HIERARCHICAL CONTROL RECONFIGURATION FOR A CLASS OF HYBRID SYSTEMS

By

Tal Pasternak

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

August, 2002

Nashville, Tennessee

Approved: Date:

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

״כל חיי גדלתי בין החכמים, ולא מצאתי לגוף טוב
אלא שתיקה. ולא המדרש הוא העיקר אלא המעשה.״

"All my life I grew up among the wise, and I have found naught good for oneself but silence. And it is not the study that is important but the practice."

Raban Shimon Ben Gamliel (Avot 1:17)

To Yehudit,

זכרתי לךְ חסד נעוּרִיךְ אַהבת כְּלוּלֹתיךְ
לכתךְ אַחרי בּמדבּר בּארץ לֹא זרוּעה

I remember for thee the affection of thy youth, The love of thine espousals;
How thou wentest after Me in the wilderness, In a land that was not sown.

Jeremiah 2:2 (AJV)

ACKNOWLEDGEMENTS

Three years ago I didn't know what a hybrid system is and now I see hybrid systems everywhere. I have learned a lot in this time. And not just about engineering.

And now for the acknowledgements. To my wife, Yehudit, thanks for suspending your career for three long years and leaving your home town, family and friends to come with me to Nashville. To my advisor, Professor Janos Sztipanovits, thanks for the encouragement, direction and focus which you provided throughout this period. Thanks to all my committee members: Professor Gautam Biswas, Professor Gabor Karsai, Professor Greg Nordstrom, and Professor Mark Sapir.

Thanks to Drs. Eric Kerrigan, Domenico Mignone, and Xenofon Koustoukos, on whose research this thesis draws a great deal. To Jonathan Sprinkle, Brandon Eames Dr. Sherif Abdelwahed. Jason Scott, Dr. Sandeep Neema, Dr. Jeff Gray, Dr. Bubba (James R. Davis), Beatrice Richardson, and all the ISIS team.

The funding provided by DARPA via grant F33615-99-C-3611 as part of the Software Enabled Control program, and by Vanderbilt University via a University Graduate Fellowship are greatly appreciated. Thanks to Boeing Corp. for collaborative work.

Last but not least, thanks to Joseph Kudish and Dr. Jonah Lavie, on whose advice and recommendation I chose to study with Professor Sztipanovits.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS iv

LIST OF FIGURES viii

LIST OF TABLES ix

LIST OF ABBREVIATIONS x

CHAPTER

I. INTRODUCTION 1

Motivation 2
The Problem 2
The Solution 5
Contributions 8
Overview 9

II. BACKGROUND 10

System Models 10
Hybrid Automata 12
Piecewise-Affine Systems 15
Control Systems 17
Example - Thermostat 18
Hybrid Control 19
Fault-Tolerant Control Architecture 20
Fault Tolerant Control 20
Passive Methods 21
Switching Control 22
Reconfigurable Control 23
Diagnosis of PWA systems 25

III. PROBLEM DEFINITION AND SCOPE 27

System Model and Assumptions 27
Problem 27
Multi-Phase Fault Accommodation 28
Problem Breakdown and Scope 29

IV. RELATED WORK 33

Model Predictive Control of Piecewise-Affine Systems 33
Model Predictive Control 33
Mixed Logic Dynamic Systems 34
Set Invariance in Control 39
System Models 39
The One-Step Set 40
Positively Invariant and Control Invariant Sets 42
Robust Controllable Sets 44
Feasibility of Model Predictive Control 45
Supervisory and Hierarchical Control 46
Supervisory Control 47
Hierarchical Control of PWA systems 49

V. ARCHITECTURE 51

Proposed Architecture 51
Motivating Example 53
Benchmark Problem 59

VI. RECONFIGURATION 65

Affine Constrained Systems 72
Robust Controllable Sets for PWA systems 73
Reconfiguration and Fault-Tolerance 74
The Reconfiguration Database 77

VII. SUPERVISORY CONTROL 79

Bisimulation 79
Quasideterminism 80
State Space Partition 81

VIII. CONCLUSIONS AND FUTURE WORK 91

Minimization of the Number of Manipulable Inputs 93
Complexity of MPC for PWA systems 93
Approximate Calculation of Invariant Sets for Hybrid Systems 94
Tools for Computational Geometry in High Dimensions 95
Piecewise-Affine-Systems Toolbox 95
Reconfiguration for Constrained Systems with Polytopic Uncertainty 95

APPENDECIES

A MATEMATICAL CONCEPTS 97

B CONVEX APPROXIMMATION 99

REFERENCES 104

LIST OF FIGURES

Figure Page

Figure 1 Three Tank System with Tank 2 Empty 3

Figure 2 Hybrid Automaton of a Bouncing Ball 14

Figure 3 PWA approximation of bouncing ball 16

Figure 4 Thermostat Control 13

Figure 5 Fault Tolerant Control Architecture 20

Figure 6 Switching Control Architecture 22

Figure 7 Single Step Reconfiguration 28

Figure 8. A multi-step reconfiguration process.. 29

Figure 9 Supervisory Control 48

Figure 10. Architecture 51

Figure 11. Simplified Aircraft Fuel System 54

Figure 12 Example Decision Logic for Fuel System Control 56

Figure 13. Architecture of the Aircraft Fuel Control System 57

Figure 14. Alternative ending to the leak scenario 62

Figure 15. Reconfiguration scenario for leak in tank 1 63

Figure 16. Refining the final partition 87

Figure 17. Envelope of Two Polyhedra 100

Figure 18 Bisecting Cutting Plane 101

Figure 19 Convex Approximation 102

Figure 20 $K_{1,200}(\Omega,T)$ for the three-tank switchover 103

LIST OF TABLES

Table Page

Fuel Quantity (lb.) for each tank in the simplified Aircraft Fuel System 56

Control Objectives and Configurations. 64

LIST OF ABBREVIATIONS

NTSB – National Transportation Safety Board

MPC – Model Predictive Control

PWA – Piecewise Affine

LTI – Linear Time Invariant

MIQP – Mixed Integer Quadratic Program

KIAS - Knots Indicated Air Speed

CHAPTER I

INTRODUCTION

It was 3 pm, a Friday, May 25th, 1979, the eve of Memorial Day Weekend. American Airlines DC 10, flight 191, carrying 258 passengers and 13 crewmembers, taxied to its holding point on runway 32R of Chicago-O'Hare Airport, prepared to depart on a non-stop flight to Los Angeles. What happened moments later is described by the investigation report of the National Transportation Safety Board (NTSB) [1]:

> "Flight 191 was taking off from Runway 32R. The weather was clear and the visibility was 15 miles. During the takeoff rotation, the left engine and pylon assembly and about 3 ft of the leading edge of the left wing separated from the aircraft and fell to the runway. Flight 191 continued to climb to about 325' above the ground and then began to roll to the left. The aircraft continued to roll to the left until the wings were past the vertical position, and during the roll, the aircraft's nose pitched down below the horizon.
>
> "Flight 191 crashed into the open field and the wreckage scattered into an adjacent trailer park. The aircraft was destroyed in the crash and subsequent fire. Two hundred and seventy-one persons on board Flight 191 were killed; two persons on the ground were killed, and two others were injured. An old aircraft hangar, several automobiles, and a mobile home were destroyed."

The investigation also found that

> "At the time of DC-10 certification, the structural separation of an engine pylon was not considered. Thus, multiple failures of other systems resulting from this single event was not considered."

In this thesis we address the problem of designing control systems to adapt to failures, so that disasters such as the flight 191 crash could be avoided.

Motivation

Several factors contributed to the crash of flight 191. In hindsight, the flight crew could have maintained control of the aircraft. The investigation found that[1]

> "The flight-crew flew the aircraft in accordance with the prescribed emergency procedure, which called for the climb-out to be flown at V2 speed. V2 was 6 KIAS below the stall speed for the left wing. The deceleration to V2 speed caused the aircraft to stall."

In other words, the flight-crew's corrective action caused the aircraft to stall. However, the aircraft was robust enough to tolerate the separation of the engine. In fact, later simulations confirmed that control of the aircraft could have been maintained if the speed was not decreased. Indeed, safety-critical systems such as aircraft possess a great deal of redundancy and fault-tolerance, often more than their designers are aware of. In one incident, an Israeli Air-Force pilot was able to land an F-15 with one wing missing [2]. Unfortunately, the system design and emergency procedures for the DC 10 were not created with the engine separation scenario in mind. The flight-crew had only a few seconds to react. Once the aircraft rolled with the wings passed the vertical position and its nosed pitched down below the horizon, the flight-crew lost control of the aircraft and the crash was inevitable.

