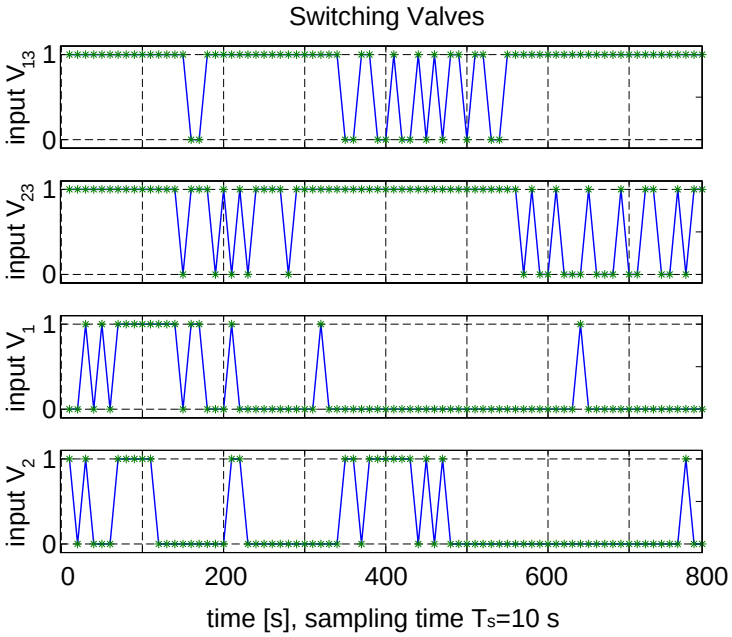


Figure 4.3: Control experiment on the three tank system: Discrete control action of the switching valves

Indeed, this experiment uses the current reference value $y_f(t)$ as a constant reference over the whole horizon in (4.1). The delays are significantly reduced, if we use the complete time-varying reference $[y_f(t), y_f(t+1), y_f(t+T-1)]$ in (4.1). We defer to (Mignone and Monachino, 2001) for a comparison.

4.1.4 Application of Model Predictive Control to the Simplified Power Plant: Simulation

The simplified model of the outflow units of a hydroelectric power plant in Section 3.3.7 has been used to perform a reference tracking control problem. We have assumed a power plant with one outflow element for each of the three outflow unit types. The simulated outflow and its reference are given in Figure 4.4. For this simulation a horizon of three steps has been used in the model predictive controller.

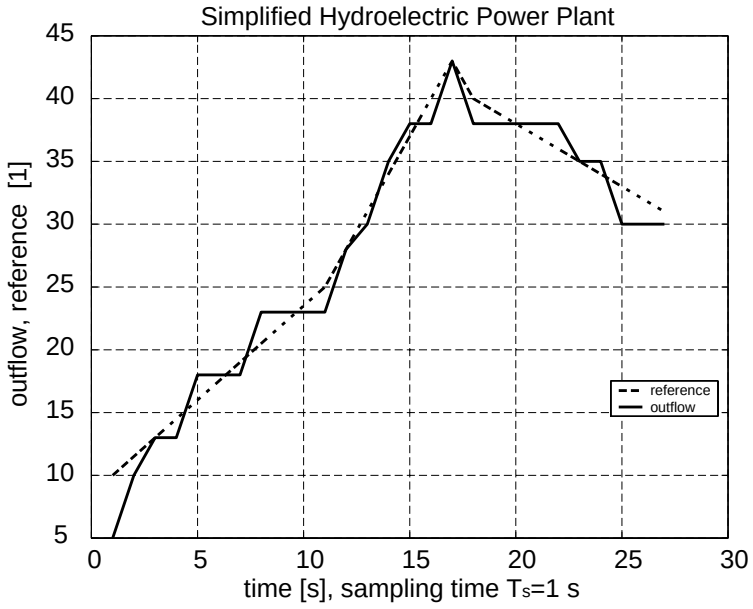


Figure 4.4: Trajectory tracking of the simplified power plant (normalized per unit representation).

In Figure 4.5 we see how the individual outflow elements are used to achieve the desired control goal.

For the successful application of the controller there are additional constraints and specifications to be taken into account. For instance if the desired outflow increases, the controller should first try to increase the openings of the turbines, rather than the openings of gates and flaps, see Table 3.12. For a complete description of such specifications we defer to (Chapuis, 1998). These specifications can be added as soft or as hard constraints in the model of the power plant and they can be systematically considered in the choice of the control moves by the model predictive controller. In Section 4.6 we outline, how to perform this inclusion without altering the type of the optimization problem to be solved.

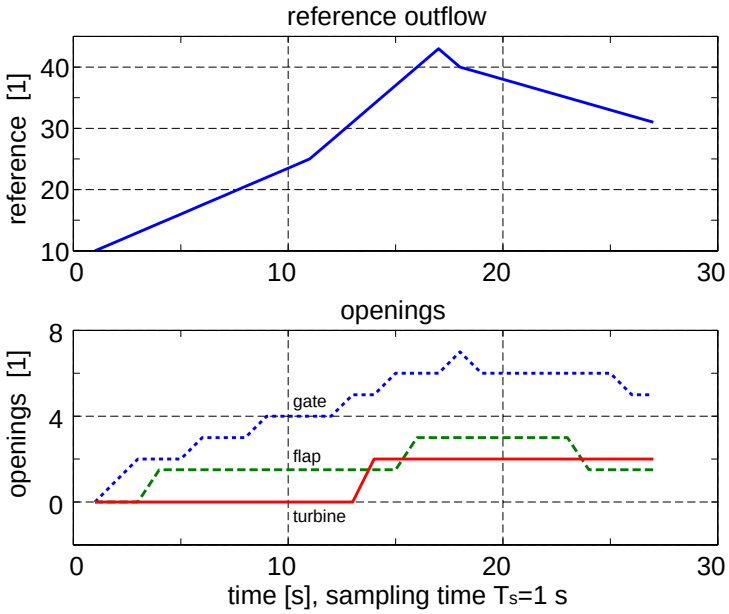


Figure 4.5: Openings of outflow elements for controlled simplified power plant (normalized per unit representation).

4.1.5 Summary

In this section we reported simulations and an application of MPC for hybrid systems to two plants. Concerning the experiments on the three tank system, we highlight some reasons contributing to the results presented here.

- The process under consideration is slow and therefore it can be operated with a relatively large sampling period. This allows to perform the optimization (4.1)-(4.4) required for controller synthesis. For the experiment in Figure 4.1 each MIQP at each time step could be solved in less than the sampling time of 10 seconds on a Pentium III, 900 MHz. The optimizations had the size mentioned in Table 4.2, which illustrates the size of affordable problems.

- The controller is robust to the plant-model mismatches introduced by the modelling procedure in Section 3.2. Moreover, the level variations stay within a small range,

number of binary variables	24
number of continuous variables	30
number of constraints	162
maximum time for solution	$\leq$ 10 sec
solver	Xpress-MP

Table 4.2: Data of the MIQPs for the experiment in Figure 4.1

where the linearized model has an acceptable accuracy.

- Except for the start-up phase, the system operates with a pump action which is always below 25 % of the full scale value. The measurement noise due to the water flowing in, is therefore in a tolerable range (Mignone and Monachino, 2001).

- The choice of weights in (4.1) is not so critical for the success of the scheme. In fact experiments with detuned weights still gave good results.

We also point out the difficulties, we were faced to, while implementing the controller.

- For fast variations of the reference, the terminal state constraint (4.3) may not be feasible and the controller fails to find a solution for the manipulated variables. In this case a larger terminal region must be defined or the reference has to be chosen such that it fulfills some additional rate constraints.

- As mentioned in Table 4.1, the control horizon was limited to three steps. Longer horizons required an increased amount of computations, violating the given sampling time frame.

4.2 Minimum Time Control

4.2.1 Introduction

The contents of this section have appeared as a technical report in (Torrisi, Mignone and Morari, 2001).

The minimum time control problem amounts to find a control action, that transfers, as fast as possible, the state of a system from an initial condition to some target region. This section presents a comparison of different heuristic and systematic approaches to solve this problem. Moreover, it shows how the MLD modeling framework allows to integrate heuristics in the systematic design. The techniques are motivated by, and applied to the three tank system.

4.2.2 Minimum Time Control of the Three Tank System

Consider the following problem: Given a plant Σ, its MLD model S, an initial state x_0 and a target set X_r on the state x of Σ, find the optimal control sequence $u(t), t = 1, \ldots, T$ such that $x(T) \in X_r$ and T is minimum.

For the three tank system, the problem can be reformulated in the following way: Assume that the tanks are initially empty and the outflow valves are closed, i.e. valves V_{L1}, V_{L2} and V_{L3} are closed. Given three constant set points r_1, r_2, r_3 for the liquid levels, find the control strategy that fills the three tanks to the desired levels in minimal time. In the laboratory set-up, the problem is encountered, if we would like to start experiments from a specific state of the system, and we would like to reach that state as fast as possible.

This problem is known in the literature as *minimum time control* or *time-optimal control*. It has been analyzed in the past for linear systems. For instance, Feldbaum (1962) shows

that for a stable, linear, n^{th}-order system with input constraints, the minimum time control signal is piecewise constant. It takes on either the upper or the lower bound on the input, i.e. it is of the bang-bang type. If the poles are real, then the control signal has at most $n-1$ switches. Sontag (1998) gives a rigorous treatment of minimum time control within the context of convex analysis. In (Rothwangl, 2001) the minimum time control problem for linear time invariant systems has been tackled by solving a number of mixed integer optimizations for different initial states, and implementing a look-up table of the results as a neural network. Each mixed integer optimization minimizes the time T needed to reach the reference by introducing a number of binary variables proportional to an upper bound to T. However, it is not clear how the neural network preserves the optimality of the control strategy and how this can scale up with the dimension and the time horizon. Gutman (1982) states the early idea for the use of mathematical programming to solve minimum time control problems.

The minimum time control problem arises in various practical situations, e.g. robotics (Ma and Watanabe, 2001) or in electronic systems (Jong and Seung, 1998).

4.2.3 Heuristic Control Strategy

In this section we show how it is possible to solve the problem by understanding the plant and stating some logic rules. Clearly this approach does not rely on any mathematical model of the plant and does not provide any guarantee of optimality. In fact, simulating and testing the control loop may leave subtle phenomena concealed. For large and complex systems, modifying such a controller may require a major effort.

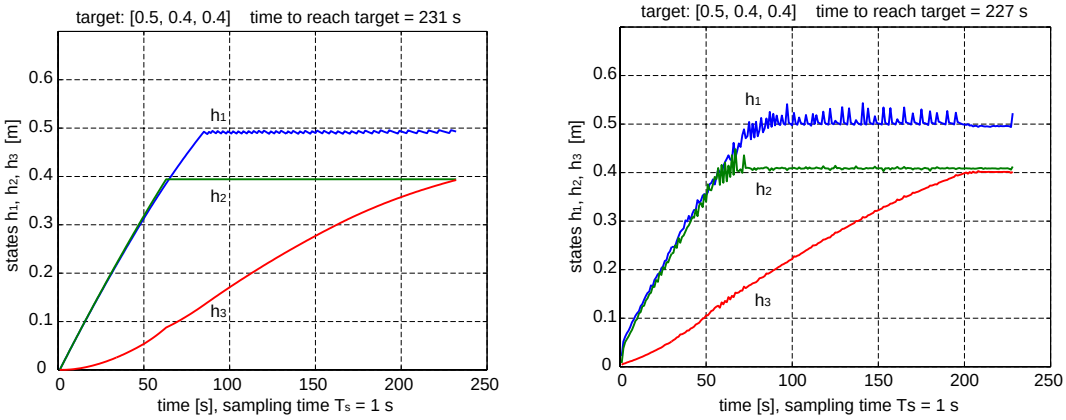
For the three tank system, we propose Algorithm 1 summarized in Table 4.3. The algorithm is basically a set of conditions that determine the control action according to some conditions on the levels and on the references. In particular Rule *3* switches both the pumps off when enough liquid has been pumped into the system. Rule *4* isolates tank 3

Algorithm 1

1 $V_1 = 1$, $V_2 = 1$, $V_{13} = 1$, $V_{23} = 1$, $Q_1 = Q_{\max}$, $Q_2 = Q_{\max}$;
2 **while** $\overline{(h_1 \geq r_1) \wedge (h_2 \geq r_2) \wedge (h_3 \geq r_3)}$
3 **if** $(h_1 + h_2 + h_3 \geq r_1 + r_2 + r_3)$ **then** $Q_1 = 0$, $Q_2 = 0$;
4 **if** $(h_3 \geq r_3)$ **then** $V_1 = 0$, $V_2 = 0$, $V_{13} = 0$, $V_{23} = 0$;
5 **if** $(h_1 \geq h_{\max})$ **then** $Q_1 = 0$;
6 **if** $(h_2 \geq h_{\max})$ **then** $Q_2 = 0$;
7 **if** $((h_1 \geq r_1) \wedge (r_3 \leq r_1) \wedge (r_3 \leq r_2)) \vee$
$((h_1 \geq r_1) \wedge (r_1 \leq r_2) \wedge (r_1 \leq r_3)) \vee$
$((h_1 \geq r_1) \wedge (r_2 \leq r_1) \wedge (r_2 \leq r_3) \wedge (h_3 \geq r_3))$ **then** $Q_1 = 0$;
8 **if** $((h_1 \geq r_1) \wedge (r_1 \leq r_2) \wedge (r_1 \leq r_3)) \vee$
$((h_1 \geq r_1) \wedge (h_3 \geq r_3) \wedge (r_2 \leq r_1) \wedge (r_2 \leq r_3))$ **then** $V_1 = 0$, $V_{13} = 0$;
9 **if** $((h_2 \geq r_2) \wedge (r_3 \leq r_2) \wedge (r_3 \leq r_1)) \vee$
$(h_2 \geq r_2) \wedge (r_2 \leq r_1) \wedge (r_2 \leq r_3)) \vee$
$((h_2 \geq r_2) \wedge (r_1 \leq r_2) \wedge (r_1 \leq r_3) \wedge (h_3 \geq r_3))$ **then** $Q_2 = 0$;
10 **if** $((h_2 \geq r_2) \wedge (r_2 \leq r_1) \wedge (r_2 \leq r_3)) \vee$
$((h_2 \geq r_2) \wedge (h_3 \geq r_3) \wedge (r_1 \leq r_2) \wedge (r_1 \leq r_3))$ **then** $V_2 = 0$, $V_{23} = 0$;

Table 4.3: Heuristic algorithm to control the three tank system

when $h_3 \geq r_3$. Rules *5* and *6* stop the inflow in tank 1 respectively tank 2 when the levels reach the physical limitations ($h_{\max}$). Note that if $r_3 < h_{\max}$ then Rule *4* guarantees that the level in tank 3 does not exceed the physical limitations. Rules *7* and *8* are somehow less immediate: They are the logical OR of different conditions, where each condition describes a possible scenario for the mutual position of the set-points r_1, r_2, r_3 and the satisfaction of the control specifications. Rule *7* switches off the pump feeding tank 1 when the tank is full and it cannot fill tank 3 any more. Rules *9* and *10* are the analogous rules for tank 2 as Rules *7* and *8*. Figure 4.6 shows the simulation and experimental results of the application of Algorithm 1.



(a) Simulation results

(b) Experimental results

Figure 4.6: Heuristics of Algorithm 1 for filling the tanks to $\underline{r} = [0.5, 0.4, 0.4]$

Note that the heuristics can be further improved by filling tank 1 over its reference r_1 to increase the flow to tank 3. This observation could be used to improve the behaviour by exploiting an overshoot in tank 1 to achieve a faster filling of tank 3. This can be done by substituting the condition for closing the valves in Rule *8* of Algorithm 1 as:

$$\begin{aligned} &((h_1 \geq r_1) \wedge (r_1 \leq r_2) \wedge (r_1 \leq r_3) \wedge (\mathbf{h_3 \geq r_1})) \quad \vee \\ &\quad ((h_1 \geq r_1) \wedge (h_3 \geq r_3) \wedge (r_2 \leq r_1) \wedge (r_2 \leq r_3)) \end{aligned} \tag{4.5}$$

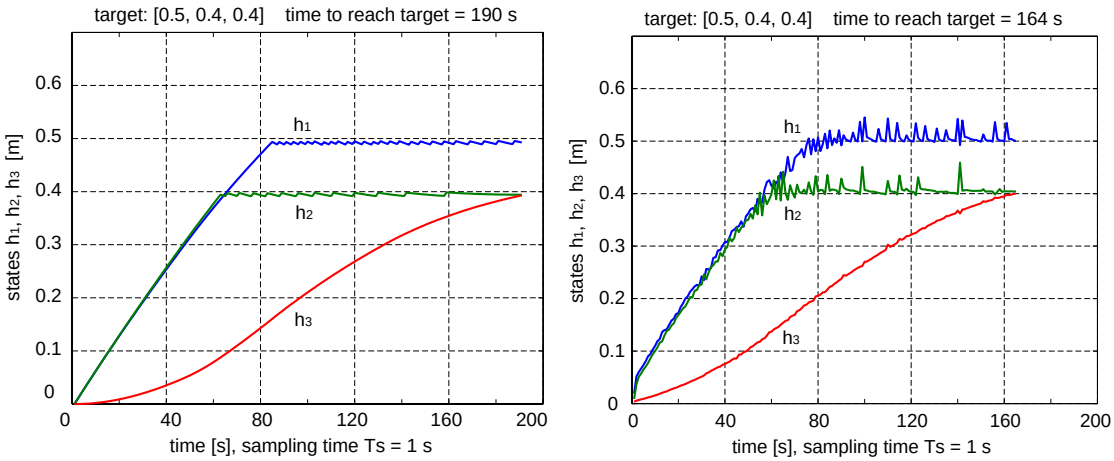
and in point (*8*) of Algorithm 1

$$\begin{aligned} &((h_2 \geq r_2) \wedge (r_2 \leq r_1) \wedge (r_2 \leq r_3) \wedge (\mathbf{h_3 \geq r_2})) \quad \vee \\ &\quad ((h_2 \geq r_2) \wedge (h_3 \geq r_3) \wedge (r_1 \leq r_2) \wedge (r_1 \leq r_3)) \end{aligned} \tag{4.6}$$

Figure 4.7 shows an experiment using the modified algorithm.

4.2.4 Open Loop Optimal Control

In this section we show how to find an open loop optimal control strategy by solving a series of MILPs.



(a) Simulation results

(b) Experimental results

Figure 4.7: Modified heuristics for filling the tanks to $\underline{r} = [0.5, 0.4, 0.4]$

First, exploit the linear structure of the MLD model (2.8) to impose the condition that the state reaches the terminal condition in T steps:

$$\begin{aligned} x(T) &= A^T x_0 + \textstyle\sum_{t=0}^{T-1} A^t \left[B_1 w(T-1-t) + B_2 \delta(T-1-t) + B_3 z(T-1-t)\right] \\ x(0) &= 0 \\ x(T) &\in X_r \\ &\text{subject to (2.8c) for } t = 1, \ldots, T. \end{aligned} \tag{4.7}$$

Clearly, all the constraints are linear. Therefore we can check if X_r is reachable from $x(T)$ by solving the associated *Mixed Integer Feasibility Test* (MIFT)[1].

Then, evaluate upper and lower bounds (T_M, T_m) on the optimal time such that the MIFT (4.7) is feasible for T_M and is infeasible for T_m. Finally run a bisection algorithm that performs a number of MIFTs to find the minimum time T such that MIFT (4.7) is feasible.

In our case study we set T_m by noting that the minimum filling time cannot be less than

[1]This is a special case of reachability analysis for MLD systems, for which an efficient algorithm exists (Bemporad, Torrisi and Morari, 2000).

the time needed by the pumps to transport the total amount of liquid specified by the reference values, i.e.

$$T_m = \frac{(r_1 + r_2 + r_3)A}{Q_{\max}}$$

where A is the cross section of the tanks. We compute T_M by simulating Algorithm 1.

To keep the computational time tractable, the trajectory produced by the heuristics, was used as initial guess for the feasibility tests. The mixed integer optimizer can take advantage of the suggested solution to direct properly the branch and bound algorithm.

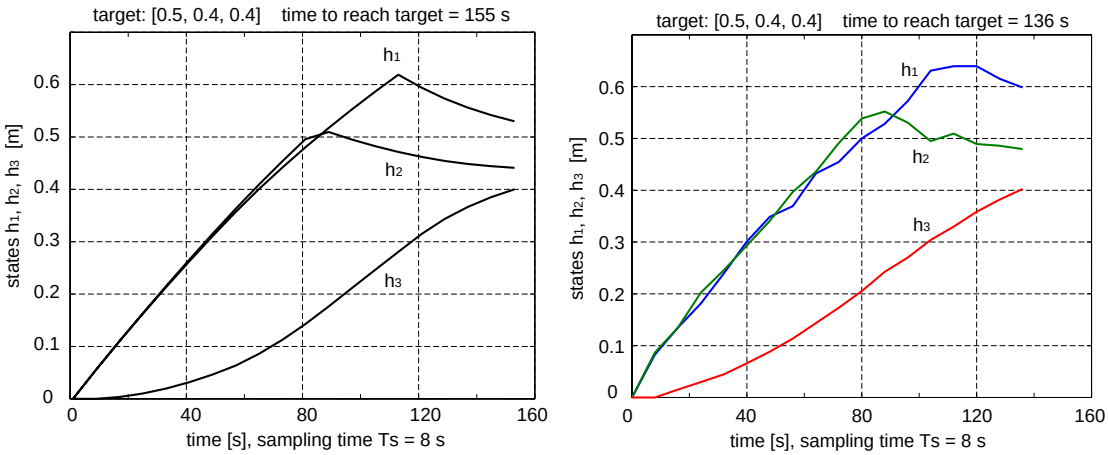
Moreover, assuming that in the optimal trajectory Valve V_1 and Valve V_{13} as well as Valve V_2 and Valve V_{23} are actuated at the same time samples, we can reduce the model by adding the constraints $V_1 = V_{13}$ and $V_2 = V_{23}$. Finally the proper choice of the sampling time is critical. Together with constraining the number of switches of the discrete inputs it can dramatically improve the computational performance.

Figure 4.8 shows the optimal open loop trajectory. In Figure 4.8(a) we show the simulation and in Figure 4.8(b) the application of the control action to the plant. The computation is indeed important as comparison, as it provides the lower bound on the filling time. The algorithm computed the optimal solution after 32 s, and the solution was proved to be optimum in 2707 s. The algorithm ran on a PC Pentium III 650 MHz using CPLEX (ILOG, Inc., 2000) on the reduced model sampled with $T_s = 8$ s.

4.2.5 Model Predictive Control

It is possible to recast the minimum time control problem as an MPC problem having the structure shown in Section 4.1.2. To enforce the minimum time requirement, we can extend the dynamical model of the plant with additional flag variables $s_i \in \{0, 1\}$ denoting the achievement of the desired references:

$$(s_i = 1) \leftrightarrow (h_i \geq r_i) \quad (i = 1, 2, 3) \tag{4.8}$$



(a) Simulation results (b) Experimental results

Figure 4.8: Open loop optimal control for filling the tanks to $\underline{r} = [0.5, 0.4, 0.4]$

and t_i denoting an acceptable final error:

$$(t_i = 1) \leftrightarrow (h_i \leq r_i + \epsilon) \quad (i = 1, 2, 3) \tag{4.9}$$

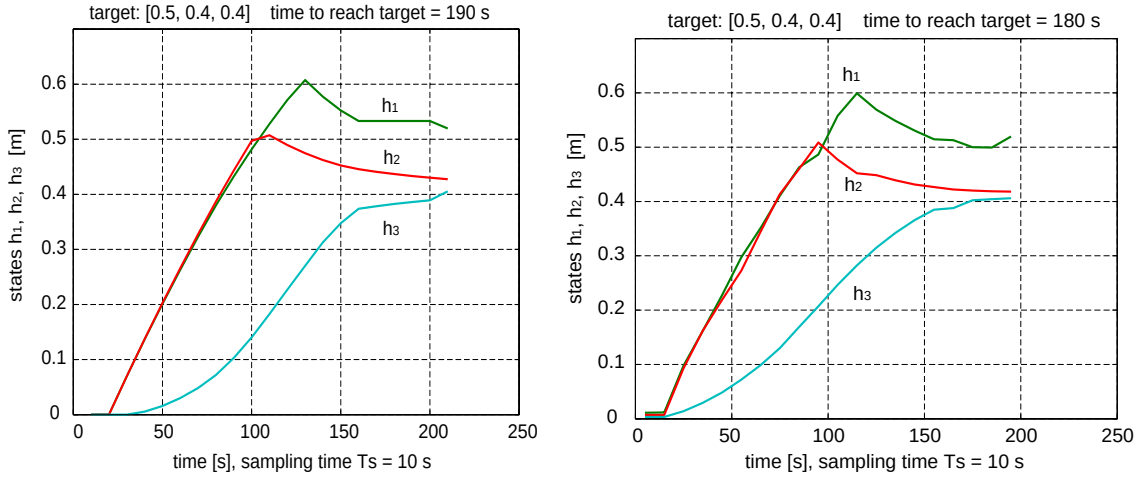
where ϵ is a tolerance. Finally, we add the variable

$$m = \overline{s_1 \wedge s_2 \wedge s_3 \wedge t_1 \wedge t_2 \wedge t_3} \tag{4.10}$$

By placing a high weight on this variable, the optimization, tries to reach the specified levels faster, at the price of large overshoots during the transient, see Figure 4.9.

4.2.6 Comparison of the Strategies

The main advantage of heuristic methods is that they don't need any mathematical model of the system and that they can be implemented on a cheap control hardware. In fact, they rely on some rules that the designer invents, once a clear understanding of the plant has been acquired. A well designed heuristics can perform extremely well, but there are only few ways to tune heuristics. Moreover they do not guarantee satisfaction of the physical



(a) Simulation results

(b) Experimental results

Figure 4.9: MPC for filling the tanks to $\underline{r} = [0.5, 0.4, 0.4]$

and operational constraints. Verification can provide the certification that the heuristics are safe, but provides few hints on how to improve them.

On the other hand, systematic methods rely on a theoretical, general purpose core that together with a model of the plant, solve the problem in an optimal way according to the chosen criteria. It is very easy to impose constraint satisfaction in optimization based methods. Due to their generality, these methods often require only minor changes in order to be ported to other systems.

MLD models allow the inclusion of heuristic rules to solve in a faster way the optimization associated with the control problem. In the tank system one such idea is to perform the optimization with the constraints $V_1 = V_{13}$ and $V_2 = V_{23}$ as shown in Sections 4.2.4 and 4.2.5. Heuristic methods can also support the solution of systematic methods by providing feasible initial solutions that tighten the bounds in the branch and bound search.

4.2.7 Summary

In this section we proposed several solutions to the minimum time control of an experimental hybrid plant using heuristic and systematic techniques. The systematic techniques rely on the MLD modeling framework and can be applied to any hybrid system modelled in MLD form. We showed the flexibility of MLD models in incorporating knowledge on the system to improve the performance of systematic methods.

4.3 Moving Horizon Estimation

Similar ideas as the model predictive control paradigm lead to the definition of a receding horizon estimator for hybrid systems. A *moving horizon estimation* (MHE) or *receding horizon estimation* scheme is an optimization based observer for the states of a system.

The ideas of MHE date back to the early nineties (Michalska and Mayne, 1992). The basic philosophy of MHE can be summarized as follows: The estimates of the states are obtained by solving a least squares problem, which penalizes the deviation between measurements and predicted outputs of a system. The data considered for the optimization are laying in a window of fixed finite length, which slides forward in time. Contrary to the control problem, the estimation horizon extends backwards in time, allowing at time t to estimate the quantities of interest at times prior to t.

MHE is appealing because of its capability to incorporate nonlinearities, and constraints on states and disturbances. Moreover, from a computational point of view, MHE algorithms are suitable for practical implementation because they amount to optimization problems of finite dimension. Rao et al. (1999) applied MHE to constrained linear systems, showing that it can guarantee stability of the estimate. This fact is important because in (Rao, 2000) and (Tyler and Morari, 1995) examples are reported, where the presence of constraints can cause instability of the classical Kalman filter. The use of an MHE scheme

for nonlinear plants was investigated in (Michalska and Mayne, 1992; Muske and Rawlings, 1995; Robertson et al., 1996) and a detailed analysis of its stability properties was given in (Rao and Rawlings, 1998) under the assumption that the state-transition and output maps are Lipschitz continuous.

Very little research has focused on observer theory for hybrid systems, although the ability to reconstruct the state of the system is fundamental for developing both state feedback control schemes and online system supervisions. In (Sontag, 1981) the conditions for the existence of a piecewise linear observer for piecewise linear systems are given. In (Alessandri and Coletta, 2001) an approach for the state estimation of hybrid systems is given based on the design of Luenberger observers. Stability of the estimation scheme is guaranteed with LMI criteria.

MHE with asymptotic convergence guarantees

Ferrari-Trecate, Mignone and Morari (2000) and Ferrari-Trecate, Mignone and Morari (2001) present a moving horizon estimation scheme for piecewise affine (PWA) systems and provide sufficient conditions on the design parameters in order to guarantee asymptotic convergence of the estimates. Here we discuss only the main results presented in those papers. As is customary in filtering theory, when noises are present on the input and the output, we consider autonomous PWA systems of the form

$$\hat{\Sigma}: \quad x(t+1) = A_i x(t) + f_i + w(t) \tag{4.11a}$$

$$y(t) = C_i x(t) + g_i + v(t), \quad \text{for } x(t) \in \mathcal{X}_i \tag{4.11b}$$

$$x \in \mathbb{X} \tag{4.11c}$$

$$w \in \mathbb{W} \tag{4.11d}$$

where $v(t) \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$ and $w(t) \in \mathbb{W} \subset \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$ is a bounded polyhedron containing the origin (i.e. there exist two matrices E_6 and E_7 such that $E_6 w \leq E_7$). The

matrices $\mathcal{X}_i$ define a polyhedral partition of the state space. The signals w and v model unmeasured input and output disturbances respectively. The condition $w \in \mathbb{W}$ takes into account constraints on the input disturbances.

We introduce the cost functional

$$J(\tau, t, w, v, x(\tau), \Gamma_\tau) \triangleq \Gamma_\tau(x(\tau)) + \sum_{k=\tau}^{t-1} \|v(k)\|_R^2 + \|w(k)\|_Q^2 \tag{4.12}$$

where $\tau, t \in \mathbb{N}$, $\tau < t$, Γ_τ is a continuous function and Q and R are positive-definite matrices of suitable dimension.

Consider a generic time instant T and assume that the output samples $y(k)$, $k = 0, \ldots, T-1$ have been measured. The *Full Information (FI)* observer is defined as the estimation of the state trajectory $x(k)$ obtained by solving the constrained minimization problem

$$\min_{x(0), w} J(0, T, w, v, x(0), \Gamma_0) \text{ subj. to } (4.11). \tag{4.13}$$

From (4.13) it is apparent that the matrices Q and R weight the deviation between measured and predicted state/output. In a probabilistic setting, they should be chosen as the inverse of the covariance matrices of the noises w and v. Similarly, Γ_0 should reflect the confidence we have about how much the true initial state of (4.11) differs from an initial guess $\bar{x}$ of the state $x(0)$. The function Γ_0 is assumed to be bounded on $\mathbb{X}$ and such that $\Gamma_0(x) \geq 0$, $\forall x \in \mathbb{X}$ and $\arg\min_{x \in \mathbb{X}} \Gamma_0(x) = \bar{x}$.

The FI observer uses all the collected output samples $y(k)$, $k = 0, 1, \ldots, T$ in order to reconstruct the state trajectory. The use of all the available information is desirable for the accuracy of estimation but is unappealing from a computational point of view. Indeed, the main drawback of the FI scheme is that, as T increases, the number of optimization variables in (4.13) grows without bounds. To overcome this problem, one can use forward dynamic programming. Consider a time instant $T > M$, where $M \in \mathbb{N}^+$ is a fixed horizon. Then one can rewrite (4.13) as

$$\min_{x(T-M), w} J(T-M, T, w, v, x(T-M), \bar{\Gamma}_{T-M}(x(T-M))) \text{ subj. to } (4.11) \tag{4.14}$$

where

$$\begin{aligned} \bar{\Gamma}_{T-M}(z) \quad &\triangleq \quad \min_{x(0),w} J(0, T-M, w, v, x(0), \Gamma_0) \\ \text{subj. to} \quad & (4.11) \\ & x(T-M) = z \end{aligned} \tag{4.15}$$

and z belongs to the reach set $R(X_0, T-M)$ at time $T-M$. For a set of initial states X_0, the *reach set* $R(X_0, t)$ is defined as the set of all states x, for which there exist a trajectory $x_0 \dots x_t$ with $x_0 \in X_0$ and $x = x_t$ (see e.g. (Varaiya, 1998)). The size of the optimization (4.14) is now bounded and the penalty $\bar{\Gamma}_{T-M}$ (the so-called *arrival cost*) relates the fixed-horizon problem (4.14) to the FI problem. As discussed in (Rao and Rawlings, 1998), the arrival cost intuitively summarizes the information carried by the past data $y(k)$, $k = 0, \dots, T-M-1$.

If the system (4.11) is linear and unconstrained, we can compute the arrival cost by using the Kalman Filter covariance update recursion. For the general PWA form, the exact calculation of the arrival cost is an involved problem. Therefore we propose an approximation. In an MHE scheme we maintain the computational advantages of (4.14) by solving, at each time instant $T \geq 0$

$$\Theta^*{}_T = \min_{x(T-M),w} J(T-M, T, w, v, x(T-M), \Gamma_{T-M}) \text{ subj. to } (4.11), \tag{4.16}$$

where, by convention $T-M$ is set to zero if $T < M$ and Γ_{T-M} is a sequence of penalties at the beginning of the horizon, called *initial penalties* that should approximate in a suitable sense the arrival cost $\bar{\Gamma}_{T-M}$. In fact, note that (4.16) and the FI estimator (4.14) coincide if $\bar{\Gamma}_{T-M} = \Gamma_{T-M}$.

In Figure 4.10 we describe the difference between the scheme outlined here and the case, where MHE is performed using the most recent data in a window of fixed length only, i.e. if $\Gamma_{T-M} = 0$. If we neglect data outside the window, we may experience sensitivity to noise or convergence problems (Rao and Rawlings, 1998). The summary of neglected

data, included in the cost function to optimize allows to use the relevant information for estimation, by keeping the size of the mathematical problem bounded.

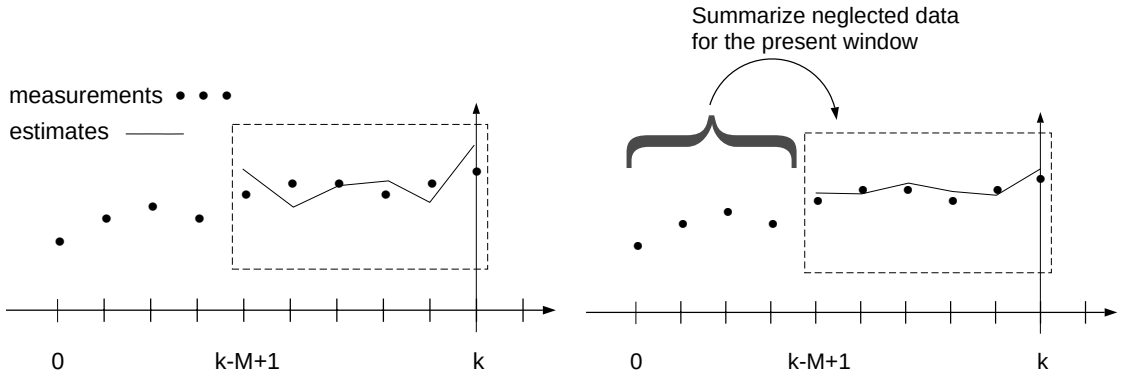


Figure 4.10: Moving horizon estimation. Left: Neglecting data outside the Estimation Window. Right: Approximating neglected data for the current estimation.

The main result of Ferrari-Trecate, Mignone and Morari (2001) shows that an approximation of $\bar{\Gamma}_{T-M}$ suffices for convergence. The authors provide methods based on Linear Matrix Inequalities (see Appendix C) to calculate such approximations in a recursive way. The computation of the initial penalties can be done quite efficiently. However, the scheme shows only a small improvement in practical estimation problems for PWA systems, compared to the case, where past data is neglected (Figure 4.10, left). An improvement is expected to be noticed if the signal to noise ratio deteriorates, as the estimation progresses. In this case the information at the beginning of the estimation is valuable. Discarding it in favour of more recent, but noisy measurements is likely to lead to bad estimates. An illustrative example for such a system is provided in (Ferrari-Trecate, Mignone and Morari, 2001).

4.4 Fault Detection of Mixed Logic Dynamical Systems

4.4.1 A Brief Survey of Fault Diagnosis Literature

Many processes are subject to faults or malfunctions. During nominal operation, sensors or actuators can break down and fail to accomplish their task. The failure of subsystems or components can partially or fully disable the controller action. This results in an unsatisfactory performance of the system, decreased availability, emergency shutdowns or even significant damages to the plant and to its environment. For this reason, recently several research activities focused on the safety aspects of control systems (Gertler, 1998; Chen and Patton, 1998). Indeed, control engineers have become aware of the need to consider possible faults already in the early design phase of a plant. The requirement for a physical system to be reliable and safe has gained importance. The system should be able to operate satisfactorily in case of malfunctions of some of its components or it should at least report the increased likelihood of an abnormal system operation.

The techniques found in the literature tackle the safety problem on different levels of detail and complexity. The field of fault tolerant control merges several disciplines into a framework with a common goal (Blanke, 1999), namely to provide tools and guidelines for coping with faults in control systems. Research activities on fault diagnosis have increased lately mainly due to the following reasons:

- There is a demand for high efficiency and availability, as well as short down-times of the plants.
- Real systems are operating under more complex and sophisticated algorithms than in earlier times.
- The reliability of the system is becoming a performance specification to be fulfilled

by the system designer.

- Advances in computing power allow the implementation and application of new algorithms and tools.

- In some cases the control system is operating satisfactorily, and controller design is not a crucial issue anymore. However, there is the need to improve safety in the control systems.

Even though there exist several books and survey articles on the field of fault diagnosis of dynamical systems, there are still some inconsistencies to what concerns the terminology used (Chen and Patton, 1998; van Schrick and Christiansen, 1997). In the context of this work, a *fault* is considered as an undesired deviation of the characteristics of a system component. Under *fault detection* we understand the act of identifying the presence of a fault in a system. *Fault isolation* is concerned with the determination of kind, location and time of detection of a fault. The term *fault diagnosis* encompasses both fault detection and isolation.

Fault diagnosis techniques can be classified in two groups: model-based approaches and model-free approaches.