The Problem

Fault-tolerant control can be achieved if the control system is robust with respect to faults, meaning that even when faults occur the control system can still maintain acceptable performance. This is not always possible, because faults can change the plant

behavior dramatically, making it impossible for the same control law to apply to the nominal and faulty system. In this case the control system must be "reconfigured" to match the faulty system. This research focuses on such reconfiguration. The problem of control reconfiguration in fault-tolerant control is concerned with changing the input-output relation between a plant and its controller in such a way that ensures the achievement of a control objective [3]. Consider for example, the three-tank system in Figure 1. Valves and pumps are used by a controller in order to achieve a set-point of fluid levels. The choice of which valves and pumps are to be used by the controller is a reconfiguration decision.

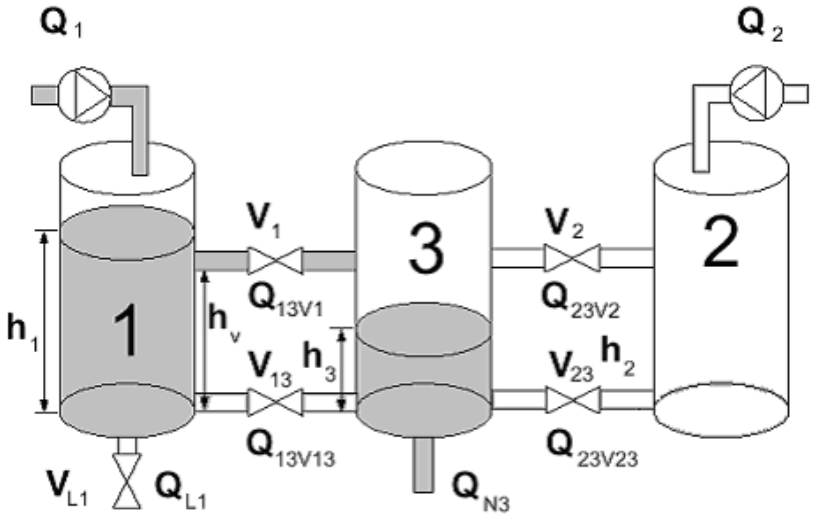

Figure 1 Three Tank System with Tank 2 Empty

The three-tank system of Figure 1 is a "hybrid system". A hybrid system is one that includes both continuous-state time-driven dynamics such as arise from physical

[1] V2 speed is the takeoff safety speed. V2 for a normal, undamaged aircraft was lower than the minimum

processes and discrete-state event-driven dynamics such as arise from the operation of software processes. Analyzing and controlling such systems is a complex task. Moreover reconfiguration in and of itself is a complex task. The problem is that the systems and procedures for reconfiguring the control system in the event of faults don't always produce the correct response for each particular fault scenario, especially when multiple faults occur. In order to address every possible fault scenario it would be necessary to do one of two things: either anticipate every scenario beforehand and prepare a response for each or equip the system with the capability to compute the right response when a situation occurs. In their simple form, both solutions are infeasible for all but extremely small problems because of complexity limitations. Pre-designing the system to react to every possible multiple-fault scenario is prohibitively complex because the number of such scenarios grows exponentially with the number of individual faults. Computing a correct response at runtime is prohibitively complex because it would require a capability to identify faults correctly, to characterize the faulty system, and then to design the new control law online, and all this within a real-time feedback loop.

Giving up on this problem by designing only for a limited number of anticipated faults scenarios is also not advisable. As Murphy's Law states, "Nature always sides with the hidden flaw"[2]. Even when the root cause of failure is a single event, it is not always

---

controllable speed for the damaged aircraft. Because of damages to the electrical system the flight crew did not have sufficient feedback to be aware of the need to increase speed.

[2] The popular culture of "Murphy's Law" attributed to the 1940's US Air Force engineer Edward Murphy Jr. stipulates that when random events are concerned, everything always goes for the worst. Incidents such as the Hubble Telescope fiasco, the Challenger disaster, the Ariane 5 self-destruction, the Mars Climate Orbiter crash, and the Mars Polar Lander disappearance, enhance the popular image of these "laws", especially when complex aerospace projects are involved.

easy to anticipate what combination of multiple induced faults will result, as with the example of flight 191.

A design a based on enumeration of faults has the advantage of robustness to all fault scenarios accounted for, but the limitations of design-time complexity caused by this enumeration permits only a high-level model of the system, and therefore low-fidelity control. On the other hand a design based runtime control synthesis permits hi-fidelity control but robustness to faults is limited because runtime complexity limits the sophistication of the fault-accommodation measures.

The two approaches – design-time enumeration of faults, and runtime control synthesis – need to be combined and balanced in a fault-tolerant control architecture. This is the subject of this thesis. The problem is limited to hybrid systems modeled as piecewise-affine systems in discrete time, and to control objectives, in which the requirement is for the state of the system to traverse a sequence of regions in the state space.

The Solution

The proposed solution is a hierarchical control architecture. At the lowest level the system is modeled as a piecewise-affine in discrete time, which is an approximation of a hybrid system. Piecewise-affine systems have been receiving increasing attention by the control community because they provide a useful modeling framework for hybrid systems. Discrete-time piecewise-affine systems are equivalent to interconnections of linear systems and finite automata [4] and to a number of other hybrid models [5]. In particular, model predictive control can be applied to piecewise-affine systems by converting them to the equivalent mixed-logic dynamic form [10]. Another approach to

control of piecewise affine systems, which is adopted in this thesis, is hierarchical control [7]. Hierarchical control includes low-level control, which may be implemented by model-predictive control, for example, and supervisory control, which operates on a discrete-event abstraction of the hybrid system. The discrete-event abstraction of the closed-loop system, which includes the low-level control and the plant, is obtained by reachability calculations that take into account the available plant inputs, which the low-level control manipulates.

In relation to the problem of control reconfiguration, hierarchical control can provide fault tolerance at both the supervisory control level and at the low level. Consider again the three-tank system in Figure 1. The objective of the control system is to regulate the fluid level in tank 3. If a leak occurs in tank 1, the supervisory controller supervises a phased process by which tank 1 is emptied and tank 2 is filled until a configuration is achieved which mirrors the original configuration. In such a multi-phased process the supervisory controller determines set points to be achieved by the low-level control while low-level control achieves these set-points using the pumps and valves. In case of a leak in tank 1, fault-tolerance is achieved by the supervisory controller at a high level by commanding the shut-down of tank 1 and its replacement by tank 2. The low-level control reconfiguration provides fault-tolerance by choosing which pumps and valves to use at each phase in such a way that set-points are reached. For example, if valve V_2 is faulty, low level control will be implemented using valve V_{23}.

At a high level of abstraction, hybrid systems can be seen as discrete systems. At this level, it is proposed by some researchers to design supervisory decision logic for fault-accommodation which can change the control structure after the occurrence of

major faults [11]. The combinatorial explosion of the number of fault combinations will affect the size of such a discrete model, which makes this practical only if a single fault is considered at a time. The problem is that a fault accommodation strategy designed in this way, while addressing a major fault, may be sensitive to occurrences of additional changes in the system, which violate the abstracted model.

In this research, hierarchical control of piecewise-affine systems is proposed, based on partitioning the state and input space. The significance of considering the inputs when generating the discrete abstraction of the hybrid system is twofold: it is important both for reconfiguration and for limiting the complexity of the low-level control. With respect to fault-tolerant control reconfiguration, the input constraints can be interpreted as control configurations (i.e. which actuators may be used and in what range) as well as fault conditions (i.e. which actuators are fixed in position due to fault). With respect to the implementation of the low-level control, the constraints imposed on the inputs affect the complexity of the problem by determining the number of control variables that can be manipulated by the low-level controller [31]. For example, in the three-tank system there are four valves and two pumps, but as will be shown in the next section, only two of these six actuators need to be used at any given time. By not having to consider the operation of the other four "stand-by" actuators, the complexity of the low-level control is reduced.

The main contribution of this paper is the reduction in complexity of low-level control, which enables improved fault-tolerant control reconfiguration strategies for hybrid systems. Therefore my thesis statement is:

The complexity of fault-tolerant online control synthesis for a discrete-time piecewise-affine system can be reduced by constraining the operating region in

terms of states and inputs and specifying a strategy for fault-adaptive control reconfiguration as a sequence of reachability problems between convex and compact polyhedral state sets.

Contributions

This dissertation presents:

- A novel architecture for hierarchical control of hybrid systems modeled as piecewise-affine systems which enables reduction in complexity of low-level control. (Chapter V).
- A method for representing actuator configuration as constraints on the actuator bounds (Chapter VI) which enables reasoning about configuration with low complexity.
- A transformation of constrained affine systems to LTI system with origin in interior of constrained space (Chapter VI). Which enables application of methods, results and tools already developed for LTI systems (with the origin in the interior of the constrained space) to affine systems.
- Computation of backwards reachability within [i,j] steps for PWA systems in discrete time (Chapter VI)
- Supervisory control with state space partition based on reachability within a finite time window for constrained PWA systems in discrete time (Chapter VII).