Model-free approaches or signal-based approaches perform fault diagnosis without relying on an explicit model of the process. The basic idea is to measure certain signals of the system that are likely to carry information about faults (Chen et al., 2001). The comparison of these signals with a database of signals reflecting different behaviours, allows to characterize the current operation with respect to faults. Model-free approaches use signal processing techniques, like e.g. neural networks, spectral power densities, frequency spectrum estimation and require an extensive training scheme prior to their application. Simpler schemes may only take into account trends, mean and limit values, or magnitudes of the measured signals. In some cases it is difficult to perform fault isolation with these

schemes (Chen and Patton, 1998).

Model-based approaches perform fault diagnosis using an explicit model of the process. Here a fault is diagnosed from the comparison of available measurements with a priori information given by the system's mathematical model. These techniques rely on the principle of consistency-based diagnosis (Chen et al., 2001). A fault can be declared, either if an inconsistency of the measured input-output sequence to the fault-free model is detected or if a consistency of the data to a faulty model can be found. The model used for fault diagnosis can be of qualitative nature; such approaches are advocated to be less fault sensitive, but they can be applied, even if an exact mathematical model of the plant is not available (Glass et al., 1995).

In most cases these approaches rely on the generation of *residuals*. Residuals are signals that determine the mismatch between the prediction of the system response given by the mathematical model and the corresponding measured signals. The two main tasks in the design of a model-based fault diagnosis scheme are the residual generation and the processing of the residuals, i.e. the decision making to get a statement about the current fault situation. A review about common methods used for these two tasks can be found in (Chen et al., 2001).

Other methods rely on parameter estimation to perform fault detection and isolation. The diagnosis scheme for MLD systems presented next is related to this idea. In this section we focus on fault detection and isolation for MLD systems. The question about the remedies to be taken in case of faults is considered in Section 4.5 on reconfiguration of hybrid systems.

4.4.2 Fault Detection of MLD Systems

The moving horizon estimation approach can be used to detect failures in a hybrid system. If an MLD model of a faulty system is available[2], as in Section 3.2.6, the least squares estimation problem over a finite horizon backwards from the current time is used to estimate the fault variables ϕ in (3.37). As in the state estimation, the resulting problem is an MIQP.

In this context we focus on faults that are modelled as binary variables, although extension to continuous valued faults are possible. The types of faults that can be considered in the MLD framework include the main classes of faults considered in practice. These are:

- Actuator faults
- Sensor faults
- Plant faults

The key idea is to model the effects of a fault on the system as logic propositions, as shown in Chapter 2 and to translate them into inequalities to be added to the MLD model of a system.

A moving horizon estimator for (3.37) can be set up with the same cost function as in (4.12). We propose here an alternative structure that neglects the arrival cost (see Section 4.3) and allows to have more influence on the solution by providing more design parameters. It is formulated as follows. At time t the estimator knows the last T input and output data $U(t)$ and $Y(t)$:

$$U(t) = [u(t-T),\ u(t-T+1),\ \ldots,\ u(t-1),\ u(t)] \tag{4.17}$$

$$Y(t) = [y(t-T),\ y(t-T+1),\ \ldots,\ y(t-1),\ y(t)] \tag{4.18}$$

[2]I.e. if the system behaviour in presence of faults is known and can be modelled within an MLD system.

and the estimates $\hat{Z}(t-1)$, $\hat{\Delta}(t-1)$, $\hat{\Phi}(t-1)$ and $\hat{X}(t-1)$ from the estimation at time $t-1$:

$$\hat{Z}(t-1) = [\hat{z}(t-T|t-1),\ \hat{z}(t-T+1|t-1),\ \dots,\hat{z}(t-2|t-1)] \tag{4.19}$$

$$\hat{\Delta}(t-1) = [\hat{\delta}(t-T|t-1),\ \hat{\delta}(t-T+1|t-1),\ \dots,\ \hat{\delta}(t-2|t-1)] \tag{4.20}$$

$$\hat{\Phi}(t-1) = [\hat{\phi}(t-T|t-1),\ \hat{\phi}(t-T+1|t-1),\ \dots,\ \hat{\phi}(t-2|t-1)] \tag{4.21}$$

$$\hat{X}(t-1) = [\hat{x}(t-T|t-1),\ \hat{x}(t-T+1|t-1),\ \dots,\ \hat{x}(t-2|t-1),\hat{x}(t-1|t-1)] \tag{4.22}$$

The quantities, we are interested in, are the fault estimates $\hat{\phi}$. At time t we can consider the following estimate evolution:

$$\begin{cases} \hat{x}(t-T|t) & \triangleq & \hat{x}(t-T|t-1) & +\Delta x(t) & \\ \hat{x}(t+k+1|t) & = & A\hat{x}(t+k|t) & +B_1u(t+k) & +B_2\hat{\delta}(t+k|t)+ \\ & & B_3\hat{z}(t+k|t) & +B_6\hat{\phi}(t+k|t) & +\xi(t+k|t) \\ \hat{y}(t+k|t) & = & C\hat{x}(t+k|t) & +D_1u(t+k) & +D_2\hat{\delta}(t+k|t)+ \\ & & D_3\hat{z}(t+k|t) & +D_6\hat{\phi}(t+k|t) & +\zeta(t+k|t) \\ E_2\hat{\delta}(t+k|t) & +E_3\hat{z}(t+k|t) & \leq E_4\hat{x}(t+k|t) & +E_1u(t+k) & +E_5+E_6\hat{\phi}(t+k|t) \end{cases} \tag{4.23}$$

for $k=-T,\dots,-1$. Let us define the optimization variable at time t as:

$$\begin{aligned} \chi_t = [&\Delta x(t),\hat{\delta}(t-T+1|t),\dots,\hat{\delta}(t-1|t),\hat{z}(t-T+1|t),\dots, \\ &\hat{z}(t-1|t),\hat{\phi}(t-T+1|t),\dots,\hat{\phi}(t-1|t), \\ &\xi(t-T+1|t),\dots,\xi(t-1|t),\zeta(t-T+1|t),\dots,\zeta(t|t)] \end{aligned} \tag{4.24}$$

and the cost function at time t as:

$$\begin{aligned} J(\chi_t) = & ||\Delta x(t)||^2_{Q9} + \\ & \textstyle\sum_{k=-T+1}^{0}\left(||\hat{y}(t+k|t)-y(t+k)||^2_{Q5}+||\zeta(t+k|t)||^2_{Q8}+||\hat{\phi}(t+k|t)||^2_{Q_{10}}\right)+ \\ & \textstyle\sum_{k=-T+1}^{-1}\left(||\hat{x}(t+k|t)-\hat{x}(t+k|t-1)||^2_{Q4}+||\xi(t+k|t)||^2_{Q7}\right)+ \\ & \textstyle\sum_{k=-T+1}^{-2}\Big(||\hat{\delta}(t+k|t)-\hat{\delta}(t+k|t-1)||^2_{Q2}+||\hat{z}(t+k|t)-\hat{z}(t+k|t-1)||^2_{Q3}+ \\ & ||\hat{\phi}(t+k|t)-\hat{\phi}(t+k|t-1)||^2_{Q6}\Big) \end{aligned} \tag{4.25}$$

where the matrices Q_i are symmetric, positive semidefinite and have appropriate dimensions. The different upper limits of the indices for the summations in (4.25) are due to the availability of the summands at a given time. The estimates at time t are obtained by solving the optimization problem:

$$\begin{aligned} &\min_{\chi_t} J(\chi_t) \\ \text{subject} \quad &\text{to} \quad (4.23) \end{aligned} \tag{4.26}$$

With the estimates χ_t and with the estimate evolution (4.23), we can reconstruct the state estimate $\hat{X}(t)$:

$$\hat{X}(t) = [\hat{x}(t-T+1|t), \ldots \hat{x}(t-1|t), \hat{x}(t|t)] \tag{4.27}$$

Note that the optimization problem is an MIQP. The effects of the weighting matrices are listed next along with advice on their usage. The advice should be taken as rules of thumb, since it is motivated by our practical experience with the case study exposed in Section 4.4.4.

Q_2,Q_3,Q_4,Q_9 Large values of these matrices give "inertia" to the estimator since variations from the estimate at time t to the estimate at time $t-1$ are penalized. We recommend to set these weights to a relatively low value, in order to get estimates with the most recent fault information, and the estimator to be fault sensitive.

Q_5 Similarly as above, this matrix represents the inertia of the output estimate. We recommend to set this weight to a medium value in order to allow enough freedom for the estimator to track the measurements but at the same time to avoid large changes of estimates at subsequent times.

Q_6 Large values of Q_6 penalize the frequent switching of the binary fault indicator variables. We recommend to set this weight to a relatively high value, reflecting the fact that faults are events that do not appear very often.

Q_8 This matrix penalizes the output error, i.e. the deviation of measured and estimated outputs. It should contain large values if the noise level on the output is low, and medium values otherwise.

Q_7 This matrix penalizes the model error. Similar to the Kalman filtering, if we allow our model to be inaccurate, Q_8 should contain smaller values than Q_7.

Q_{10} This matrix penalizes the occurrence of fault estimates. Large values in Q_{10} cause the estimator to be "cautious" in detecting faults and it reflects our wish to fit the data using the nominal model, rather than the fault dynamics.

Tuning the relatively large set of design parameters can be a difficult task. A scaling of the system variables to a common range can help to compare the contribution of the individual terms of the cost function. Note that the form (3.37) allows to describe systems, whose behavior depends on the faults in a nonlinear manner. It is possible to include multiplicative faults or actuator failures, as it is shown for the three tank system. The key idea is to express the occurrence of these faults as a propositional logic statement involving binary and continuous variables.

In analogy to classical fault detection schemes, we can interpret the value of the cost function (4.25) as the magnitude of a residual signal. On one hand this allows to provide a measure of reliability of the estimates. On the other hand this allows to detect the presence unmodelled faults. In the latter case the scheme is only able to perform fault detection, leaving the task of fault isolation to further analysis. Note that a classical residual signal is a signed quantity, whereas the value of the cost function (4.25) is nonnegative. In this sense it carries less information than a residual signal.

In the form presented in this work the moving horizon estimator does not address all relevant issues in state estimation and fault detection. For example, it does not explicitly take into account any stochastic aspects. Nevertheless, it is a new and promising method to deal with state estimation and fault detection for the broad class of hybrid systems.

4.4.3 Application to the Three Tank System: Simulation

To illustrate the effectiveness of the fault diagnosis scheme, we applied the method to the three tank benchmark system of Section 3.2. In the simulation of Figures 4.11-4.12 we simulated all possible combinations of faults that can occur in the system, according to Table 4.4. This simulation has been performed with a model parameterized with the values of the original benchmark system.

time			faults
0	-	245	none
250	-	370	ϕ_1
375	-	495	none
500	-	620	ϕ_2
625	-	745	none
750	-	870	ϕ_3
875	-	995	none
1000	-	1120	ϕ_1
1125	-	1245	ϕ_1, ϕ_2
1250	-	1370	ϕ_1
1375	-	1495	ϕ_1, ϕ_3

Table 4.4: Fault sequence in the simulation of Figures 4.11, 4.12

In Table 4.5 we report some parameters of the simulation in Figures 4.11-4.12. To what concerns the timing, we remark that the solver used is the code `miqp.m` described in Section 5.5. The code is not optimized for speed, therefore the duration of the simulation has to be understood as a loose upper bound to what is achievable with commercial solvers. The simulation ran on a Sun Sparc Ultra 10 workstation (333 MHz, SunOS 5.7) using Matlab R12.1. In this case the average computational time exceeds the sampling time

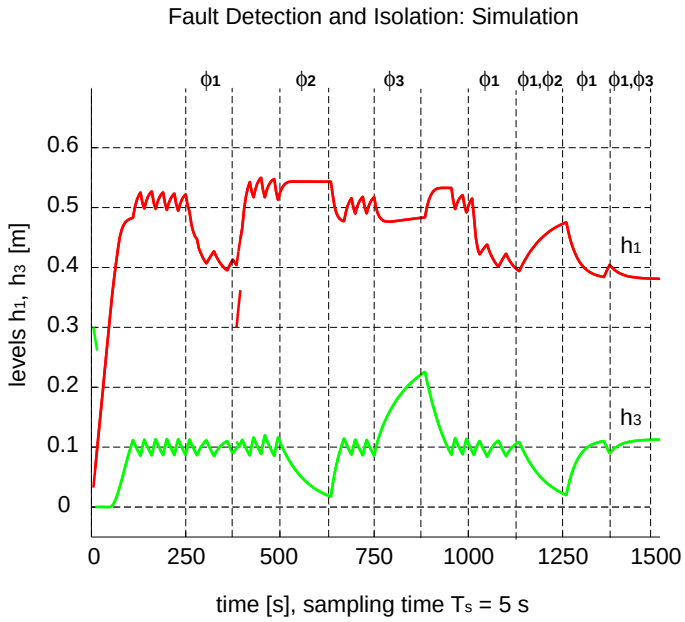


Figure 4.11: State trajectories during fault detection simulation; the sequence of faults is given on top of the plot

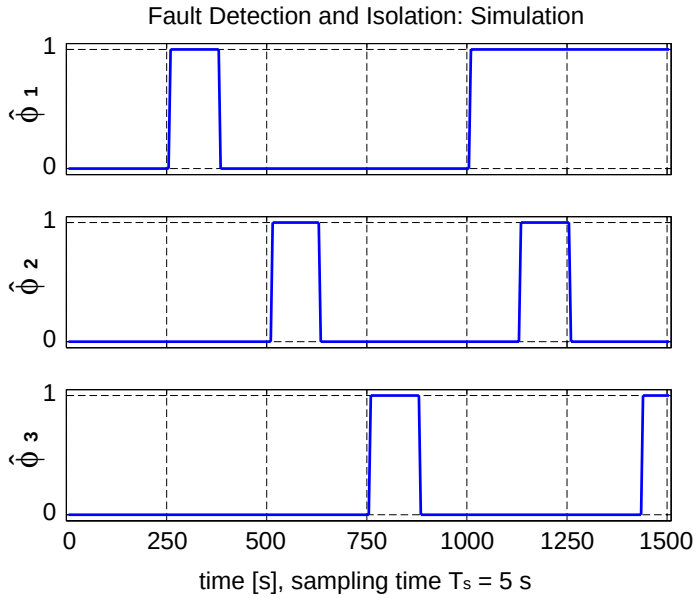


Figure 4.12: Fault estimates during fault detection simulation. The sequence of faults is given in Table 4.4

of 10 seconds. In the experimental set-up this problem is overcome thanks to the faster computer used there. The number of QPs to be solved for each MIQP lies below 2% of the worst case number.

Number of binary variables for each MIQP	12
Number of continuous variables for each MIQP	26
Number of MILPs solved (Simulation)	303
Number of MIQPs solved (Fault Detection)	300
Total execution time	4213 sec
Average time for each computational step	13.9 sec
Total number of QPs solved	21'416
Average number of QPs for each step	72
Maximum number of QPs over all steps	169
Minimum number of QPs over all steps	1
Worst case number of QPs for each MIQP	8191
Number of constraints for each MIQP	82
Horizon length	3 steps

Table 4.5: Parameters in the simulation of Figures 4.11, 4.12

The MLD framework for fault detection allows to include heuristic knowledge in the estimation procedure that can be used to avoid false alarms. In the three tank system we can formulate one such simple rule as

$$([h_1 \leq h_v] \wedge [h_3 \leq h_v]) \Rightarrow [\overline{\phi_2} \wedge \overline{\phi_3}] \tag{4.28}$$

The proposition (4.28) determines that faults concerning valve V_1 being blocked cannot be identified, if the liquid level in tanks 1 and 3 are both lower than the location of the valve itself. Indeed, in this case there is no information available allowing to detect a failure of valve V_1. In the simulation of Figure 4.13 we simulated an occurrence of ϕ_1

for $(200 \leq t \leq 350)$, an occurrence of ϕ_2 for $(350 \leq t \leq 500)$ and an occurrence of ϕ_3 for $(500 \leq t \leq 600)$. Moreover we artificially limited the time allowed for computations, simulating hard real time constraints. In Figure 4.13 (left) we see that the scheme reports a false alarm and it detects fault ϕ_3 only at $t = 600$, despite its earlier occurrence. On the right plot we used the same design parameters but enhanced the model with additional heuristics. We note that the false alarm has disappeared. The detection of ϕ_3 is still delayed, but it occurs one step earlier than before.

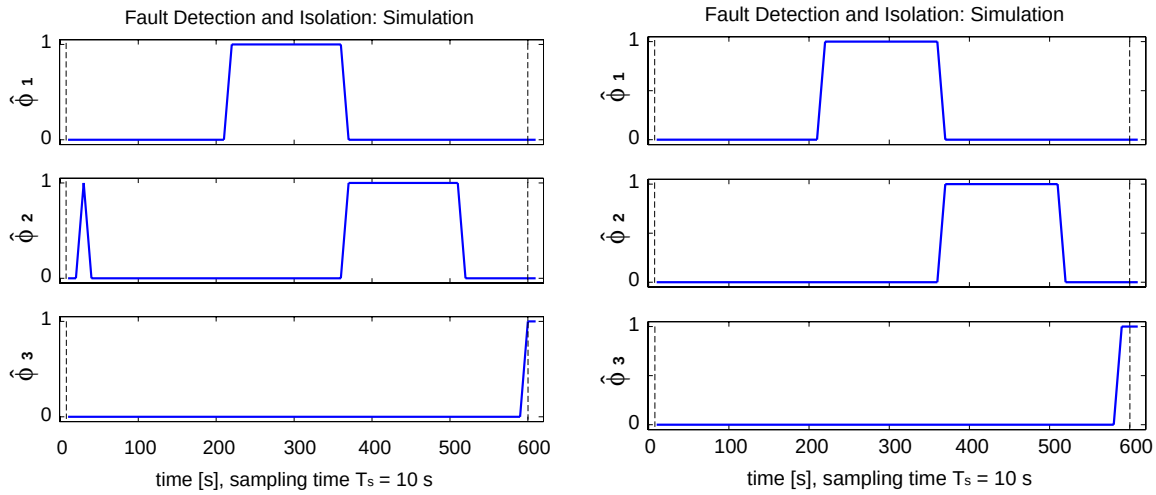


Figure 4.13: Fault estimates during fault detection simulation with limited computational time. Left: A False alarm occurs. Right: The false alarm disappears thanks to a heuristic rule added to the model.

4.4.4 Application to the Three Tank System: Experiment

The scheme is able to produce acceptable results when working with measurements on the real plant. The set-up for the experiments shown in this section is reported in (Mignone and Monachino, 2001). All experiments shown in Figures 4.14-4.16 have been performed with the same set of design parameters in (4.25). The scheme was able to correctly detect each fault with only 2 to 3 time steps of delay. The search for a set of weights that avoid false

Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}
$0.001 \cdot I$	$0.001 \cdot I$	$0.001 \cdot I$	$1 \cdot I$	$1000 \cdot I$	$1000 \cdot I$	$10 \cdot I$	$0.001 \cdot I$	$6 \cdot I$

Table 4.6: Weights for fault detection experiments in Figures 4.14-4.16, I denotes the identity matrix of appropriate dimensions.

alarms and do not exhibit missed detections is a delicate procedure that has to be performed by extensive simulation, following the hints given in Section 4.4.2. The experiments have been implemented with the weights given in Table 4.6. Further parameters are listed in Table 4.7. The choice of the sampling time and the horizon length is motivated by the amount of computations required to solve the MIQPs at each time step. In these experiments the fault detection procedure has been performed in real time, without missing the given time frame. We used the solver `miqp.m`, see Section 5.5.

estimation horizon	3 steps
sampling time (= 1 step)	10 s
cases, where time frame was missed	none
instant of fault occurrence	10 steps

Table 4.7: Parameters of the fault detection experiments in this section

The state estimates in the left plots of Figures 4.14-4.16 show the measurements as solid lines and the state estimates as short superposed lines. The latter represent the state estimates at different time instances, as it is depicted in Figure 4.17. Each line segment of the estimates has the length of the estimation horizon.

In our experience the detection of faults is not equally difficult for each of the three fault scenarios. The tuning of the algorithm for correct detection of fault ϕ_1 is more delicate. In fact, for a leak of small size, the detectability is bad, since the effect on the measured variables can hardly be noticed on the output.

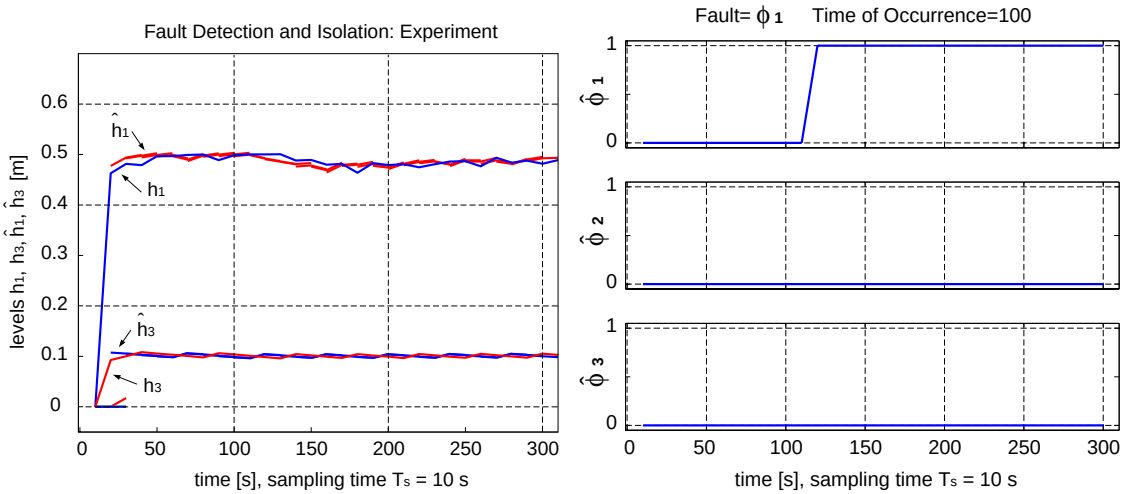


Figure 4.14: Detection of fault ϕ_1 on the laboratory model: The fault occurred at time step $t = 100$. Left: Measured and estimated state trajectories; Right: Estimated fault

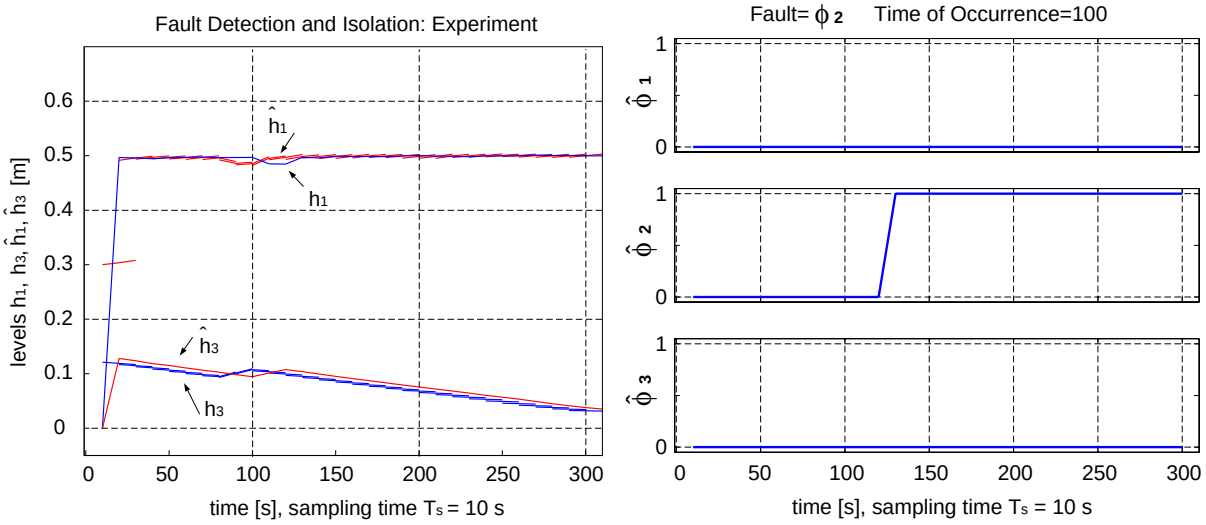


Figure 4.15: Detection of fault ϕ_2 on the laboratory model: The fault occurred at time step $t = 100$. Left: Measured and estimated state trajectories; Right: Estimated fault

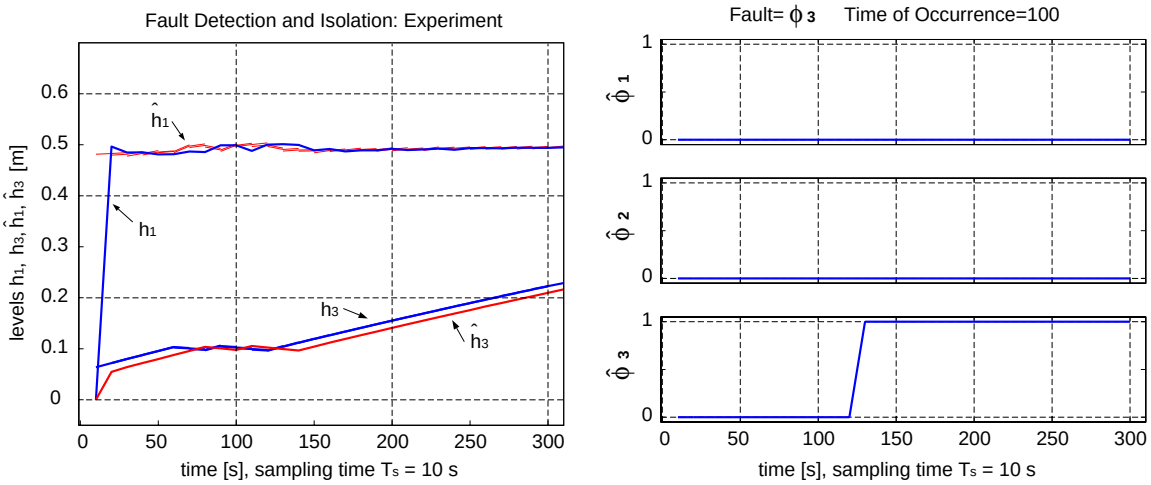


Figure 4.16: Detection of fault ϕ_3 on the laboratory model: The fault occurred at time step $t = 100$. Left: Measured and estimated state trajectories; Right: Estimated fault

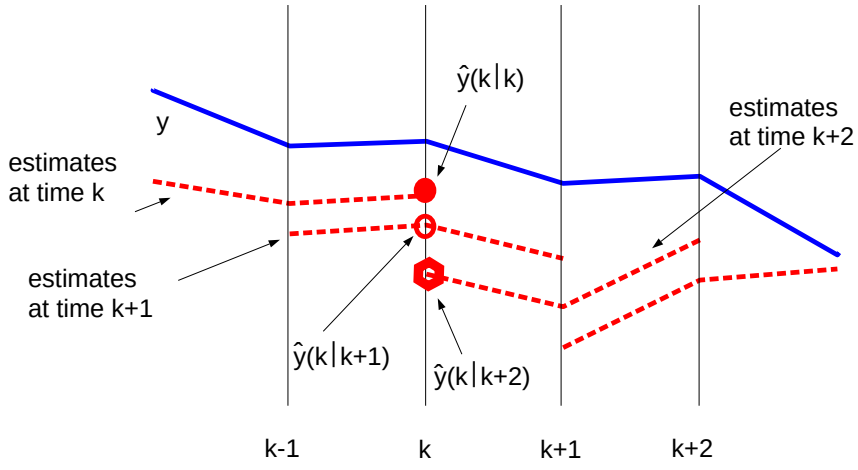


Figure 4.17: Magnified view of a state estimate plot in moving horizon estimation

4.4.5 Summary

The success of the fault detection and isolation experiments can be attributed to a number of reasons that are discussed in detail in (Mignone and Monachino, 2001). We notice for instance that the pump action to keep the system at the reference value is low, consequently the measurements are corrupted by noise of negligible magnitude. Moreover, the asymmetric characteristics of the valves is not affecting fault detection, since the flows are in one direction only for each valve. We remark that the estimation horizon was limited by the available computing power. Indeed, for horizons longer than three steps, computations lasted too long and the time frame was missed. The simulation in Figure 4.13 outlines a strength of using MLD systems for fault detection. Unlike other fault detection schemes where fine tuning is performed by extensive training schemes, for MLD system the formulation of heuristic rules supporting fault detection can be directly integrated in the models. False alarms due to incomplete modelling and inappropriate choice of the weights can be elegantly corrected in some cases. On the other hand our experience shows that choosing the weights avoiding false alarms and missed detections is in general a difficult task. The guidelines we gave in this section aim at relieving this problem. Moreover, the mathematical problems we are generating are sometimes badly conditioned and hard to solve, despite the robustness of the available tools. We recommend a normalization of the optimization problem in such cases.

4.5 Reconfiguration of Mixed Logic Dynamical Systems

4.5.1 Introduction

In this section we consider the reconfiguration problem using the MLD framework. After some general remarks about fault tolerant control systems in Section 4.5.2, we outline the reconfiguration problem for the three tank system in Section 4.5.3. The main contribution is in Sections 4.5.5, 4.5.6 and 4.5.7, where we present three strategies for reconfiguration of MLD systems. Some simulations and experiments are reported in Sections 4.5.8 and 4.5.9. We conclude with a comparison of the methods in Section 4.5.10.

4.5.2 Fault Tolerant Control Systems

A fault tolerant control system is able to maintain stability and acceptable performance in the presence of a set of predefined failures in its components. There are two approaches to achieve these goals. On one hand techniques are available based on *passive fault tolerance*. These systems are designed in order to be robust against the occurrence of failures. Usually these systems carry on their operation even in presence of faults and no particular qualitative changes are required for the control scheme. However, since the system behaviour and its operating conditions in case of faults can span over a wide range, the design is often conservative. Therefore, the performance degradation can be significant. Some works in this area include (Veillette et al., 1992), where the authors propose a reliable control scheme with guaranteed stability and H_∞ performance, both in the nominal case and in the case of sensor or actuator failures. Zhou and Ren (2001) propose a control structure, where a given control loop is enhanced with an additional controller that renders the overall loop robust against model uncertainties or faults.

On the other hand, the field of *active fault tolerant systems* encompasses those systems, where the information about the presence of faults is used to reconfigure the control scheme. The controller structure and the controller parameters may be changed after detection of anomalies. Here the system is designed to be adapted to the current plant operation. This can be done either through online-computations deciding in real-time about the best remedy to be taken, or via precomputed strategies and look-up tables that are set up correspondingly. In order to devise an active fault tolerant system the concept of redundancy is of key importance. For a system to be reconfigurable, a certain amount of hardware and/or software redundancy has to be present. Contrary to the analytical redundancy[3], we require that the system contains redundant sensors or actuators, subsystems with similar functions, or alternative control loops. Practical applications of reconfiguration strategies can be found e.g. in safety critical flight control systems (Bajpani et al., 2001). Izadi-Zamanabadi (1999) uses a Finite State Machine to structure and guide the action to be taken as reconfiguration. After having gained informations about fault propagation, remedies are determined and governed by the supervisory logic. While this approach requires detailed knowledge of the fault effects, a more automated framework is provided by model predictive control, which is advocated as a promising tool for fault tolerant controllers (Maciejowski and Lemos, 2001). Considering classical linear MPC evidences the main reasons:

- The internal model can be updated to represent the faulty system behaviour. For instance, the constraints can be modified to reflect different operating ranges of system components.
- Control objectives can be adapted to the current capabilities of the plant.

In the context of MLD systems we focus on active fault accommodation. The main con-

[3]Analytical redundancy is a concept in fault detection and isolation, where fault diagnosis is performed by checking the consistency of the measurements with a model output, in order to detect deviations from the nominal behaviour.

tribution of this section is to propose several solutions for the decision problem about the choice of redundant hardware, in particular redundant inputs, using mathematical optimization. We refer to this decision simply as *reconfiguration* of the control scheme.

There are four main reasons for formulating active fault accommodation within the MLD framework.

1. The MLD framework allows to model several classes of practical systems and a large set of fault scenarios can be covered within the same model.
2. Often there are user's preferences concerning the usage of system components. The priorities on redundant components can be directly included in the MLD model and can therefore be systematically considered during computations, see Section 4.6.
3. The formulation of the mathematical optimization problems for solving the reconfiguration problem can be tackled by solvers, which are commercially available, see Chapter 5.
4. The decision about the usage of redundant inputs is an inherently binary problem, therefore the modelling with binary variables is appropriate and does not alter the structure, which is already present in MLD models.

4.5.3 Reconfiguration of the Three Tank System

The three tank system described in Section 3.2 has been used to illustrate the proposed ideas. The reconfiguration problem for the three tank system has been defined as a COSY benchmark problem, see (Lunze, 1998; Heiming and Lunze, 1999) and (COSY). In presence of the faults defined in Section 3.2.6, the goal of the reconfiguration algorithms is:

1. To keep the level h_3 around 0.1 m, in order to provide a constant outflow from tank 3.

2. To minimize the liquid loss from tank 1, if a leak occurs in tank 1.

In the latter case, tank 1 should ideally be emptied as fast as possible and not used anymore. There are some aspects that complicate the reconfiguration for this system:

- There are several possibilities for the choice of the redundant hardware. To check and evaluate all combinations represents a large combinatorial problem.
- The choice of redundant hardware has to be made fulfilling a hierarchy of preferences. On one hand, the use of some actuator is preferred over the use of some other actuator with similar functions. For instance valve V_1 is preferred over valve V_{13}, even though both are interconnecting the same two tanks. On the other hand, we want to involve as little equipment as possible. For instance it is desirable to avoid using tank 2 if the control goal can be reached with tanks 1 and 3 only.
- The plant exhibits typical characteristics of a hybrid system. In fact, the dynamics change, if the liquid reaches the height of the upper valves. Moreover, the model has to take into account discrete inputs because of the presence of switching valves.
- The control goal in nominal operation can only be reached by providing a sequence of periodic inputs to the system. For some reference values no constant steady state exists, if pump Q_1 and valve V_1 are used, see Section 2.8.
- The system is subject to constraints on the liquid levels $h_i \quad (i = 1, 2, 3)$ and on the inflows $Q_j \quad (j = 1, 2)$.

Previous works on the three tank system are reported in (Askari et al., 1999) and (Lunze and Schröder, 1999), where schemes that only take into account qualitative information about the state of the plant are considered. In (Rato and Lemos, 1999) the reconfiguration is formulated as a multi-model switching control task. Approaches using neural networks are reported in (Marcu et al., 1999).

The supervision scheme we are considering consists of two stages. The conceptual scheme is shown in Figure 4.18.

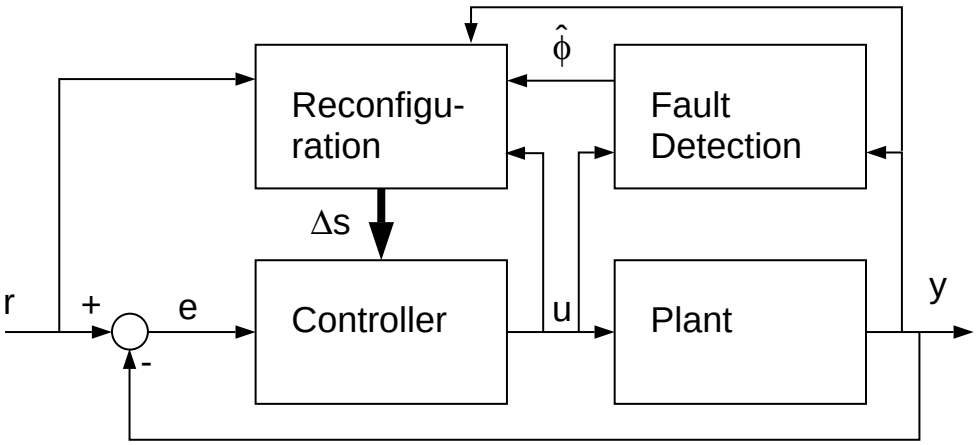


Figure 4.18: Reconfiguration scheme

In a first stage a fault detection and isolation algorithm is required to monitor the operation of the plant. This part should estimate at each time step which faults occurred or that no fault was detected. We assume that this stage is already implemented, for instance by using the tools described in Section 4.4. The second stage is the actual reconfiguration device, which decides on the system modification that is most suitable in order to achieve the control goals, or the best performance in terms of some value function. The reconfiguration block takes as inputs the reference r, the control action u, the output signal y and the fault information $\hat{\phi}$. The output is a modification of the controller structure Δs, which can be the inclusion or exclusion of some actuators or sensors or a switch to a different controller. We require that in absence of faults no reconfiguration occurs, i.e.

$$\left[\hat{\phi} = 0\right] \;\Rightarrow\; [\Delta s = 0] \tag{4.29}$$

4.5.4 Reconfiguration of MLD Systems

We present various strategies for performing the decision on redundant hardware in case of faults. We characterize the manipulated variables of an MLD system as follows:

The set V_N (Nominal Inputs) is the set of all manipulated variables used in nominal operation.

The set V_R (Redundant Inputs) is the set of all other manipulated variables, i.e., those inputs that can be used additionally for reconfiguration.

The sets V_i (Candidate Inputs) $(i = 1, \ldots, n_f)$ are the sets of manipulated variables that are proposed to be used by the reconfiguration scheme when fault ϕ_i is detected. n_f is the number of fault scenarios considered.

We assume that all manipulated variables have been partitioned into the sets V_N and V_R and that the following statements hold.

$$V_R \neq \emptyset \tag{4.30}$$

$$V_R \cap V_N = \emptyset \tag{4.31}$$

$$V_i \subset (V_R \cup V_N) \quad (i = 1, \ldots, n_f) \tag{4.32}$$

4.5.5 Reconfiguration of MLD Systems as a Control Problem

One way to handle the reconfiguration problem is to treat it as a control problem within a receding horizon framework as described in (4.1)-(4.4). As soon as a fault has been detected, the reconfiguration block modifies the model of our system, such that it henceforth describes the system behaviour in presence of the fault. Typically the occurrence of a fault involves the failure of actuators, sensors or subsystems. We assume that such a fault can be modelled as a set of constraints or additional logic propositions for the MLD model.

For MLD models with faults, we can extend the Model Predictive Control algorithm (4.1)-(4.4) and the steady state finding procedure (2.75)-(2.78) by including fault estimates $\hat{\phi}$. We treat the faults $\hat{\phi}$ as exogenous variables. Instead of (4.2), we get for $t = 0 \dots T-1$

$$x(t+1, x_0, u_0^t) = Ax(t, x_0, u_0^t) + B_1 u(t) + B_2 \delta(t, x_0, u_0^t) + B_3 z(t, x_0, u_0^t) + B_6 \hat{\phi}(t) \quad (4.33)$$

Instead of (4.4), we get for $t = 0 \dots T$ the constraints:

$$E_2 \delta(t, x_0, u_0^t) + E_3 z(t, x_0, u_0^t) \leq E_1 u(t) + E_4 x(t, x_0, u_0^t) + E_5 + E_6 \hat{\phi}(t) \quad (4.34)$$

Similarly, the optimization (2.75) - (2.78) becomes for faulty MLD systems:

$$\min_{x_f, u_f, \delta_f, z_f} ||y_f - r|| + ||x_f||_{\rho_4} + ||u_f||_{\rho_1} + ||z_f||_{\rho_3} + ||\delta_f||_{\rho_2} \quad (4.35)$$

s.t.

$$\begin{aligned} x_f &= Ax_f + B_1 u_f + B_2 \delta_f + B_3 z_f + B_6 \hat{\phi} & (4.36) \\ E_2 \delta_f + E_3 z_f &\leq E_1 u_f + E_4 x_f + E_5 + E_6 \hat{\phi} & (4.37) \end{aligned}$$

Once a fault has been detected, we carry on solving for the MPC control law with the model that includes the corresponding fault information. In faultless mode we impose the usage of the set V_N of manipulated variables by adding prioritizations in the control synthesis optimization, as mentioned in Section 4.6. This idea is summarized in the following reconfiguration algorithm.

Algorithm 4.1

1. *Consider the complete model (3.37) of the plant, including the complete set of manipulated variables $V_N \cup V_R$.*

2. *Run the fault detection scheme and the control scheme in parallel, as shown in Figure 4.18.*

3. *While $\phi = 0$, apply the MPC control algorithm in Section 4.1.2 to the complete model. To avoid the usage of the inputs V_R in nominal operation include prioritizations on the inputs V_N.*

4. *If a fault ϕ_i is detected, update $\hat{\phi}(t)$ and use it in the MPC optimization for all future time steps where the fault is active.*

In step 4 the prioritizations on V_N might have to be modified or dropped, especially if they are formulated as hard constraints. Multiple occurrences of faults can be handled as well by setting ϕ_i accordingly in step 4. The advantage of this algorithm is its optimality in terms of the cost function (4.1) for the controller. The drawback is the large size of the optimizations, done at each time instant over the complete model. The large number of variables is a difficulty that is particularly pronounced in the mixed integer optimizations of MLD systems.

We can prove stability of this reconfiguration scheme as in (Bemporad and Morari, 1999a, Theorem 1), since it is a direct application of the control scheme presented there.

Theorem 4.1 *Let (x_f, u_f) be an equilibrium pair and (δ_f, z_f) definitely admissible in the sense of (Bemporad and Morari, 1999a). Let $(x_f, u_f, \delta_f, z_f)$ be unique and reachable. Assume further that a constant fault ϕ_f is present and that the initial state $x(0)$ is such that a feasible solution of problem (4.1),(4.33),(4.3),(4.34) exists at time $t = 0$.*

Then $\forall Q_1 = Q_1^T > 0$, $Q_2 = Q_2^T \geq 0$, $Q_3 = Q_3^T \geq 0$, $Q_4 = Q_4^T > 0$, and $Q_5 = Q_5^T \geq 0$ the control law (4.1),(4.33),(4.3),(4.34) stabilizes the system in that

$$\begin{aligned}
\lim_{t\to\infty} x(t) &= x_f \\
\lim_{t\to\infty} u(t) &= u_f \\
\lim_{t\to\infty} \|\delta(t) - \delta_f\|_{Q_2} &= 0 \\
\lim_{t\to\infty} \|z(t) - z_f\|_{Q_3} &= 0 \\
\lim_{t\to\infty} \|y(t) - y_f\|_{Q_5} &= 0
\end{aligned}$$

Proof. The MLD system with faults can be rewritten as an MLD system without faults by redefining the vector of auxiliary binary variables as:

$$\tilde{\delta} = \begin{bmatrix} \delta \\ \phi_f \end{bmatrix} \tag{4.38}$$

Since the fault ϕ_f is assumed to be constant, its value can be formulated as additional, time invariant constraints in (4.34). Stability then follows from stability of MPC for MLD systems. □

4.5.6 Reconfiguration of MLD Systems on Two Decision Levels

In order to avoid the large size of the online optimization problems involved in receding horizon control for reconfiguration purposes, we propose three methods, on how to lower the computational complexity of (4.1),(4.33),(4.3),(4.34). In this context we measure the complexity by the number of manipulated variables used in the optimization. We assume that the receding horizon control scheme is operating with the inputs V_N, as long as no fault is detected. If a fault ϕ_i occurs, the first task of the reconfiguration procedure is to decide, which inputs V_i have to be used for control. Note that for Algorithm 4.1 this question did not arise, since the choice of manipulated variables was implicitly given by the optimization over all available manipulated variables in $V_N \cup V_R$. The second task consists of *applying the receding horizon control strategy using the manipulated variables V_i determined in the first task*. The three methods described in this section differ from each other in the first decision level, i.e., in the decision about V_i.

Reconfiguration with Look Up Table

In this strategy we propose to determine the manipulated variables V_i offline. A look up table is set up that denotes the set V_i for each possible fault scenario ϕ_i. The following algorithm describes the procedure:

Algorithm 4.2

1. *Build up a look up table with the assignment of a set V_i to each possible fault scenario ϕ_i.*

2. *Run the fault detection scheme and the control scheme in parallel, as shown in Figure 4.18.*

3. *For each fault scenario ϕ_i detected, switch the model used in control according to the look up table, i.e., determine the candidate inputs V_i from the look up table.*

4. *Apply the MPC algorithm using the candidate inputs V_i as manipulated variables.*

5. *Go to step 3. and use the most recent fault information.*

The lookup table with V_i can be found offline via physical considerations or reachability analysis.

Choice by physical motivation The look up table is found by analyzing all possible fault scenarios. The sets of candidate inputs V_i are found by gaining physical insight into the plant, by considering experience and a priori plant knowledge.

Choice by reachability analysis The look up table is found using tools from verification (Bemporad and Morari, 1999b) and observability/reachability analysis (Bemporad, Ferrari-Trecate and Morari, 1999). A given set of inputs is considered as candidate input V_i, if it allows to reach the desired reference value from a set of initial states, despite the influence of ϕ_i.

Reconfiguration with Static Steady State Analysis

We assume that the system has a constant control target that can be reached without requiring cycling states (see Section 2.8), and that the steady state obtained by (4.35) to (4.37) is reachable. If a fault ϕ_i is detected, the candidate inputs V_i are chosen based on the steady state optimization (4.35)-(4.37). V_i is chosen as the set of inputs that are different from zero in this optimization. The search for the steady state can be modified to include further constraints or prioritizations. This method requires a low computational load, however it lacks guarantees of optimality for the choice of V_i in the transients. The algorithm can be summarized as follows:

Algorithm 4.3

1. *Run the fault detection scheme and the control scheme in parallel, as shown in Figure 4.18.*
2. *If a fault ϕ_i is detected, run the steady-state finding procedure (4.35)-(4.37) on the model with inputs $V_N \cup V_R$.*
3. *Choose those inputs, which are different from zero in steady state, as candidate inputs V_i.*
4. *Apply the MPC algorithm using the candidate inputs V_i as manipulated variables.*
5. *Repeat step 2. at the next time step using the most recent fault information.*

Reconfiguration with Polytopic Steady State Analysis

The algorithm presented here is a modification of the method shown previously. Its motivation is given by the possibility that a system does not exhibit one single steady state, but

rather tracks a cycle of states in order to stay close to the reference. We assume that the cycle consists of reachable states. While in the Reconfiguration with Static Steady State we extracted the candidate inputs V_i from a single steady state, for systems with cycling states we propose to extract V_i from multiple steady states. For a system with n_y outputs we solve $M = 2^{n_y}$ optimizations, given by (4.35)-(4.37), subject to exactly one additional constraint:

$$\mu_k y_f \;\leq\; \mu_k r \quad (k \in 1 \dots 2^{n_y}) \tag{4.39}$$

where

$$\mu_k \in \left\{ diag(d) \mid d \in \left\{ [\xi_1, \dots, \xi_{n_y}] \mid \xi_i \in \{-1, 1\} \right\} \right\} \tag{4.40}$$

Example 4.1:

Consider for instance a system with $n_y = 2$ outputs. The $M = 2^2 = 4$ steady states are found by solving (4.35)-(4.37) subject to one of the following four additional constraints, respectively:

$$k = 1: \quad \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} y_f \;\leq\; \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} r \tag{4.41}$$

$$k = 2: \quad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} y_f \;\leq\; \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} r \tag{4.42}$$

$$k = 3: \quad \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} y_f \;\leq\; \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} r \tag{4.43}$$

$$k = 4: \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} y_f \;\leq\; \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} r \tag{4.44}$$

□

Let Y_i be the set of outputs generated by the M steady states computed above. We assume that

$$r \in \text{conv}(\{Y_i | i = 1 \dots M\}) \tag{4.45}$$

i.e., the desired reference value is element of the convex hull of the M steady state outputs. If (4.45) is fulfilled, we define M tentative sets of inputs $V_{ik} \quad (k = 1 \dots M, i = 1 \dots n_f)$ as those inputs that are different from zero at steady state. The candidate set of inputs V_i is then chosen such that $V_{ik} \subset V_i \quad (k = 1 \dots M, i = 1 \dots n_f)$. The set of inputs V_i allows to keep the system at each of the M steady state values. Therefore, if r is within the polytope spanned by Y_i, it is reasonable to use these inputs to achieve the actual goal of staying close to the reference r.

Example 4.2:

We visualize graphically the $M = 2^2$ steady states for the previous example in Figure 4.19. Assuming that the axes are normalized to have the desired reference $r = [r_1 \quad r_2]^T$ at the origin, the equilibria Y_i $(i = 1 \dots 4)$ fulfill the conditions denoted in the corresponding quadrants. The shaded area marks the convex hull of Y_i $(i = 1 \dots 4)$ and the reference r fulfills (4.45). □

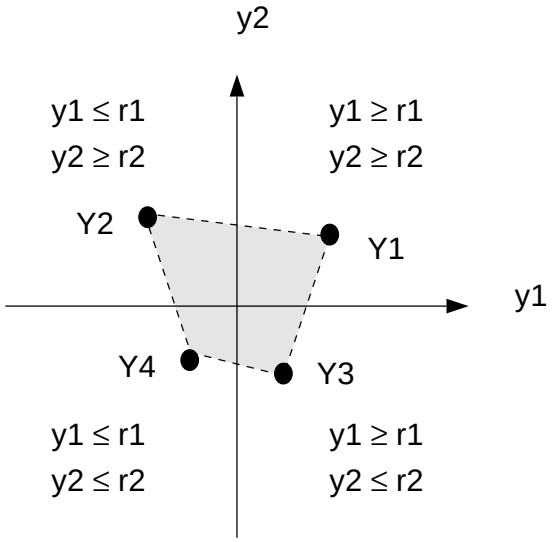


Figure 4.19: Outputs Y_i of steady states: The origin is the desired reference value $r = [r_1 \quad r_2]^T$, the outputs Y_1, Y_2, Y_3, Y_4 result as steady state outputs from the optimization (4.35)-(4.37), (4.39). The origin r is an element of the convex hull of Y_1, Y_2, Y_3, Y_4

The reconfiguration algorithm is summarized next:

Algorithm 4.4

1. *Run the fault detection scheme and the control scheme in parallel, as shown in Figure 4.18.*

2. *If the fault ϕ_i is detected, find M steady states with property (4.45). The steady states are found with (4.35)-(4.37) on the model with inputs $V_N \cup V_R$. For each steady state define V_{ik} as the set of manipulated variables that are nonzero in steady state.*

3. *Find the set of candidate inputs V_i from V_{ik}.*

4. *Apply the MPC algorithm using the candidate inputs V_i as manipulated variables.*

5. *Repeat step 2. at the next time step using the most recent fault information.*

Some remarks about this approach are discussed next.

Finding a common set of inputs V_i from V_{ik}: An obvious choice for V_i is given by

$$V_i = \cup_{k=1}^{M} V_{ik}$$

However, in terms of the number of manipulated variables, this can be suboptimal. We defer to (Tsuda et al., 2000) for more details, where the aggregation of V_{ik} ($k = 1 \dots M$) is considered. The idea there is that the set of manipulated variables V_{ik} is in general not unique[4]. This nonuniqueness can be exploited to find V_i, such that its cardinality is as small as possible.

[4]The complete set of admissible manipulated variables is obtained by solving the feasibility test (4.36)-(4.37),(4.39) rather than the optimization problem (4.35)-(4.37),(4.39).

Number of vertices M**:** The method requires to find $M = 2^{n_y}$ steady states for each possible fault. Therefore it has to be limited to systems with a low number of outputs.

Collapsing steady states: Solving for the $M = 2^{n_y}$ steady states, it can occur that some steady state fulfills several sets of constraints (4.36)-(4.37),(4.39). This is the case, if one or more components of the reference are reached exactly by some steady state value, see Figure 4.20. For these cases the number of different steady states drops below 2^{n_y}.

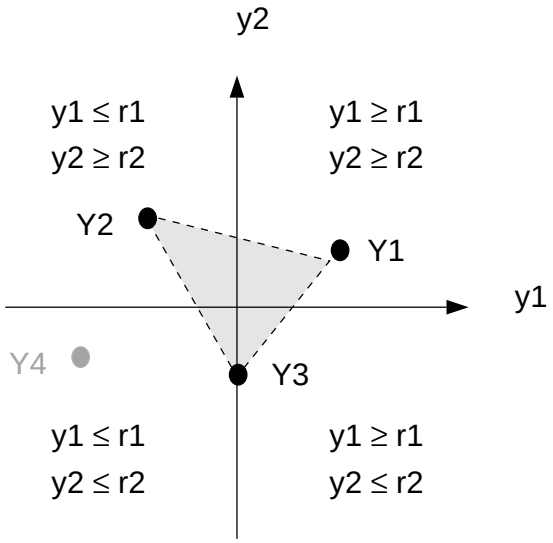


Figure 4.20: Collapsing of steady states: When solving for Y_3 we see that Y_3 satisfies also the constraints for Y_4. Therefore, Y_4 can be neglected. The origin r is an element of the convex hull of Y_1, Y_2, Y_3

Infeasibilities: The additional constraints in the steady state finding procedure can lead to infeasibilities. In this case the set of steady states Y_i contains $M' < 2^{n_f}$ elements. This is not a limitation to the method, as long as (4.45) is satisfied.

In Section 4.5.9 we apply this method to the three tank system.

4.5.7 One Step Compensation

In the previous section, we first determined the manipulated variables V_i to be used and then ran the MPC controller with those variables. In this approach we set $V_i = V_N \cup V_R$, i.e. we use the complete set of manipulated variables and we propose a strategy to compute the values of V_i in case of faults, avoiding to run the MPC optimization over all manipulated variables. We use the redundant inputs V_R to minimize at each time step the effect of faults, i.e. we propose a scheme for fault compensation. The MLD model is rewritten as:

$$x(t+1) = Ax(t) + B_1 u(t) + B_2 \delta(t) + B_3 z(t) + \bar{B}_1 \bar{u}(t) + B_6 \hat{\phi}(t) \tag{4.46a}$$

$$y(t) = Cx(t) + D_1 u(t) + D_2 \delta(t) + D_3 z(t) + \bar{D}_1 \bar{u}(t) + D_6 \hat{\phi}(t) \tag{4.46b}$$

$$E_2 \delta(t) + E_3 z(t) \leq E_1 u(t) + E_4 x(t) + E_5 + \bar{E}_1 \bar{u}(t) + E_6 \hat{\phi}(t) \tag{4.46c}$$

Here we explicitly split up the inputs used in nominal operation and the inputs for reconfiguration purposes as $u \in V_N$ and $\bar{u} \in V_R$, respectively. In nominal operation, we have $\hat{\phi} = 0$ and $\bar{u} = 0$. Running the MPC optimization we obtain at each time step t a triple $(u_n(t), \delta_n(t), z_n(t))$. The state evolution in nominal operation is then given by

$$x(t+1) = Ax(t) + s_n(t) \tag{4.47}$$

where $s_n(t)$ is determined online by the MPC controller and

$$s_n(t) = B_1 u_n(t) + B_2 \delta_n(t) + B_3 z_n(t) \tag{4.48}$$

Note that at each t, $s_n(t)$ is determined by the MPC algorithm on the nominal model, and is therefore known. In faulty operation the state update is different from $s_n(t)$. In fact in presence of a fault, we use the reconfiguration inputs V_R as well. The state evolution is then given by:

$$x(t+1) = Ax(t) + s_e(t) \tag{4.49}$$

The total state update can therefore be expressed as:

$$s_e(t) = B_1 u_e(t) + B_2 \delta_e(t) + B_3 z_e(t) + B_6 \hat{\phi}(t) + \bar{B}_1 \bar{u}_e(t) \tag{4.50}$$

The key idea of this method is to make $s_e(t)$ as close to $s_n(t)$ as possible by exploiting the freedom we have to choose the redundant inputs $\bar{u}(t)$. This specification can be translated into the following optimization problem:

$$J_s = \min_{u_e, \bar{u}_e, \delta_e, z_e} \parallel s_n(t) - s_e(t) \parallel_{Q_{os}} \tag{4.51}$$

$$\text{s.t. } E_2\delta(t) + E_3 z(t) \leq E_1 u(t) + E_4 x(t) + E_5 + \bar{E}_1 \bar{u}(t) + E_6 \hat{\phi}(t) \tag{4.52}$$

In other words, when a fault is detected, we reproduce as best as possible the state update of the system, as it would be in nominal operation. The algorithm is described as follows.

Algorithm 4.5

1. *Run the fault detection scheme and the control scheme in parallel, as shown in Figure 4.18.*
2. *Compute the MPC control moves with the faultless model, yielding* $u_n(t), \delta_n(t), z_n(t)$ *and determine* $s_n(t)$ *from equation (4.48).*
3. *If a fault* ϕ_i *is detected, run the optimization (4.51), (4.52) to obtain* $u_e(t)$, $\bar{u}_e(t)$, $\delta_e(t)$, $z_e(t)$. *If no fault is detected, set* $u_e(t) = u_n(t), \bar{u}_e(t) = 0, \delta_e(t) = \delta_n(t), z_e(t) = z_n(t)$.
4. *Apply* $u_e(t)$ *and* $\bar{u}_e(t)$ *to the system*
5. *Repeat 1. at the next time step.*

A couple of remarks about this algorithm are given next.

- Online we solve two optimization problems. The first one is the standard MPC optimization with faultless model and manipulated variables V_N in order to determine $s_n(t)$. The second one is (4.51)-(4.52) and represents a correction step, which yields

the actual inputs to apply to the plant, i.e. $u_e(t)$ and $\bar{u}_e(t)$. The computational advantage over the complete optimization mentioned in Section 4.5.5 is that the full model is only used for an optimization over one step (4.51)-(4.52), instead of the optimization over the complete MPC horizon.

- Note that in general $\delta_n(t) \neq \delta_e(t)$ as well as $z_n(t) \neq z_e(t)$. This is due to the influence of the faults ϕ on the set of feasible points for the auxiliary variables δ and z.

- The partition of the inputs in V_N and V_R is done, such that in faultless operation the reconfiguration inputs V_R are never used, i.e.

$$\left[\hat{\phi} = 0\right] \;\Rightarrow\; [\bar{u}_n = 0] \tag{4.53}$$

which explicitly satisfies requirement (4.29). Since the triple $(u_n(t), \delta_n(t), z_n(t))$ satisfies the constraints in (4.52), we note that the optimum is $J_s = 0$, which can be reached by choosing

$$(u_e(t), \delta_e(t), z_e(t)) = (u_n(t), \delta_n(t), z_n(t)) \tag{4.54}$$

Therefore we see that in faultless operation the reconfiguration procedure (4.51)-(4.52) is bypassed, in the sense that the nominal control action is not modified. This justifies the action in the fault free case of step 3 in Algorithm 4.5.

- One Step Compensation is not necessarily linked to MPC. Other control algorithms like e.g. output feedback can be used as control schemes. To determine the quantity (4.48) necessary for One Step Compensation a simulation step is required to determine $\delta_n(t)$ and $z_n(t)$.

- The main decision variables are the weighting coefficients of Q_{os} in the norm of Equation (4.51). They determine the relative importance of tracking the individual state components.

The correction step done at each single time instant can be a drawback of the method, since in presence of a fault, it might not be the best strategy to try to follow the nominal

state update as close as possible. The method is however fast, and in Section 4.5.9 we show that it can provide the correct remedy to fault scenarios of the three tank system.

4.5.8 Application to the Three Tank System: Simulation

In this section we simulate the system behaviour for an occurrence of fault ϕ_2, i.e. valve V_1 is blocked closed. We apply the One Step Compensation strategy in Section 4.5.7 and compare it to the MPC controller using the complete model of Section 4.5.5. The fault information is assumed to be exact, i.e. the fault is not estimated, but its type and time of occurrence is known exactly. The fault is assumed to occur at $t = 150$, which would result in the faulty system trajectories of Figure 4.21.

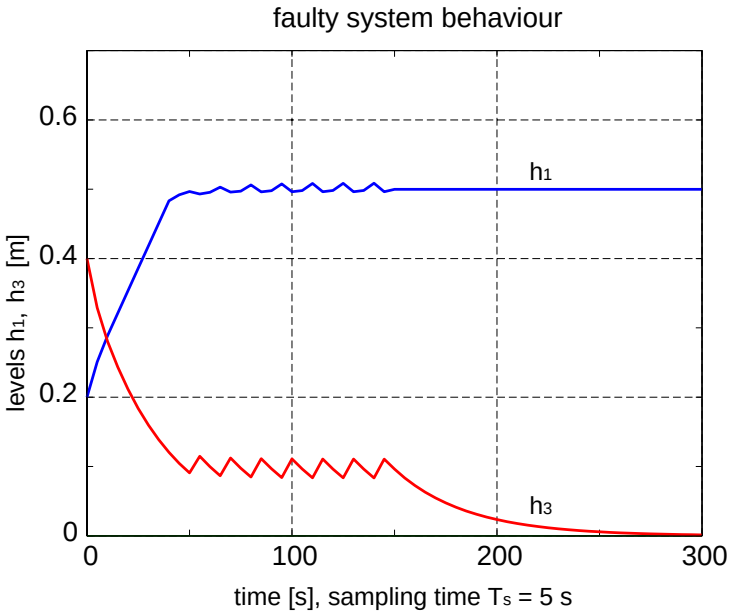


Figure 4.21: Faulty system behaviour for fault ϕ_2: The model parameters correspond to those of the original benchmark system with increased cross sectional areas of the pipes

In Figure 4.22 we use MPC with a reduced model of the system[5] before the fault ϕ_2 occurs

[5]including only Q_1 and V_1 as manipulated variables

and the reconfiguration block switches to the complete model, as soon as the fault occurs. The scheme decides to isolate tank 1, i.e. not use the lower valve V_{13}. Instead it uses tank 2 to control level h_3. The oscillations of h_3 disappear, since level h_2 is controlled at 0.2 m. The strategy obtained with One Step Compensation is different, as it is shown in Figure 4.23. The scheme decides to use valve V_{13}. Tank 2 is used to buffer the increased amount of liquid flowing from tank 1 to tank 3 due to the usage of V_{13} instead of V_1. Pump 2 is not used and the oscillations in tank 3 remain, as in nominal behaviour.

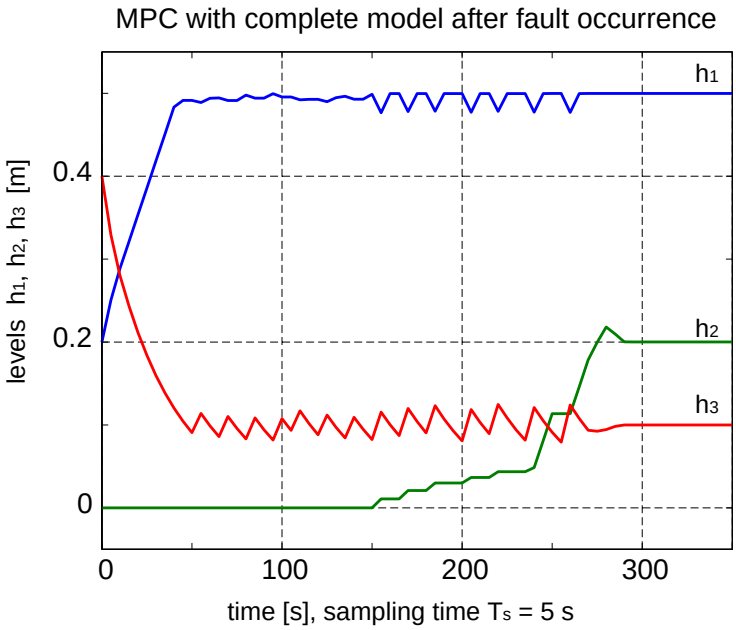


Figure 4.22: Reconfiguration with complete model in MPC after fault occurrence at $t = 150$

4.5.9 Application to the Three Tank System: Experiments

We show an application of the reconfiguration based on polytopic steady state analysis and on One Step Compensation on the laboratory model in Figure 3.5. For all experiments the assumption is that the fault information is exact, i.e. the fault is not estimated, but its type and time of occurrence is assumed to be known exactly.

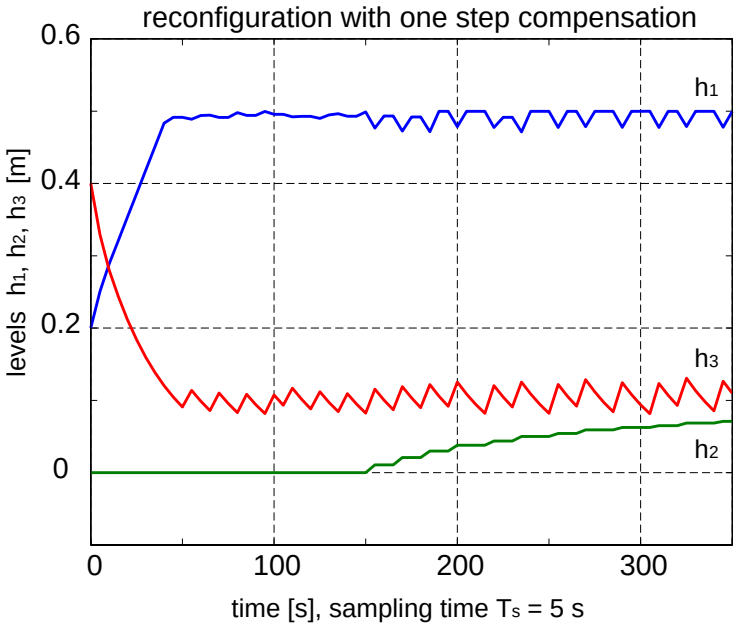


Figure 4.23: Reconfiguration with one step compensation after fault occurrence at $t = 150$

Polytopic Steady State Analysis

The three tank system exhibits cycling trajectories when it is controlled to the reference $[0.5, 0, 0.1]$. Therefore, we don't apply static steady state reconfiguration, but rather extract the information about the inputs to be used from the cycle of states, the system goes through. We solved the steady state finding procedure (4.35)-(4.37) four times, adding in each case one of the constraints shown in Table 4.8.

It is easy to verify that the reference $r_1 = 0.5$ and $r_3 = 0.1$ lies in the convex hull of the steady states in Table 4.8. Considering the third column of Table 4.8 we find that the manipulated variables to be used are those different from zero in the steady states, i.e. $[Q_1, Q_2, V_1, V_{23}]$. Repeating the analysis for ϕ_2 and ϕ_3 gives the suggested manipulated variables in Table 4.9.

We implemented the MPC scheme on the three tank system, where in faultless mode we use the nominal manipulated variables $\{Q_1, V_1\}$. The controller switches model and uses

Additional constraints to (4.35)-(4.37)	Resulting steady state x_f	Manipulated variables $[Q_1, Q_2, V_{13}, V_{23}, V_1, V_2]$ in steady state
$y_{1f} \geq r_1, \quad y_{3f} \geq r_3$	$[0.5, 0.2, 0.1]$	$[0.27 \cdot 10^{-4}, 0.13 \cdot 10^{-4}, \ 0, 1, 0, 0]$
$y_{1f} \geq r_1, \quad y_{3f} \leq r_3$	$[0.62, 0.3, 0.1]$	$[0.42 \cdot 10^{-4}, 0.01 \cdot 10^{-4}, \ 0, 0, 1, 0]$
$y_{1f} \leq r_1, \quad y_{3f} \geq r_3$	$[0.34, 0.21, 0.1]$	$[0.20 \cdot 10^{-4}, 0.13 \cdot 10^{-4}, \ 0, 1, 0, 0]$
$y_{1f} \leq r_1, \quad y_{3f} \leq r_3$	$[0.49, 0.2, 0.1]$	$[0.27 \cdot 10^{-4}, 0.13 \cdot 10^{-4}, \ 0, 1, 0, 0]$

Table 4.8: Steady states in presence of ϕ_1, where $y_f = [y_{1f}, y_{2f}, y_{3f}]$, $r_1 = 0.5$, $r_3 = 0.1$

Fault	Manipulated variables to be used
ϕ_1	$[Q_1, Q_2, V_1, V_{23}]$
ϕ_2	$[Q_2, V_{23}]$
ϕ_3	$[Q_1, Q_2, V_{23}]$

Table 4.9: Look up table: Manipulated variables to be used in case of faults, suggested by polytopic steady state analysis

the variables in Table 4.9, when a fault occurs. Since the scheme suggests to use tank 2, we introduce a reference value $r_2 = 0.2$ when the model is switched. The experiments for the three fault scenarios are shown in Figures 4.24 - 4.26.

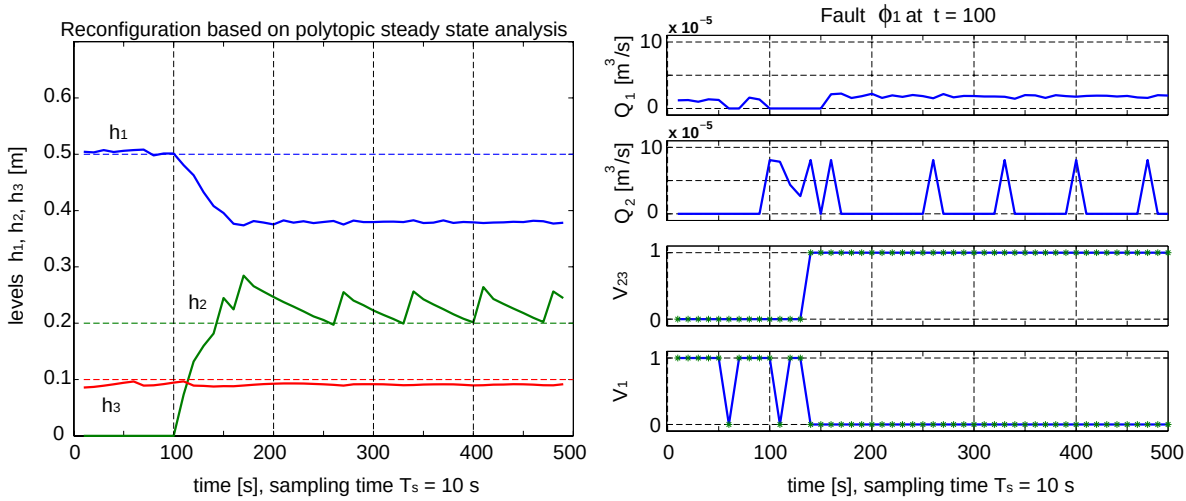


Figure 4.24: Reconfiguration based on polytopic steady state analysis of Table 4.9, Fault ϕ_1 at time step $t = 100$: State trajectories (left), reconfiguration inputs Q_1, Q_2, V_{23}, V_1 (right)

If valve V_1 is blocked closed (fault ϕ_2, Figure 4.25), the system decides to isolate tank 1 and uses tank 2 to control h_3. If valve V_1 is blocked open (fault ϕ_3), it is not possible to keep both the reference for h_1 and h_3. Indeed, if V_1 is blocked open, keeping h_1 at $r_1 = 0.5$ causes a large amount of liquid flowing in tank 3 and $h_3 \geq r_3$. On the other hand, to keep r_3 at h_3 requires that the amount of liquid in tank 1 is about at $h_1 = 0.4$, i.e. lower than its reference. Both behaviours can be obtained with the manipulated variables given in Table 4.9 by appropriate choice of the weights in the MPC cost function. In Figure 4.26 we show an experiment, where the scheme decides to lower the target value for h_1. For fault ϕ_1, the suggested manipulated variables result in a state trajectory that does not minimize the liquid loss in tank 1, as it is shown in Figure 4.24. If we impose the usage of tank 2 only, using the manipulated variables Q_2, V_2, V_{23}, we obtain the desired draining of tank 1,

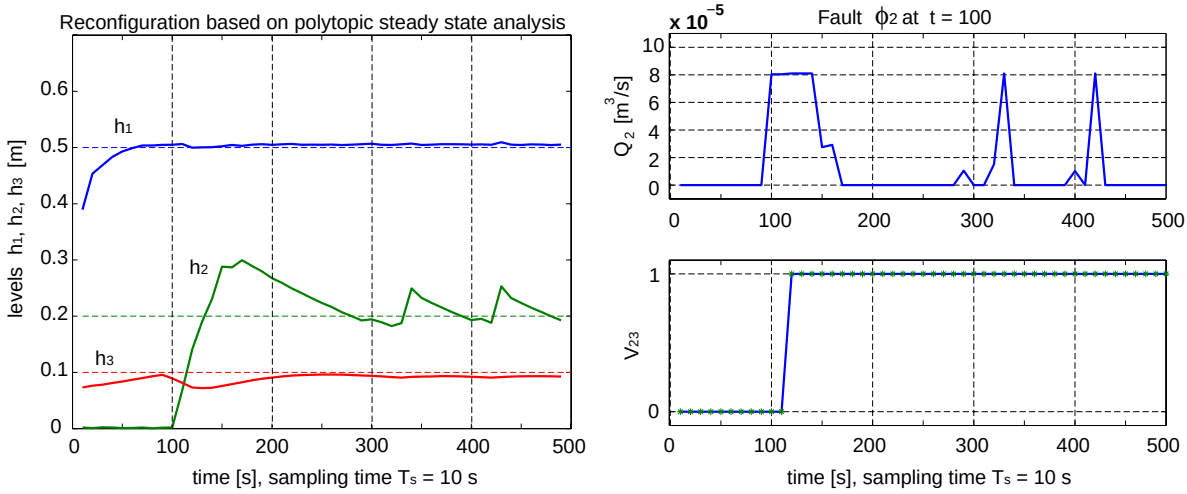


Figure 4.25: Reconfiguration based on polytopic steady state analysis of Table 4.9, Fault ϕ_2 at time step $t = 100$: State trajectories (left), reconfiguration inputs Q_2, V_{23} (right)

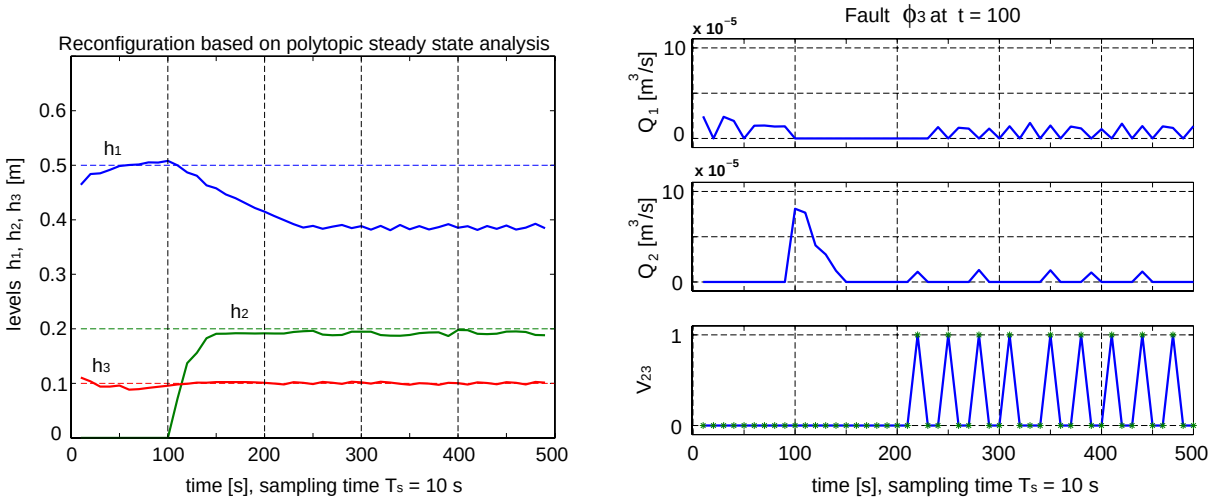


Figure 4.26: Reconfiguration based on polytopic steady state analysis of Table 4.9, Fault ϕ_3 at time step $t = 100$: State trajectories (left), reconfiguration inputs Q_1, Q_2, V_{23} (right)

see Figure 4.27. Explicitly setting $r_1 = 0$ in the polytopic steady state finding procedure allows to find the same set of manipulated variables, confirming the choice of Q_2, V_2, V_{23}.

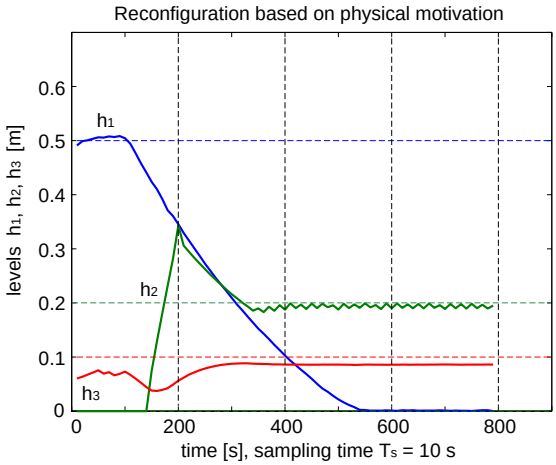


Figure 4.27: Reconfiguration based on physical motivation, Fault ϕ_1 at time step $t = 100$, these system trajectories result imposing the usage of tank 2.

One Step Compensation

In Figure 4.28 we applied one step compensation for the fault scenario ϕ_1 on the three tank system. Despite the fault occurrence at time $t = 100$, the state trajectories do not show any visible effect. Indeed, the One Step Compensation scheme decides to increase the pump action of Q_1 in order to force the output to stay at the reference value of $[0.5, 0, 0.1]$. This has been achieved by increasing the power to pump Q_1, as we show in Figure 4.29.

One Step Compensation automatically decides to use valve V_{13}, if V_1 is blocked closed, i.e. if fault ϕ_2 occurs. The oscillations of h_3 in Figure 4.30 are larger than in nominal behaviour, because using V_{13} the liquid flow from tank 1 to tank 3 cannot be dosed as finely as using V_1. To compensate for this, the scheme decides to allow a liquid flow into tank 2.