Overview

Chapter II introduces the topics of fault-tolerant control, reconfigurations, and hybrid systems. In Chapter III the problem is defined and scoped. Three major topics of related work are presented in Chapter IV: invariant sets, hierarchical control, and model-predictive control. These provide the foundation for the architecture suggested in Chapter V, which is the proposed solution to the problems defined in Chapter III. Chapters VI and VII provide further detail about how reconfiguration and supervisory control are designed and performed. Conclusions and future work are suggested in Chapter VIII. Appendix A provides some mathematical definitions of concepts used in the body of this thesis. Appendix B outlines a method for generating a convex under-approximation of a union of convex polyhedra. As discussed in Chapter VIII, this topic needs further investigation.

CHAPTER II

BACKGROUND

System Models

Complex systems, which include computers and their controlling software in addition to mechanical, electrical, or chemical sub-systems are known as computer-based systems. In this sense computer-based systems are a hybrid of hardware/software and physical systems. As the cost of computing platforms decreases and software technology advances, more applications are created which embed software into physical systems. Following advances in computer communications infrastructure, computations become more distributed and their interactions more complex.

The analysis and design of dynamic systems and their behavior necessitates a modeling language with which to create models that describe the system. Control Theory traditionally focused on continuous systems, especially Linear Time-Invariant systems, modeled by a set of differential equations. For physical systems such as electrical or mechanical systems, which are governed by Newtonian physics, this is generally a suitable modeling language.

Digital computers operate in a finite and discrete domain. Finite State Automata are usually used in the realm of computer science to model behavior of computer programs. The use of Finite-State Automata dates back to the early days of computer science. More refined methods to model software behavior, such as Statecharts [12] are built on the same foundation.

When the need arises to model a computer-based system incorporating a physical system controlled by a discrete-state controller, the choice of either a discrete-state abstraction or a continuous state-abstraction of the system may not be sufficient. For a certain class of systems it may be necessary to use a hybrid of both modeling approaches. Such systems are called Hybrid Systems. Hybrid systems such as Web caching server farms, computer-controlled chemical plants, and automated highway systems all require fault-tolerant control to maintain availability and safety.

Following are some definitions regarding systems theory. A simple model of a discrete-event system is a finite automaton.

Definition 2.1 (Finite-State Automaton with Inputs). A (non-deterministic) finite state automaton with inputs is three-tuple (I, Q, E) where Q is a finite set of states and $E \subseteq I \times Q \times Q$ is the discrete transition relation. Given an initial state $q_0 \in Q$, and a sequence of inputs $\{i(t) \in I\}_{t=0}^{t=N-1}$ any sequence of states $\{q(t) \in Q\}_{t=0}^{t=N}$ that satisfies

$$\begin{aligned} & q(0) = q_0 \\ & \langle i(t), \quad q(t), \quad q(t+1) \rangle \in E \end{aligned} \tag{1}$$

is a possible state sequence of the system.

Continuous systems operate in a continuous state space, as defined below.

Definition 2.2 (Continuous Time-Invariant System). A discrete-time continuous-state time-invariant system with *n* states and *m* inputs is given by a vector field $f(\cdot,\cdot) : R^n \times R^m \rightarrow R^n$. Where R^n is the state space R^m is the input

space. Given an initial state $x_s \in R^n$ and a sequence of inputs $\{u(t) \in R^m\}_{t=0}^{t=N-1}$ the sequence of states $\{x(t) \in R^m\}_{t=0}^{t=N}$ is given by the solution of the difference equation

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= f\big(x(t), u(t)\big) \end{aligned} \tag{2}$$

Definition 2.3 (LTI System). A discrete-time linear time-invariant (LTI) system with *n* states and *m* inputs is a continuous time-invariant system given by matrices $A \in R^{n\times n}$ and $B \in R^{n\times m}$. Given an initial state $x_s \in R^n$ and a sequence of inputs $\{u(t) \in R^m\}_{t=0}^{t=N-1}$ the sequence of states $\{x(t) \in R^m\}_{t=0}^{t=N}$ is given by the solution of the difference equation

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= Ax(t) + Bu(t) \end{aligned} \tag{3}$$

Hybrid Automata

To model the continuous and discrete behavior of a hybrid system, various formalisms have been proposed [16]. A widely used model for hybrid systems is a hybrid automaton [17]. Hybrid automata are a marriage, so to speak, of finite state automata and continuous systems in continuous time. A hybrid automaton is a closed system with a discrete decision logic determining when and how the system switches between its various discrete modes, where the continuous behavior in each discrete mode is governed by a vector field.

Definition 2.4 (Hybrid Automaton). A hybrid Automaton (Q, X, E, φ, I , R , G). Q is a finite set of modes.

X is a connected subset of R^n. $E \subseteq Q \times Q$ is the discrete transition relation.

φ: $Q \times X \times \mathbb{R}$ is a flow on X, giving rise to continuous dynamics in mode q.

Where $\mathbb{R}$ represents the valuation set for the time variable.

I: $Q \rightarrow 2^X$ assigns a set of invariant states for mode $q \in Q$

R: $E \times X \rightarrow 2^X$ is a reset relation defining the possible successors $x' \in R(q, q', x)$ of a point $x \in X$ upon switching from q to q'.

G: $E \rightarrow 2^X$ assigns to each e=(q,q') a guard condition.

The combination of discrete and continuous formalisms in a hybrid automaton together gives rise to various problems relating to their interaction. Existence and uniqueness of executions is not guaranteed. Switching in a hybrid automaton is assumed to be immediate, while the time spent in each discrete state may vary. This can create a situation where infinitely many switches occur at a singular point in time, creating a cycle in which time stops. This can occur, for example, if the guard conditions for moving from on to off states in a thermostat (see Figure 2) are identical.

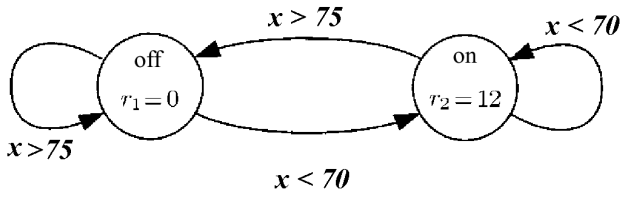

Figure 2 Thermostat Control

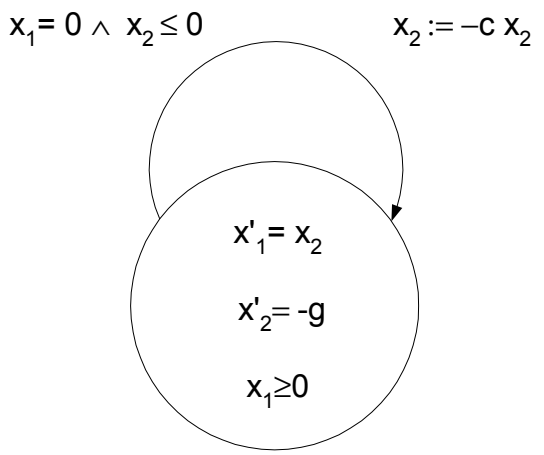

Figure 3 Hybrid Automaton of a Bouncing Ball

A more subtle but equally problematic situation is that of "Zeno" behavior[3], in which the progression of time is stopped because the hybrid automaton permits infinitely many switches to occur in a finite interval of time. The bouncing ball is an example of a Zeno system. The hybrid automaton for this system is shown in Figure 3. With a coefficient of restitution $0 < c < 1$, the time intervals between impacts will become shorter, the apex reached by the ball after each impact will be come lower, causing an infinite number of switches in a finite time interval. Time will approach a limit.

In all, hybrid systems can exhibit very complex behaviors. It was shown [18] that even for simple configurations stability and reachability analysis is an NP-hard problem or un-decidable.

[3] The Greek philosopher Zeno of the fifth Century BCE claimed that motion is impossible. Zeno's argument was that a prerequisite for reaching any point, is reaching halfway; this prerequisite can never be fulfilled, because an interval in time and space can always be partitioned into two smaller ones.

Piecewise-Affine Systems

Piecewise-affine system models are models that can approximate hybrid systems by modeling their behavior in discrete (sampled) time as well as linearizing the state evolution within each discrete mode.

Definition 2.5 (Piecewise-Affine System). A Piecewise-Affine (PWA) system is a system with continuous states $X \subseteq R^n$, and inputs $U \subseteq R^m$ operating in a hybrid state space $Q \times X$ which is described by a set of $|Q|$ affine state-space difference equations. PWA systems are described by

$$x(t+1)= A^q x(t)+ B^q u(t)+f^q \text{ if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \chi_q \tag{4}$$

where is $\chi_q \subseteq X \times U$ are convex polyhedra (i.e. given by a finite number of linear inequalities) in the state and input space. The variables $x(t) \in X$ and $u(t) \in U$ denote state and input, respectively, at time t. A PWA system is called *well-posed* if $x(t+1)$ is uniquely solvable once $x(t)$, $u(t)$ are specified.

Piecewise-affine systems can serve as reasonable approximations of hybrid automata as shown in the next example.