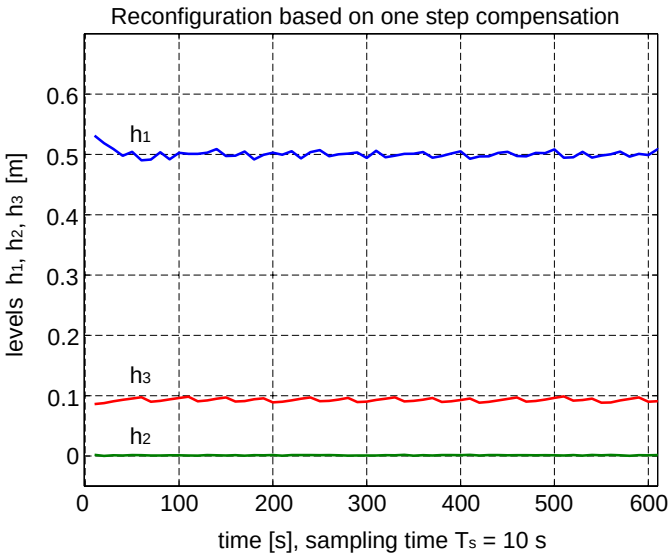


Figure 4.28: Reconfiguration based on one step compensation for fault ϕ_1. The fault occurs at $t = 100$.

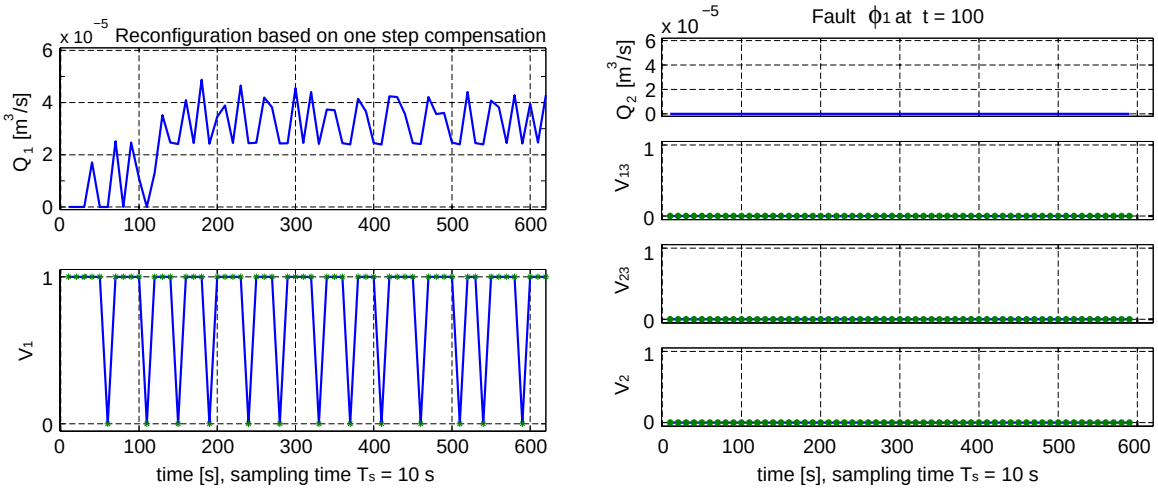


Figure 4.29: Reconfiguration based on one step compensation for fault ϕ_1: Nominal actuator action (left) and reconfiguration actuator action (right). The fault occurs at $t = 100$.

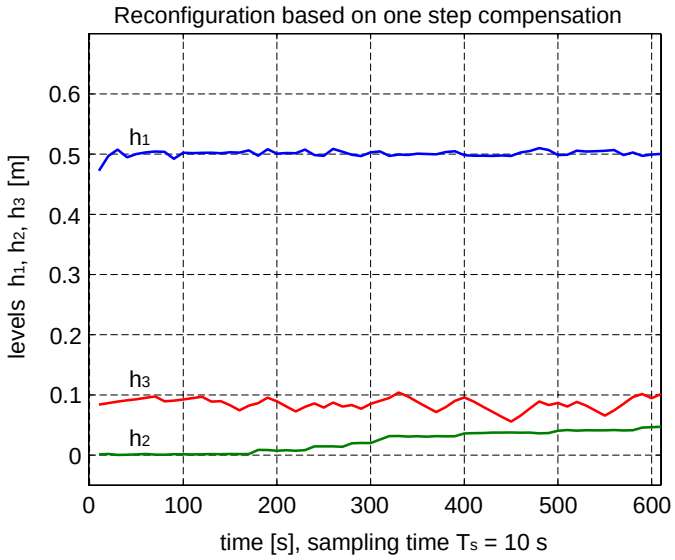


Figure 4.30: Reconfiguration based on one step compensation for fault ϕ_2. The fault occurs at $t = 100$.

In Figure 4.31 we show the case where V_1 is blocked open, i.e. fault ϕ_3 occurs. The scheme decides to use V_{23} to redirect the exceeding amount of liquid to tank 2. However, in this case the scheme is not able to recognize the need to lower the reference for h_1, and ultimately both tanks 2 and 3 settle at a level of around 0.2 m.

In Figure 4.32 we report an experiment, where we modified slightly the set-up of the reconfiguration problem by opening the outflow of tank 2, when ϕ_3 occurred. Note that One Step Compensation performs better than in Figure 4.31 in this case. The opening in tank 2 provides a further possibility for liquid outflow and the scheme correctly decides to let liquid flow out through tank 2, see Figure 4.32.

4.5.10 Summary

Table 4.10 summarizes the main differences between the reconfiguration algorithms. The methods presented in this section aim at providing suggestions on automated system recon-

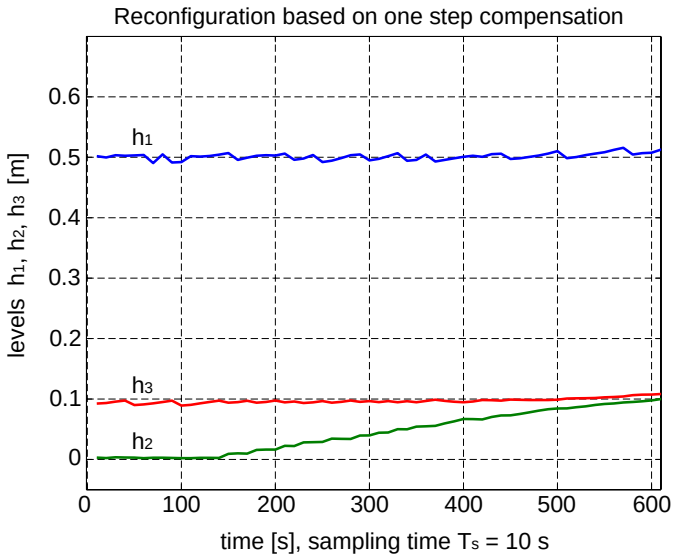


Figure 4.31: Reconfiguration based on one step compensation for fault ϕ_3. The fault occurs at $t = 100$. The trajectories of h_2 and h_3 settle at about 0.2 m.

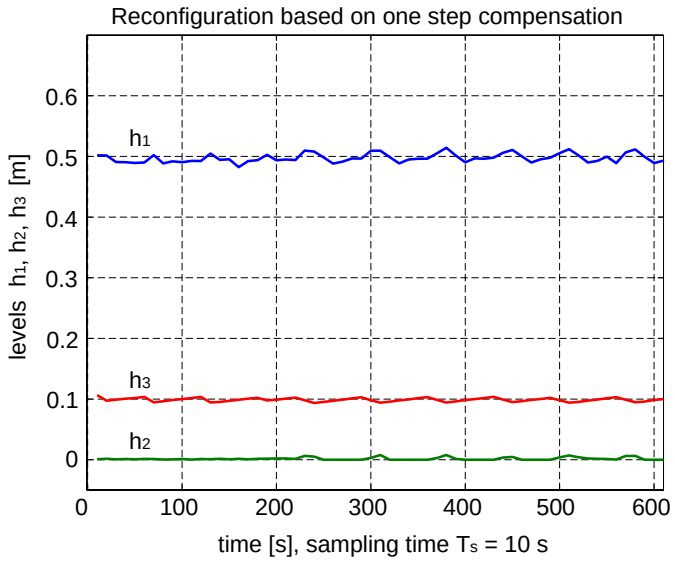


Figure 4.32: Reconfiguration based on one step compensation for fault ϕ_3 and additional opening of tank 2. The fault occurs at $t = 100$.

Algorithm	Choice of candidate inputs V_i in MPC optimization	Value of candidate inputs V_i in MPC optimization given by
4.1 on page 157	$V_i = V_N \cup V_R$	MPC over V_i
4.2 on page 160	look up table	MPC over V_i
4.3 on page 161	static steady state	MPC over V_i
4.4 on page 164	polytopic steady state	MPC over V_i
4.5 on page 167	$V_i = V_N \cup V_R$	correction (4.51)-(4.52)

Table 4.10: Summary of the reconfiguration algorithms

figuration in case of failures. We showed that the procedures are able to produce reasonable suggestions for the three tank system benchmark model, after some tuning of the design parameters. We also pointed out the limitations of the techniques, especially in cases where a thorough system understanding leads to more plausible strategies. At the present stage the methods are meant to be rather a decision support, than an autonomous, intelligent reconfiguration mechanism.

4.6 Constraints Prioritizations

4.6.1 Introduction

So far, all topics in this work involved in one way or another a constrained optimization to solve the desired analysis or synthesis problem. In some cases we would like to assign a different level of importance to the constraints, in the sense that we might be willing to accept their violation, to enlarge the search space of the optimizer. While this is not meaningful for hard physical constraints, like e.g. the maximum volume of containers, some constraints defining performance requirements can be relaxed without any danger for the plant, like e.g. the size of a target region around a reference value. In a reconfiguration

algorithm we would like to avoid using some "expensive", redundant components, unless it is necessary to achieve the specifications. When these components are used, we would like to impose a hierarchy of their usage by prioritizing some of them over the others. In (Bemporad and Morari, 1999a) it is shown that such constraint prioritizations can be directly specified within the MLD framework. We restrict the exposition of these ideas to the controller design problem, even though the framework can in principle be applied to any constrained optimization scheme. This section summarizes the paper by Kerrigan et al. (2000).

4.6.2 Soft Constraints and Their Hierarchy

The key idea is to formulate the hierarchy of priorities as a set soft constraints in the MPC optimization (4.1),(4.2),(4.3),(4.4). This MIQP can be written as an optimization of the following form:

$$\begin{aligned} \min_{\theta} \quad & \theta^T H \theta + f^T \theta \\ s.t. \quad & A\theta \leq b \end{aligned}$$

where $\theta = (\underline{u}, \underline{\delta}, \underline{z})$, H and f are the Hessian and the linear term of the cost function, and all constraints (4.2),(4.3),(4.4) are summarized in the matrix A and vector b. The prioritizations of some hardware components are formulated as soft constraints in the above MIQP, yielding:

$$\begin{aligned} \min_{\theta,\epsilon,\delta} \quad & \theta^T H \theta + f^T \theta + \epsilon^T S \epsilon + \rho M_p^T \delta \\ s.t. \quad & A\theta \leq b + C\epsilon \\ & 0 \leq \epsilon \leq M_\epsilon \end{aligned}$$

where ϵ is a vector of slack variables, representing the constraint violations, C is a matrix of appropriate dimension, whose rows are 0 or e_k^T according to whether the k^{th} constraint is hard or soft, e_k^T is the k^{th} canonical basis vector, $S = S^T > 0$ is a weighting matrix which decides the trade-off between cost and constraint violation.

In order to express the hierarchy, how we allow constraint violations, we define the vector $\delta = [\delta_1, \ldots, \delta_r]^T \in \{0,1\}^r$, associating its components with constraint violations at r levels of priority. We use the logical statements

$$\begin{aligned} [\delta_1 = 0] &\rightarrow [\epsilon_{11} + \ldots + \epsilon_{1c_1} = 0] \\ \vdots &\quad\ \vdots \\ [\delta_r = 0] &\rightarrow [\epsilon_{r1} + \ldots + \epsilon_{rc_r} = 0] \end{aligned} \tag{4.55}$$

where ϵ_{ij} represents the constraint violation of the j^{th} constraint on the i^{th} priority level, c_i is the number of constraints involved in the i^{th} priority level. The constraints on the first level have the highest priority. Constraint violation is indicated by $\delta_i = 1$. The upper bound M_ϵ on the slack variables is necessary, in order for the scalar

$$\rho > \max_{\theta,\epsilon} \theta^T H \theta + f^T \theta + \epsilon^T S \epsilon \tag{4.56}$$

to be well-defined. This choice of ρ implies that any constraint violation costs more than the constraint fulfillment. These logical statements can be converted into mixed integer linear inequalities, see Chapter 2, and included in the constraints of the MIQP.

The key to this approach is the choice of the vector M_p. If the priority is assigned in decreasing order from δ_1 to δ_r, then e.g. the choice

$$M_p = [2^{r-1}, 2^{r-2}, \ldots, 2^1, 2^0]^T \tag{4.57}$$

guarantees that the satisfaction of higher prioritized constraints always results in a greater reduction in cost than the satisfaction of any combination of lower prioritized constraints (Kerrigan et al., 2000). Indeed, it can be verified that for M_p as in (4.57) it holds that

$$M_p(i) > \sum_{j=i+1}^{r} M_p(j) \tag{4.58}$$

where $M_p(i)$ is the i^{th} coefficient of M_p. The definition of ρ in Equation (4.56) requires that

$$\min_i M_p(i) = M_p(r) \geq 1$$

4.6.3 Increasing the Number of Satisfied Constraints

The approach in Equations (4.55) assigning one binary variable for each priority level hides a potential undesired effect. Indeed, if one constraint on a certain priority level cannot be satisfied, the choice (4.55), (4.57) will not guarantee the satisfaction of all other constraints on the *same* priority level. A possible solution that guarantees the satisfaction of a maximum number of constraints, consists of using one binary variable for each constraint on each priority level:

$$\begin{array}{cc} [\delta_{i,j} = 0] & \rightarrow \quad [\epsilon_{ij} = 0] \\ i = 1 \dots r & \qquad j = 1 \dots c_i \end{array}$$

The first index i in $\delta_{i,j}$ denotes the priority level and the second index j the constraint on that priority level. The vector δ now contains $\sum_{i=1}^{r} c_i$ elements as

$$\delta = [\delta_{1,1}, \dots, \delta_{1,c_1}, \delta_{2,1}, \dots, \delta_{i,1}, \dots, \delta_{i,c_i}, \dots, \delta_{r,c_r}]^T$$

The choice of the weights M_p is such that each $\delta_{i,j}$ on the same priority level is assigned the same weight m_i, therefore

$$M_p = [m_1 1_{c_1}, \dots, m_r 1_{c_r}]^T \tag{4.59}$$

where 1_j is a vector of ones of length j. We can normalize Equation (4.59) by setting the weight corresponding to the least priority $m_r = 1$. As in Equation (4.58) we require that the weights at a given priority level be larger than the sum of all weights at lower priority levels. This condition ensures that whenever there is a trade-off between satisfaction of one high prioritized constraint instead of all lower prioritized constraints, the former is chosen to be satisfied. Moreover, if a priority level is violated because one constraint can't be fulfilled, the satisfaction of as many as possible of the others at the same level results in a reduced cost. One possible choice is

$$m_i = 1 + \sum_{j=i+1}^{r} c_j m_j \tag{4.60}$$

Example 4.3:

Assume that we assign the priorities listed in Table 4.11 to the usage of manipulated variables in the three tank system. For $u = [Q_1, V_1, V_{13}, Q_2, V_2, V_{23}]$

manipulated variables	priority level i	number of constraints c_i	weight m_i
Q_1, V_1	1	$c_1 = 2$	$m_1 = 8$
V_{13}	2	$c_2 = 1$	$m_2 = 4$
Q_2, V_2, V_{23}	3	$c_3 = 3$	$m_3 = 1$

Table 4.11: Example: Priorities assigned to the manipulated variables of the three tank system

we can add the soft constraints

$$I_6 u \leq I_6 \epsilon \tag{4.61}$$

where I_6 is the identity matrix of dimension 6. Equation (4.61) would force all inputs to 0, if it were formulated as hard constraint. We allow the violation of this constraint, i.e. the usage of the corresponding manipulated variable, according to the hierarchy of Table 4.11. In this example we have

$$M_p \;=\; \begin{bmatrix} 8 & 8 & 4 & 1 & 1 & 1 \end{bmatrix}^T$$

as well as the logic propositions defining the constraint violations

$$\begin{aligned}
[\delta_{1,1} = 0] &\;\rightarrow\; [\epsilon_{1,1} = 0] \\
[\delta_{1,2} = 0] &\;\rightarrow\; [\epsilon_{1,2} = 0] \\
[\delta_{2,1} = 0] &\;\rightarrow\; [\epsilon_{2,1} = 0] \\
[\delta_{3,1} = 0] &\;\rightarrow\; [\epsilon_{3,1} = 0] \\
[\delta_{3,2} = 0] &\;\rightarrow\; [\epsilon_{3,2} = 0] \\
[\delta_{3,3} = 0] &\;\rightarrow\; [\epsilon_{3,3} = 0]
\end{aligned}$$

□

Extensions to these concepts, concerning e.g. the optimization of the duration of constraint violation can be found in (Kerrigan, 2000).

Chapter 5

Computational Aspects

This chapter is dedicated to the mathematical problems occurring in analysis and synthesis of MLD systems.

5.1 Introduction

In Chapter 4 we have shown, how to cast practical analysis and synthesis problems for MLD systems as mathematical optimization problems. Due to the presence of both integer variables and continuous valued variables, the resulting optimizations are *Mixed Integer Programs*. In this work we focus on the *Mixed Integer Quadratic Program* (MIQP) formulation (Fletcher and Leyffer, 1998; Lazimy, 1985; Volkovich et al., 1987). The choice of a quadratic cost function is motivated by the same reasons as in the continuous case, namely the relatively easy solution, the uniqueness of a solution for positive definite functions and the desire to penalize large deviations stronger than small ones. The usage of a linear cost function, resulting in a *Mixed Integer Linear Program* (MILP) is currently under investigation and is expected to have computational advantages in terms of solution times (Bemporad, Borrelli and Morari, 2000a).

The usual procedure for solving mixed integer programs is to rely on an iterative search scheme. We solve a set of relaxed problems, the solutions of which approach the solution of interest. For general MIQPs the average solution time can be significantly influenced by the choice of the search procedure. The main goal of this chapter is to propose a strategy to solve mixed integer programs, especially tailored for the problems occurring for MLD systems.

After reviewing some standard facts about mixed integer optimization in Section 5.2, we consider in Section 5.3 in detail the branch and bound method, the search procedure of which can be represented by a tree data structure. A new algorithm, which exploits structural characteristics of the optimizations problems for MLD systems is presented in Section 5.3.3. We estimate the computational complexity of the algorithm and give suggestions about its implementation. The effect of this algorithm is illustrated with simulations on the three tank system of Section 3.2. Some commercially available solvers are described in Section 5.4. A free solver for Matlab that has been coded during this work is described in Section 5.5.

5.2 Mixed Integer Optimization

Mixed integer programming problems are classified in general as $\mathcal{NP}$-hard, which means that with the available algorithms in the worst case, the solution time grows exponentially with the number of integer variables (Raman and Grossmann, 1991). Despite this combinatorial nature, several algorithmic approaches have been proposed and applied successfully to medium and large size application problems (Fletcher and Leyffer, 1998), the four major ones being: *Cutting plane* methods, where new constraints ("cuts") are generated and added to reduce the feasible domain until the optimal solution is found; *decomposition* methods, where the mathematical structure of the models is exploited via variable partitioning, duality, and relaxation methods; *Logic-based* methods, where disjunctive con-

straints or symbolic inference techniques are utilized, which can be expressed in terms of binary variables; *branch and bound* (B&B) methods , where the 0-1 combinations are explored through a binary tree, the feasible region is partitioned into sub-domains systematically, and valid upper and lower bounds are generated at different levels of the binary tree. For MIQP problems, Fletcher and Leyffer (1998) mention Generalized Benders' Decomposition (Lazimy, 1985), Outer Approximation, LP/QP based branch and bound, and B&B as the major solver methods. See (Volkovich et al., 1987) for a review of these methods. Several authors agree on the fact that B&B methods are the most successful for mixed integer quadratic programs (Fletcher and Leyffer, 1998).

There exist some exceptions of mixed integer programs having polynomial complexity in the number of binary variables. A simple case consists of the presence of exclusive-or constraints over all integer variables (Mignone, 1999). Moreover, there exist integer linear programs, the solution of which can be found by solving one single linear program. This occurs, e.g., if the constraints are totally unimodular (Nemhauser and Wolsey, 1988; Chandru and Hooker, 1999), since then they define a polyhedron with integral vertices. This implies that the relaxation of the constraints $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$ leads to a continuous optimization problem, whose solution is integral for x_i. After having formulated the mixed integer optimization according to Chapter 4 it is worthwhile checking a posteriori, if the optimization has a form allowing an easy solution. However, most likely this will not be the case. At present it is not clear either, under which conditions on the model the resulting optimization problem lies in such a class, and when the results for integer programs mentioned above can be used for mixed integer programs.

On the other hand, the optimization problems stemming from MLD systems exhibit some clear structural properties, like e.g. the repetition of some blocks of constraints over the optimization vector. The latter phenomenon is due to the evolution of a system over some given horizon, both in control and estimation problems. Devising particular B&B algorithms that exploit the problem structure can be useful to reduce the amount of com-

putations on average. We will measure the complexity of a solution method for an MIQP by the number of relaxed QPs (Quadratic Programs) that have to be solved during B&B. This is only a rough measure of complexity since it does not take into account the complexity of the single QPs. It is however a measure that often reflects the time required for finding a solution.

5.3 The Branch and Bound Method

One disadvantage of MIQPs is the computational complexity that in the worst case is exponentially increasing with the number of binary optimization variables δ and ϕ, as they appear e.g. in the faulty MLD system (3.37). However, in general the complexity is inherent to the problem and not a particular drawback of the proposed analysis or synthesis methods. At present the only method to guarantee the fulfillment of hard timing constraints and the global optimal solution for the optimization is explicit enumeration of all possible combinations of the integer variables. This brute-force approach is not recommended due to large computations precluding an online application, already for small problem sizes. On average, branch and bound algorithms are an efficient way to solve MIQPs (Nemhauser and Wolsey, 1988; Floudas, 1995).

Problem specific knowledge can be incorporated in the search strategy of B&B methods in order to speed up the computations. We present in the next sections a technique that is particularly suited for the control and estimation problems considered in this work.

5.3.1 Branch and Bound Algorithms for Mixed Integer Quadratic Programs

A *Mixed Integer Quadratic Program* (MIQP) has the following form

$$\begin{aligned} \min_x \quad & 0.5x^TQx + b^Tx && (5.1)\\ \text{subject to} \quad & Cx + d \leq 0 \\ & x = \left[\begin{smallmatrix} x_c \\ x_d \end{smallmatrix}\right],\ x_c \in \mathbb{R}^{n_c} \\ & x_d \in \{0,1\}^{n_d} && (5.2) \end{aligned}$$

and differs from a standard QP through the integrality constraint (5.2)[1]. If $Q = 0$, the optimization is a *Mixed Integer Linear Program* (MILP).

5.3.2 Representation of Mixed Integer Quadratic Programs as Trees

The main idea in solving MIQPs with branch and bound methods lies in the solution of *subproblems* that are relatively easy to tackle. Their solution approaches the desired optimum or represents a bound thereof. The key concepts in branch and bound are therefore:

- *separation* of a problem, referring to the generation of new subproblems from a given one
- *relaxation* of the integrality constraints (5.2), i.e. integer variables are allowed to span the interval $[0, 1]$. The subproblems of an MIQP are therefore QPs.

The optimal values of the subproblems, if they exist, represent lower bounds on the optimal value of the original MIQP (Floudas, 1995). A graphical representation of the concepts

[1] In a more general setup any integer value is allowed, but we restrict ourselves to the 0/1 case here.

relaxation and separation in branch and bound algorithms can be drawn with the help of k-ary trees. Figure 5.1 depicts a binary tree.

We recall here some standard concepts from tree data structure terminology (Massey, 1996). A *tree* consists of *nodes* (or *vertices*) and *branches* (or *arcs*). Exactly one node of a tree is characterized as the *root*. Each node except the root has a unique *father*, i.e. a unique predecessor towards the root. Each node including the root can have none, one or more subsequent nodes, called the *children* of the node. Nodes without children are called *leaves*. A tree, where each node except the leaves has exactly k children is called a *k-ary tree*. The *depth* of a node is the number its predecessors towards the root, i.e. the number of its 'fathers' and 'grandfathers'. The *q-th level* of a tree is the set of all nodes with depth q. The depth of the root is 0. The *length* of a tree is the maximum depth over all its nodes. A *full k-ary tree of length N* is a k-ary tree with k^N leaves, each at depth N. The tree obtained from a node ν by deleting the branch to its father and taking ν as root of a new tree is the *subtree of node ν*.

The representation of MIQPs as trees is done as follows. Let $\xi \in \{0,1\}^{n_d}$ be a vector having the same dimension n_d as the vector of binary variables x_d, and let the symbol $\star$ mean that the corresponding entry of ξ is relaxed, i.e. free to span the interval $[0,1]$. We associate the original MIQP (5.1) where all integrality constraints are relaxed to the interval $[0,1]$ with

$$\xi_0 = \underbrace{[\star\ ,\ \star\ ,\ \ldots\ ,\ \star]}_{n_d \quad times} \tag{5.3}$$

The vector ξ_0 is assigned to the root of a k-ary tree. The separation of the original MIQP or any subproblem into relaxed QPs is done by setting selected integer variables to 0 or 1. The resulting new QP problems are assigned to the children of the node.

Example 5.1:

For instance, separating the root $[\star\ ,\ \star\ ,\ \ldots\ ,\ \star]$ on the second variable results

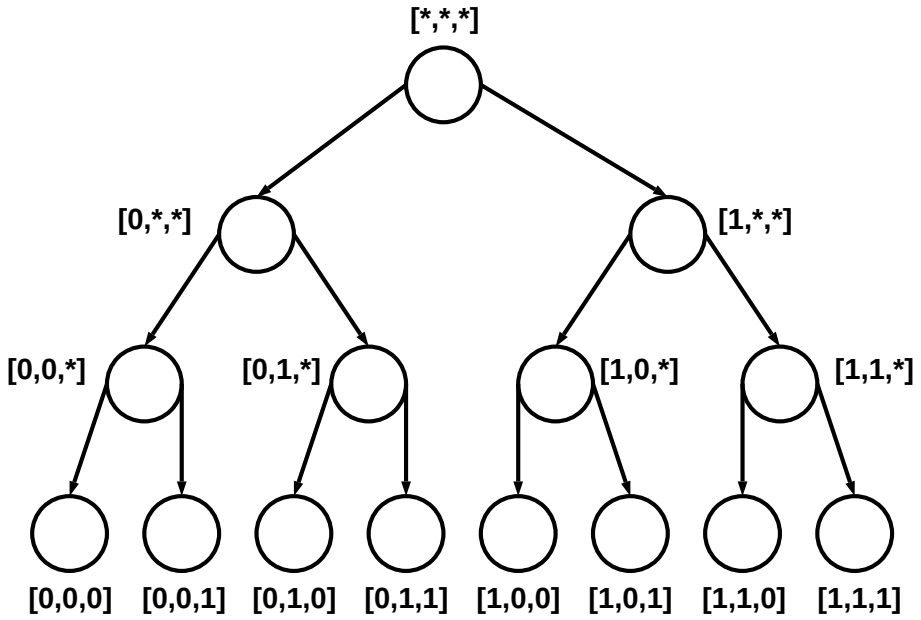


Figure 5.1: The binary tree for a MIQP with 3 integer variables. Each node is marked with the corresponding vector ξ_j.

in the two new subproblems:

$$\xi_1 = [\star\ ,\ 0\ ,\ \star\ ,\ \ldots\ ,\ \star] \tag{5.4}$$

$$\xi_2 = [\star\ ,\ 1\ ,\ \star\ ,\ \ldots\ ,\ \star] \tag{5.5}$$

In Figure 5.2 we sketch the separation of the root on the second variable. □

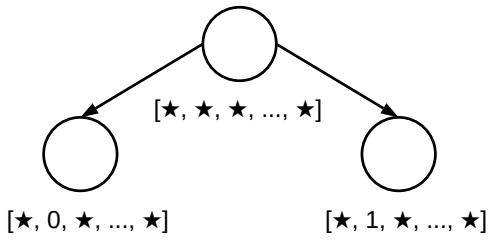


Figure 5.2: Separation of the root on the second variable

We denote each child by a vector ξ_j, $\xi_j \in \{\star, 0, 1\}^{n_d}$. If the i-th component $\xi_j^i = 0$ (or $\xi_j^i = 1$), then the QP corresponding to that node is solved by setting the i-th integer variable to 0 (or 1). If $\xi_j^i = \star$, then the i-th integer variable of ξ_j is regarded as free within $[0, 1]$ in the corresponding QP.

Example 5.2:

The tree of an MIQP with 3 binary variables is shown in Figure 5.1. Note that the number of continuous variables n_c is arbitrary in this representation. The number of optimization variables in subproblem ξ_j is

$$n_c + \#\{k \in [1, n_d] : \xi_j(k) = \star\}$$

As another illustration, consider the node marked with $[0, 1, \star]$. This node denotes the following QP:

$$\begin{aligned} \min_x \quad & 0.5x^T Qx + b^T x \\ \text{subject to} \quad & Cx + d \leq 0 \\ & x = \left[\begin{smallmatrix} x_c \\ x_d \end{smallmatrix}\right],\ x_c \in \mathbb{R}^{n_c} \\ & x_d(1) = 0 \quad x_d(2) = 1 \quad x_d(3) \in [0, 1] \end{aligned}$$

□

Any MIQP with n_d binary variables can be associated to the $2^{n_d+1} - 1$ QPs in the tree structure described above. A general B&B algorithm for an MIQP can be stated as follows (Floudas, 1995):

Algorithm 5.1 *Branch and Bound*

1. *Initialize a list of subproblems with the MIQP to be solved. Set*

$$f_{opt} = +\infty \quad \text{and} \quad x_{opt} = [\infty, \ldots, \infty]$$

2. *If the list of subproblems is empty, terminate with f_{opt} and x_{opt}. If $f_{opt} == +\infty$, the MIQP is infeasible.*

3. *Select one problem in the list of subproblems to become the current problem and delete it from the list of subproblems.*

4. *Relax the current problem, solve it, and denote its solution as f_R and its optimizer as x_R.*

5. - *If the current problem is infeasible, then go to point 2. This step is called* fathoming by infeasibility.
   - *Else if $f_R \geq f_{opt}$, then go to point 2. This step is called* fathoming by higher cost
   - *Else if x_R satisfies the integer constraints and $f_R \leq f_{opt}$, then update the current best solution by setting*

     $$f_{opt} = f_R \quad \text{and} \quad x_{opt} = x_R$$

     and go to point 2.

6. *Generate two new subproblems out of the current problem and add them to the list of subproblems. This step is called* separation. *Go to point 2.*

Strategies for B&B

The two decisions that determine the average performance of the B&B algorithm are taken in steps 6 and 3. In step 6 separation is performed by setting one selected binary optimization variable to its two possible values. The chosen variable is called *branching variable* and its selection defines the *branching rule*. The principle according to which the next current node is chosen in step 3 is called *tree exploring strategy*. A good B&B algorithm aims at quickly fathoming large subtrees, therefore avoiding the solution of many

subproblems. The most common choices for the branching rule and the tree exploring strategy are listed next.

The Branching Rule

Some possible branching rules are:

First Free Variable: Among the relaxed integer variables (denoted by $\star$), choose the one with the smallest index.

Fractional Part-Based Branching: Solving the relaxed QPs, the solution for the variables that are constrained to be binary will usually have a fractional part. Choose the variable that has the largest or the smallest distance to the next integer value as the next branching variable. Choosing to branch on the maximal fractional part, the index of the branching variable is therefore determined as:

$$j = \arg\max_i(\min_{\delta_i}\{\delta_i, 1-\delta_i\}) \tag{5.6}$$

and branching on the minimal fractional part chooses the index as

$$j = \arg\min_i(\min_{\delta_i}\{\delta_i, 1-\delta_i\}) \tag{5.7}$$

User Prioritization: The sequence of branching variables is user-specified. This case occurs for instance, if we decide to branch on those variables first that have a small time index over the horizon.

Branching on Multiple Variables: Groups of d variables are chosen simultaneously according to the rules above. This gives a 2^d-ary tree for the MIQP.

The Tree Exploring Strategy

Some standard tree exploring strategies are:

Depth First Strategy: The QPs are solved by a last-in first-out (LIFO) rule.

Breadth First Strategy: The problems at depth N are not solved before all problems at depth $N-1$ have been solved.

Best Bound Strategy: The node is chosen with the best value for the corresponding relaxed QP.

Normalized or Alternative Best Bound Strategy: The node is chosen with the best normalized cost for the relaxed QP, where the normalization is performed with the depth of the node, i.e. the number of variables already set.

In order to illustrate the tree exploring strategies, consider again an MIQP with 3 binary variables. The order in which the problems are solved for the depth first and the breadth first strategy are given in Figure 5.3.

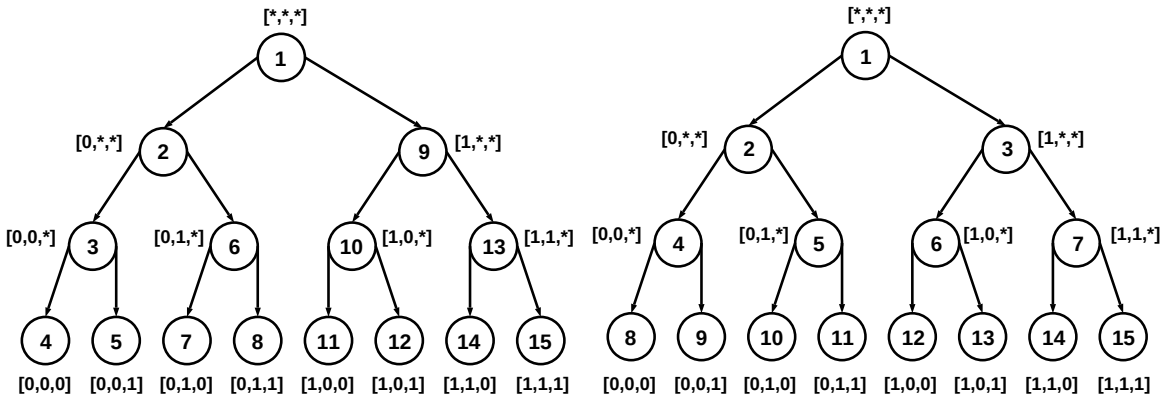


Figure 5.3: Order how problems are solved in the depth first tree exploring strategy (left), and the breadth first tree exploring strategy (right).

5.3.3 The Outside First Tree Exploring Strategy

We present in this section an approach that belongs to the class of B&B methods. It is tailored to optimal control or estimation problems, like (4.1) for MLD systems (2.8) where the time sequence of binary variables switches rarely over the time horizon. In this case, the integer optimization vector x_d in (5.2) contains samples of $\delta(t)$ taken at different time instants, viz.

$$x_d = [\delta(t), \delta(t+1), \ldots, \delta(t+n_d-1)]$$

The main motivation of the algorithm stems from the observation that in some cases the binary variables $\delta(t)$ seldom change their value over the time horizon T. In fact, typically binary variables $\delta(t)$ are associated with conditions on continuous states $x(t)$, for instance

$$[\delta(t) = 0] \leftrightarrow [x(t) \geq x_0] \tag{5.8}$$

This occurs for instance in the three tank system, where Equation (5.8) gives rise to a trajectory for δ as in Figure 5.4.

Because the continuous components satisfy dynamic equations, in general, their inertia will prevent frequent switches of the indicator variable $\delta(t)$. This phenomenon is even more pronounced, when integer variables represent the occurrence of a fault, which involves an irreversible physical damage, because in this case the integer variable will switch at most once its value over the horizon $[t-T, t]$.

We wish to take account of these simple engineering considerations in the B&B algorithm. When we suspect that the system exhibits this kind of behaviour, we should try to solve first the QPs, where the integer variables do indeed describe a limited number of switches. However, we do not want to impose a limited number of switchings over the prediction horizon as a model constraint. In fact, although unlikely, an arbitrary number of switches might indeed occur.

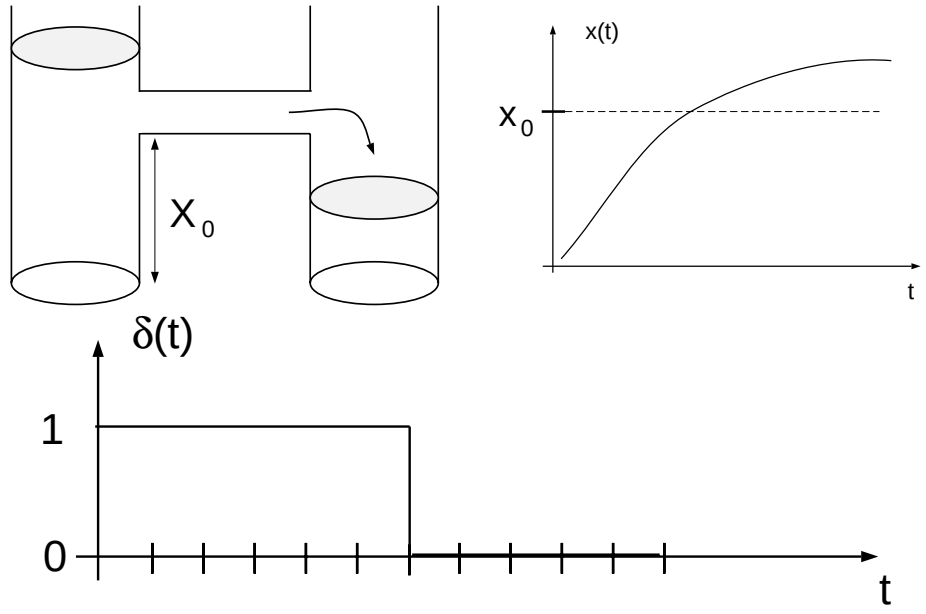


Figure 5.4: Motivation for the outside first tree exploring strategy: Representation of a binary level indicator variable switching its value according to the continuous valued height

Guaranteed Switches

Given a subproblem marked by ξ, let I be the ordered m-tuple collecting the indices i for which $\xi^i \neq \star$,

$$I \triangleq [i_1, i_2, \ldots i_m] \quad \text{such that} \quad \xi^{i_j} \neq \star \ \ \forall i_j \ \text{ and } \ i_1 < i_2 < \cdots < i_m$$

We define the number D of *guaranteed switches* as the number of indices i_q in I such that $\xi^{i_q} \neq \xi^{i_{q+1}}$.

Example 5.3:

For instance, assume that

$$\xi = [0, 0, \star, 1, \star, \star, 1, 0, 0, \star]$$

then with the notation introduced before we have $I = [1, 2, 4, 7, 8, 9]$, and $D = 2$.