Example 2.1 (Bouncing Ball) consider the following PWA system with no inputs approximating the height and velocity of a bouncing ball.

$$Q = \{up, down, bounce\} \tag{5}$$

x_1 is interpreted as vertical displacement and x_2 as velocity.

The polyhedral regions are

$$\chi_{up} = \{x_1, x_2 \mid x_2 > 0\} \tag{6}$$

$$\chi_{down} = \{x_1, x_2 \mid x_2 < 0, x_1 + x_2 > 0.5\} \tag{7}$$

$$\chi_{bounce} = \{x_1, x_2 \mid x_2 < 0, x_1 + x_2 < 0.5\} \tag{8}$$

$$A_{bounce} = [1\ -c;\ 0\ -c] \tag{9}$$

$$A_{up} = A_{down} = [1\ 1;0\ 1] \tag{10}$$

$$f_{up} = f_{down} = f_{bounce} = [-0.5;\ -1]; \tag{11}$$

Figure 4 shows the system trajectory with an initial condition of (1000,0) and a coefficient of restitution $c = 0.8$

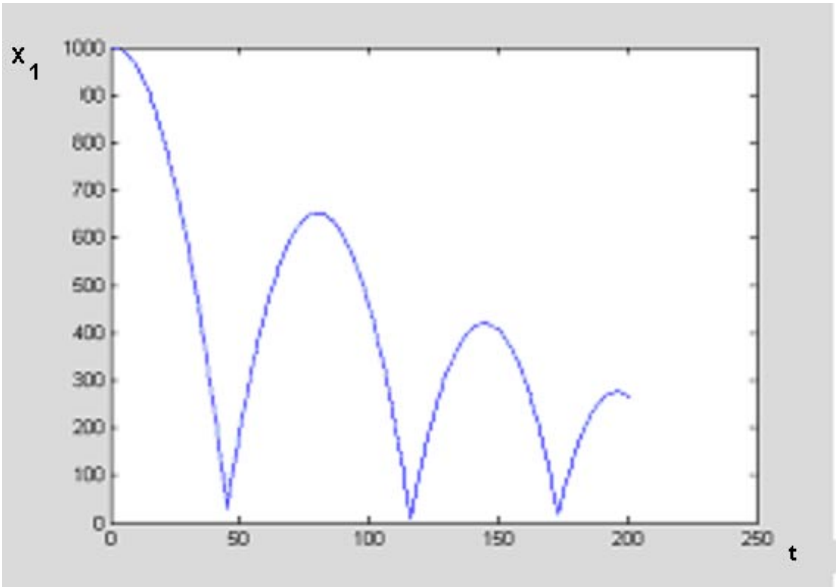

Figure 4 PWA approximation of bouncing ball

Evidently, the trajectory of Figure 4 approximates the behavior of a bouncing ball Hybrid Automaton of Figure 3. This is only an approximation, note that in Figure 4 the state does not reach the $x=[0,0]^T$ surface.

Interconnection of Automata and Discrete-Time Linear Systems

Piecewise-Affine systems can be represented as interconnections of linear systems and finite automata [4]. Consider the system (4) as an interconnection of |Q| affine systems and an automaton with finite state space Q and input-value space T.

$$\begin{aligned} x(t+1) &= A^q x(t) + B^q u(t) + f^q \\ q &= \delta(q, h(x(t), u(t))) \end{aligned} \tag{12}$$

The function h: $R^n \times R^m \rightarrow T$ provides the interface between the automaton and the continuous-state systems. The state-transition function of the automaton is $\delta : Q \times T \rightarrow Q$. By defining the functions h and δ as

$$\begin{aligned} h(x,u) &= (p_1,\ldots p_k) \quad : \forall q, A^q x + B^q u + f^q \in \chi_{p_q} \\ \delta(q,(p_1,\ldots p_k)) &= p_q \end{aligned} \tag{13}$$

The resulting system is equivalent to the PWA system (4).

Control Systems

A control system is a device that regulates a process or sequence of events. The system in which the controlled process occurs is often called the plant. Physical processes are time driven: the passage of time dictates the transfer of energy in the system. A control system interfacing with a physical plant must have means to measure the plants physical output, and produce physical input to the plant. Historically, control theory focused on control

systems, which process information using physical analog devices. Control systems which are implemented by software may include decision logic and signal processing functions. The information processing based on decision logic is more appropriately seen as an event-driven rather than a time-driven process. While the signal processing can be modeled as a time-driven process on a discrete (sampled) time line.

The combination of a time-driven physical plant with an event-driven software controller lends itself to modeling as a hybrid system. Because by such modeling the important aspects of the system's behavior are preserved without necessitating the introduction of undue complexity that would result from using a single-layer modeling approach.

Example - Thermostat

The following example is taken from [13]. The hybrid system in this example consists of a typical thermostat and furnace, which are used to control the temperature in a room. Assuming the thermostat is set at 70ºF, the system behaves as follows. If the room temperature falls below 70º F, the furnace starts and remains on until the room temperature reaches 75º. At 75º, the furnace shuts off. For simplicity, we will assume that when the furnace is on it produces a constant amount of heat per unit time. The plant in the thermostat/furnace hybrid control system is made up of the furnace and room. It can be modeled with the following differential equation:

$$\dot{x} = 0.0042(T_0 - x) + 0.1r \qquad \textbf{(14)}$$

The plant state x is the temperature of the room in degrees Fahrenheit, the input r is the voltage on the furnace control circuit, and is the outside temperature. The units for time are minutes.

Figure 2 shows the finite-state machine, which describes the operation of the controller.

Hybrid Control

A hybrid system executes in a sequence of modes along a trajectory within a hybrid continuous-discrete state space. Each mode includes a continuous evolution followed by a discrete, instantaneous transition. In a system consisting of a controller and a plant, both may have continuous and discrete dynamics

Definition 2.6 (Hybrid State Space). Let Q be a finite set of discrete states, and $\mathrm{X} \subseteq \mathbb{R}^{\mathrm{n}}$ a finite set of continuous states. Q and X respectively are the valuation sets for discrete and continuous state variables. The space $Q \times \mathrm{X}$ is a hybrid state space.

Definition 2.7 (Hybrid Trajectory) A discrete-time trajectory of a hybrid system in a space $\mathrm{Q} \times \mathrm{X}$ is a finite or infinite sequence $\{q_i \in Q, x_i \in X\}_i$

Hybrid control seeks to construct a controller such that the system, which consists of the controller and the plant, shall achieve a prescribed objective, in the sense that the hybrid trajectory of the plant shall satisfy a specified property.

Fault-Tolerant Control Architecture

Fault Tolerant Control may be obtained by a supervisory controller through online diagnosis, and subsequent remedial action. Figure 5 from [14] shows the general schematic arrangement appropriate to many fault-tolerant control systems.

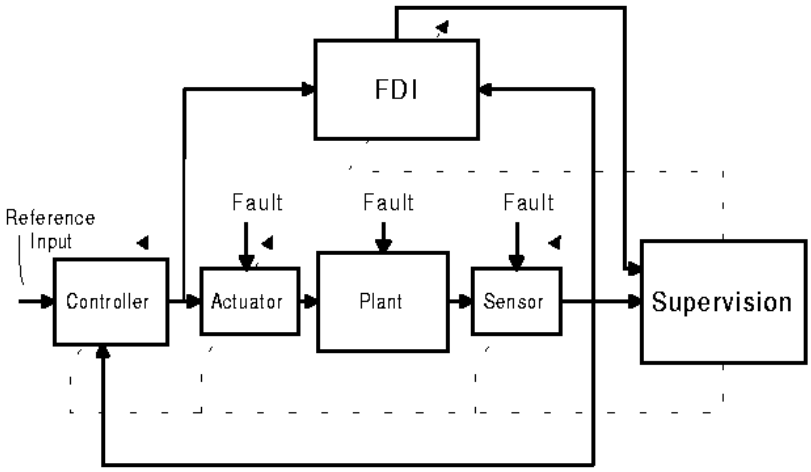

Figure 5 Fault Tolerant Control Architecture

The four main components are: the plant with its associated sensors and actuators, the fault detection and isolation (FDI) unit, the controller, and the supervisory system. The solid line represents signal flow, and the dashed line represents adaptation (tuning, scheduling, reconfiguration or restructuring).

Fault Tolerant Control

Approaches to fault tolerance can be divided into passive methods, which achieve fault tolerance by robust design, and active methods including switching control, which

achieves fault-tolerance by switching between alternative control systems and reconfigurable control, which achieves fault-tolerance via changes to the control system.

Passive Methods

Passive approaches to fault-tolerant control make use of robust control techniques to ensure that a closed loop system remains insensitive to certain faults using constant controller parameters without use of on-line fault information. The system is made robust to a restricted set of anticipated faults.

Robust control traditionally aims to maximize the permissible deviations in the performance of the plant while maintaining the control objectives. Since the passive approach to fault-tolerant control implicitly includes the function of fault-diagnosis, the scheme must also be robust against small deviations, which are harder to detect. As Patton [14] notes that this point is sometimes overlooked.

Another method of passive fault-tolerance is adaptive control. By tracking the change in system behavior the controller can adapt to degradation in system behavior and compensate for it. However, when the degradation continues, it eventually becomes impossible to compensate for the faults by means of adaptive control, and control reconfiguration or switching becomes necessary. One of the challenges in this situation is to detect degradation while still in the region where the controller is able to compensate for it; but in this region the effects of the fault may be masked by the control loop and are therefore hard to detect.

Switching Control

Switching control is a method of control by online selection between predesigned controllers.

Definition 2.8 (Switching Controller) A switching controller consists of a finite state automaton γ with a finite set of states Q, a collection of smooth vector fields $\{f_q\}_{q\in Q}$ and a switching function f: $Q \rightarrow f_q$. The switching control problem is to define γ and f, given $\{f_q\}_{q\in Q}$ such that the resulting hybrid automaton shall satisfy a specified control objective.

The switching control architecture is shown in Figure 6

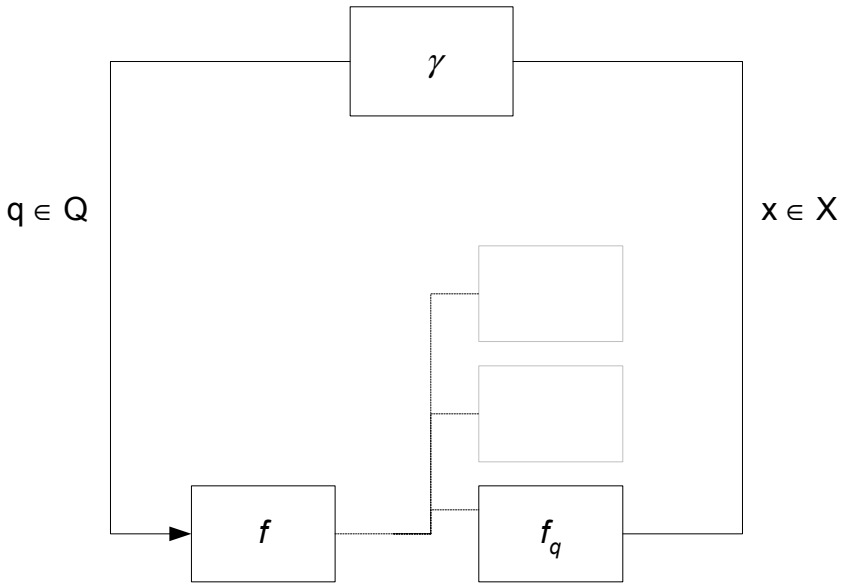

Figure 6 Switching Control Architecture

Non-linear systems can sometimes be represented as a collection of local linear models which each apply to certain operating regimes defined under constraints.

Switching control is known outside of the hybrid systems community as an approach for controlling such systems by switching controllers when an operating regime changes [20].

Morse [29] describes a high level controller called a "supervisor" which is similar to the switching controller described here. The control problem is for the output of a plant, modeled by a continuous single-input single-output linear system, to approach and track a constant reference input. The high level controller is capable of switching a sequence of controllers into the plant's feedback path so as to achieve the control objective.

The switching controller architecture can be used in the context of fault tolerant control. Asarin et al. [19] present a methodology for synthesizing switching controllers for the safe operation of systems described by linear differential equations. The approach is based on reachability analysis and the iterative computation of reachable states. They formulate the synthesis problem as finding the conditions upon which a controller should switch the behavior of the system from one "mode" to another in order to avoid a set of bad states. If we examine a hybrid system, with some of the modes considered as fault modes [26], then this methodology is applicable for recovery from such faults.

Reconfigurable Control

Reconfiguration differs from switching control in that the controllers are designed online rather than selected from a predesigned set. Following ideas from [32] the different kinds of reconfiguration can be classified according to whether or not there is a change in each of the following:

- The control objective

- The control law
- Controller-plant input output relations

The control objective is a predicate which tells which trajectories starting from the current state are acceptable. It can be in the form of a safety requirement such as never to allow the state to leave a safe envelope; it can be in the form of a reachability requirement, such as to reach a target state set. An example objective is "decrease the climb-out speed to V2". When a major fault occurs, it may be necessary to alter the control objectives accordingly. For example, for flight 191 the objective for setting the speed of the aircraft should have been different.

Achieving a new system configuration may entail several reconfiguration steps, where at each step a different set of local control objectives apply. The problem of finding the sequence of control objectives to achieve a global objective, is a planning problem. The planning problem consists of finding a way to achieve a goal, or objective, by a sequence of sub-goals.

The control law is the functional relationship between the system state (or outputs) and its control inputs. The control law can be designed based on feedback or feed-forward techniques. In feedback, the inputs are calculated as a function of the current error, defined as the distance between the actual state and target state of the system. This is a delayed reaction, since the current state is a function of previous inputs over a period of time. In feed-forward, the inputs are calculated based on projected future behavior of the system, which, in turn is based on the current and previous measurements of the state and a model of the system. Feed-forward control has the potential to be more accurate than feedback control because the additional reasoning performed by predicting

future behavior introduces more sophistication in the choice of control actions[4]. However, feed-forward control depends critically on the model which is used for forecasting.

The implementation of a control law depends on the input-output relationship between the plant and the controller. In a narrow sense this means the selection of actuators and sensors which provide the inputs to the plants and direct or indirect measurements of its state. In a broader sense, the activation of plant inputs may be tied to activation of entire subsystems of the plant. For example, a switch which turns on or off, engages or disengages a subsystem system, is a system input. Mathematically, if the system is modeled as a piecewise-affine system, the enabling of certain inputs entails the enabling of corresponding discrete modes of the system which may define entirely different behaviors for the system.

Diagnosis of PWA systems

Fault detection and isolation (FDI) constitutes a key point in active approaches to fault-tolerant control. Before activating reconfiguration, the fault has to first be detected, isolated, and its severity evaluated. Detection is the information processing task of determining whether a system or sub-system conforms to expectations. Isolation involves locating the fault sources, and evaluation consists of estimating the attributes of a fault (e.g. the degree to which a system parameter deviates from its nominal value).

[4] Consider how a human driver uses a stick-shift control while taking into account the road ahead as opposed to automatic gear system which bases its decision to switch gears only on the current situation

A decision on fault accommodation can be taken upon the availability of FDI information. The success of an active fault-accommodation method depends to a great degree on the effectiveness of the FDI unit.

Summary

This chapter introduced hybrid systems in general and the discrete-time piecewise-affine model of a hybrid system. The relevance of hybrid systems to software-based control was demonstrated. The functional elements of fault-tolerant control were discussed. These include: supervisory control, fault-diagnosis, reconfiguration and control. Some approaches to fault-tolerant control were discussed, including passive approaches that rely on robustness of the control law, and active approaches in which the control law is changed in response to results of online diagnosis and state estimation.

CHAPTER III

PROBLEM DEFINITION AND SCOPE

System Model and Assumptions

The plant is modeled as a piecewise-affine system in discrete time with additive state disturbance. It is assumed that the disturbance is bounded. It is also assumed that the state is observable, at least to the degree that thresholds can be detected.

> **Definition 2.5 (Piecewise-Affine System with Additive Disturbance).** A Piecewise-Affine (PWA) system with additive disturbance is a PWA system with discrete states Q, continuous states $X \subseteq R^n$, inputs $U \subseteq R^m$, and disturbances $D \subseteq R^n$ acting additively on the state. The system is described by
>
> $$x(t+1)= A^q x(t)+ B^q u(t) + d(t)+f^q \text{ if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \chi_q \qquad (15)$$
>
> $$d(t) \in D \qquad (16)$$
>
> where $\chi_q \subseteq X \times U$ are convex polyhedra. The set D bounds the disturbance for all discrete states $q \in Q$ of the system.

Problem

The purpose of fault-tolerant design is to enable the control system to achieve its task in the presence of faults. In the event that a fault occurs, the system moves from its nominal trajectory to a state which is not necessarily on any nominal trajectory, but may still be recoverable. The problem of fault-adaptive control is to find and implement the control

actions that would guide the system from that point back to a trajectory that is sufficiently close to a nominal trajectory satisfying the control objectives.

Multi-Phase Fault Accommodation

The architecture shown in Figure 5 suggests that two separate functions of control and control reconfiguration are involved in recovery from a fault. The supervisory controller initiates control reconfiguration in response to information from the FDI unit. After reconfiguration the reconfigured control system continues to guide the plant on a trajectory that satisfies the requirements. Figure 7 shows this concept visually. The dashed line (a) shows the trajectory that would be followed in the absence of a fault and dashed line (b) shows the trajectory that would be followed in the event of a fault and in the absence of reconfiguration.