□

column -1 original problem	column 0 no guaranteed switches	column 1 1 guaranteed switch	column 2 2 guaranteed switches
$[\star, \star, \star]$	$[0, 0, 0]$	$[0, 0, 1]$	$[0, 1, 0]$
	$[0, 0, \star]$	$[0, 1, 1]$	$[1, 0, 1]$
	$[0, \star, \star]$	$[1, 1, 0]$	
	$[1, 1, 1]$	$[1, 0, 0]$	
	$[1, 1, \star]$	$[0, 1, \star]$	
	$[1, \star, \star]$	$[1, 0, \star]$	

Table 5.1: Classification of subproblems according to guaranteed switches in the binary variables for $n_d = 3$

Example 5.4:
By considering again an MIQP with 3 binary variables (Figure 5.1) we can partition the $2^{3+1} - 1 = 15$ QPs occurring in the branch and bound tree into four classes, according to the guaranteed switches of binary variables in each QP. This classification is given in Table 5.1. □

Outside First Tree Exploring Strategy

The proposed tree exploring strategy solves a QP in column i of Table 5.1 only after all problems in the columns up to $i - 1$ have been solved. The order of the QPs among one column is arbitrary and offers an additional degree of freedom.

According to this rule, the tree with 3 binary variables of the example above is explored in the order denoted in Figure 5.5. We call the strategy *outside first*, since the MIQP tree is explored from the outside to the inside. The root will be assigned by definition -1 guaranteed switches. This technical convention simplifies the assessment of the computational

complexity in Section 5.3.4.

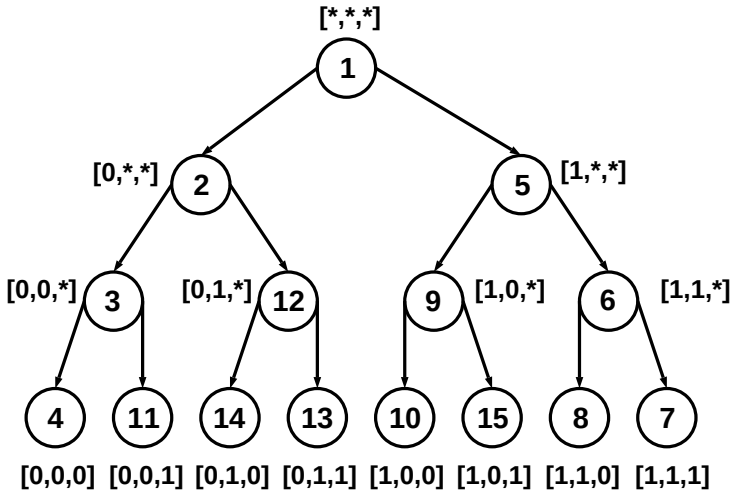


Figure 5.5: Order how problems are solved in the outside first strategy, assuming the "first free variable" branching rule.

We adapt the general B&B algorithm to the outside first strategy and obtain the following algorithm:

Algorithm 5.2 *Branch and Bound with outside first tree exploring strategy*

1. *Initialize a list of subproblems with the MIQP to be solved. Set*

$$f_{opt} = +\infty \quad \text{and} \quad x_{opt} = [\infty, \ldots, \infty]$$

2. *If the list of subproblems is empty, terminate with f_{opt} and x_{opt}. If $f_{opt} == \infty$, the MIQP is infeasible.*

3. *Determine k^o as the minimum of guaranteed switches among the problems in the list of subproblems. Select one problem with k^o guaranteed switches to become the current problem and delete it from the list of subproblems.*

4. *Relax the current problem, solve it, and denote its solution as f_R and its optimizer as x_R.*

5. • *If the current problem is infeasible, then go to point 2.*

   • *Else if $f_R \geq f_{opt}$, then go to point 2.*

   • *Else if x_R satisfies the integer constraints and $f_R \leq f_{opt}$, then update the current best solution by setting*

   $$f_{opt} = f_R \quad \text{and} \quad x_{opt} = x_R$$

   and go to point 2.

6. *Generate two new subproblems out of the current problem, determine their guaranteed switches and add them to the list of subproblems. Go to point 2.*

Often hard restrictions on the time available to perform the required computations severely limit the chances to find a global minimizer for (5.1)–(5.2), especially for large problems with many binary variables. In this case, the MIQP optimization should aim at providing good suboptimal solutions. The problem is then to choose and solve the problems in the MIQP tree that are most likely to give a value of the cost function that is close to the global minimum. The search for suboptimal solutions can be systematized with the outside first tree exploring strategy by defining a maximum number of guaranteed switches k_{max}. Only those subproblems are solved that have a number of guaranteed switches smaller than k_{max}. The outside first approach allows therefore to select and solve the QPs that are most promising in giving good suboptimal solutions, provided that the integer variables do not switch their value often. For MPC of MLD systems local minima are tolerable, in fact, Bemporad and Morari (1999a) prove that stability is not altered by local minima, though the convergence properties of the controller deteriorate.

In principle, the outside first tree exploring strategy can be applied in combination with any branching rule. However, if we consider the interpretation of the binary variables given above, the most natural choice for the branching rule is the 'User Prioritization'. In fact in order to give a physical meaning to the guaranteed switches, the variables have to be

selected, such that they represent subsequent time instances of the same binary system variable.

If the MLD model contains more than one binary variable δ_i, it is appropriate to define blocks of binary variables, which collect the same system variable at subsequent time steps. The number of guaranteed switches in each block is determined independently of the other blocks, and the total number of guaranteed switches is given by the sum of the guaranteed switches in each block.

Example 5.5:
Assume the MPC controller is implemented with a horizon of $T = 3$ steps for a model with two binary variables $\delta = [\delta_1, \delta_2]$. The optimization at time k in Equation (4.1) is done over the vector

$$\begin{aligned} \delta &= [\delta(k), \delta(k+1), \delta(k+2)] \\ &= [\delta_1(k), \delta_2(k), \delta_1(k+1), \delta_2(k+1), \delta_1(k+2), \delta_2(k+2)] \end{aligned}$$

The count of guaranteed switches has to be done defining two blocks b_1 and b_2, as

$$\begin{aligned} b_1 &= [\delta_1(k), \delta_1(k+1), \delta_1(k+1)] \\ b_2 &= [\delta_2(k), \delta_2(k+1), \delta_2(k+1)] \end{aligned}$$

The guaranteed switches are counted separately for each block and added to yield the total number of guaranteed switches. For instance, if a subproblem in the branch and bound tree has $\xi = \delta = [1, 1, 0, \star, \star, 0]$, the number of guaranteed switches is 2, since $b_1 = [1, 0, \star]$ and $b_2 = [1, \star, 0]$, which have both one guaranteed switch. □

5.3.4 Computational Complexity of the Outside First Tree Exploring Strategy

In this section we analyze the computational complexity associated with the outside first tree exploring strategy by using combinatorial arguments. We focus on the case, where computations are stopped, as soon as all problems with less than k_{max} switches are solved. We determine the complexity by counting the nodes in the binary tree having a given number k of guaranteed switches.

Isoswitches – Nodes with the Same Number of Switches

Given a full binary tree A_n of length n and a root with -1 switches, denote by $M_n(k)$ the number of nodes having k guaranteed switches. The same quantity for a subtree having a root with 0 switches is denoted by $\tilde{M}_n(k)$. In Figure 5.6 we marked the number of guaranteed switches for an MIQP tree with 3 integer variables. To simplify the nomenclature we call *node with k switches* the node corresponding to the relaxed QP with k guaranteed switches in the integer variables.

Lemma 5.1 *Assume that the root of A_n has 0 switches. Then*

$$\tilde{M}_n(k) = \binom{n+1}{k+1}, \ \forall k = 0, \dots, n \tag{5.9}$$

Proof. We prove (5.9) by induction on n. For $n = 1$, it is easy to verify that $\tilde{M}_1(0) = 2$, $\tilde{M}_1(1) = 1$ (see e.g. Figure 5.7). Assume now that (5.9) holds. By induction on the tree of length n, it holds that

$$\tilde{M}_{n+1}(k) = \begin{cases} \tilde{M}_n(k) + \tilde{M}_n(k-1) & (k = 1 \dots n+1) \\ \tilde{M}_n(0) + 1 & (k = 0) \end{cases} \tag{5.10}$$

The induction step from n to $n+1$ is performed by adding a new root to a binary tree of length n. Equation (5.10) states that the number of nodes with

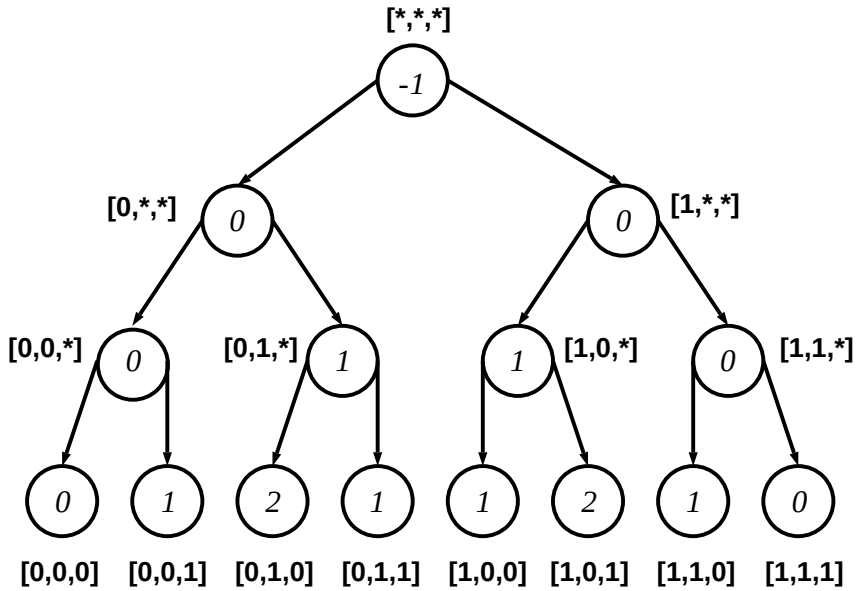


Figure 5.6: Number of switches for each subproblem in the outside first tree. The root has by definition -1 switches.

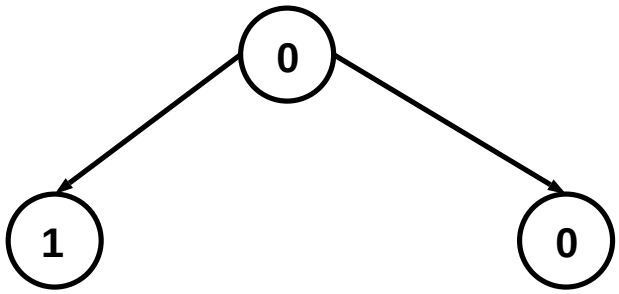


Figure 5.7: Tree of depth 1 with a root of 0 switches

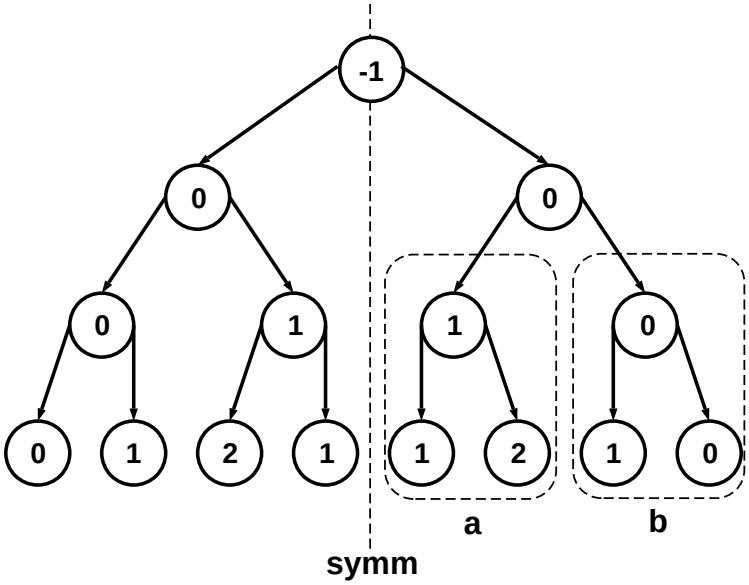


Figure 5.8: Number of switches per node marked in a binary tree. Note the symmetry with respect to the 'symm' - axis. The subtrees denoted by a and b have the same differences between the nodes, but in a the values are augmented by 1.

k switches in a tree of length $n+1$ is the sum of the nodes with k switches in one of the main subtrees of length n and the nodes with $k-1$ switches in a subtree of length n. The latter subtree experiences an increase of the number of switches in each node, as soon as its root becomes a node and gets a new father, cfr. subtree a in Figure 5.8. It is now straightforward to verify that:

$$\begin{aligned}
\tilde{M}_n(k) + \tilde{M}_n(k+1) &= \binom{n+1}{k+1} + \binom{n+1}{k} \\
&= \frac{(n+1)!}{(k+1)!(n-k)!} + \frac{(n+1)!}{k!(n-k+1)!} \\
&= \frac{(n+1)!}{k!(n-k)!}\left(\frac{1}{k+1} + \frac{1}{n+1-k}\right) \\
&= \frac{(n+2)!}{(k+1)!(n-k+1)!} \\
&= \binom{n+2}{k+1}
\end{aligned}$$

which completes the proof. □

In Lemma 5.1 we have assumed that the root has 0 switches. The binary trees we want to consider in the MIQP context, have by definition a root with -1 switches. Exploiting the symmetry of guaranteed switches in the nodes of the tree, we prove the following theorem.

Theorem 5.1 *Assume that the root of A_n has -1 switches. Then:*

$$M_n(k) = 2\binom{n}{k+1} \tag{5.11}$$

for $k = 0 \ldots n-1$.

Proof. The root has -1 switches. Therefore we have to count only the switches in the 2 subtrees of the root. Both subtrees have a root with 0 switches, and therefore by Lemma 5.1 $M_n(k) = 2\tilde{M}_{n-1}(k)$. □

Nodes with Less Than k_{max} Switches

Since each tree has exactly one root, it holds that

$$M_n(-1) = 1$$

The number of nodes with less than k_{max} switches in a full binary tree of length n is denoted by $C_n(k_{max})$:

$$C_n(k_{max}) = \sum_{k=-1}^{k_{max}} M_n(k) \tag{5.12}$$

$$= 1 + 2\sum_{k=0}^{k_{max}} \binom{n}{k+1} \tag{5.13}$$

k_{max}	$C_n(k_{max})$	complexity
0	$1+2n$	$O(n)$
1	n^2+n+1	$O(n^2)$
2	$\frac{n^3}{3}+\frac{5}{3}n+1$	$O(n^3)$

Table 5.2: Number of nodes with less than k_{max} switches in a tree of length n

Note that (5.13) is consistent with the total number of nodes in a full binary tree of length n, since:

$$\begin{aligned} C_n(n-1) &= 1+2\sum_{k=0}^{n-1}\binom{n}{k+1} \\ &= 1+2\left(\sum_{k=1}^{n}\binom{n}{k}\right) \\ &= 1+2\left(2^n-1\right) \\ &= 2^{n+1}-1 \end{aligned}$$

In Table 5.2 we illustrate that the complexity of the outside first algorithm is polynomial in the number of binary variables once the number of switches is limited to a fixed $k_{max} \ll n$.

5.3.5 Implementation Scheme

Each tree exploring strategy requires a data structure to store the subproblems generated during the tree search. In fact, the depth first tree exploring strategy can be implemented with a *stack* or a LIFO-list. Similarly the breadth first strategy results from a storage of the problems in a *first-in first-out* (FIFO) list. These standard data structures are appealing, since they simplify the implementation of the search procedures.

The outside first tree exploring strategy can be implemented in a similar way, by using multiple stacks. Each stack contains the subproblems marked by the same number of guaranteed switches. The correct order, in which the subproblems should be solved, is

obtained by emptying the stack first that contains the subproblems with fewer guaranteed switches.

Branching on one variable results in at most three stacks to be used simultaneously. This can be seen by noting that branching can add 0, 1 or 2 guaranteed switches, see Figure 5.10 for a case where three stacks are necessary. If the branching rule is the first free variable, only two stacks are required, since in this case 2 guaranteed switches can never be added to the number of guaranteed switches of the current subproblem, see Figure 5.9.

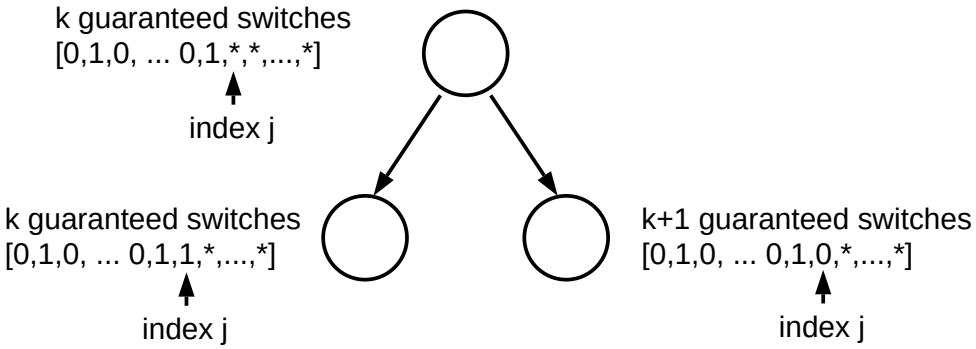


Figure 5.9: Branching on the first free variable (here index j) generates subproblems of either k, or $k+1$ switches, if the node to be separated has k guaranteed switches

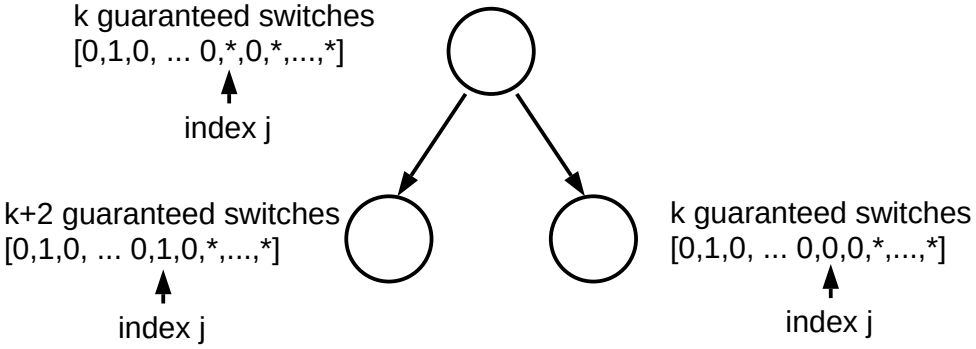


Figure 5.10: Branching according to other criteria (here index j) generates subproblems of either k, $k+1$ or $k+2$ switches, if the node to be separated has k guaranteed switches

5.3.6 Extension to Optimal Control

In the previous sections we have formulated the outside first tree exploring strategy, suggesting to solve those problems first that have few switches in the sequence of binary variables. The assumption is justified for binary indicator variables or for faults. In a receding horizon context, the outside first idea can be extended to solve those problems first that exhibit few differences in the binary variables compared to the optimal solution at the previous time instant. In Figure 5.11 it is illustrated, how to get the sequence of binary variables δ_{guess} by shifting δ_{opt} one step forward and padding the last entry with an arbitrary value. In the modified outside first strategy we decide to solve those problems first, the binary variables of which have a small *Hamming distance*[2] to δ_{guess}. In other words, instead of counting guaranteed switches, we count the Hamming distance between δ_{guess} and the vector of binary variables ξ in the MIQP tree.

5.3.7 Example: Control of the Three Tank System

We have applied the outside first tree exploring strategy to a model predictive control scheme of the three tank system in Section 3.2. For this simulation we have excluded tank 2 from the model. We compared the outside first algorithm with the breadth first and the depth first tree exploring strategies.

In Figure 5.12 we show the number of QPs that are solved at each time step in order to compute the global minimum, using different tree exploring strategies. The standard breadth first and depth first strategies require a significantly higher number of QPs than the outside first strategy at the beginning of the control experiment. After about 8 steps the system is in steady state. In this phase we remark a smaller number of QPs solved by the outside first strategy. In Table 5.3 we report the total number of QPs solved during

[2]The Hamming distance of 2 vectors $x_1, x_2 \in \{0,1\}^{n_d}$ is the number of indices, in which x_1 and x_2 differ.

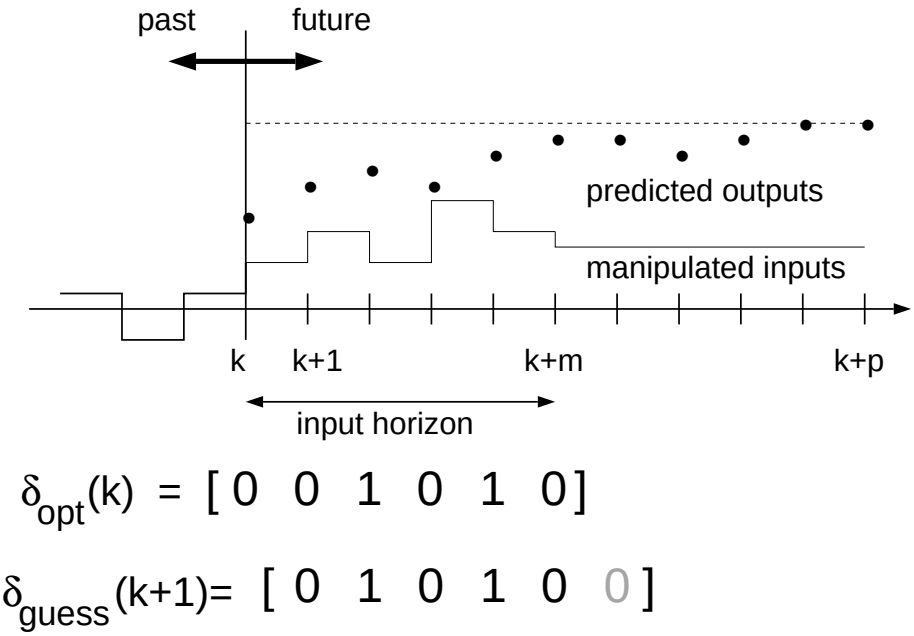


Figure 5.11: Obtaining the vector δ_{guess} in the modified outside first tree exploring strategy. Here, δ_{guess} is obtained by padding it with a 0.

the simulation.

Considering the numbers of Table 5.3, we have repeated the simulation of Figure 5.12 by imposing an artificial upper bound on the number of QPs to be solved for each MIQP. This is a similar specification as limiting the number of switches, as suggested in Section 5.3.4, however it allows an easier comparison to the other tree exploring strategies. We assumed that at each time step, the solver is allowed to solve at most 100 QPs. This simulates hard timing constraints on the control scheme. The solver reports the currently best integer feasible solution when this limit is reached. If no integer feasible solution could be found, the solver reports the shifted solution from the previous simulation.

When the number of QPs is limited, the outside first approach is the only strategy allowing to reach the steady state for both levels h_1 and h_3, as it can be seen in Figure 5.13.

We note in Figure 5.13 that the breadth first strategy fails to reach the target for h_1 and

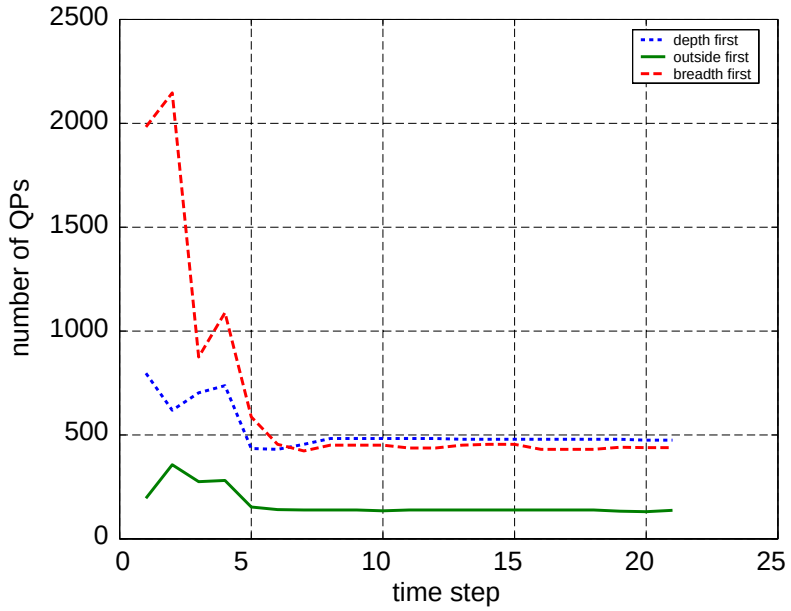


Figure 5.12: Number of relaxed QPs to be solved at each time for controlling the three tank system, using different tree exploring strategies

strategy	total number of QPs	maximum and minimum number of QPs for 1 step	
depth first	10'895	797	431
breadth first	13'759	2147	423
outside first	3467	357	131
worst case	1'376'235	65'535	(1)

Table 5.3: Number of QPs during the simulation of Figure 5.12

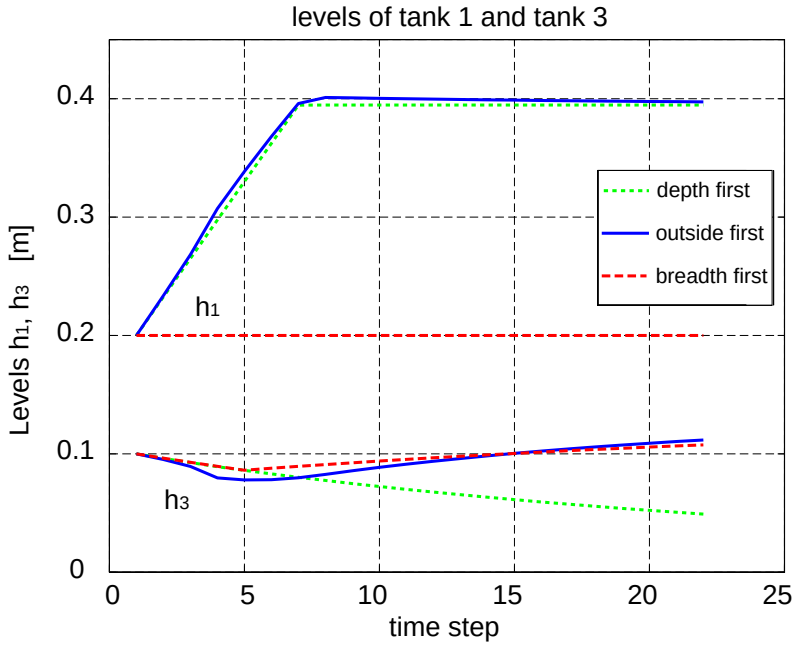


Figure 5.13: Trajectories of the states of the three tank system, when the number of relaxed QPs is limited at 100 for each time step, using different tree exploring strategies

the depth first strategy causes tank 3 to run dry. Apparently, the outside first strategy exploits the available computation time by solving first the most promising subproblems among the open problems, and achieves the control goal for both levels.

When limiting the computational time, a relevant issue for a given problem is the investigation, at which point of the tree exploration the optimal solution is found. This knowledge can be acquired offline by simulations and can be used online to limit the number of QPs accordingly. In fact it is known that in B & B often many computations are done merely in order to verify that the present candidate solution is optimal.

For the simulation in Figure 5.12 the first three strategies get almost down to the worst case. This means that the optimum is found almost at the last QP. The outside first approach not only involves fewer QPs, but also reaches the optimum after roughly half the QPs at steady state. Limiting the number of QPs to about this value would therefore

further decrease the complexity.

5.3.8 Example: Fault Detection of the Three Tank System

In Figure 5.14 we compare the number of QPs solved during a fault detection experiment for the outside first and the depth first strategy. All three faults occur sequentially, and we assume that one fault is repaired, as soon as a new fault occurs. Contrary to the

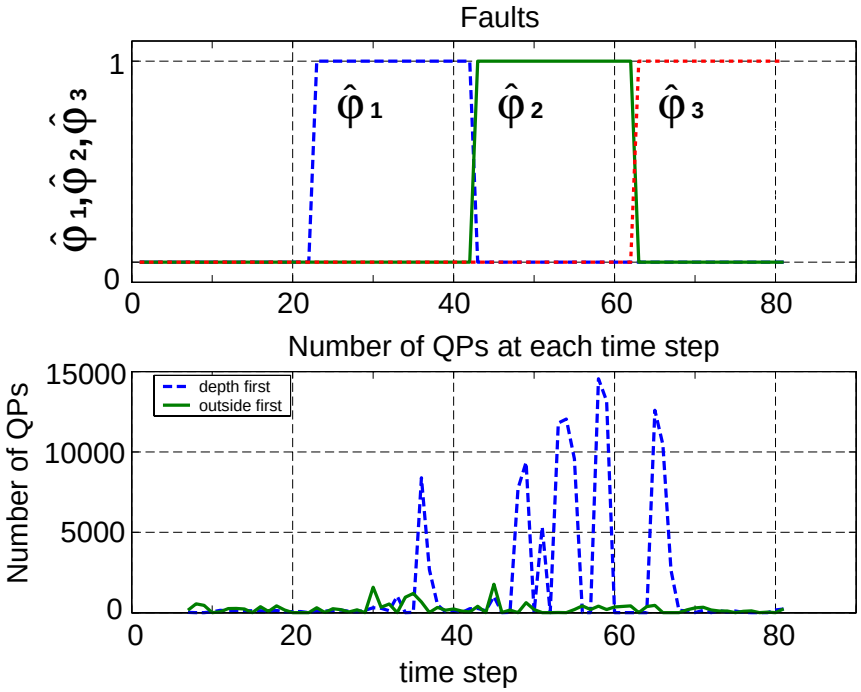


Figure 5.14: Number of QPs during a fault detection simulation

control experiment in Section 5.3.7, we see that the outside first strategy is not uniformly better than the depth first strategy, since it requires to solve more QPs at some time steps. Indeed, in the simulation of Figure 5.14 there are 41 time steps, where the depth first strategy copes with fewer QPs, but only 30 time steps, where the outside first strategy prevails. In the overall simulation the outside first strategy computes about a factor of 7 less QPs (see Table 5.4) because there is a large difference in QPs, when the outside first

strategy	total number of QPs	maximum and minimum number of QPs for 1 step	
depth first	127'657	14'553	1
outside first	18'475	1769	1
worst case	$40.27 \cdot 10^9$	$536.9 \cdot 10^6$	1

Table 5.4: Number of QPs during the simulation of Figure 5.14

strategy is better. In our example, this difference can be up to 14'000 QPs.

5.3.9 Summary

The outside first tree exploring strategy is a heuristic rule. It exploits structural properties of the MLD optimizations at computational level. There is no guarantee that for all systems and for all resulting mixed integer optimizations, the scheme provides faster solutions. Its effectiveness relies on an assumption about the switching properties of the binary variables involved in the MLD model.
The outside first tree exploring strategy is searching for the global optimum, provided that computations are not interrupted after all problems of a fixed number of switches have been solved. In the latter case we have illustrated that the scheme is able to report good suboptimal solutions compared to standard tree exploring strategies.

5.4 Solvers for Mixed Integer Continuous Optimizations

Several tools are available for mixed integer continuous optimization. The following list is by no means complete and shows only the tools considered during this work.

CPLEX (ILOG, Inc., 2000) is a package for solving linear and mixed integer linear programs. The solver is very fast and reliable. Unfortunately it does not include routines for solving MIQPs.

BARON (Sahinidis, 2000) is a tool for solving nonconvex optimization problems to global optimality. Purely continuous, purely integer, and mixed integer nonlinear problems can be solved with the software. Therefore its scope encompasses both MIQPs and MILPs. BARON requires routines from mathematical libraries, like MINOS (Murtagh and Saunders, n.d.) or CPLEX.

Xpress-MP (Dash Associates, 1999) is an optimizer for linear programming and integer programming problems. Starting from release 12 of the software MIQPs are also handled.

miqp.m (Bemporad and Mignone, 2000) is a Matlab m-function for solving MIQPs and MILPs within Matlab. The code is free, although it is slow, since its coding is not optimized for efficiency. The emphasis has been put rather on the readability and the portability of the routine. In Section 5.5 we briefly describe the solver.

NAG toolbox (NAG, 2001): The Numerical Algorithms Group (NAG) offers a series of numerical and statistical libraries, used to solve different mathematical problems. The routines on optimization represent only one small fraction of the capabilities. In the release "Mark 5" a solver for MIQPs and MILPs has been introduced. Its code-name is `h02bbc`.

MIQP solver: Fletcher and Leyffer (1994) describe a branch and bound solver, which can exploit sparsity. The solver computes lower bounds efficiently during the tree search using the techniques shown in (Fletcher and Leyffer, 1998).

The environment for all experiments and simulations in this work is Matlab. For most of the packages listed above an interface to Matlab is available at `http://www.aut.ee.ethz.ch/~hybrid/internal/ComputationalTools/panmip.m`.

5.5 miqp.m: A Mixed Integer Quadratic Programs Solver for Matlab

The Matlab function `miqp.m` allows to solve mixed integer quadratic programs and mixed integer linear programs. The solver is implemented using a branch and bound technique and allows the user to specify various options, like tree exploring strategies, branching variable selection rules, and many more. The code is freeware and can be downloaded from `http://www.aut.ee.ethz.ch/~hybrid/miqp/`. The documentation and user manual can be found in (Bemporad and Mignone, 2000).

The Matlab function `miqp.m` solves the following MIQP:

$$\begin{aligned} \min_x \quad & 0.5x^T Hx + f^T x \\ \text{subject to} \quad & Ax \leq b \\ & A_{eq}x = b_{eq} \\ & v_{lb} \leq x \leq v_{ub} \\ & x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_d} \\ & x(i_{vartype}) \in \{0,1\}^{n_d} \end{aligned}$$

The length of the optimization vector x is $n = n_c + n_d$. The variables indexed by $i_{vartype}$, which is a subset of $\{1, \ldots, n_c + n_d\}$, are constrained to be binary. The matrix $H \in \mathbb{R}^{n \times n}$ is positive semidefinite. The special case, where $H = 0$ is an MILP and it can also be handled by `miqp.m`. The matrix $A \in \mathbb{R}^{m \times n}$ and the vector $b \in \mathbb{R}^m$ define linear inequality constraints on the optimization variables. Linear equality constraints are given by $A_{eq} \in \mathbb{R}^{m' \times n}$ and $b_{eq} \in \mathbb{R}^{m'}$ Bounds on x can be specified by the vectors $v_{lb} \in \mathbb{R}^n, v_{ub} \in \mathbb{R}^n$.

`miqp.m` was not conceived to compete with available optimized solvers for MIQPs, like they are described in (Dash Associates, 1999; Fletcher and Leyffer, 1994; Sahinidis, 2000). In fact some simulations show that `miqp.m` is rather slow compared to these solvers. More emphasis has been put on the possibility to choose design parameters and solution strategies

in an easy way, allowing the user to easily write his/her own decision routines, if necessary. `miqp.m` comes "ready to use" in the sense that choosing default settings allows a quick start in the solution of MIQPs.

Chapter 6

Mixed Logic Dynamical and Piecewise Affine Systems

In this chapter we consider the stability analysis for piecewise affine systems.

6.1 Introduction

This chapter is an extended version of the paper (Ferrari-Trecate, Cuzzola, Mignone and Morari, 2002). We present various algorithms both for stability and performance analysis of discrete-time Piece-Wise Affine (PWA) systems. For stability, different classes of Lyapunov functions are considered and it is shown how to compute them through Linear Matrix Inequalities that take into account the switching structure of the systems. We also show that the continuity of the Lyapunov function is not required in the discrete-time case. Moreover, the tradeoff between the degree of conservativeness and computational requirements is discussed. Finally, by using arguments from the dissipativity theory for nonlinear systems, we generalize our approach to analyze the l_2-gain of PWA systems. For some stability analysis methods, we derive the corresponding stabilizing state feedback synthesis

expressions.

Piece-Wise Affine (PWA) systems have been receiving increasing attention by the control community because they provide a useful modelling framework for a class of hybrid systems. The study of PWA system is motivated by the interest in this class of systems by itself, but also for the actual goal of considering more general classes of systems, for which piecewise affine systems are a suitable approximation. Moreover, discrete-time PWA systems are equivalent to interconnections of linear systems and finite automata (Sontag, 1996), to linear complementarity systems (Heemels et al., 2001) and also hybrid systems in the MLD form, see Chapter 2.

The key idea is to exploit this equivalence to use tools and methods that were originally developed for PWA systems, also for MLD systems, and viceversa. If we are able to solve a problem using one modelling framework, we get in principle a solution for a system described in the other modelling framework with little effort. The modelling capabilities of MLD systems and their transformation to PWA systems are recalled in Section 2.6.

Another reason for considering PWA systems is that in (Bemporad, Morari, Dua and Pistikopoulos, 2000) the explicit form of Model Predictive Control (MPC) for linear constrained systems was derived and, besides providing an algorithm for its computation, it was shown that the closed-loop system has a PWA structure. In (Bemporad, Borrelli and Morari, 2000a) such results were further generalized to obtain the explicit form of MLD/PWA systems regulated with an MPC scheme. Also in this case the closed-loop system is a PWA model.

An important feature of PWA models is that the state-update map can be discontinuous along the boundary of the regions. For instance, when considering PWA systems stemming from hybrid systems in the MLD form, discontinuities can arise from the representation of logic conditions.

The control problem of MLD systems and consequently PWA systems is computationally

difficult: in (Bemporad and Morari, 1999a) a Mixed Integer Quadratic Programming approach is proposed in order to solve the control problem of MLD systems by means of MPC techniques. The computational complexity of this approach may increase exponentially with the prediction horizon considered. The use of *Linear Matrix Inequalities* (LMI) techniques, for which computationally advantageous and numerically reliable algorithms, as well as toolboxes are available, is a valuable alternative. In Appendix C we give a short summary on LMIs.

One aim of this chapter is to develop tests for checking the stability of the origin of a PWA system. The main motivation for this research is that there is no easy way to check the stability even of an autonomous PWA system defined over two polyhedral regions. Indeed, Blondel and Tsitsiklis (1999) showed that in general this problem is either $\mathcal{NP}$-complete or undecidable[1]. Even the boundedness of trajectories described by a piecewise linear dynamics is undecidable in general. Moreover, it is not possible to deduce the stability or the unstability of a PWA system from the stability or unstability of its component subsystems (Branicky, 1998). These results highlight that, in order to derive practical stability tests, one must either resort to a restricted subclass of PWA system or look for *sufficient* conditions. An algorithm to test stability for discrete-time PWA systems was proposed in (Bemporad, Torrisi and Morari, 2000) by exploiting verification and robust simulation ideas. However, this test applies only to PWA systems where the origin is strictly contained in one region and it gives conclusive results only if the state trajectory enters an invariant region around the origin in a number of steps that must be fixed a priori. Nevertheless the method in (Bemporad, Torrisi and Morari, 2000) can be viewed as complementary to the tests proposed in this chapter.

Johansson and Rantzer (1998) investigate the stability of a continuous-time PWA system by using piecewise quadratic (PWQ) Lyapunov functions that can be computed by solving suitable LMIs. This approach obviously leads to sufficient conditions, i.e. if the LMIs have

[1] A problem is undecidable if there is no algorithm which, upon input of the data associated with an instance of the problem, provides a yes-no answer after finitely many steps.

a solution, the origin of the system can be classified as asymptotically stable but nothing can be said if no solution is found. In particular, their approach hinges on the use of *continuous* Lyapunov functions. A Matlab toolbox implementing some of the techniques shown in (Johansson and Rantzer, 1998; Johansson, 1999) is described in (Hedlund and Johansson, 1999). These results have been generalized in (Pettersson and Lennartson, 1997; Pettersson and Lennartson, 1999) to a wider class of continuous-time hybrid systems showing that stability can be also checked by exploiting PWQ *discontinuous* Lyapunov functions. Ezzine and Kavranoglyu (1997) present stability criteria of discrete time jump linear systems. This class of systems is represented by similar dynamic equations as a PWA system[2] with the only difference that the system matrices change according to a finite state Markov chain (Romanovsky, 1970). The stability criterion is formulated as a LMI problem (Boyd et al., 1994). Contrary to the approach shown in this chapter, it contains probabilistic elements.

We propose LMI based procedures similar to (Johansson, 1999) in order to analyze the stability of discrete-time PWA systems via either PWQ or more general Lyapunov functions. We do not require continuity of the Lyapunov functions across the boundaries and the a major difference between the continuous and the discrete-time case is the phenomenon that in discrete-time switching can also occur between non-adjacent regions and this fact must be properly handled in the analysis algorithms. It is worthwhile pointing out that the use of LMI techniques for the analysis of discrete-time affine parameter-dependent systems is proposed in (Slupphaug et al., 2000). The PWA systems considered here have continuous states and inputs. An extension to PWA systems with Boolean states and inputs is reported in (Ferrari-Trecate, Cuzzola and Morari, 2002).

Concerning stability analysis of discrete-time PWA systems, in Sections 6.2 and 6.3 we propose various stability tests for PWA systems that exhibit different degrees of flexibility. The more general procedures are computationally more demanding. In particular, in Sec-

[2] see Equation (6.1)

tion 6.3.5 we propose a relaxation strategy (analogous to the one proposed in (Johansson and Rantzer, 1998; Jakubovic, 1977)) for the LMIs involved in the stability analysis through PWQ Lyapunov functions that allows us to consider displacements in the PWA system. These relaxations along with a proper handling of the switching structure, aim at reducing conservativeness of the criteria. Moreover, in Section 6.4, we discuss the degree of conservativeness of the analysis algorithms and compare our technique with the one proposed in (Johansson and Rantzer, 1998) for the continuous-time case. Finally, in Section 6.5 we generalize our approach to the synthesis of storage functions for testing the l_2-gain of PWA systems. The rationale for our derivation hinges on the use of passivity theory for nonlinear systems (Lin and Byrnes, 1996). An application of the performance tests is the possibility to check *a posteriori* the performance of MPC for both linear and hybrid systems in the MLD form. This can be done by applying the techniques we propose to the explicit PWA form of the closed-loop system.

6.2 Stability of Discrete-Time Piecewise Affine Systems

In this section we focus on analysis problems for autonomous PWA systems with the following form:

$$x_{k+1} = A_i x_k + a_i, \quad \text{for } x_k \in \mathcal{X}_i \tag{6.1}$$

where $x \in \mathbb{X} \subseteq \mathbb{R}^n$. For the autonomous PWA system (6.1) the cells $\{\mathcal{X}_i\}_{i=1}^s$ represent a polyhedral partition of the set of states $\mathbb{X}$, i.e. each set $\mathcal{X}_i$ is a (not necessarily closed) convex polyhedron such that

$$\mathcal{X}_i \bigcap \mathcal{X}_j = \emptyset, \quad \forall i \neq j, \tag{6.2}$$

$$\bigcup_{i=1}^{s} \mathcal{X}_i = \mathbb{X}. \tag{6.3}$$

Moreover we assume that the origin belongs to $\mathbb{X}$.
According to the notation in (Johansson and Rantzer, 1998), we call $\mathcal{I} = \{1, \ldots, s\}$ the set of indices of the state space cells of the autonomous PWA system (6.1). $\mathcal{I}$ is partitioned as $\mathcal{I} = \mathcal{I}_0 \cup \mathcal{I}_1$, where $\mathcal{I}_0$ are the indices of the cells whose closure contains the origin $x = 0$, and $\mathcal{I}_1$ are the indices of the cells whose closure does not contain the origin. We assume that $x = 0$ is an equilibrium of system (6.1) and we focus on the stability of the origin. This implies that $a_i = 0$, $\forall i \in \mathcal{I}_0$ in the PWA form (6.1).

Definition 6.1 *The equilibrium state $x = 0$ of a system $x_{k+1} = f(x_k)$ is* **stable** *if, for any $\epsilon > 0$, there exists a $\delta(\epsilon) > 0$, such that $\| x_0 \| < \delta(\epsilon) \implies \| x_k \| < \epsilon, \quad \forall k > 0$.*

If, in addition, $\lim_{k \to +\infty} \|x_k\| = 0$ the origin is **asymptotically stable**.

Finally, let $\mathbb{X}_0 \subseteq \mathbb{X}$ such that $0 \in \mathbb{X}_0$. The origin $x = 0$ is **asymptotically stable on $\mathbb{X}_0$** *if it is asymptotically stable for any initial state $x_0 \in \mathbb{X}_0$.*

Definition 6.2 *Let $\mathbb{X}_0 \subseteq \mathbb{X}$ such that $0 \in \mathbb{X}_0$. The origin $x = 0$ is* **exponentially stable on $\mathbb{X}_0$** *if there exist two coefficients $K > 0$, $0 < \gamma < 1$ and a time instant $\bar{k}$ such that $\forall x_0 \in \mathbb{X}_0$ then $\|x_k\|^2 \leq K\gamma^k \|x_0\|^2$, $\forall k > \bar{k}$.*

Note that in (6.1), the set $\mathbb{X}$ is possibly defined by polyhedral constraints on the state trajectory x_k. Therefore, when focusing on stability on a set $\mathbb{X}_0$ it is natural to introduce the next assumption.

Assumption 6.1 *The free evolution of the state trajectory for the PWA system (6.1) stemming from $x_0 \in \mathbb{X}_0$ satisfies the condition $x_k \in \mathbb{X}$ $\forall k \in \mathbb{N}_+$.*

In order to analyze the stability of the origin, we exploit a particular class of Lyapunov functions with the structure:

$$V(x) = x^T P_i(x) x, \quad \forall x \in \mathcal{X}_i \tag{6.4}$$

where

$$x^T P_i(x) x > 0, \quad \forall x \in \mathcal{X}_i \backslash \{0\}, \ \forall i \in \mathcal{I} \tag{6.5}$$

and $\forall i \in \mathcal{I}$,

$$\sup_{x \in \mathcal{X}_i} |\lambda_{max}(P_i(x))| < +\infty$$
$$\sup_{x \in \mathcal{X}_i} |\lambda_{min}(P_i(x))| < +\infty$$

where $P_i(x) = P_i(x)^T$ and $\lambda_{max}(P)$ and $\lambda_{min}(P)$ denote the largest and the smallest eigenvalues, respectively, of a real and symmetric matrix P. Note that (6.4) does not imply that the matrices $P_i(x)$ are positive definite because the inequality is required to hold only for the points in the i-th region. Moreover the assumption on the maximum and minimum eigenvalues avoids that $|x_k^T P_i(x_k) x_k|$ goes to infinity for some sequence $x_k \in \mathcal{X}_i$, $k = 0, \dots, +\infty$ that converges to a limit point belonging to the boundary of $\mathcal{X}_i$. Obviously, in order to guarantee these requirements on the eigenvalues, possible choices of matrices $P_i(x)$ are

1. continuous functions of x on $\mathcal{X}_i$ if the cell $\mathcal{X}_i$ is bounded;

2. constant matrices if the cell $\mathcal{X}_i$ is unbounded.

Note that the functions $V(x)$ we consider can be discontinuous across the cell boundaries. More precisely, as stated in Theorem 6.1, we do not need to require the continuity of $V(x)$ on the whole state-space $\mathbb{X}$ to prove the stability, as long as the number of cells is finite. For this reason we introduce the following assumption.

Assumption 6.2 *The cardinality $|\mathcal{I}|$ of the set $\mathcal{I}$ is finite.*

For stability, we can prove the following result.

Theorem 6.1 *Let Assumption 6.2 hold. Let $\mathbb{X}_0$ be a set of initial states satisfying Assumption 6.1. The equilibrium $x = 0$ of (6.1) is exponentially stable on $\mathbb{X}_0$ if there exists a function $V(x)$ as in (6.4) possessing a negative forward difference $\Delta V(x_{k+1}, x_k) = V(x_{k+1}) - V(x_k)$:*

$$\Delta V(x_{k+1}, x_k) = x_{k+1}^T P_i(x_{k+1}) x_{k+1} - x_k^T P_j(x_k) x_k < 0, \quad x_k \in \mathcal{X}_j \backslash \{0\},\ x_{k+1} \in \mathcal{X}_i. \quad (6.6)$$

Proof. In view of the assumption that the minimal and maximal eigenvalues of each matrix $P_i(x)$ are bounded, $\forall x \in \mathcal{X}_i$, and the assumption on the cardinality of the set $\mathcal{I}$, there exist constants $\alpha, \beta > 0$ such that $\alpha I < P_i(x) < \beta I,\ \forall x \in \mathcal{X}_i,\ \forall i \in \mathcal{I}$. Moreover, by recalling (6.6), there exists $\gamma > 0$ arbitrarily small such that:

$$\begin{aligned} \Delta V(x_{k+1}, x_k) &= V(x_{k+1}) - V(x_k) \\ &\leq -\gamma x_k^T x_k \\ &\leq -\frac{\gamma}{\beta} x_k^T P_i(x_k) x_k \\ &= -\frac{\gamma}{\beta} V(x_k) \end{aligned} \quad (6.7)$$

where i is the index for which the condition $x(k) \in \mathcal{X}_i$ is satisfied.
Since the coefficient γ can be chosen arbitrarily small, we can select it such that $0 < 1 - \frac{\gamma}{\beta} < 1$. On the other hand, from (6.7) we can deduce that:

$$V(x_k) \leq \left(1 - \frac{\gamma}{\beta}\right)^k V(x_0),\ \forall k \in \mathbb{N}_+ \quad (6.8)$$

and consequently that

$$\|x_k\|^2 \leq \frac{\beta}{\alpha} \left(1 - \frac{\gamma}{\beta}\right)^k \|x_0\|^2,\ \forall k \in \mathbb{N}_+. \quad (6.9)$$

□

Remark 1 *Exponential stability*

Note that if, for a fixed $0 < \gamma \leq 1$, we can substitute the condition (6.6) with

$$V(x_{k+1}) - \gamma V(x_k) = x_{k+1}^T P_i(x_{k+1}) x_{k+1} - \gamma x_k^T P_j(x_k) x_k < 0, \quad x_k \in \mathcal{X}_j \backslash \{0\},\ x_{k+1} \in \mathcal{X}_i \tag{6.10}$$

then, by following the same line of reasoning used to prove Theorem 6.1, it is possible to state that there exists a coefficient $\bar{\gamma}$, $0 < \bar{\gamma} < \gamma$ such that

$$\forall k \in \mathbb{N}_+, \quad \|x_k\|^2 < K \bar{\gamma}^k \|x_0\|^2 \tag{6.11}$$

where K is a suitable positive coefficient. Therefore, γ represents a strict upper bound on the degree of exponential stability. □

6.3 Linear Matrix Inequalities Algorithms for Exponential Stability Analysis

In this section we investigate numerical procedures to check the stability conditions (6.5) and (6.6). We first impose a convenient structure to the Lyapunov function used in (6.5) and (6.6). More precisely, we will adopt one of the following alternatives:

- Q-stability in Section 6.3.1
- PWQ-stability in Section 6.3.2
- Stability with parameterized Lyapunov functions in Section 6.3.3

6.3.1 Q-stability

Here, the choice of $P_i(x)$ is:

$$P_i(x) = P \qquad \forall x \in \mathbb{X}, \forall i \in \mathcal{I} \tag{6.12}$$

These conditions lead to the so-called Quadratic Lyapunov stability (Q-stability) widely studied in the past for systems subject to uncertainties described by polytopic affine inclusions (see e.g. (Boyd et al., 1994)). The Lyapunov function reads

$$V(x) = x^T P x \tag{6.13}$$

In (Kantner, 1997) it is recalled that sufficient conditions on P for asymptotic stability are given by the LMIs:

$$P > 0 \tag{6.14}$$

$$A_i^T P A_i - P < 0 \quad \forall i \in \mathcal{I} \tag{6.15}$$

If such a P exists, the function (6.13) is called a common Lyapunov function for the matrices $\{A_1, \ldots, A_s\}$. Narendra and Balakrishnan (1994) show that a sufficient condition for the existence of a quadratic Lyapunov function satisfying (6.14) and (6.15), is that all the matrices A_i, $i \in \mathcal{I}$ commute pairwise.

With the exception of linear time-invariant systems, there are few necessary and sufficient conditions for stability based on Lyapunov arguments. One of the first results of this kind for piecewise linear systems is given in (Shorten and Narendra, 1999). The result is however very restrictive, since it applies only to systems of second order having $m = |I| = 2$. There it is proven that a necessary and sufficient condition for the existence of a quadratic Lyapunov function for a piecewise linear system of order two with two regions is that the pencils $\sigma[A_1, A_2]$ and $\sigma[A_1, A_2^{-1}]$ are both Hurwitz. A pencil is defined as:

$$\sigma[A_1, A_2] = \{A : A = \alpha A_1 + (1-\alpha)A_2, \quad \text{with} \quad \alpha \in [0,1]\} \tag{6.16}$$

A pencil is Hurwitz, if all matrices in it have their eigenvalues in $\mathbb{C}^-$. For switching systems of second order with 2 dynamics, necessary and sufficient conditions are given by Xu and Antsaklis (1999) using geometric properties of vector field in the plane $\mathbb{R}^2$. The generalization to higher dimensions of these criteria is not immediate because of the different topological structure of $\mathbb{R}^n$ with $n > 2$.

State Feedback Synthesis

We consider the synthesis of a piecewise linear state feedback

$$u_k = K_i x_k \qquad \forall x_k \in \mathcal{X}_i. \tag{6.17}$$

for the piecewise linear system with input

$$x_{k+1} = A_i x_k + B_i u_k, \qquad \text{for } x_k \in \mathcal{X}_i \tag{6.18}$$

that stabilizes the origin by means of a quadratic Lyapunov function (6.13). In other words we look for the matrices P and K_i, $\forall i \in \mathcal{I}$ that satisfy the conditions

$$P > 0 \tag{6.19}$$

$$(A_i + B_i K_i)^T P (A_i + B_i K_i) - P < 0 \quad \forall i \in \mathcal{I} \tag{6.20}$$

Since both variables P and K_i are unknown, the matrix inequality (6.20) is apparently nonlinear. However, as for LTI discrete-time systems, it can be rewritten as an LMI, using Schur complements, see Appendix C.3. By using the equivalence between (C.6) and (C.8), inequality (6.15) can be rewritten as:

$$P - A_i^T P A_i > 0 \;\Leftrightarrow\; P^{-1} - A_i P^{-1} A_i^T > 0 \tag{6.21}$$

Instead of considering inequalities (6.19) and (6.20) for asymptotic stability we can consider the conditions

$$Q - (A_i + B_i K_i) Q (A_i + B_i K_i)^T > 0 \tag{6.22}$$

$$Q > 0 \tag{6.23}$$

with $Q = P^{-1}$. Since Q is symmetric and positive definite, these (nonlinear) matrix inequalities are equivalent to

$$\begin{pmatrix} Q & (A_i Q + B_i K_i Q) \\ (A_i Q + B_i K_i Q)^T & Q \end{pmatrix} > 0 \tag{6.24}$$

where we used the equivalence between (C.6) and (C.4). Using again the fact that Q is positive definite, we can introduce new variables Y_i as

$$Y_i = K_i Q \tag{6.25}$$

and solve for Y_i instead of K_i. The resulting LMI is

$$\begin{pmatrix} Q & (A_i Q + B_i Y_i) \\ (A_i Q + B_i Y_i)^T & Q \end{pmatrix} > 0, \tag{6.26}$$

and the state feedback vector can be recovered as

$$K_i = Y_i Q^{-1} \tag{6.27}$$

To what concerns the region of attraction of the state-feedback derived so far, let $\bar{V}_1(x) = x^T P x$ and choose $\mathbb{X}_0$ as the largest level set $\{x \in \mathbb{X} : \bar{V}_1(x) \leq \xi, \xi > 0\}$ contained in $\mathbb{X}$. Then every state-trajectory of the controlled system starting from $\mathbb{X}_0$, besides converging to the origin, does not leave the set $\mathbb{X}$. In summary, we have proved the following result

Lemma 6.1 *Let Q and Y_i be the solutions of the LMIs (6.23), (6.26) and K_i be computed as in (6.27). Then, the piecewise linear state feedback $u_k = K_i x_k$, $\forall x_k \in \mathcal{X}_i$ stabilizes asymptotically the origin of (6.18) on $\mathbb{X}_0$.*

Remark 2 Note that the stability of the linear system

$$x_{k+1} = A_i x_k \tag{6.28}$$

is equivalent to the stability of the dual system

$$x_{k+1} = A_i^T x_k \tag{6.29}$$

in the following sense: If $x^T P x$ is a Lyapunov function for (6.28), then $x^T P^{-1} x$ is a Lyapunov function for (6.29). This can be seen with the rules in Theorem C.1. Choose for $P > 0$ for instance $R = P$, $S = A$ and $Q = P^{-1}$ in Equation (C.8). Because of the equivalence to Equation (C.6), we obtain Equation (6.21). □

Bounded Inputs

The synthesis of the state feedback controller did not take into account any constraints on the control action so far. If u is constrained, the values of K_i can possibly result in input values violating the bounds present in (6.18). According to (Boyd et al., 1994, p. 103) we can add further LMI's that take into account these constraints. Assuming that the initial condition x_0 satisfies $x_0^T Q^{-1} x_0 \leq 1$, where Q is the same unknown appearing in Lemma 6.1, we can enforce the constraint

$$\max_{k\geq 0} \parallel u_k \parallel \leq \mu \tag{6.30}$$

with the LMIs

$$\begin{bmatrix} 1 & x_0^T \\ x_0 & Q \end{bmatrix} \geq 0 \tag{6.31}$$

$$\begin{bmatrix} Q & Y_i^T \\ Y_i & \mu^2 I \end{bmatrix} \geq 0 \quad \forall i \in \mathcal{I} \tag{6.32}$$

Note that LMI (6.32) is more restrictive than condition (6.30), since it holds for those states as well that are outside the region, where the i-th controller is active. The matrix Q is chosen such that the set $\{x : x^T Q^{-1} x \leq 1\}$ is invariant.

6.3.2 PWQ-stability

The choice of $P_i(x)$ is:

$$P_i(x) = P_i \quad \forall x \in \mathbb{X} \tag{6.33}$$

This class of Lyapunov functions, which leads to the so-called piecewise quadratic (PWQ) stability, has been studied for continuous-time PWA systems in (Johansson and Rantzer, 1998).

A piecewise quadratic Lyapunov function for a PWA system can be defined as

$$V(x) = x^T P_i x \quad \forall x \in \mathcal{X}_i \tag{6.34}$$

We do not have to require continuity of $V(x)$ to prove stability, as long as the number of cells is finite (Mignone et al., 2000b). For stability, it has to hold that $V(x)$ is positive-definite in a neighborhood of the origin and that Equation (6.6) holds. Assuming that $x_{k+1} \in X_i$ and $x_k \in X_j$, we have

$$\begin{aligned} \Delta V(x_{k+1}, x_k) &= x_{k+1}^T P_i x_{k+1} - x_k^T P_j x_k \\ &= x_k^T \left(A_j^T P_i A_j - P_j \right) x_k \end{aligned} \tag{6.35}$$

The LMIs to satisfy in this case are

$$A_j^T P_i A_j - P_j < 0 \quad \forall (j,i) \in \mathcal{W}_{all} \tag{6.36}$$

$$P_i > 0, \quad \forall i \in \mathcal{I} \tag{6.37}$$

where $\mathcal{W}_{all} = \mathcal{I} \times \mathcal{I}$. When the LMIs (6.36)-(6.37) are feasible, we term the PWA system $\mathcal{W}_{all}$-PWQ stable.

Switching Regions

The main difficulty, compared to the continuous-time case, is that we have to satisfy the LMIs (6.36) for all the pairs (i, j) because in principle the state may switch in one step between non adjacent cells. Without further analysis of the system we have to take into account all possible switches $\mathcal{W}_{all}$ from each state space region to each other region. Therefore the number of possible switches grows quadratically with the number of cells. The conservativeness introduced by this approach can be relaxed because usually not all the transitions between cells $\mathcal{X}_i$ and $\mathcal{X}_j$ are allowed. Let $\mathcal{W}$ be the set of all ordered pairs (j, i) of indices, denoting the possible switches from cell j to cell i:

$$\mathcal{W} = \{(j,i) : j, i \in \mathcal{I} \text{ and } \exists k \in \mathbb{N}_0, \text{ such that } x_k \in \mathcal{X}_i \text{ and } x_{k-1} \in \mathcal{X}_j\} \tag{6.38}$$

The set $\mathcal{W}$ can be determined via reachability analysis for MLD systems (Bemporad, Ferrari-Trecate and Morari, 2000; Bemporad and Morari, 1999b). Since for exponential stability it is enough that (6.36)-(6.37) holds for $(j,i) \in \mathcal{W}$, we have the following result.

Theorem 6.2 *The origin of the PWA (6.1) is exponentially stable on* $\mathbb{X}$ *if there exist s matrices* P_i*, such that the following LMIs are satisfied:*

$$P_i > 0 \qquad \forall i \in \mathcal{I} \tag{6.39}$$

$$A_j^T P_i A_j - P_j < 0 \qquad \forall (j,i) \in \mathcal{W} \tag{6.40}$$

In this case the PWA system is termed $\mathcal{W}$*-PWQ stable*. In $\mathcal{W}$ there are in general pairs of the form (i,i) for each cell that is not left in one step. For these cells Equation (6.40) states that A_i must be stable. This means that we cannot show stability of a stable piecewise linear system, whose components are unstable, with a piecewise quadratic Lyapunov function (6.34), if the system stays in the same unstable cell for more than one time step.

The set $\mathcal{W}$ represents all possible switches in a single time-step and an easy procedure for finding a set of switches $\bar{\mathcal{W}} \supseteq \mathcal{W}$ is the following. First, note that it is possible to represent each closed polyhedron $\mathrm{Cl}(\mathcal{X}_i)$ as

$$\mathcal{X}_i = \left\{ x : \bar{E}_i \begin{bmatrix} x \\ 1 \end{bmatrix} = [E_i \; e_i] \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \right\} \tag{6.41}$$

where E_i and e_i are suitable matrices and the inequality holds componentwise. Then, the pair (i,j) is included in $\bar{\mathcal{W}}$ if there is at least one point $x \in \mathcal{X}$ satisfying

$$E_i A_j x \leq e_i - E_i a_j \tag{6.42}$$

$$E_j x \leq e_j \tag{6.43}$$

In fact (6.43) represents the condition $x \in \mathrm{Cl}(\mathcal{X}_j)$ and (6.42) the condition $A_j x + a_j \in \mathrm{Cl}(\mathcal{X}_i)$. Note that $\bar{\mathcal{W}}$ does not coincide with $\mathcal{W}$ for the only reason that the regions $\mathcal{X}_i$ are replaced with their closures. However, this allows having non strict inequalities in the

feasibility test (6.42)-(6.43) that can be implemented by using standard linear programming solvers. The computation of $\bar{\mathcal{W}}$ requires $s^2 \cdot \mathrm{lp}(n, n_c)$, where $\mathrm{lp}(n, n_c)$ is the complexity of solving a linear program in n unknowns and n_c constraints (Goodman and O'Rourke, 1997). In practice we found that the computational burden for the determination of $\bar{\mathcal{W}}$ is usually much less than the one needed for solving the LMIs (6.66)-(6.67). Moreover this complexity could be reduced by computing $\bar{\mathcal{W}}$ in a different way, namely by resorting to the equivalence between PWA and MLD systems and by exploiting switching detection procedures available for MLD systems and based on mixed integer linear programming, see (Bemporad, Ferrari-Trecate and Morari, 2000). Finally, a more trivial possibility is to resort to LMI constraints of type (6.64)-(6.65) defined for each couple (i, j) contained in the set $\mathcal{W}_{all} = \mathcal{I} \times \mathcal{I}$. In this way we take into account all conceivable switches of a PWA system. Clearly, the stability test obtained by exploiting $\mathcal{W}_{all}$ instead of $\mathcal{W}$ or $\bar{\mathcal{W}}$ can be more conservative since it can comprise LMI constraints that are unnecessary.

In order to distinguish between these types of stability analysis, we will refer to as $\mathcal{W}$-PWQ stability, the stability condition associated with the set $\mathcal{W}$. The stability condition obtained for the set $\mathcal{W}_{all}$ will be referred to as $\mathcal{W}_{all}$-PWQ stability. If the additional condition $P_i = P, \forall i \in \mathcal{I}$ is imposed, it is not necessary to distinguish between the stability test associated with the set $\mathcal{W}$ and the stability test associated to the set $\mathcal{W}_{all}$. This explains the usage of the term Q-stability to refer to the corresponding stability condition.

Systems with Switchings in Both Directions

There are PWA systems where the set $\mathcal{W}$ contains both pairs of indices (i, j) and (j, i). This occurs for instance if the system trajectories point to the boundary of a state space cell, allowing the system to change from cell i to the cell j as well as from cell j to cell i. We define $\mathcal{W}_2$ as the set of such indices:

$$\mathcal{W}_2 = \{(i, j) \in \mathcal{W} : (j, i) \in \mathcal{W} \text{ and } i \neq j\} \tag{6.44}$$

We can state a necessary condition for the existence of a PWQ Lyapunov function (6.34) for systems having a nonempty set $\mathcal{W}_2$.

Theorem 6.3 *Assume that for a PWA system (6.1), the set $\mathcal{W}_2$ is nonempty and contains $2N$ elements. If a PWQ Lyapunov function (6.34) satisfying (6.39)-(6.40) exists, then the $2N$ matrices $A_{i_1}A_{j_1}$, $A_{j_1}A_{i_1}$, ..., $A_{i_N}A_{j_N}$, $A_{j_N}A_{i_N}$ must have all eigenvalues inside the unit circle.*

Proof. Given a system where boundary crossings can occur in both directions for cell i and j, the existence of a PWQ Lyapunov function implies the fulfillment of the LMI's

$$\begin{aligned} A_j^T P_i A_j - P_j &< 0 \\ A_i^T P_j A_i - P_i &< 0 \end{aligned}$$

Therefore,

$$P_j > \; A_j^T P_i A_j \geq \; A_j^T A_i^T P_j A_i A_j$$

where we made use of the fact that for matrices of appropriate dimensions, it holds that

$$C > D \;\; \rightarrow \;\; B^T C B \geq B^T D B$$

Analogously,

$$P_i \;\; > \;\; A_i^T A_j^T P_i A_j A_i$$

This means that both $A_i A_j$ and $A_j A_i$ must be stable matrices. These considerations apply to each pair $(i, j) \in S_2$ separately. Therefore the statement of the Lemma follows. □

Note that for square matrices A, B of equal dimensions it holds that AB and BA have the same eigenvalues, see eg. (Kailath, 1980). To verify the nonexistence of a piecewise quadratic Lyapunov function with Theorem 6.3, we therefore only have to check the eigenvalues of at most N matrices.

We can generalize the result of Lemma 6.3 to systems that can cycle across an arbitrary number of cells:

$$j_1 \to j_2 \to j_3 \to \ldots \to j_n \to j_1 \tag{6.45}$$

In this case the existence of a PWQ Lyapunov function implies the stability of all products of matrices denoting the cells, the system can cycle through. Define the set of ordered n-tuples of indices

$$\mathcal{W}_n = \{(j_1, \ldots, j_n) \in \mathcal{I}^n : \{(j_1, j_2), \ldots, (j_{n-1}, j_n), (j_n, j_1)\} \subset \mathcal{W} \text{ and } j_i \text{ all different}\} \tag{6.46}$$

and define the set of products of matrices

$$\tilde{A} = \cup_{\ell=2}^{\infty} \{A : A = \prod_{\tau=1}^{\ell} A_{i_\tau}, (i_1, \ldots, i_\ell) \in \mathcal{W}_\ell\} \tag{6.47}$$

We have the following corollary:

Corollary 6.1 *Assume that for a PWA system (6.1), the sets $\mathcal{W}_n$ $(n = 2 \ldots \infty)$ are not all empty. If a PWQ Lyapunov function (6.34) exists, then all matrices in $\tilde{A}$ must have all eigenvalues inside the unit circle.*

State Feedback Synthesis

The same rationale used in Section 6.3.1 applies also to the case of a piecewise quadratic function. We can formally exploit the equivalence between Equations (C.8) and (C.9) as

well as the equivalence between Equations (C.4) and (C.6), yielding

$$P_j - (A_j + B_j K_j)^T P_i (A_j + B_j K_j) > 0 \tag{6.48}$$

$$\Leftrightarrow \quad P_i^{-1} - (A_j + B_j K_j) P_j^{-1} (A_j + B_j K_j)^T > 0 \tag{6.49}$$

$$\Leftrightarrow \quad P_i^{-1} - (A_j + B_j K_j) P_j^{-1} P_j P_j^{-1} (A_j + B_j K_j)^T > 0 \tag{6.50}$$

$$\Leftrightarrow \quad P_i^{-1} - (A_j P_j^{-1} + B_j K_j P_j^{-1}) P_j (A_j P_j^{-1} + B_j K_j P_j^{-1})^T > 0 \tag{6.51}$$

$$\Leftrightarrow \quad \begin{pmatrix} P_i^{-1} & (A_j P_j^{-1} + B_j K_j P_j^{-1}) \\ (A_j P_j^{-1} + B_j K_j P_j^{-1})^T & P_j^{-1} \end{pmatrix} > 0 \tag{6.52}$$

To obtain an LMI, we substitute

$$W_j = K_j P_j^{-1} \quad \text{or} \quad K_j = W_j P_j \tag{6.53}$$

which gives

$$\begin{pmatrix} P_i^{-1} & (A_j P_j^{-1} + B_j W_j) \\ (A_j P_j^{-1} + B_j W_j)^T & P_j^{-1} \end{pmatrix} > 0 \quad (\forall (j,i) \in \bar{\mathcal{W}}), \tag{6.54}$$

where the set $\bar{\mathcal{W}}$ will be defined next. We still have to impose the condition

$$Q_i = P_i^{-1} > 0, \quad \forall i \in \mathcal{I} \tag{6.55}$$

The LMIs (6.54), (6.55) provide the synthesis of a piecewise linear controller that stabilizes the origin, as can be shown with the PWQ Lyapunov function (6.34).

As for stability analysis, the LMI (6.54) must be fulfilled for all possible pairs (j,i) corresponding to regions, the system can switch to in one step. However, the switching must be predicted in *closed loop*. Lacking of better knowledge we can always choose $\bar{\mathcal{W}} = \mathcal{W}_{all} = \mathcal{I} \times \mathcal{I}$. This takes into account all conceivable pairs of cells in a system with s cells, introducing conservativeness of the approach.

To what concerns the region of attraction of the controlled system, let $\bar{V}_2(x) = x^T P_i x$, $\forall x \in \mathcal{X}_i$ and choose $\mathbb{X}_0$ as the largest level set $\{x \in \mathbb{X} : \bar{V}_2(x) \leq \xi, \xi > 0\}$ contained in $\mathbb{X}$. Then, even if $\bar{V}_2$ is only piecewise continuous, it is easy to prove that every state-trajectory

of the controlled system starting from $\mathbb{X}_0$, besides converging to the origin, does not leave the set $\mathbb{X}$. This is due to the fact that the number of regions $\mathcal{X}_i$ is finite. In summary, we have the following Lemma:

Lemma 6.2 *If $\bar{\mathcal{W}}$ contains all possible pairs of indices corresponding to the cells the closed-loop PWA system can switch in one step, a stable piecewise linear state feedback that stabilizes asymptotically the origin on $\mathbb{X}_0$ can be found solving the LMIs (6.54) and (6.55) for Q and Y_i. K_i is then given by Equation (6.53).*

6.3.3 Stability with Parameterized Lyapunov Functions

The choice of $P_i(x)$ is:

$$P_i(x) = \sum_{j=0}^{N} P_{i(j)} \rho_{i(j)}(x) \tag{6.56}$$

where

$$\rho_{i(j)} : \mathcal{X}_i \longrightarrow \mathbb{R}, \quad j = 1, 2, \ldots, N \tag{6.57}$$

are (bounded) basis-functions (Johansen, 2000) for $P_i(x)$ and $P_{i(1)}, P_{i(2)}, \ldots, P_{i(N)}$ are parameter matrices. A similar class of Lyapunov functions has been proposed for continuous-time nonlinear systems in (Johansen, 2000).

6.3.4 Comparison

The stability tests that one obtains in these three cases decrease in their degree of conservativeness. We point out that all these tests can be translated into an LMI form (Scherer et al., 1997; Gahinet et al., 1994) by replacing (6.5) and (6.6) with the following more

conservative conditions:

$$P_i(x) > 0, \quad \forall x \in \mathcal{X}_i \tag{6.58}$$

$$(A_j x_k + a_j)^T P_i (A_j x_k + a_j)(A_j x_k + a_j) - \gamma x_k^T P_j(x_k) x_k < 0, \tag{6.59}$$

$$x_k \in \mathcal{X}_j \backslash \{0\}, \quad x_{k+1} = A_j x_k + a_j \in \mathcal{X}_i$$

where $0 < \gamma \leq 1$ represents a strict upper bound on the minimal degree of exponential stability. As we have outlined in the previous sections, inequalities (6.58)-(6.59) are LMI conditions in all the three cases $i) - iii)$. Unfortunately, in general and even when the state-space $\mathbb{X}$ is bounded one has to deal with an infinite-dimensional LMI problem or, in other words, it is necessary to deal with a Parameterized LMI (PMI) problem that can be reduced to a finite-dimensional LMI problem by resorting to various techniques (Tuan et al., 2001; Gahinet et al., 1996). One of these techniques is called grid method of the parameter space and can be straightforwardly adapted to our case. An alternative approach, which is particularly advantageous from a computational point of view in the case $iii)$, is based on the so-called multiconvexity concepts (Gahinet et al., 1996) and can be developed starting from Equations (6.58)-(6.59) by taking advantage of the boundedness of the basis functions $\rho_{i(j)}(x)$.

Example 6.1:

In order to analyze the conservativeness of the stability tests corresponding to the cases $i) - iii)$, we consider the following numerical example:

$$x_{k+1} = \begin{cases} \begin{bmatrix} -0.999 & 0 \\ -0.139 & 0.341 \end{bmatrix} x_k & x_k \in (0,1] \times [-1,0) \\ \begin{bmatrix} 0.436 & 0.323 \\ 0.388 & -0.049 \end{bmatrix} x_k & x_k \in [0,1] \times [0,1] \\ \begin{bmatrix} -0.457 & 0.215 \\ 0.491 & 0.49 \end{bmatrix} x_k & x_k \in [-1,0] \times [-1,0] \backslash \{0\} \\ \begin{bmatrix} -0.022 & 0.344 \\ 0.458 & 0.271 \end{bmatrix} x_k & x_k \in [-1,0) \times (0,1] \end{cases} \tag{6.60}$$

The regions $\{\mathcal{X}_i\}_{i=1}^4$ partition in four quadrants the state-space $\mathbb{X} = [-1,1] \times [-1,1]$. The partitioning corresponds to the four quadrants of the state-space

$$\mathbb{X} = \left\{ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \mid x_1 \in [-1,1], x_2 \in [-1,1] \right\}$$

i.e.

$$E_1 = E_2 = E_3 = E_4 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}^T \tag{6.61}$$

and

$$\begin{aligned} e_1 &= \begin{bmatrix} -1 & 0 & 0 & -1 \end{bmatrix}^T, \quad e_2 = \begin{bmatrix} 0 & -1 & 0 & -1 \end{bmatrix}^T \\ e_3 &= \begin{bmatrix} -1 & 0 & -1 & 0 \end{bmatrix}^T, \quad e_4 = \begin{bmatrix} 0 & -1 & -1 & 0 \end{bmatrix}^T. \end{aligned}$$

It is possible to check that if $x_0 \in \mathbb{X}$ then $x_k \in \mathbb{X}, \forall k \in \mathbb{N}_+$. We have applied the stability tests corresponding to the three cases $i) - iii)$ for different values of the coefficient γ. In particular, according to Johansen (2000), the case $iii)$ has been tackled by means of the following basis-functions:

$$\rho_{i(j)}(x) = \frac{\mu_{i(j)}(x)}{\sum_{k=1}^4 \mu_{i(k)}(x)} \in [0,1] \tag{6.62}$$

where $\mu_{i(j)}(x) = exp\left(-0.5(x - x_{i(j)})\right)$ and $x_{i(j)}$ (with $i,j = 1,2,3,4$) are the vertices of each cell $\mathcal{X}_i$. The corresponding LMIs (6.58)-(6.59) were solved by exploiting gridding techniques (Tuan et al., 2001; Gahinet et al., 1996). The test corresponding to the case $i)$ is feasible for $\gamma \geq 0.72$, the test corresponding to the case $ii)$ is feasible for $\gamma \geq 0.35$ and, finally, the test corresponding to the case $iii)$ is feasible for $\gamma \geq 0.14$. This simple experiment highlights that the last stability test is the least conservative and can be used to obtain better information about the true minimal degree of exponential stability of the considered system. □

6.3.5 Relaxation of Finite-Dimensional LMI Tests for Exponential Stability

In this subsection we concentrate on the tests associated with Lyapunov functions of type (6.13) and (6.34). In fact, for these cases, it is possible to obtain stability tests that do not require gridding and that are computationally less expensive. For the sake of simplicity, we first assume that:

$$a_i = 0, \quad \forall i \in \mathcal{I} \tag{6.63}$$

and later we will show how to remove this assumption.

It is easy to verify that conditions (6.58)-(6.59) are satisfied, for a given γ ($0 < \gamma \leq 1$), if the following alternative inequalities are met:

$$P_i > 0, \quad \forall i \in \mathcal{I} \tag{6.64}$$

$$A_j^T P_i A_j - \gamma P_j < 0, \quad \forall (j,i) \in \mathcal{W} \tag{6.65}$$

where $\mathcal{W}$ is defined as in (6.38). The LMI test represented by inequalities (6.64)-(6.65) corresponds to the case (6.34) when the matrices P_i are allowed to be different (PWQ stability), whereas the case (6.13) corresponds to the more demanding condition $P_i = P,\ \forall i \in \mathcal{I}$ (Q-stability).