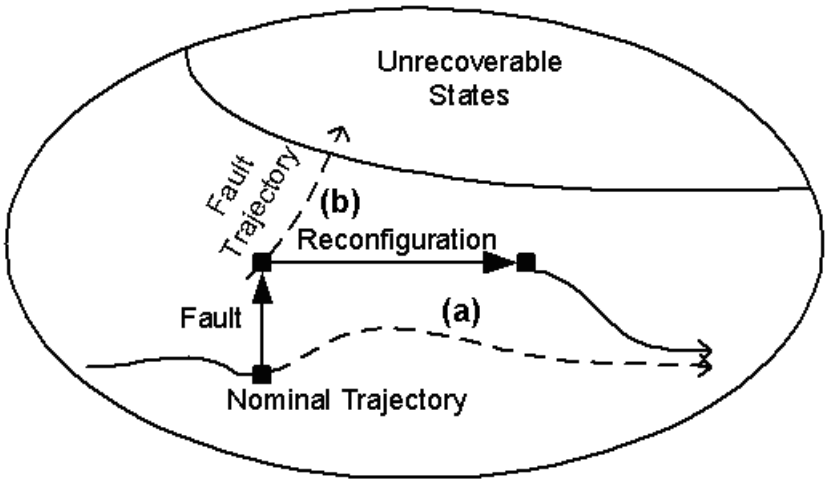

Figure 7 Single Step Reconfiguration

The picture shown in Figure 7 is in fact a simplistic one. It shows the recovery action as composed of two steps: a reconfiguration initiated by the supervisory controller followed by the control actions of the controller. A complex fault-recovery process may require more than one such iteration in response to a single fault. In absence of a one-step reconfiguration action that would produce a suitable control configuration capable of putting the system on a desired trajectory, the supervisory controller may have to engage in a planning problem of finding a sequence of reconfiguration actions, at the end of which the system is placed on a desired trajectory. See Figure 8.

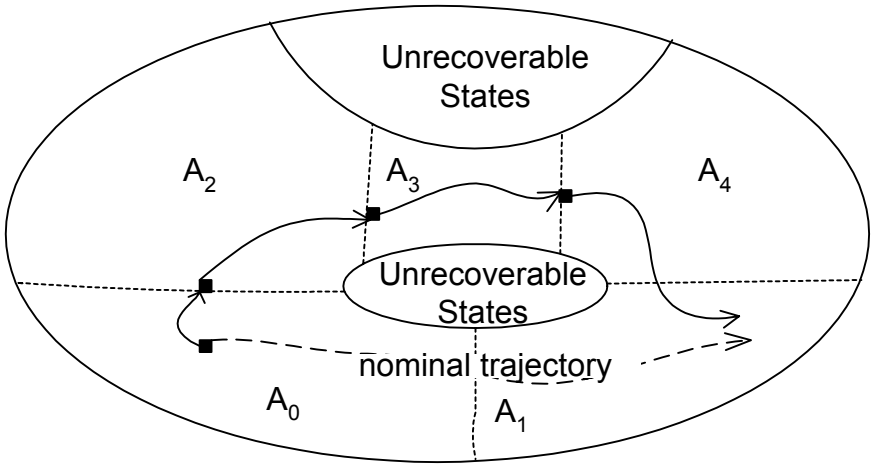

Figure 8. A multi-step reconfiguration process. Due to a fault the nominal trajectory cannot be followed. The supervisory controller dictates a multi-step process of guiding the system through regions A_2, A_3 and A_4, until the trajectory of the system returns to its regular course. At each step reconfiguration may be required.

Problem Breakdown and Scope

As stated earlier, the problem of finding the sequence of control objectives to achieve a global objective is a planning problem.

Problem 3.1 (Planning) Given a control objective O, the planning problem is to find a sequence of control objectives ($o_1,o_2,\ldots o_n$) such that when achieved sequentially, the objective O is achieved.

Planning can be performed either by an online automated planning module or a human operator, or it can be prescribed at design time. The latter shall be assumed, and the planning problem is beyond the scope of this research. The consequence of this assumption is that the design is aimed at the ability of the system to execute a limited set of pre-designed plans (i.e. sequences of objectives).

Definition 3.1 (Global Control Objective) Given a set Bad $\subseteq$ X and a finite collection of sets $A_k \subseteq X$, $k \in K$, that includes an initial set $A_0 \subseteq X$, with $A_k \cap$ Bad = $\varnothing$, a set valued map next: $K \rightarrow 2^K$ and a function time: $K \times K \rightarrow Z^+$ the global control objective is that for the system with initial conditions $x \in A_0$ the continuous state will remain in any set A_k for t time steps and subsequently leave A_k and cross into set A_k' for some $k' \in$ next(k) for which $t \leq$ time(k,k'). Formally:

$$x(t_0) \in A_k \Rightarrow \tag{17}$$

$$\exists k' \in \text{next}(k):\ \exists t \leq \text{time}(k,k')\ :$$

$$x(t_0+t) \in A_{k'} \ \wedge \forall\ 0 \leq t' < t,\ x(t_0+t') \in A_k$$

Remark 1. Definition 3.1 (Global Control Objective) applies for the nominal case. In case of a fault, which necessitates reconfiguration, a degraded performance is assumed to be acceptable in which time constraints do not apply. In this case the global control objective requires an event sequence

specified by the next relation, while the constraints specified by the function "*time"* do not apply.

Remark 2. Definition 3.1 (Global Control Objective) does not depend on the model of the system. Whether or not a control objective is achieved is an assertion about the actual system, not about its model.

The global control objective can be achieved by a series of control objectives.

Definition 3.2 (Control Objective). For a system operating in state space X, given state sets T, $\Omega \subseteq X$, a control objective is a 3-tuple (T, Ω, t). The control objective for the system at time t_0 is to reach a state $x(t_0+k) \in T$, with $x(t_0+j) \in \Omega$, $\forall\ 1 \leq j \leq k\text{-}1$, for some $1 \leq k \leq t$.

Assuming that a (possibly prioritized) list of control objectives is available and that a set $\bar{U} \subseteq 2^U$ represents all possible configurations, where U is the valuation set for the plant inputs. The problem of achieving one of the control objectives is broken down into two levels.

Problem 3.2 (Reconfiguration). Given system (1), a set of control objectives O, a state and fault detection X_e, D, U_f, and a set of possible input constraints, $\bar{U} \subseteq 2^U$ determine input constraints $U_l \in \bar{U}$ and a control objective (T, Ω, t) ∈ O, such that system (1) with constraints $u \in U_l$ and disturbance set D can be driven to target set T, within k time steps, with $1 \leq k \leq t$ while staying in Ω for the first k-1 time steps and that $u \in U_l \Rightarrow u \in U_f$.

Problem 3.3 (Low-Level Control). Determine input values $u(t) \in U_l$ needed to reach T within k time steps, with $1 \leq k \leq t$ while staying in Ω for the first k-1 time steps.

The main problem that this research addresses is how to maintain fault-tolerance while keeping the complexity of Problem 3.2 (Reconfiguration) and Problem 3.3 (Low-Level Control) low. Note that half the solution lies in the fact that Problem 3.1 (Planning) is assumed to be solved so that the control objective which needs to be addressed is relatively simple.

Summary

In this Chapter the problem with which this thesis is concerned was defined and put into context. The problem is how to achieve reconfiguration, while lowering the complexity of low-level control The context in which a multi-phase fault-accommodation scenario takes place provides the motivation to study the reconfiguration problem, where a supervisory controller is present which directs the phases of a fault-accommodation strategy, while reconfiguration provides fault-tolerance at each phase.

CHAPTER IV

RELATED WORK

Model Predictive Control of Piecewise-Affine Systems

Model Predictive Control

Model predictive control (MPC) [8] is a feed-forward optimal control technique which determines the values of control inputs to a plant by solving an optimization problem online. The objective function in the optimization problem represents a measure of forecasted behavior of the plant; the decision variables are the manipulable inputs; and the constraints represent the system dynamics. Discrete-time state-space models are commonly used to represent system dynamics because they are convenient for computer-based implementation. These system models include the standard difference equations and also constraints on the admissible values for the states and inputs. Typical measures represented in the objective function include distance of the plant output values from a target and the energy expended in the inputs.

In the following problem formulation the state vector is assumed to be identical to the output vector.