Following the rationale of Johansson and Rantzer (1998) we note that the LMIs (6.64)-(6.65) are actually valid on the whole state space, even though they would be required to hold in a single cell only *i.e.* for $x \in \mathcal{X}_i$. We can remove some conservativeness by deriving from (6.5) and (6.6) some alternative conditions exploiting the so-called S-Procedure (see *e.g.* (Jakubovic, 1977; Iwasaki et al., 2000)). More precisely, it is possible to reduce

conservativeness if we can find matrices F_i and G_{ij} such that

$$\begin{aligned} P_i - F_i &> 0, \quad \forall i \in \mathcal{I} && (6.66)\\ A_j^T P_i A_j - \gamma P_j + G_{ij} &< 0, \quad \forall (j,i) \in \mathcal{W} && (6.67)\\ x^T F_j x &\geq 0, \quad x^T G_{ij} x \geq 0 \quad \text{if} \quad x \in \mathcal{X}_j. && (6.68) \end{aligned}$$

Note that we do not require the matrices F_j and G_{ij} to be positive or negative definite, we only require that the quadratic forms take on the signs mentioned above in the corresponding regions. From the assumptions on the matrices F_j and G_{ij} it follows that the fulfillment of the LMIs (6.66)-(6.67) implies the fulfillment of (6.58)-(6.59). An effective procedure to find suitable matrices F_j and G_{ij} meeting the requirement (6.68) can be provided by exploiting the representation (6.41) of the cells $\mathrm{Cl}(\mathcal{X}_i)$ in form of inequalities. Here, we assume

$$e_i = 0, \quad \forall i \in \mathcal{I}. \tag{6.69}$$

This condition will be removed together with the assumption (6.63) on the vectors a_i. By exploiting (6.69), as shown in (Johansson and Rantzer, 1998), a possible choice of the matrices F_j and G_{ij} is given by:

$$\begin{aligned} F_j &= E_j^T U_j E_j && (6.70)\\ G_{ij} &= E_j^T Z_{ij} E_j && (6.71) \end{aligned}$$

where U_j and Z_{ij} are matrices of suitable dimensions with non-negative entries.

Remark 3 Removing the assumptions on the vectors a_i and e_i

In this remark we provide a simple procedure to remove the assumptions on the vectors a_i and e_i (Equations (6.63) and (6.69), respectively). We introduce the extended state $\bar{x}_k = [\ x_k^T \quad 1\ ]^T$ and rewrite the original system (6.1) as follows:

$$\bar{x}_{k+1} = \bar{A}_i \bar{x}_k, \quad \text{for } x_k \in \mathcal{X}_i \tag{6.72}$$

where

$$\bar{A}_i = \begin{bmatrix} A_i & a_i \\ 0 & 1 \end{bmatrix}. \tag{6.73}$$

The system (6.72) has the same structure of the original system (6.1), i.e. without displacements. However, the matrices $\bar{A}_i$ have an eigenvalue at 1. This means that the LMIs (6.64)-(6.65), necessary for PWQ stability, are in general not satisfied. There are two possible remedies for this problem: the first is to resort to a more complex family of Lyapunov functions, as in point iii) of Section 6.3. The second is analogous to the procedure proposed in (Johansson and Rantzer, 1998). We exploit PWQ Lyapunov functions together with relaxations (6.66). □

Following the same rationale used in (Johansson and Rantzer, 1998), we can formulate the following stability test.

Lemma 6.3 *Let $\mathbb{X}_0$ be the set of initial states satisfying Assumption 6.1. If Assumption 6.2 holds and the LMIs*

$$P_i - \bar{E}_i^T \bar{U}_i \bar{E}_i > 0 \qquad \forall i \in \mathcal{I} \tag{6.74}$$

$$\bar{A}_j^T P_i \bar{A}_j - \gamma P_j + \bar{E}_j^T \bar{Z}_{ij} \bar{E}_j < 0 \qquad \forall (j,i) \in \mathcal{W}. \tag{6.75}$$

admit a solution in $P_i = P_i^T$, $\bar{U}_i$ and $\bar{Z}_{ij}$ (where $\bar{U}_i$, $\bar{Z}_{ij}$ are matrices of suitable dimensions with non-negative entries and $\bar{E}_j = [E_j \; e_j]$), then the origin of (6.1) is exponentially stable on $\mathbb{X}_0$ with a degree of stability γ.

Each system satisfying the stability test of Lemma 6.3 based on the use of S-Procedure, will be referred to as *$\mathcal{W}$-PWQ stable with relaxations* (or *$\mathcal{W}_{all}$-PWQ stable with relaxations* in case $\mathcal{W}_{all}$ is used instead of $\mathcal{W}$). As in (Johansson and Rantzer, 1998) one can impose some structure to the matrices U_i and Z_{ij}, for instance by taking them diagonal or lower triangular. This decreases the flexibility of the relaxation procedure but reduces the number of unknowns in the LMIs (6.74)-(6.75).

Example 6.2:

To illustrate the analysis and synthesis methods described in the previous sections we consider the following PWA system:

$$x_{k+1} = \begin{cases} \begin{bmatrix} -0.04 & -0.461 \\ -0.139 & 0.341 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k & E_1 x_k \geq 0 \\ \begin{bmatrix} 0.936 & 0.323 \\ 0.788 & -0.049 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k & E_2 x_k \geq 0 \\ \begin{bmatrix} -0.857 & 0.815 \\ 0.491 & 0.62 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k & E_3 x_k \geq 0 \\ \begin{bmatrix} -0.022 & 0.644 \\ 0.758 & 0.271 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k & E_4 x_k \geq 0 \end{cases} \tag{6.76}$$

The partitioning corresponds to the four quadrants of the two dimensional $x_1 - x_2$ plane, i.e.

$$E_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad E_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad E_3 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad E_4 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

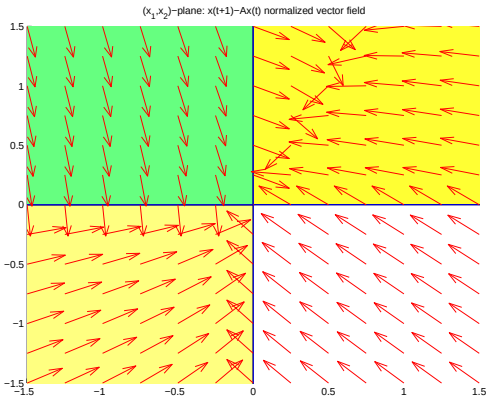
For this system the stability analysis gave conclusive results only if we were looking for a piecewise quadratic Lyapunov function with minimal set $\mathcal{W}$ and relaxations. All the other more conservative stability tests did not give any results. The vector field of the system in the two-dimensional plane can be seen in figure 6.1-a. The system trajectories starting from some nonzero initial condition are in figure 6.1-b.

The controller synthesis succeeded in the search for a piecewise linear state feedback showing stability with a quadratic Lyapunov function. The closed loop matrices are given by the expressions:

$$A_{cli} = A_i + B_i K_i^T \tag{6.77}$$

where the feedback gains are given by:

$$K_1 = \begin{bmatrix} 0.04 \\ 0.461 \end{bmatrix} \quad K_2 = \begin{bmatrix} -0.936 \\ -0.323 \end{bmatrix} \quad K_3 = \begin{bmatrix} 0.857 \\ -0.815 \end{bmatrix} \quad K_4 = \begin{bmatrix} 0.022 \\ -0.644 \end{bmatrix} \tag{6.78}$$



(a) (x_1, x_2)-plane: $x(t+1)-Ax(t)$ normalized vector field.

(b) Trajectories of the open loop and the controlled system.

Figure 6.1: Open and closed loop behaviour of system (6.76).

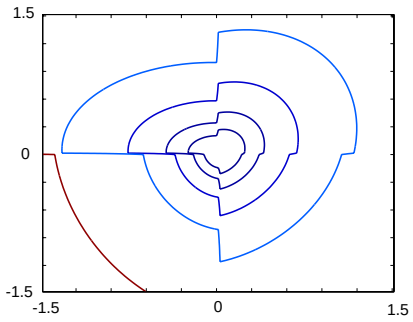
The contour-lines and a 3-D plot of the Lyapunov function can be seen in figure 6.2 whereas the state trajectories of the controlled system is depicted in figure 6.1-b.

□

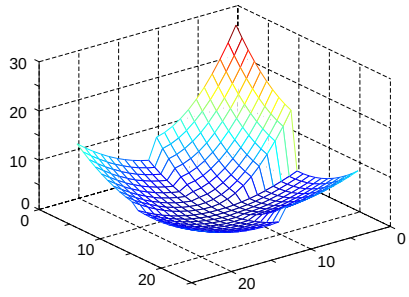
6.4 Conservativeness of the Various Stability Analysis Algorithms

The stability tests presented in this section exhibit different levels of conservativeness and complexity in their applications. For instance it is clear that Q-stability implies $\mathcal{W}_{all}$-PWQ stability of the system. The relations become more complex if one considers relaxations and minimal switching sets $\mathcal{W}$. For instance systems can be found, that are not $\mathcal{W}_{all}$-PWQ stable, but are both $\mathcal{W}_{all}$-PWQ stable with relaxations and $\mathcal{W}$-PWQ stable.

This is summarized in Figure 6.3. In general, a higher degree of conservativeness of the



(a) Contour Plot of the Lyapunov function.



(b) 3D plot of the Lyapunov Function.

Figure 6.2: Open loop Lyapunov function for system (6.76).

Lyapunov function corresponds to lower computational requirements. The classification in Figure 6.3 has been obtained by considering several examples of piecewise linear systems with two states. It is worth noticing that all the regions of Figure 6.3 are non-empty as one can prove by means of examples.

In some cases, the stability of PWA systems cannot be revealed by means of PWQ Lyapunov functions of the type (6.34) even in the discrete-time case (see e.g. (Hassibi et al., 1999)). In fact, roughly speaking, these Lyapunov functions cannot capture the path-dependence of the "stored energy" in hybrid dynamical systems. The structure of the Lyapunov function of type *iii*) is more flexible from this point of view. In fact, there exist PWA systems whose stability cannot be proved by exploiting Lyapunov functions of type *ii*) but only by resorting to those of type *iii*). Therefore, Lyapunov functions of type *iii*) exploiting relaxations and the switching set $\mathcal{W}$ embrace all the $\mathcal{W}$-PWQ stability tests.

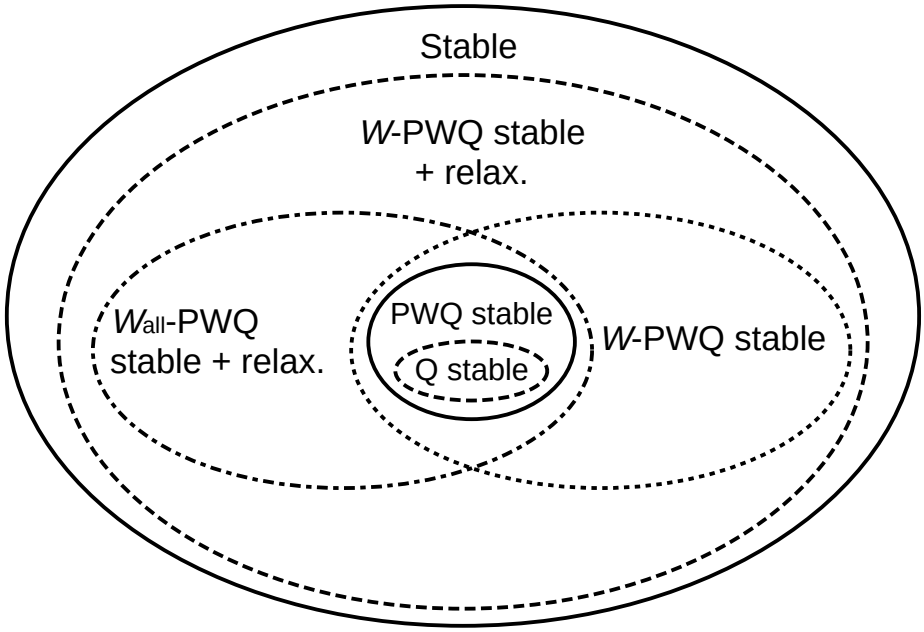


Figure 6.3: Conservativeness of the different stability analysis approaches.

6.5 Performance Analysis Techniques

In this section we propose algorithms for performance analysis of discrete-time PWA systems. More precisely, we focus on the l_2-gain of PWA systems, showing that it can be analyzed by resorting to LMI-based algorithms. The rationale for our derivation hinges on the use of passivity theory for nonlinear systems (Lin and Byrnes, 1996). The proposed LMI techniques can be adapted to consider other performance measures. An application of the H_∞ and Generalized H_2 performance analysis tests is the performance *analysis* of MPC for both linear (Bemporad, Morari, Dua and Pistikopoulos, 2000) and MLD (Bemporad, Borrelli and Morari, 2000a) systems. This can be done by exploiting the explicit PWA form of the closed-loop system.

6.5.1 l_2-gain of PWA Systems

Consider the augmented PWA system

$$\begin{aligned} x_{k+1} &= A_i x_k + B_i w_k + a_i \\ z_k &= C_i x_k + D_i w_k \end{aligned} \qquad x_k \in \mathcal{X}_i \tag{6.79}$$

where $w_k \in \mathbb{R}^r$ is a disturbance signal and $z_k \in \mathbb{R}^s$ is a penalty output that can model, for instance, tracking errors. The cells $\mathcal{X}_i$ define a polyhedral partition of the state-space $\mathbb{X}$. Assumption 6.1 has to be modified as follows, in order to take into account the effects of the disturbance signal w_k.

Assumption 6.3 *The evolution of the state trajectory for the PWA system (6.79) starting form $x_0 = 0$ and due to a disturbance signal $w_k, k \in \mathbb{N}_+$ satisfies the condition $x_k \in \mathbb{X}$ $\forall k \in \mathbb{N}_+$.*

Analogously to the stability analysis, we first assume $a_i = 0, \forall i \in \mathcal{I}$ and then we remove this assumption by resorting to the procedure described in Remark 3

In this subsection we focus on the disturbance attenuation problem in an l_2 framework: given a real number $\gamma > 0$, the exogenous signal w is attenuated by γ if, starting from $x_0 = 0$, for each integer $N \geq 0$ and for every $w \in l_2([0, N], \mathbb{R}^r)$

$$\sum_{k=0}^{N} \|z_k\|^2 < \gamma^2 \sum_{k=0}^{N} \|w_k\|^2. \tag{6.80}$$

A discrete-time nonlinear system is *strictly dissipative* with supply rate $W : \mathbb{R}^s \times \mathbb{R}^r \to \mathbb{R}$ (Byrnes and Lin, 1995) if there exists a non-negative function $V : \mathbb{R}^n \to \mathbb{R}$ termed *storage function* such that

$$\forall w_k \in \mathbb{R}^r, \forall k \geq 0, \quad V(x_{k+1}) - V(x_k) < W(z_k, w_k) \tag{6.81}$$

and $V(0) = 0$. Condition (6.81), is the so-called *dissipation inequality* that can be equivalently represented through the condition (Lin and Byrnes, 1996; Willems, 1972):

$$\forall w_k, \forall N \geq 0, \forall x_0 \in \mathbb{X}, \quad V(x_{N+1}) - V(x_0) < \sum_{k=0}^{N} W(z_k, w_k). \tag{6.82}$$

Hereafter, we concentrate on finite gain dissipative PWA systems with the following supply rate

$$W_\infty(z, w) \;=\; \gamma^2 \|w\|^2 - \|z\|^2, \quad \gamma > 0 \tag{6.83}$$

In fact, $W_\infty(z, w)$ is related to the l_2-gain of the PWA system. Other types of performance analysis procedures can be derived by considering alternative types of supply rates (Scherer et al., 1997). An important issue is represented by the structure of the storage function used to test this performance criterion. The considerations reported in Section 6.3 about the structure of a Lyapunov function for stability tests, are valid also for the choice of a storage function.

6.5.2 l_2-gain Analysis for PWA Systems

We first establish some preliminary facts on the l_2-gain of a PWA system for which the origin is stable and stability can be checked with a storage function of type (6.4)-(6.5).

Lemma 6.4 *Let Assumptions 6.2 and 6.3 hold. Assume that the stability of the origin of system (6.79) can be guaranteed with a storage function of type (6.4)-(6.5) and set $x_0 = 0$. Then, the l_2 constraint (6.80) is satisfied $\forall \gamma > \gamma_0$ where*

$$\gamma_0 = \left(\tilde{C}^2 \bar{\gamma}^2 + \tilde{D}^2\right)^{1/2} \tag{6.84}$$

and

$$\tilde{C} \triangleq \sup_{i \in \mathcal{I}} \|C_i\|, \qquad \tilde{D} \triangleq \sup_{i \in \mathcal{I}} \|D_i\| \tag{6.85}$$

$$\bar{\gamma} \triangleq \frac{\bar{L}_1 + (\bar{L}_1^2 + 4\bar{L}_2)^{1/2}}{2} \tag{6.86}$$

$$\bar{L}_1 \triangleq \frac{2L_1\bar{P}}{\sigma}, \qquad \bar{L}_2 \triangleq \frac{L_2\bar{P}}{\sigma} \tag{6.87}$$

$$L_1 \triangleq \sup_{i\in\mathcal{I}} \|A_i\|\|B_i\| \tag{6.88}$$

$$L_2 \triangleq \sup_{i\in\mathcal{I}} \|B_i\|^2 \tag{6.89}$$

$$\bar{P} \triangleq \sup_{i\in\mathcal{I}} \sup_{x\in\mathcal{X}_i} \|P_i(x)\|. \tag{6.90}$$

Proof. The proof is reported in (Ferrari-Trecate, Cuzzola, Mignone and Morari, 2000). □

The next result, which is a generalization of the classical Bounded Real Lemma (Lin and Byrnes, 1996) to PWA systems, allows to analyze the H_∞ performance of a PWA system.

Lemma 6.5 *Let Assumptions 6.2 and 6.3 hold. Consider the system (6.79) with zero initial condition $x_0 = 0$. If there exists a function $V(x) = x^T P_i(x) x \ \forall x \in \mathcal{X}_i$ of type (6.4) satisfying the LMIs*

$$\forall k \in \mathbb{N}_+, \qquad \begin{bmatrix} x_k^T & w_k^T \end{bmatrix} M_{ji}(x_k, w_k) \begin{bmatrix} x_k^T & w_k^T \end{bmatrix}^T < 0 \tag{6.91}$$

where

$$M_{ji}(x_k, w_k) \triangleq \begin{bmatrix} A_j^T P_i(x_{k+1}) A_j - P_j(x_k) + C_j^T C_j & \left(D_j^T C_j + B_j^T P_i(x_{k+1}) A_j\right)^T \\ D_j^T C_j + B_j^T P_i(x_{k+1}) A_j & B_j^T P_i(x_{k+1}) B_j + D_j^T D_j - \gamma^2 I \end{bmatrix}. \tag{6.92}$$

then, the dissipativity inequality (6.81) with supply rate (6.83) is met with and the l_2 performance condition (6.80) is satisfied. Furthermore, the system (6.79) is exponentially stable.

Proof. The proof is reported in (Ferrari-Trecate, Cuzzola, Mignone and Morari, 2000). □

The conservativeness of the performance condition (6.83) depends strictly on the structure imposed on the matrices $P_i(x)$: some possible choices are reported in Section 6.3.
Note that (6.83) (or equivalently the condition (6.91)) is satisfied if the following inequality is met

$$\begin{aligned} & M_{ji}(x_k, w_k) < 0 \\ & x_k \in \mathcal{X}_j \backslash \{0\}, \qquad x_{k+1} = A_j x_k + B_j w_k \in \mathcal{X}_i. \end{aligned} \tag{6.93}$$

Obviously, it is possible to reduce the condition (6.93) to a PMI problem that, as pointed out in Section 6.3, can be solved resorting to classical techniques like the gridding method. Furthermore, by assuming that the matrices $P_i(x)$ are independent of the state (cases i) and ii) of Section 6.3), and by using relaxations similar to those discussed in Section 6.3.5, condition (6.93) becomes:

$$P_i - E_i^T U_i E_i > 0, \quad \forall i \in \mathcal{I} \tag{6.94}$$

$$\bar{M}_{ji} \triangleq \begin{bmatrix} A_j^T P_i A_j - P_j + C_j^T C_j + E_i^T Z_{ij} E_i & \left(D_j^T C_j + B_j^T P_i A_j\right)^T \\ D_j^T C_j + B_j^T P_i A_j & B_j^T P_i B_j + D_j^T D_j - \gamma^2 I \end{bmatrix}, \forall (i,j) \in \mathcal{W} \tag{6.95}$$

where U_i and Z_{ij} are matrices of suitable dimensions with non-negative entries.

6.6 Summary

We have developed an LMI based technique to test stability and performance of discrete-time hybrid systems in PWA form. The use of LMIs has been exploited in oder to obtain a numerically reliable procedure. In this chapter we focused on three main points: first, we

highlighted the importance of the switching structure of the PWA system, second, we made an extensive comparison between several types of tests in terms of their conservativeness, and third, we extended our results to the l_2 analysis problem. by means of classical dissipative concepts for nonlinear discrete-time systems. Such LMI procedures can be further extended to the synthesis of state-feedback regulators. Some preliminary results concerning the control of PWA systems, whose switching structure depends on the state only, is reported in (Ferrari-Trecate, Cuzzola, Mignone and Morari, 2001) and generalizations were derived in (Cuzzola and Morari, 2001), where the authors considered controlling PWA systems whose switches depend not only on the state but also on the control input.

Chapter 7

Conclusions and Outlook

Even though the tools and methods dealing with MLD systems are still at an early stage, an increasing number of practical applications and theoretical results have been reported. Some of them are presented in this thesis. It is the author's opinion that the MLD formalism provides a promising framework for formulating and solving a large number of practical problems for hybrid systems. One of the main strengths of MLD systems is the possibility to handle several questions of practical relevance within one common framework in a systematic way.

The major challenge for the future consists in overcoming the computational problems that prevent a scaling of the methods to arbitrary dimensions for general MLD systems. This can be achieved by suitable problem formulations, by using improved and reliable solvers, or by combining mixed integer optimization techniques with other computational methods, like constraint logic programming (Jain and Grossmann, 2002). In view of the results presented in this work, there are some directions in which further research is appropriate to gain better understanding of the MLD paradigm and the analysis and synthesis methods connected to it.

Integral Vertices of Polytopes in the Unit Hypercube in Section 2.4.1: The translation of logic propositions into inequalities in Section 2.4.1 leads in some cases to sets of constraints, describing polytopes with exclusively integral vertices. The appealing property of these cases concerns the solution of the corresponding integer linear program that can be performed by solving the problem with relaxed integrality constraints, i.e. an LP. The recognition of these problems and the specific modeling of systems such that these features are reached can be further analyzed. In particular, a possible application for mixed integer optimizations can be envisioned.

In Section 2.4.1 we have also shown that using the CNF method to find the inequalities representing a logical proposition may lead to polytopes, which are larger than necessary. This problem can be avoided using the truth table method. However, we still have to provide an application, where the benefits of the truth table method over the CNF method can be seen.

Geometrical Method for Mixed Logic-Continuous Propositions in Section 2.4.2: The geometric interpretation of Boolean relations in Section 2.4.1 provides a method to find the inequalities corresponding to those relations. For mixed logic-continuous propositions the analogous procedure in Section 2.4.2 on page 51 is not as systematic as for Boolean relations. Further research can be dedicated to this topic. The goal is the development of an algorithm for this translation, which is possibly based on computational geometry tools, as listed in Section 2.4.1 for the convex hull computations. In particular, it is not clear yet, whether the polytopes obtained with the generalized conjunctive normal form method exhibit unnecessary vertices, as it was the case for the CNF of Boolean relations. Another goal of a geometric approach consists in finding an algorithm that avoids such vertices.

Different Norms in Moving Horizon Estimation in Section 4.3: The moving horizon estimation scheme in this work uses a quadratic cost function. From a computational point of view, the software tools to solve MILPs are more advanced than for MIQPs. A reformulation of the estimation problem with a 1-norm cost function allows to use these tools. In the author's experience, this approach leads, however, to bad estimator properties. If used for fault detection and isolation purposes, the estimates exhibit several false alarms. It is not yet clear, whether the usage of nonquadratic norms for estimation of MLD systems has any inherent limitations.

Observability Analysis for the Fault Detection Scheme in Section 4.4.2: The method shown in Section 4.4.2 provides fault estimates using moving horizon estimation. The choice of the design parameters influences the success of fault detection and isolation. However, in some cases the faults are difficult to detect, because their effect on the measured variables is too small. The FDI scheme may interpret these effects as disturbances ζ and ξ in (3.37), rather than as faults ϕ. In case of such false alarms or missed fault detections, we should have a method that helps us to determine, whether the design parameters are badly chosen or the FDI problem is inherently difficult to solve.

As a possible answer to these question, we suggest a quantitative observability analysis for MLD systems, as it was first investigated by Bemporad, Ferrari-Trecate and Morari (2000). The goal is to find out, whether the degree of observability of the faults in a given model is high enough to allow fault diagnosis. If the degree of observability lies above a certain threshold[1], then we are likely to face a problem of inadequate choice of design parameters in case of a malfunction of the FDI scheme. On the other hand, if the degree of observability lies below a threshold, then we should consider a modified modeling of the faulty MLD system or additional measurements.

[1] which has to be determined for the system under consideration

Stochastic Set-Up of the Fault Detection Scheme in Section 4.4.2: The fault detection set-up introduced in this work does not consider any probability distribution of the fault occurrence. In principle this type of information can be included in the problem at least in three ways:

- In the MLD model: One possibility is to define one additional continuous state x_{fi} with range $[0, 1]$ for each fault ϕ_i. The states x_{fi} keep track of the probability of a fault occurrence at the current time step, given the past values of the fault estimates. The evolution of x_{fi} may be dependent on conditional probability distribution functions. Since x_{fi} change in time, we can use them to introduce time-varying penalty terms in the moving horizon estimation cost function (4.25), using the modeling techniques shown in Chapter 2.
- In the weights of the cost function: In general, the weights on the fault estimates ϕ in (4.25) reflect the probability of fault occurrence. If this probability is known a priori, it can be used for the choice of the weights.
- In the optimization scheme, i.e. in the branch and bound procedure: Especially for schemes with limited computational time, we can use the knowledge on the fault probability distribution to explore those subproblems first, that correspond to the most likely fault combinations. This idea aims at quickly find the optimal solution in the branch and bound iterations.

Training of the Fault Detection Scheme in Section 4.4.2: It is a common practice to train a fault detection scheme with faultless data. In our case, sets of faultless data can be used to tune the weighting matrices Q_i in the cost function of the estimator, in order to avoid false alarms and missed fault detections.

Combination of Fault Detection and Reconfiguration in Section 4.5: The experiments on the three tank system for reconfiguration, shown in Section 4.5.9 assume that

the fault information is exact: there is no underlying fault estimation, but the faults and their time of occurrence are assumed to be known precisely. The combination of a fault detection scheme with a reconfiguration scheme has still to be performed and tuned in order to give satisfactory results on the experimental set-up.

Piecewise Affine Observers for Piecewise Affine Systems in Chapter 6: We reported some methods based on piecewise quadratic Lyapunov functions to stabilize piecewise linear systems. The dual problem consists in finding a piecewise linear observer for piecewise linear models, which stabilizes the estimation error. In analogy to the state feedback problem, the observer gain is allowed to switch according to the current plant operating condition.

Appendix A

Boolean Functions and Clausal Inequalities

This appendix refers to Section 2.4.1. We report a Boolean function $f : [0,1]^4 \to [0,1]$ for which P_{CH} has a nonclausal inequality. f is defined by the truth table in Table A.1.

δ_1, δ_2, δ_3, δ_4 are independent variables: $\delta_5 = f(\delta_1, \delta_2, \delta_3, \delta_4)$

δ_1	δ_2	δ_3	δ_4	δ_5
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Table A.1: Truth table of a Boolean function: δ_5 is the dependent variable

The convex hull of the rows of Table A.1 in $[0,1]^5$ is described by the hyperplanes

$$\begin{aligned}
4-\delta_1-\delta_2-\delta_3-\delta_4-\delta_5 &\geq 0\\
1-\delta_1+\delta_3-\delta_4+\delta_5 &\geq 0\\
1-\delta_2+\delta_3-\delta_4+\delta_5 &\geq 0\\
1-\delta_1+\delta_2-\delta_4+\delta_5 &\geq 0\\
1-\delta_1+\delta_3+\delta_4-\delta_5 &\geq 0\\
1-\delta_2+\delta_3+\delta_4-\delta_5 &\geq 0\\
1+\delta_1-\delta_2+\delta_4-\delta_5 &\geq 0\\
1-\delta_1-\delta_3+\delta_4+\delta_5 &\geq 0\\
+\delta_2-\delta_3+\delta_4+\delta_5 &\geq 0\\
-1+\delta_1+\delta_2+\delta_4+\delta_5 &\geq 0\\
1+\delta_1-\delta_2-\delta_4+\delta_5 &\geq 0\\
1-\delta_1+\delta_2-\delta_3+\delta_5 &\geq 0\\
1+\delta_1+\delta_2-\delta_4-\delta_5 &\geq 0\\
1-\delta_1+\delta_2-2\delta_3+\delta_4+2\delta_5 &\geq 0
\end{aligned}$$

Note that the last inequality is not clausal since it contains coefficients with value 2. For this set of constraints, it is the only inequality that is not clausal. Using Karnaugh maps, the minimal CNF of this function and therefore P_{CNF} can be found. By construction, P_{CNF} contains all vertices represented by the rows of Table A.1. However, it contains also 7 nonintegral vertices. This confirms that in general $P_{CH} \neq P_{CNF}$ even restricting the considered Boolean relations to Boolean functions.

Appendix B

Description of the Three Tank System in HYSDEL

The HYSDEL listing of the three tank system in Chapter 3 is reported next. The model has the characteristics mentioned in the header of the file in Table B.1.

```
/*********************************************************************/
/* file:           tank03.hys                                        */
/* system:         Three tank system with faults                     */
/* states:         3 continuous =  levels                            */
/*                 0 logic                                           */
/* input:          2 continuous =  pumps                             */
/*                 7 logic      =  V13, V23, V1, V2, phi1, phi2, phi3 */
/* parameters:     estimated from laboratory model                   */
/* outflows:       leak in tank 1 as phi1       position o1-b        */
/*                 nominal outflow from tank 3  position o1-b        */
/* author:         Domenico Mignone (mignone@aut.ee.ethz.ch)         */
/* date:           2001.09.04                                        */
/* hysdel version: 0.76 or higher                                    */
/*********************************************************************/
```

Table B.1: HYSDEL description of the three tank system, part 1

```
IMPLEMENTATION {

SYSTEM tank03 {

INTERFACE {
   STATE {
      REAL h1, h2, h3;             /* continuous liquid levels      */ }

   INPUT {
      REAL Q1, Q2;                                /* pumps             */
      BOOL V13, V23, V1, V2, phi1, phi2, phi3; /* valves and faults */ }

   OUTPUT {
      REAL y1, y2, y3;   }

   PARAMETER {
      REAL az    = 1.0;       /* Flow Correction Term          */
      REAL Area  = 0.0143;    /* Cross-Area of tank            */
      REAL g     =  9.81;     /* Gravity Constant              */
      REAL s13   = 10.9e-6;   /* Cross Section area of valves  */
      REAL s23   =  8.89e-6;  /* Cross Section area of valves  */
      REAL s2    =  5.54e-6;  /* Cross Section area of valves  */
      REAL s1    =  9.36e-6;  /* Cross Section area of valves  */
      REAL sl    =  8.62e-6;  /* Cross Section area of valves  */
      REAL sn3   =  9.05e-6;  /* Cross Section area of valves  */
      REAL dT    = 10;        /* sampling time                 */
      REAL hv    =  0.3;      /* m                             */
      REAL hmax  =  0.62;     /* m                             */
      REAL Q1max =  0.1e-3;   /* Maximum flow through Pump 1   */
      REAL Q2max =  0.1e-3;   /* Maximum flow through Pump 2   */

      REAL k1  = az*s1*sqrt(2*g/(hmax-hv));
      REAL k2  = az*s2*sqrt(2*g/(hmax-hv));
      REAL k13 = az*s13*sqrt(2*g/hmax);
      REAL k23 = az*s23*sqrt(2*g/hmax);
      REAL kl1 = az*sl*sqrt(2*g/hmax);
      REAL kn3 = az*sn3*sqrt(2*g/hmax);
      REAL e   = 1e-6;        /* epsilon of inequalities           */
      REAL TdA = dT/Area;     /* (sampling time (s) / Area (m^2) )  */ }

} /* end interface */
```

Table B.2: HYSDEL description of the three tank system, part 2

```
IMPLEMENTATION {

   AUX {
      REAL z01, z02, z03, z1, z2, zl1, z13, z23;
      BOOL d01, d02, d03, f1;
   }

   LOGIC {
      f1 = (~phi2 & V1) | (phi3);
   }

   AD {
      d01 = -h1 + hv <= 0.0 [hv, -hmax+hv,  e];
      d02 = -h2 + hv <= 0.0 [hv, -hmax+hv,  e];
      d03 = -h3 + hv <= 0.0 [hv, -hmax+hv,  e];
   }

   DA {
      z01 = {IF d01  THEN h1-hv   [hmax-hv,   -hv, 0] };
      z02 = {IF d02  THEN h2-hv   [hmax-hv,   -hv, 0] };
      z03 = {IF d03  THEN h3-hv   [hmax-hv,   -hv, 0] };
      z1  = {IF f1   THEN z01-z03 [   hmax, -hmax, 0] };
      z2  = {IF V2   THEN z02-z03 [   hmax, -hmax, 0] };
      zl1 = {IF phi1 THEN h1      [   hmax,     0, 0] };
      z13 = {IF V13  THEN h1-h3   [   hmax, -hmax, 0] };
      z23 = {IF V23  THEN h2-h3   [   hmax, -hmax, 0] };
   }

   CONTINUOUS {
      h1 = h1 + TdA*(Q1 -k1*z1 -k13*z13 -kl1*zl1) ;
      h2 = h2 + TdA*(Q2 -k2*z2 -k23*z23 );
      h3 = h3 + TdA*(    k1*z1 +k2*z2   +k13*z13 +k23*z23 -kn3*h3);
   }

   MUST {
      -h1        <= 0;
      -h2        <= 0;
      -h3        <= 0;
      h1 - hmax  <= 0;
      h2 - hmax  <= 0;
      h3 - hmax  <= 0;
      -Q1        <= 0;
      -Q2        <= 0;
      Q1 - Q1max <= 0;
      Q2 - Q2max <= 0;
   }
} /* end implementation */
} /* end system */
```

Table B.3: HYSDEL description of the three tank system, part 3

Appendix C

Linear Matrix Inequalities

In this section we give a brief introduction in the theory of Linear Matrix Inequalities that are used in Chapter 6.

C.1 Definition

A Linear Matrix Inequality (LMI) is a constraint of the following form:

$$F(\underline{x}) \;=\; F_0 + x_1 F_1 + \ldots + x_p F_p \;>\; 0 \tag{C.1}$$

where $\underline{x} = (x_1, \ldots, x_p) \in \mathbb{R}^p$ are variables, F_i are given symmetric matrices. The inequality sign in (C.1) denotes positive definiteness. One of the best known LMIs is Lyapunov's stability criterion for linear time invariant systems $\dot{x}(t) = Ax(t)$. The system is stable if and only if there exists a symmetric matrix P such that

$$A^T P + PA \;<\; 0 \tag{C.2}$$

$$P \;>\; 0 \tag{C.3}$$

By choosing a basis for symmetric matrices and by merging (C.2) and (C.3) to one LMI, we can rewrite the criterion as (C.1).

In an *LMI Feasibility Problem* we are given an LMI $F(\underline{x}) > 0$ and wish to find an $\underline{x}_{feas}$ such that $F(\underline{x}_{feas}) > 0$ or prove that the LMI is infeasible.
A *Semidefinite Program* (SDP) is defined by a linear optimization problem constrained by LMIs as:

$$\begin{aligned} &\min_{\underline{x}} \underline{c}^T \underline{x} \\ \text{subject to } &F(\underline{x}) > 0 \end{aligned}$$

The first monograph on this topic is by Boyd et al. (1994), which contains a comprehensive list of tasks that can be recast as LMI problems.

C.2 Why Linear Matrix Inequalities?

The main practical reason for formulating problems in the LMI framework is the efficiency, these problems can solved with. On one hand the problem of interest might be difficult to solve by other means or can lack an analytical solution, at all. On the other hand an LMI formulation is often a convenient way to specify a constraint, which arises in several situations, like e.g. in stability or performance requirements.

There are several problems in system and control theory that can be formulated as an SDP or as an LMI feasibility test. From a practical point of view, a problem that can be reduced to an LMI problem can be solved efficiently and reliably (Boyd et al., 1994). Here we list only a couple of such problems:

- Search for Lyapunov functions for linear and some classes of nonlinear systems (e.g. piecewise linear systems)
- H_∞ control synthesis
- Multiobjective controller and system design

- Finding the largest domain of attraction of a polytopically constrained linear system that can be guaranteed with a quadratic Lyapunov function
- Robust stability analysis of linear systems
- Gain-Scheduling synthesis

We remark that LMIs are encountered also in several fields, other than systems and control theory. For instance in VLSI circuit design, we encounter semidefinite programs that are formulating the trade-off, between speed, power consumption and size of the circuits. In economical applications the choice of a robust portfolio composition, managing risk and expected return can be formulated with the help of LMIs.

C.3 Schur Complements

One tool often used to formulate LMI constraints is the application of Schur complements (Boyd et al., 1994). It is used to transform matrix inequalities that are apparently nonlinear into LMIs. We state here the main theorem.

Theorem C.1 *The following LMIs are equivalent:*

- $$\begin{pmatrix} Q & S \\ S^T & R \end{pmatrix} > 0 \tag{C.4}$$
- $$R > 0 \tag{C.5}$$
$$Q - SR^{-1}S^T > 0 \tag{C.6}$$

- $$Q > 0 \tag{C.7}$$
$$R - S^T Q^{-1} S > 0 \tag{C.8}$$

- $$\begin{pmatrix} R & S^T \\ S & Q \end{pmatrix} > 0 \tag{C.9}$$

C.4 Software Tools

In the late 1980s interior point methods for solving LMI problems were introduced. This allowed to develop efficient solvers that have polynomial complexity in the number of variables and the number of rows in the LMIs. We defer to (Vandenberghe and Boyd, 1996) for a review of the theory. Two software packages using these techniques for solving LMI problems are:

- LMI control toolbox, which is distributed as a Matlab toolbox (Gahinet et al., 1994)
- LMItool (El Ghaoui and Commeau, 1998) is a package for LMI optimization that acts as an interface for the following Semidefinite Programming solvers:
  - SP developed by L. Vandenberghe and S. Boyd
  - SDP developed by Alizadeh, J.-P. Haeberly, M. Nayakkankuppam, M.L. Overton
  - SDPHA developed by Florian A. Potra, Rongqin Sheng and Nathan Brixius

Appendix D

List of Abbreviations

abbreviation	:	meaning	occurrences (pages)
B & B	:	Branch and Bound	187
CNF	:	Conjunctive Normal Form	31
ELC	:	Extended Linear Complementarity	58
F	:	False	21
FDI	:	Fault Detection and Isolation	77
FI	:	Full Information	133
FIFO	:	First-In First-Out	206
GBF	:	Generalized Boolean formula	42
GCNF	:	Generalized Conjunctive Normal Form	42
HYSDEL	:	Hybrid Systems Description Language	61, 68, 261
LC	:	Linear Complementarity	58
LIFO	:	Last-In First-Out	195
LMI	:	Linear Matrix Inequality	219, 265
LP	:	Linear Program	187

Table D.1: List of abbreviations, part 1

abbreviation	:	meaning	occurrences (pages)
MHE	:	Moving Horizon Estimation	131
MIFT	:	Mixed Integer Feasibility Test	29, 127
MILP	:	Mixed Integer Linear Program	185, 189
MIQP	:	Mixed Integer Quadratic Program	116, 185, 189
MLD	:	Mixed Logic Dynamical	12, 20
MLDF	:	Mixed Logic Dynamical with Faults	78
MMPS	:	Max-Min-Plus-Scaling	58
MPC	:	Model Predictive Control	115
PWA	:	Piecewise Affine	57, 217
PWQ	:	Piecewise Quadratic	219
Q	:	Quadratic	225
QP	:	Quadratic Program	188
SDP	:	Semidefinite Program	266
T	:	True	21
$\wedge$	:	logical conjunction (AND)	21
$\vee$	:	logical disjunction (OR)	21
$\bar{\cdot}$	:	logical negation (NOT)	21
$\oplus$	:	logical exclusive-or (EXOR)	21
$\rightarrow$	:	logical implication	21
$\leftrightarrow$	:	logical equivalence (IFF)	21

Table D.2: List of abbreviations, part 2

Appendix E

Curriculum Vitae of Domenico Mignone

March 23, 1972	Born in Zürich, Switzerland
April, 1979 to April, 1987	Primary and Secondary School in Zürich, Switzerland
April, 1987 to September, 1991	High School, Wirtschaftsgymnasium, Kantonsschule Enge, Zürich, Switzerland
October, 1991 to October, 1992	Studies of Mathematics, ETH Zürich, Switzerland
October, 1992 to April, 1997	Diploma in Electrical Engineering, ETH Zürich, Switzerland
Since May, 1997	Postgraduate Student at the Automatic Control Laboratory, ETH Zürich, Switzerland
November, 1999	Postdiploma in Information Technology ETH Zürich, Switzerland

Bibliography

Alessandri, A. and Coletta, P.: 2001, Design of Luenberger Observers for a Class of Hybrid Linear Systems, *in* M. Di Benedetto and A. Sangiovanni-Vincentelli (eds), *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 4^{th} *International Workshop on Hybrid Systems*, number 2034 in *Lecture Notes in Computer Science*, Springer-Verlag, Roma, Italy, pp. 7–18.

Alur, R., Courcoubetis, C., Henzinger, T. and Ho, P.: 1993, Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems, *in* R. Grossmann, A. Nerode, A. Ravn and H. Rischel (eds), *Hybrid Systems*, number 736 in *Lecture Notes in Computer Science*, Springer-Verlag, pp. 209–229.

Anlauff, M., Bemporad, A., Chakraborty, S., Kutter, P., Mignone, D., Morari, M., Pierantonio, A. and Thiele, L.: 1999, From Ease in Programming to Easy Maintenance: Extending DSL Usability with Montages, *submitted to Usenix, DSL 99, 2nd conference on domain specific languages, Austin Texas.*

Antsaklis, P. and Nerode, A. (eds): 1998, *IEEE Transactions on Automatic Control – Special Issue on Hybrid Control Systems*, IEEE. Volume 43, Number 4.

Askari, J., Heiming, B. and Lunze, J.: 1999, Controller Reconfiguration Based on a Qualitative Model: A Solution of Three-Tanks Benchmark Problem, *Proceedings of the European Control Conference, Karlsruhe, Germany* . Session CM5-3.

Åström, K., Albertos, P., Blanke, M., Isidori, A., Schaufelberger, W. and Sanz, R. (eds): 2001, *Control of Complex Systems*, Springer-Verlag.

Avis, D.: 1997, *User's Guide for lrs - Version 3.2*, McGill University Montreal, Canada. `http://www.cs.mcgill.ca/~avis`.

Bajpani, G., Chang, B. and Lau, A.: 2001, Reconfiguration of flight control systems for actuator failures, *IEEE Aerospace and Electronic Systems Magazine* **16**(9), 29–34.

Barber, C., Dobkin, D. and Huhdanpaa, H.: 1996, The quickhull algorithm for convex hulls, *ACM Trans. on Mathematical Software* . `http://www.geom.umn.edu/software/qhull/`.

Bemporad, A., Borrelli, F. and Morari, M.: 2000a, Optimal controllers for hybrid systems: Stability and piecewise linear explicit form, *Proceedings of 39th Conference on Decision and Control* .

Bemporad, A., Borrelli, F. and Morari, M.: 2000b, Piecewise Linear Optimal Controllers for Hybrid Systems, *Proceedings of the American Control Conference* .

Bemporad, A., Ferrari-Trecate, G., Mignone, D., Morari, M. and Torrisi, F. D.: 1999, Model Predictive Control - Ideas for the Next Generation, *Proceedings of the European Control Conference, Karlsruhe, Germany.*

Bemporad, A., Ferrari-Trecate, G. and Morari, M.: 1999, Observability and Controllability of Piecewise Affine and Hybrid Systems, *Proceedings of the Conference on Decision and Control* .

Bemporad, A., Ferrari-Trecate, G. and Morari, M.: 2000, Observability and Controllability of Piecewise Affine and Hybrid Systems, *IEEE Transactions on Automatic Control* **45**(10), 1864–1876.

Bemporad, A. and Mignone, D.: 2000, `miqp.m`: A matlab function for solving mixed integer quadratic programs, *Technical report*, Automatic Control Laboratory, ETH Zürich, `http://www.aut.ee.ethz.ch/~hybrid/miqp/`.

Bemporad, A., Mignone, D. and Morari, M.: 1999a, An Efficient Branch and Bound Algorithm for State Estimation and Control of Hybrid Systems, *Proceedings of the European Control Conference, Karlsruhe, Germany* .

Bemporad, A., Mignone, D. and Morari, M.: 1999b, Moving Horizon Estimation for Hybrid Systems and Fault Detection, *Proceedings of the American Control Conference*.

Bemporad, A. and Morari, M.: 1999a, Control of Systems Integrating Logic, Dynamics, and Constraints, *Automatica* **35**(3), 407–427.

Bemporad, A. and Morari, M.: 1999b, Verification of hybrid systems via mathematical programming, *in* F. Vaandrager and J. van Schuppen (eds), *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 2^{nd} *International Workshop on Hybrid Systems*, number 1569 in *Lecture Notes in Computer Science*, Springer-Verlag, Berg en Dal, The Netherlands.

Bemporad, A., Morari, M., Dua, V. and Pistikopoulos, E. N.: 1999, The explicit linear quadratic regulator for constrained systems, *Technical Report AUT99-16*, Automatic Control Lab, ETH Zürich, Switzerland.

Bemporad, A., Morari, M., Dua, V. and Pistikopoulos, E. N.: 2000, The explicit linear quadratic regulator for constrained systems, *American Control Conference*, Chicago, IL. pdf available at `http://control.ethz.ch/~bemporad/dsi/mp_frame.html`.

Bemporad, A., Torrisi, F. and Morari, M.: 2000, Optimization-based verification and stability characterization of piecewise affine and hybrid systems, *in* N. Lynch and B. Krogh (eds), *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 3^{rd} *International Workshop on Hybrid Systems*, number 1790 in *Lecture Notes in Computer Science*, Springer-Verlag, Pittsburgh, PA, USA, pp. 45–58.

Bemporad, A., Torrisi, F. and Morari, M.: 2001, Discrete-time hybrid modeling and verification of the batch evaporator process benchmark, *European Journal of Control* **7**(4), 382–399.

Benveniste, A., Le Borgne, M. and Le Guernic, P.: 1993, Hybrid Systems: the SIGNAL approach, *in* R. Grossmann, A. Nerode, A. Ravn and H. Rischel (eds), *Hybrid Systems*, number 736 in *Lecture Notes in Computer Science*, Springer-Verlag.

Blanke, M.: 1999, *Fault-tolerant Control Systems*, Vol. Advances in Control - Highlights of ECC'99, Springer-Verlag, pp. 171–196.

Blondel, V. and Tsitsiklis, J.: 1999, Complexity of Stability and Controllability of Elementary Hybrid Systems, *Automatica* **35**(3), 479–490.

Borrelli, F.: 2002, *Optimal Control of Constrained Systems*, PhD thesis, ETH Zürich, Switzerland. to appear.

Borrelli, F., Bemporad, A., Fodor, M. and Hrovat, D.: 2001, A hybrid approach to traction control, *in* M. Di Benedetto and A. Sangiovanni-Vincentelli (eds), *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 4^{th} *International Workshop on Hybrid Systems*, number 2034 in *Lecture Notes in Computer Science*, Springer-Verlag, Roma, Italy, pp. 162–174.

Boyd, S., El Ghaoui, L., Feron, E. and Balakrishnan, V.: 1994, *Linear Matrix Inequalities in System and Control Theory*, SIAM.

Branicky, M.: 1995, *Studies in Hybrid Systems: Modeling, Analysis, and Control*, PhD thesis, Massachussets Institute of Technology.

Branicky, M.: 1998, Multiple Lyapunov functions and other analysis tools for switched and hybrid systems, *IEEE Transactions on Automatic Control* **43**(4), 475–482.

Branicky, M., Borkar, V. and Mitter, S.: 1998, A unified framework for hybrid control: model and optimal control theory, *IEEE Transactions on Automatic Control* **43**(1), 31–45.

Byrnes, C. I. and Lin, W.: 1995, Passivity and absolute stabilization of a class of discrete-time nonlinear systems, *Automatica* **31**(2), 263–268.

Camacho, E. and Bordons, C.: 1999, *Model Predictive Control*, Springer-Verlag.

Castagnoli, D.: 2000, *Modellizzazione di una centrale idroelettrica ad acqua fluente mediante sistemi ibridi*, Master's thesis, Dipartimento di ingegneria elettrica, Universitá degli Studi di Pavia, Italia.

Cavalier, T., Pardalos, P. and Soyster, A.: 1990, Modeling and integer programming techniques applied to propositional calculus, *Computers and Operations Research* **17**(6), 561–570.

Chandru, V. and Hooker, J.: 1999, *Optimization Methods for Logical Inference*, Wiley-Interscience Series in Discrete Mathematics and Optimization.

Chapuis, J.: 1998, *Modellierung und neues Konzept für die Regelung von Laufwasserkraftwerken*, Diss. ETH Nr. 12765, ETH Zürich.

Chapuis, J. and Kraus, F.: 1999, Application of fuzzy logic for selection of turbines and weirs in hydro power plants, *Proc. 14th IFAC World Congress*.

Chen, J., Frank, P., Kinnaert, M., Lunze, J. and Patton, R.: 2001, *Control of Complex Systems*, Springer-Verlag, chapter : Fault Detection and Isolation, pp. 191–207.

Chen, J. and Patton, R.: 1998, *Robust Model-Based Fault Diagnosis for Dynamic Systems*, Kluwer Academic.

Christiansen, D.: 1997, *Electronics Engineers' Handbook, 4th edition*, IEEE Press/ McGraw Hill, Inc.

Clocksin, W. F. and Mellish, C. S.: 1981, *Programming in Prolog*, Springer-Verlag.

COSY: n.d., European science foundation research program: Control of complex systems, `http://www.esf.org/physical/pp/cosy/cosyb.htm`.

Cuzzola, F. and Morari, M.: 2001, A generalized approach for analysis and control of discrete-time piecewise affine and hybrid systems, *in* M. Di Benedetto and A. Sangiovanni-Vincentelli (eds), *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 4^{th} *International Workshop on Hybrid Systems*, number 2034 in *Lecture Notes in Computer Science*, Springer-Verlag, Roma, Italy, pp. 189–203.

Dash Associates: 1999, *XPRESS-MP User Guide*. `http://www.dashopt.com`.

David, R. and Alla, H.: 1994, Petri nets for modeling of dynamic systems - a survey, *Automatica* **30**(2), 175–202.

Di Benedetto, M. D. and Sangiovanni-Vincentelli, A. (eds): 2001, *Hybrid Systems: Computation and Control*, number 2034 in *Lecture Notes in Computer Science*, Springer-Verlag, Roma, Italy.

El Ghaoui, L. and Commeau, J.-L.: 1998, *LMITOOL-2.0 package*, ENSTA Optimization and Control Group, `http://www.ensta.fr/uer/uma/gropco/`.

Engell, S., Kowalewski, S. and Zaytoon, J. (eds): 2000, *ADPM 2000: The* 4^{th} *International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, Dortmund, Germany.

Evans, R. and Savkin, A. (eds): 1999, *Systems and Control Letters - Special Issue on hybrid control systems*, number 3, Elsevier. Volume 38.

Ezzine, J. and Kavranoglyu, D.: 1997, On Almost-Sure Stabilization of Discrete-Time Jump Parameter Systems: an LMI Approach, *International Journal of Control* **68**(5), 1129–1146.

Fabian, G.: 1999, *A Language and Simulator for Hybrid Systems*, PhD thesis, Technische Universiteit Eindhoven.

Feldbaum, A. A.: 1962, *Rechengeräte in automatischen Systemen*, Oldenbourg.

Ferrari-Trecate, G., Cuzzola, F. A., Mignone, D. and Morari, M.: 2000, Analysis and Control with Performance of Piecewise Affine and Hybrid Systems, *Technical Report AUT00-23*, Automatic Control Laboratory, ETH Zürich, Switzerland, `http://control.ethz.ch/`.

Ferrari-Trecate, G., Cuzzola, F. A., Mignone, D. and Morari, M.: 2002, Analysis of discrete-time piecewise affine and hybrid systems, *Automatica* **to appear**.

Ferrari-Trecate, G., Cuzzola, F. A. and Morari, M.: 2002, Analysis of Discrete-Time PWA Systems with Boolean Inputs, Outputs and States, *to appear in "Hybrid Systems: computation and control (HSCC), Proceedings of the* 5^{th} *international workshop on Hybrid Systems"* .

Ferrari-Trecate, G., Cuzzola, F., Mignone, D. and Morari, M.: 2001, Analysis and Control with Performance of Piecewise Affine and Hybrid Systems, *Proceedings of the American Control Conference* .

Ferrari-Trecate, G., Mignone, D., Castagnoli, D. and Morari, M.: 2000, Mixed Logic Dynamical Model of a Hydroelectric Power Plant, *Proceedings of the 4th International Conference: Automation of Mixed Processes: Hybrid Dynamic Systems ADPM, Dortmund, Germany* .

Ferrari-Trecate, G., Mignone, D. and Morari, M.: 2000, Moving Horizon Estimation for Piecewise Affine Systems, *Proceedings of the American Control Conference* .

Ferrari-Trecate, G., Mignone, D. and Morari, M.: 2001, Moving Horizon Estimation for Hybrid Systems, *IEEE Transactions on Automatic Control* **to appear**.

Fletcher, R. and Leyffer, S.: 1994, A mixed integer quadratic programming package, *Technical report*, University of Dundee, Dept. of Mathematics, Scotland, U.K.

Fletcher, R. and Leyffer, S.: 1998, Numerical Experience with Lower Bounds for MIQP Branch-And-Bound, *SIAM Journal on Optimization* **8**(2), 604–616. `http://epubs.siam.org/sam-bin/dbq/toclist/SIOPT`.

Floudas, C.: 1995, *Nonlinear and Mixed-Integer Optimization*, Oxford University Press.

Gahinet, P., Apkarian, P. and Chilali, M.: 1996, Affine Parameter-Dependent Lyapunov Functions and Real Parametric Uncertainty, *IEEE Transactions on Automatic Control* **41**(3), 436–442.

Gahinet, P., Nemirowski, A., Laub, A. J. and Chilali, M.: 1994, *LMI Control Toolbox*, The MathWorks Inc.

Garcia, C., Prett, D. and Morari, M.: 1989, Model Predictive Control: Theory and Practice – a Survey, *Automatica* **25**(3), 335–348.

Gertler, J.: 1998, *Fault Detection and Diagnosis in Engineering Systems*, Marcel Dekker, Inc.

Glass, A., Gruber, P., Roos, M. and Tödtli, J.: 1995, Qualitative model-based fault detection in air-handling units, *IEEE Control Systems Magazine* **15**(4), 11–22.

Glover, F.: 1975, Improved Linear Integer Programming Formulation of Nonlinear Integer Problems, *Management Science* **22**(4), 455–460.

Goodman, J. and O'Rourke, J. (eds): 1997, *Handbook of discrete and computational Geometry, Discrete Mathematics and its applications*, CRC Press, New York.

Grogg, F.: 1991, Signale und Systeme I, Lecture Notes for Electrical Engineers ETH Zürich. Introduction in digital circuit design, in german.

Grossmann, R., Nerode, A., Ravn, A. and Rischel, H. (eds): 1993, *Hybrid Systems*, number 736 in *Lecture Notes in Computer Science*, Springer-Verlag, New York.

Gutman, P.: 1982, *Controllers for Bilinear and Constrained Linear Systems*, PhD thesis, Department of Automatic Control Lund Institute of Technology, Lund.

Hassibi, A., Boyd, S. P. and How, J. P.: 1999, A Class of Lyapunov Functionals for Analyzing Hybrid Dynamical Systems, *Proceedings of the American Control Conference 1999*. Session TM18-1.

Hayes, J.: 1993, *Introduction to Digital Logic Design*, Addison-Wesley Publishing Company, Inc.

Hedlund, S. and Johansson, M.: 1999, A toolbox for computational analysis of piecewise linear systems, *Proceedings of the European Control Conference, Karlsruhe, Germany* .

Heemels, W., De Schutter, B. and Bemporad, A.: 2001, Equivalence of hybrid dynamical models, *Automatica* **37**(7), 1085–1093.

Heiming, B. and Lunze, J.: 1999, Definition of the Three-Tank Benchmark Problem for Controller Reconfiguration, *Proceedings of the European Control Conference, Karlsruhe, Germany* .

Henzinger, T. and Sastry, S. (eds): 1998, *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 1^{st} *International Workshop on Hybrid Systems*, number 1386 in *Lecture Notes in Computer Science*, Springer-Verlag, Berkeley, CA, USA.

Huerlimann, T.: 2001, *Reference Manual for the LPL Modeling Language, Version 4.42*, Departement for Informatics, Université de Fribourg, Switzerland, `http://www2-iiuf.unifr.ch/tcs/lpl/TonyHome.htm`.

ILOG, Inc.: 2000, *CPLEX 7.0 User Manual*, Mountain View, CA, USA; `www.ilog.com`.

Iwasaki, T., Meinsma, G. and Fu, M.: 2000, Generalised S-Procedure and Finite Frequency KYO Lemma, *Mathematical Problems in Engineering* **6**(2-3), 305–320.

Izadi-Zamanabadi, R.: 1999, *Fault-tolerant Supervisory Control - System Analysis and Logic Design*, PhD thesis, Aalborg University, Denmark.

Jain, V. and Grossmann, I.: 2002, Algorithms for hybrid MILP/CLP models for a class of optimization problems, *INFORMS Journal of Computing* . to appear.

Jakubovic, V.: 1977, The S-Procedure in Nonlinear Control Theory, *Vestnik Leningrad Univ. Math.* **4**.

Johansen, T. A.: 2000, Computation of Lyapunov functions for smooth nonlinear systems using convex optimisation, *Automatica* **36**(11), 1617–1626.

Johansson, M.: 1999, *Piecewise Linear Control Systems*, PhD thesis, Lund Institute of Technology, Sweden, Department of Automatic Control.

Johansson, M. and Rantzer, A.: 1998, Computation of Piecewise Quadratic Lyapunov Functions for Hybrid Systems, *IEEE Transactions on Automatic Control* **43**(4), 555–559.

Jong, W. C. and Seung, K. S.: 1998, Generalized solution of minimum time current control in three-phase balanced systems, *IEEE Transactions on Industrial Electronics* **45**(5), 738–44.

Kailath, T.: 1980, *Linear Systems*, Prentice Hall.

Kantner, M.: 1997, Robust Stability of Piecewise Linear Discrete Time Systems, *Proceedings of the American Control Conference* .

Kennedy, M.: 1993, Three Steps to Chaos – Part I: Evolution and Part II: A Chua's Circuit Primer, *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications* **40**(10), 640–674.

Kerrigan, E.: 2000, *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*, PhD thesis, Control Group, Department of Engineering, University of Cambridge.

Kerrigan, E., Bemporad, A., Mignone, D., Morari, M. and Maciejowski, J. M.: 2000, Multi-objective Prioritisation and Reconfiguration for the Control of Constrained Hybrid Systems, *Proceedings of the American Control Conference* .

Kohavi, Z.: 1978, *Switching and Finite Automata Theory*, McGraw-Hill.

Krebs, V. and Schnieder, E. (eds): 2000, *at Automatisierungstechnik - Special Issue on Hybrid Systems I : Modeling and Control*, number 9, Oldenbourg.

Krebs, V. and Schnieder, E. (eds): 2001, *at Automatisierungstechnik - Special Issue on Hybrid Systems II: Analysis, Modeling and Verification*, number 2, Oldenbourg.

Krivchenko, G.: 1994, *Hydraulic Machines: Turbines and Pumps - Second Edition*, Lewis Publishers.

Labinaz, G., Bayoumi, M. and Rudie, K.: 1997, A Survey of Modeling and Control of Hybrid Systems, *Annual Reviews of Control* **21**, 79–92.

Lahlou, M.: 1994, *Modélisation des canaux hydrauliques et application au règlage de niveau*, Institut d' électronique industrielle, EPF Lausanne, Switzerland.

Lazimy, R.: 1985, Improved algorithm for mixed-integer quadratic programs and a computational study, *Mathematical Programming* **32**, 100–113.

Lee, J. and Cooley, B.: 1997, Recent advances in model predictive control, *Chemical Process Control - V*, Vol. 93, no. 316, AIChE Symposium Series - American Institute of Chemical Engineers, pp. 201–216b.

Lennartson, B., Tittus, M., Egardt, B. and Pettersson, S.: 1996, Hybrid Systems in Process Control, *IEEE Control Systems Magazine* **16**(5), 45–56.

Letizia, P.: 2001, *Controllo di impianti cogenerativi mediante sistemi ibridi*, Master's thesis, Dipartimento di ingegneria elettrica, Universitá degli Studi di Pavia, Italia.

Levis, A., Marcus, S., Perkins, W., Kokotovic, P., Athans, M., Brockett, R. and Willsky, A.: 1987, Challenges to Control: A collective View; Report of the Workshop Held at the University of Santa Clara on September 18-19, 1986, *IEEE Transactions on Automatic Control* **32**(4), 275–285.

Lin, W. and Byrnes, C. I.: 1996, H_∞ Control of Discrete-Time Nonlinear Systems, *IEEE Transactions on Automatic Control* **41**(4), 494–510.

Lions, J.: 1996, Ariane 5 – flight 501 failure, *Technical report*, Ariane 501 Inquiry Board report, `http://www.esa.int/export/esaCP/Pr_33_1996_p_EN.html`.

Lunze, J.: 1998, Laboratory Three Tanks System — Benchmark for the Reconfiguration Problem, *Technical report*, Tech. Univ. of Hamburg-Harburg, Inst. of Control. Eng., Germany. `http://www.tu-harburg.de/rts/software/cosy/`.

Lunze, J., Askari-Marnani, J., Cela, A., Frank, P., Gehin, A.-L., Heiming, B., Lemos, J., Marcu, T., Rato, L. and Staroswiecki, M.: 2001, *Control of Complex Systems*, Springer-Verlag, chapter Three-Tank Control Reconfiguration, pp. 241–283.

Lunze, J. and Schröder, J.: 1999, Qualitative Diagnosis of the 3-Tanks System, *Proceedings of the European Control Conference, Karlsruhe, Germany* .

Lygeros, J. and Lynch, N.: 1998, String of vehicles: modeling and safety conditions, *in* T. Henzinger and S. Sastry (eds), *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 1^{st} *International Workshop on Hybrid Systems*, number 1386 in *Lecture Notes in Computer Science*, Springer-Verlag, Berkeley, CA, USA.

Lynch, N. and Krogh, B. (eds): 2000, *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 3^{rd} *International Workshop on Hybrid Systems*, number 1790 in *Lecture Notes in Computer Science*, Springer-Verlag, Pittsburgh, PA, USA.

Ma, S. and Watanabe, M.: 2001, Minimum-time control of coupled tendon-driven manipulators, *Advanced Robotics* **15**(4).

Maciejowski, J.: 2002, *Predictive Control*, Prentice Hall.

Maciejowski, J. and Lemos, J.: 2001, *Control of Complex Systems*, Springer-Verlag, chapter Predictive Methods for FTC, pp. 229–240.

Marcu, T., Matcovschi, M. H. and Frank, P.: 1999, Neural Observer-Based Approach to Fault-Tolerant Control of a Three-Tank System, *Proceedings of the European Control Conference, Karlsruhe, Germany* . Session CM5-5.

Massey, J.: 1996, Applied Digital Information Theory 2, Lecture Notes ETH Zürich, Switzerland.

Mat: n.d., *Matlab - The Language of Technical Computing*. `http://www.mathworks.com`.

Matveev, A. and Savkin, A.: 2000, *Qualitative Theory of Hybrid Dynamical Systems*, Birkhäuser.

Mayne, D.: 1997, Nonlinear model predictive control: an assessment, *Chemical Process Control - V*, Vol. 93, no. 316, AIChE Symposium Series - American Institute of Chemical Engineers, pp. 217–231.

Mayne, D., Rawlings, J., Rao, C. and Scokaert, P.: 2000, Constrained model predictive control: Stability and optimality, *Automatica* **36**(6), 789–814.

Mendelson, E.: 1964, *Introduction to mathematical logic*, Van Nostrand.

Michalska, H. and Mayne, D.: 1992, Moving horizon observers, *in* M. Fliess (ed.), *Nonlinear control systems design symposium (NOLCOS), IFAC*, Bordeaux, France.

Mignone, D.: 1999, *Moving Horizon Estimation and Fault Detection of Mixed Logic Dynamical Systems*, Postdiploma Thesis (Nachdiplomstudium Informationstechnik), Automatic Control Laboratory, ETH Zürich, Switzerland.

Mignone, D.: 2001, The really big collection of logic propositions and linear inequalities, *Technical report*, AUT01-11, Automatic Control Laboratory, ETH Zürich, Switzerland.

Mignone, D., Bemporad, A. and Morari, M.: 1999, A Framework for Control, Fault Detection, State Estimation and Verification of Hybrid Systems, *Proceedings of the American Control Conference* .

Mignone, D., Ferrari-Trecate, G. and Morari, M.: 2000a, Stability and Stabilization of Piecewise Affine and Hybrid Systems: An LMI Approach, *Proceedings of the Conference on Decision and Control* .

Mignone, D., Ferrari-Trecate, G. and Morari, M.: 2000b, Stability and Stabilization of Piecewise Affine and Hybrid Systems: An LMI Approach, *Technical Report AUT00-12*, Automatic Control Laboratory, ETH Zürich, Switzerland, `http://control.ethz.ch/`.

Mignone, D. and Monachino, N.: 2001, The total three tank tutorial text, *Technical Report AUT01-21*, Automatic Control Laboratory, ETH Zürich, Switzerland.

Miller, R. E.: 1965, *Switching Theory. Volume 1: Combinational Circuits*, John Wiley.

Minnick, R. C.: 1961, Linear-Input Logic, *IRE Transactions on Electronic Computers* pp. 6–16.

Morari, M., Bemporad, A. and Mignone, D.: 1999a, A Framework for Control, State Estimation, Fault Detection, and Verification of Hybrid Systems, *at Automatisierungstechnik* **8**, 374–381. Special issue at the ECC 99.

Morari, M., Bemporad, A. and Mignone, D.: 1999b, A Framework for Control, State Estimation, Fault Detection, and Verification of Hybrid Systems, *International Workshop "Scientific Computing in Chemical Engineering II"*, Vol. 2 (Simulation, Image Processing, Optimization, and Control), Springer-Verlag.

Morse, A., Pantelides, C., Sastry, S. and Schumacher, J. (eds): 1999, *Automatica - A Special Issue on Hybrid Systems*, number 3, Pergamon. Volume 35.

Murtagh, B.: 1981, *Advanced linear programming : computation and practice*, McGraw-Hill.

Murtagh, B. and Saunders, M.: n.d., *Minos 5.5*, `http://www.sbsi-sol-optimize.com/Minos.htm`.

Muske, K. and Rawlings, J.: 1995, *Nonlinear Moving Horizon State Estimation*, Kluwer Academic Press, pp. 349–365.

NAG: 2001, *NAGWare Foundation Toolbox*. `http://www.nag.com`.

Narendra, K. and Balakrishnan, J.: 1994, A Common Lyapunov Function for Stable Systems with Commuting A-Matrices, *IEEE Transactions on Automatic Control* pp. 2469–2471.

Nemhauser, G. and Wolsey, L.: 1988, *Integer and Combinatorial Optimization*, Wiley.

Pettersson, S. and Lennartson, B.: 1997, Exponential stability of hybrid systems using LMIs, *Proceedings of the Conference on Decision and Control* .

Pettersson, S. and Lennartson, B.: 1999, Exponential Stability of Hybrid Systems Using Piecewise Quadratic Lyapunov Functions Resulting in LMIs, *IFAC, 14th Triennial World Congress, Beijing, P.R. China*.

Qin, S. and Badgewell, T.: 1997, An overview of industrial model predictive control technology, *Chemical Process Control - V*, Vol. 93, no. 316, AIChE Symposium Series - American Institute of Chemical Engineers, pp. 232–256.

Raman, R. and Grossmann, I.: 1991, Relation between MILP modeling and logical inference for chemical process synthesis, *Computers and Chemical Engineering* **15**(2), 73–84.

Raman, R. and Grossmann, I.: 1992, Integration of logic and heuristic knowledge in MINLP optimization for process synthesis, *Computers and Chemical Engineering* **16**(3), 155–171.

Rao, C. V.: 2000, *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*, PhD thesis, University of Wisconsin-Madison.

Rao, C. V. and Rawlings, J. B.: 1998, Nonlinear Moving Horizon Estimation, *International Symposium on Nonlinear Model Predictive Control: Assessment and Future Directions (Ascona, Switzerland)*.

Rao, C. V., Rawlings, J. B. and Lee, J. H.: 1999, Stability of Constrained Linear Moving Horizon Approach, *Proceedings of the American Control Conference 1999* .

Rato, L. and Lemos, J.: 1999, Multimodel Based Fault Tolerant Control of the 3-Tank System, *Proceedings of the European Control Conference, Karlsruhe, Germany* . Session CM5-4.

Robertson, D., Lee, J. and Rawlings, J.: 1996, A moving horizon-based approach for least-squares estimation, *AIChE Journal* **42**, 2209–2224.

Romanovsky, V.: 1970, *Discrete Markov Chains*, Wolters-Noordhoff Publishing, Groningen, The Netherlands.

Rothwangl, H. P.: 2001, Numerical Synthesis of the Time Optimal Nonlinear State Controller via Mixed Integer Programming, *Proceedings of the American Control Conference* pp. 3201–3205.

Sahinidis, N. V.: 2000, BARON — Branch And Reduce Optimization Navigator, *Technical report*, University of Illinois at Urbana-Champaign, Dept. of Chemical Engineering, Urbana, IL, USA. `http://archimedes.scs.uiuc.edu/baron/manuse.pdf`.

Sasao, T.: 1999, *Switching Theory for Logic Synthesis*, Kluwer Academic.

Scherer, C., Gahinet, P. and Chilali, M.: 1997, Multiobjective Output-Feedback Control via LMI Optimization, *IEEE Transactions on Automatic Control* **42**(7), 896–911.

Schrijver, A.: 1986, *Theory of Linear and Integer Programming*, Wiley-Interscience.

Shorten, R. and Narendra, K.: 1999, Necessary and Sufficient Conditions for the Existence of a Common Quadratic Lyapunov Function for Two Stable Second Order Linear Time-Invariant Systems, *Proceedings of the American Control Conference 1999* .

Slupphaug, O. and Foss, B.: 1997, Model Predictive Control for a class of Hybrid Systems, *Proceedings of the European Control Conference*, Brussels, Belgium.

Slupphaug, O., Imsland, B. and Foss, B.: 2000, Uncertainty modelling and robust output feedback control of nonlinear discrete systems: a mathematical programming approach, *International Journal of Robust and Nonlinear Control* **10**(13), 1129–1152.

Slupphaug, O., Vada, J. and Foss, B.: 1997, MPC in Systems with Continuous and Discrete Control Inputs, *Proceedings of the American Control Conference*, Albuquerque, NM, USA.

Sontag, E.: 1981, Nonlinear Regulation: The Piecewise Linear Approach, *IEEE Transactions on Automatic Control* **26**(2), 346–358.

Sontag, E.: 1996, Interconnected automata and linear systems: A theoretical framework in discrete-time, *in* R. Alur, T. Henzinger and E. Sontag (eds), *Hybrid Systems III: Verification and Control*, number 1066 in *Lecture Notes in Computer Science*, Springer-Verlag, New Brunswick, NJ, USA, pp. 436–448.

Sontag, E.: 1998, *Mathematical Control Theory*, 2^{nd} edn, Springer-Verlag.

Spedicato, M.: 2001, *Modellizzazione di impianti cogenerativi mediante sistemi ibridi*, Master's thesis, Dipartimento di ingegneria elettrica, Universitá degli Studi di Pavia, Italia.

Steffen, T. and Lunze, J.: 2001, COSY Benchmark Problem, *Technical report*, Arbeitsbereich Regelungstechnik, TU Hamburg-Harburg, Germany, `http://pcweb.rts.tu-harburg.de/inter/homepage.nsf/`.

Stram, O. B.: 1961, Arbitrary Boolean Functions of N Variables Realizable in Terms of Threshold Devices, *Proceedings of the IRE* **49**, 210–220.

Tomlin, C.: 1998, *Hybrid Control of Air Traffic Management Systems*, PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley.

Torrisi, F.: 2002, *Modeling and Verification of Hybrid Systems*, PhD thesis, ETH Zürich, Switzerland. to appear.

Torrisi, F., Bemporad, A. and Mignone, D.: 2000, HYSDEL — a tool for generating hybrid models, *Technical report*, AUT00-03, Automatic Control Laboratory, ETH Zürich, Switzerland.

Torrisi, F., Mignone, D. and Morari, M.: 2001, Time-optimal control of hybrid systems: Heuristic vs systematic design, *Technical Report AUT01-18*, Automatic Control Laboratory, ETH Zürich, Switzerland.

Tsuda, K., Mignone, D., Ferrari-Trecate, G. and Morari, M.: 2000, Reconfiguration for mixed logic dynamical systems, *Technical report*, Automatic Control Laboratory, ETH Zürich, Number AUT00-24.

Tsuda, K., Mignone, D., Ferrari-Trecate, G. and Morari, M.: 2001, Reconfiguration Strategies for Hybrid Systems, *Proceedings of the American Control Conference* .

Tuan, H. D., Ono, E., Apkarian, P. and Hosoe, S.: 2001, Nonlinear H_∞ Control for an Integrated Suspension System via Parametrized Linear Matrix Inequality Characterizations, *IEEE Transactions on Control System Technology* **9**(1), 175–185.

Turner, J.: 1968, *Digital Computer Analysis*, Charles E. Merrill Publishing Company.

Tyler, M. and Morari, M.: 1995, Stability of Constrained Moving Horizon Estimation Schemes, *Technical Report AUT95-05*, Automatic Control Laboratory, ETH Zürich, `http://www.control.ethz.ch/`.

Tyler, M. and Morari, M.: 1999, Propositional Logic in Control and Monitoring Problems, *Automatica* **35**(4), 565–582.

Vaandrager, F. and van Schuppen, J. (eds): 1999, *Hybrid Systems: Computation and Control (HSCC), Proceedings of the* 2^{nd} *International Workshop on Hybrid Systems*, number 1569 in *Lecture Notes in Computer Science*, Springer-Verlag, Berg en Dal, The Netherlands.

van der Schaft, A. and Schumacher, H.: 1999, *An Introduction to Hybrid Dynamical Systems*, Lecture Notes in Control and Information Sciences 251, Springer-Verlag.

van der Schaft, A. and Schumacher, J.: 1998, Complementarity modeling of hybrid systems, *IEEE Transactions on Automatic Control* **43**(4), 483–490.

van Schrick, D. and Christiansen, D.: 1997, Remarks on terminology in the field of supervision, fault detection and diagnosis, Electronics Engineers' Handbook, 4^{th} edition, *Proceedings of the IFAC Symposium on Fault Detection, Supervision, and Safety for Technical Processes: SAFEPROCESS '97* .

Vandenberghe, L. and Boyd, S.: 1996, Semidefinite programming, *SIAM Review* **38**(1), 49–95.

Varaiya, P.: 1998, Reach Set Computation Using Optimal Control, *School on Computational Aspects and Applications of Hybrid Systems, KIT Workshop on Verification of Hybrid Systems*, Grenoble.

Veillette, R., Medanić, J. V. and Perkins, W. R.: 1992, Design of reliable control systems, *IEEE Transactions on Automatic Control* **37**(3), 290–304.

VHS: n.d., `http://www-verimag.imag.fr/VHS/`. Verification of Hybrid Systems, ESPRIT-LTR Project 26270.

Volkovich, O., Roshchin, V. A. and Sergienko, I.: 1987, Models and methods of solution of quadratic integer programming problems, *Cybernetics* **23**, 289–305.

Willems, J. C.: 1972, Dissipative dynamic systems, *Arch. Rational Mechanics Analysis* **45**, 321–393.

Williams, H.: 1977, Logical problems and integer programming, *Bulletin of the Institute of Mathematics and Its Applications* **13**, 18–20.

Williams, H.: 1993, *Model Building in Mathematical Programming*, John Wiley & Sons, Third Edition.

Witsenhausen, H.: 1966, A Class of Hybrid-State Continuous-Time Dynamic Systems, *IEEE Transactions on Automatic Control* **AC-11**(2), 161–167.

Xu, X. and Antsaklis, P. J.: 1999, Stabilization of second-order LTI switched systems, *Proceedings of the 38th Conference on Decision and Control* pp. 1339–1344.

Zhou, K. and Ren, Z.: 2001, A new controller architecture for high performance, robust, and fault-tolerant control, *IEEE Transactions on Automatic Control* **46**(10), 1613–1618.

Ziegler, G.: 1995, *Lectures on Polytopes*, Springer-Verlag.

Index