Problem 4.1 (Model Predictive Control) given $x(k)$, find $u(k)$,…u(k+P-1)

such that

$$\min F(x(P+k)) + \sum_{i=0}^{P-1} L(x(i+k), u(i+k)) \tag{18}$$

is achieved subject to constraints

$$x(k+l+1)=f(x(k+l), u(k+l)) \tag{19}$$

$$x(k+l) \in \mathrm{X},\ u(k+l) \in \mathrm{U} \qquad l=0,..,P-1 \tag{20}$$

$$x(k+P) \in \mathrm{T} \tag{21}$$

The variable P is the prediction horizon. T is the terminal constraint set, $\mathrm{T} \subseteq \mathrm{X}$. The system model is specified by $f(\cdot\ ,\cdot)$, X, and U. P, $F(\cdot)$ and $L(\cdot\ ,\cdot)$ and T are design variables. X and T are closed and U is compact. It is assumed that the origin is an equilibrium point contained in T and in the interior of X. The origin in the input space is contained in the interior of U. The aim of the control is to regulate the states and inputs to the origin. $L(\cdot\ ,\cdot)$ is a continuous non-negative, time-invariant function defined on $\mathrm{X} \times \mathrm{U}$ and $F(\cdot)$ is a continuous non-negative, time-invariant function defined on X. $F(\cdot)$ and $L(\cdot\ ,\cdot)$ achieve their minimum at the regulation point.

Mixed Logic Dynamic Systems

The Mixed-Logic Dynamical (MLD) system model [10] is a discrete-time model of a dynamic system, which is equivalent to the Piecewise Affine model [5]. The MLD model incorporates both the continuous dynamics and logic conditions as a set of mixed-integer linear constraints. This unification allows reasoning about the system dynamics using mixed-integer optimization techniques for the purpose of control, state estimation and verification [6], specifically it enables the application of Model-Predictive Control to Piecewise-Affine systems.

Hybrid systems are generally governed by interdependent physical laws, logic rules, and operating constraints. The MLD framework is based upon transforming propositional logic statements into linear inequalities involving integer variables and continuous variables. Logic propositions can be represented as integer inequalities involving integer variables that are constrained to values of 0 and 1 [37]. For example, the logical proposition $x_1 \vee x_2$ is equivalent to the integer inequality $\delta_1 + \delta_2 \geq 1$, where the integer values of 0,1 encode the logical values of *false* and *true* respectively. Other logical connectives follow in a similar fashion.

Conditions on real functions over a bounded domain are encoded by integer variables in the following manner. Given a function $f:\Re^n \rightarrow \Re$, and bounds $X^n \subset \Re^n$ on the domain, the constants $M = \max_{x \in X} f(x)$, $m = \min_{x \in X} f(x)$ are defined. A continuous inequality $f(x) \geq 0$ can be represented by an integer variable δ, such that $\delta=1$ if and only if $f(x) \geq 0$ by introducing the mixed integer inequalities:

$$f(x) \leq M(\delta\text{-}1) \tag{22}$$

$$f(x) \geq \varepsilon + (m\text{-}\varepsilon)\,\delta \tag{23}$$

where ($\varepsilon>0$) represents the numerical precision of the computation.

Example. Let $X = \{x \mid -2 \leq x \leq 4\}$, $f(x) = x^2 - 6$. The bounds on $f(x)$ are $M = \max_{x \in X} f(x) = 10$ and $m = \min_{x \in X} f(x) = -6$. Let the numerical precision be $\varepsilon=0.001$. Then the inequalities which enforce $\delta=1 \Leftrightarrow f(x) \geq 0$ are

$$f(x) \leq 10\,\delta - 10 \tag{24}$$

$$f(x) \geq -6.001\,\delta + 0.001 \tag{25}$$

Defining an auxiliary variable $z=\delta f(x)$ allows for representing functions involving both logical and continuous parts. In this relation $\delta=1$ implies $z=0$ and $\delta=0$ implies $z=f(x)$. The nonlinear equation $z=\delta f(x)$ is enforced by the following set of linear mixed-integer inequalities:

$$z \le M\delta \tag{26}$$

$$z \ge m\delta \tag{27}$$

$$z \le f(x) - m(1-\delta) \tag{28}$$

$$z \ge f(x) - M(1-\delta) \tag{29}$$

Integer variables representing each discrete mode $q \in Q$ of a PWA system are defined by associating a logical variable $\delta i(t)$ with each discrete mode, i.e. $[\delta_i(t)=1] \Leftrightarrow \left[\begin{bmatrix} x \\ u \end{bmatrix} \in \chi_i\right]$, and imposing an exclusive-or condition $\oplus_{i=1}^{s} [\delta_i(t)=1]$. By using auxiliary variables as shown above, the system equations for each mode can be activated only when the system is in the corresponding discrete mode. A more detailed explanation of the construction of mixed-integer inequalities from the logical and dynamical equations is found in [37]. The resulting model is the MLD model of the hybrid system, shown in equation (30).

$$\begin{aligned} x(t+1) &= Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) \\ E_2\delta(t) + E_3z(t) &\le E_1u(t) + E_4x(t) + E_5 \end{aligned} \tag{30}$$

Where each of the vectors x(state), u (input), is partitioned into discrete and continuous components, e.g.

$$x = \begin{bmatrix} x_c \\ x_l \end{bmatrix}, \quad x_c \in \mathrm{R}^{n_c}, \quad x_l \in \{0,1\}^{n_l}, \quad n \equiv n_c + n_l \quad \delta \in \{0,1\}^{r_l} \quad z \in R^{r_c} \tag{31}$$

x are the continuous and binary states, u are the continuous and binary inputs, δ and z represent binary and continuous variables respectively. Outputs can be added to the model as an additional vector $y(t)$ and an additional set of equations. Fault conditions can be added as an additional input vector $\phi(t)$, with corresponding terms in the equations and inequalities [6]. A well posed MLD system results in a unique solution to the inequalities effectively defining the system model ($f(\cdot\ ,\ \cdot)$, X, U) which can be used for Model Predictive Control. Equations (32), (33) constitute a model-predictive control problem with a MLD system model

$$\min J = \sum_{t=0}^{P-1} \left\| u(t) - u_f \right\|_{Q_1}^2 + \left\| \delta(t) - \delta_f \right\|_{Q_2}^2 + \left\| z(t) - z_f \right\|_{Q_3}^2 + \left\| x(t) - x_f \right\|_{Q_4}^2 \tag{32}$$

subject to

$$\begin{aligned} & x(t+1) = Ax(t) + B_1 u(t) + B_2 \delta(t) + B_3 z(t) \\ & E_2 \delta(t) + E_3 z(t) \le E_1 u(t) + E_4 x(t) + E_5 \\ & x(P-1) = x_f \end{aligned} \tag{33}$$

where

$$\|x\|_{Q_4}^2 = x^T Q x\ ,\ Q_i = Q_i^T \ge 0,\ i = 1, \ldots 5, \tag{34}$$

are given weight matrices.

The target values x_f, u_f, δ_f, z_f, are chosen to be consistent values corresponding the desired regulation point.

The complexity of the MPC optimization problem is a function of the prediction horizon P, the number of manipulable inputs u, and the complexity of the MLD model. For a given system, the complexity can be reduced by reducing the number of manipulable variables (i.e. choosing fixed values for non-manipulable variables) and

reducing the prediction horizon. This will reduce complexity and also lead to a less optimal solution, and possibly to infeasibility of reaching the desired set point within the required time.

Tsuda et al. [31] propose using MPC for reconfiguration. The MLD model is extended to include faults and controller configurations. In the event of a detected fault, the MPC algorithm searches for the best possible values of all input variables, rather than just the input variables which are used under nominal control conditions. There are two problems with this approach: one which is addressed in [31] is that the complexity of the MPC problem is increased by expanding the number of manipulable variables. The second problem is that the prediction which may be needed for reconfiguration might be further than the prediction needed for nominal control. Unless the reconfiguration consists of replacing failed actuators with stand-by actuators performing almost the same function, it is likely that a change in set-point will be needed. Recovery from a fault may involve a multi-stage process by which short-term goals are compromised in favor of long term goals. A reconfiguration method, which seeks to optimize the same objective function as is used for control, will not be suited to such a situation. The MPC-based reconfiguration alone is therefore limited to short-term reasoning, and can benefit from a reduction of the search space in the form of a reduced set of manipulated input variables.

The issue is therefore how to reduce complexity while maintaining feasibility of the model-predictive control solution. The next section presents the theory of invariant sets which is needed to address the feasibility issues. The following section deals with hierarchical control, which is employed in this research as a means to deal with the complexity issues.

Set Invariance in Control

In control theory robustness to disturbances and uncertainty has often been studied by assuming a noise model and analyzing the behavior of a system based on random signal theory. A different approach, which was studied as far back as 1971 [9] is that of determining the system behavior under all allowable disturbance sequences. More specifically, determining the subset of the state space from which the system can be steered to a given target state set, while guaranteeing that the state and input constraints will be satisfied for all allowable disturbance sequences. A comprehensive survey of papers on set invariance is given in [38]. The notation used here follows that of [15].

System Models

The following discrete-time system models are defined.

Definition 4.1 (Constrained System) A constrained continuous-state discrete-time time-invariant system is given by a vector field $f(\ ,\ ,\ )$, a state set $X \subseteq R^n$, an input set $U \subset R^m$ and a disturbance set $D \subset R^l$. The set U is compact and the sets X and D are closed. The evolution of the system is given by

$$x(t+1) = f(x(t), u(t), d(t)) \tag{35}$$

with $t \in Z$. The following constraints apply for all $t \in Z$

$$\begin{array}{ll} \text{state} & x(t) \in X \\ \text{input} & u(t) \in U \\ \text{disturbance} & d(t) \in D \end{array} \tag{36}$$

Definition 4.2 (Allowable Disturbance) An allowable disturbance sequence for a constrained system is one that satisfies the constraint $d(t) \in D$

Definition 4.3 (Admissible Input) An admissible input for a constrained system is one that satisfies the input constraint $u(t) \in D$

Definition 4.4 (Constrained System with Additive State Disturbance) A constrained continuous-state discrete-time time-invariant system with additive state disturbance is given by a vector field $f(\cdot,\cdot)$, a state set $X \subseteq R^n$, an input set $U \subset R^m$ and a disturbance set $D \subset R^n$. The set U is compact and the sets X and D are closed. The evolution of the system is given by

$$x(t+1) = f(x(t), u(t)) + d(t) \tag{37}$$

Definition 4.5 (Autonomous System). An autonomous system with disturbance is given by

$$\begin{aligned} x(t+1) &= f\big(x(t), d(t)\big) \\ d(t) &\in D \end{aligned} \tag{38}$$

The One-Step Set

The one-step set represents backwards reachability of a system.

Definition 4.6 (One-Step Set). [15] For a constrained system with inputs $u(t) \subseteq U_l$ and disturbances $d(t) \equiv 0$, the *one-step set* $Q(\Omega)$ is the set of states in X

for which an admissible control input exists which will drive the system to Ω in one step

$$Q(\Omega) = \{ x \in X \mid \exists u \in U: f(x, u) \in \Omega \}. \tag{39}$$

For a piecewise-affine system the one step set is

$$Q(\Omega) = \{ x \in X \mid \exists u \in U, \exists q \in Q : (x, u) \in \chi_q, A_q x + B_q u + f_q) \in \Omega \}. \tag{40}$$

Proposition 4.1 [23] if Ω is given by the union

$$\Omega = \bigcup_i \Omega_i \tag{41}$$

then

$$Q(\Omega) = \bigcup_i Q(\Omega_i) \tag{42}$$

Definition 4.7 (Robust One-Step Set). [15] For a constrained system, with inputs $u(t) \subseteq U_l$ and disturbances $d(t) \in D$, the *robust one-step set* $\tilde{Q}(\Omega)$ is the set of states in X for which an admissible control input exists which will drive the system to Ω in one step, for any disturbance $d(t) \in D$ i.e

$$\tilde{Q}(\Omega) = \{ x \in X \mid \exists u \in U: f(x, u) \in \Omega \}. \tag{43}$$

Proposition 1 [23] For an LTI system with additive state disturbance, the robust one-step set has the following properties:

1. If Ω is compact then the set $\tilde{Q}(\Omega)$ is closed.

2. If Ω is convex then the set $\tilde{Q}(\Omega)$ is convex.

3. If Ω is a polyhedron then the set $\tilde{Q}(\Omega)$ is a polyhedron.

4. If A is non singular and Ω is compact, then the set $\tilde{Q}(\Omega)$ is also compact.

Proposition 4.3 [23] For all Ω_1, Ω_2

$$\Omega_1 \subseteq \Omega_2 \Rightarrow \tilde{Q}(\Omega_1) \subseteq \tilde{Q}(\Omega_2) \tag{44}$$

Positively Invariant and Control Invariant Sets

Positively invariant sets and control invariant sets are sets in which an autonomous or controlled system, respectively, can remain for all disturbances.

Definition 4.8 (Robust Positively Invariant Set). [38] The set $\Omega \subset R^n$ is a *robust positively invariant set* for an autonomous system if and only if

$$x(t) \in \Omega \Rightarrow x(t+1) \in \Omega \quad \forall d \in D \tag{45}$$

Proposition 4.5 The union of two robust positively invariant sets is a positively invariant set.

Definition 4.9 (Maximal Robust Positively Invariant Set). [23] The set $\tilde{O}(\Omega) \subset R^n$ is the *maximal robust positively invariant set* for an autonomous

system if and only if it is robust positively invariant and contains all the robust positively invariant sets contained in Ω.

Definition 4.10 (Robust Control Invariant Set). [38] The set $\Omega \subset R^n$ is a *robust control invariant set* for a constrained system if and only if there exists a feedback control law $u(t)=h(x(t))$ such that Ω is a robust positively invariant set for the closed-loop system

$$x(t+1)=f(x(t),h(x(t)),d(t)) \tag{46}$$

and

$$u(t) \in U \; \forall \; x(t) \in \Omega \tag{47}$$

Remark 4.1 The phrase 'exists a feedback control law' does not necessarily mean to imply any structure on the control law. The function $h(x(t))$ can be constructed to serve as a feedback law by finding for each $x(t) \in \Omega$ a value $u(t) \in U$ such that $x(t+1)=f(x(t), u(t), d(t)) \; \forall \; d(t) \in D$.

Proposition 4.5 The union of two robust control invariant sets is a control invariant set.

Definition 4.11 (Maximal Robust Control Invariant Set). [40] The set $\tilde{C}(\Omega) \subset R^n$ is the *maximal robust control invariant set* for a constrained system if and only if it is robust control invariant and contains all the robust control invariant sets contained in Ω.

The following result follows immediately from the definitions.

Proposition 4.6 Given a constrained system, there exists an admissible control law such that the state constraints can be satisfied for all time $t \in \mathrm{N}$ and for all allowable disturbance sequences if and only if the initial state $x_0 \in \tilde{C}(X) \subseteq X$.

The following is simple condition that checks if a set is control invariant..

Theorem 4.1 (Geometric Condition for Invariance) [41] The set $\Omega \subset R^n$ is a robust control invariant set for a constrained system if and only if

$$\Omega \subseteq \tilde{Q}(\Omega) \tag{48}$$

Corollary 4.1 The set $\Omega \subset R^n$ is a robust control invariant if and only if

$$\Omega = \tilde{Q}(\Omega) \cap \Omega \tag{49}$$

Robust Controllable Sets

Controllable sets are sets from which a system can be driven to a target set.

Definition 4.12 (*i*-step Robust Controllable Set). [23] The set $\tilde{K}_i(\Omega,T) \subseteq R^n$ is a *robust controllable set* for a constrained system if and only if the system can be driven to T in *i* steps while not leaving Ω for the first *i*-1 steps, for all allowable disturbance sequences.

Remark 4.2 $K_i(\Omega,T)$ denotes $\tilde{K}_i(\Omega,T)$ for $d(t) \equiv 0$.

Feasibility of Model Predictive Control

A model predictive control problem may be infeasible if there is no solution within the constraints specified by the sets X, T, and U. The feasible set $X_F(T, P)$ of the MPC control problem is the set of states $x(k)$, for which a feasible control sequence to the MPC control problem exists. A feasible control sequence is a sequence of inputs $u(k+l) \in U$, l =0,..,P-1 driving the system $x(k+l+1)=f(x(k+l), u(k+l))$ from to $x(k)$ to $x(k+P)$ such that $x(k+l) \in X$ and $x(k+P) \in T$.

Theorem 4.2 (Feasibility of Model Predictive Control)[5] [15] The feasible set $X_F(T, P)$ of the MPC problem is given by

$$X_F(T, P)=K_P(X,T) \tag{50}$$

Proof From the constraints of the MPC problem, the solution has to satisfy $x(k+P) \in T$, it is also required that $u(k+l) \in U$ and $x(k+l) \in X$ for all l=0,..,P-1. It follows that there exists a control sequence of length p, such that these constraints can be satisfied if and only if $x(k) \in K_p(X,T)$, where $K_p(X,T)$ is the p-step robust controllable set.

The above theorem can be extended to the case of robust feasibility: the MPC problem is robustly feasible if it is feasible for all disturbance sequences within given constraints and the feasible set for the MPC problem is given by the robust controllable set. Given the feasible set for the nominal MPC problem, robust feasibility of the MPC problem for PWA systems with polyhedral constraints and additive state disturbance can

be tested without the need to compute the robust controllable sets [24]. This is based on the observation that the MPC controller is robustly feasible if and only if the nominal feasible set is a robustly positively invariant set for the closed loop system. By this observation, given the nominal feasible set, it is only necessary to test one-step reachability.

Supervisory and Hierarchical Control

Many control design problems for complex dynamic systems can approached by modeling the plant as a Discrete Event System. It is possible that the same system can be modeled as a continuous system, a hybrid system, or a discrete-event system, all depending on the purpose that the model serves. The starting point for modeling an actual plant is a model that is identified experimentally, derived from first principles or both. This model is referred to as the 'real' system model. The model of the 'real' system can then be further *abstracted* to fit the task at hand. Loss of model precision in the abstraction process is permitted to the degree that it does not affect the purpose of the model. On the other hand, it is necessary to decrease precision in order to meet computational complexity requirements. An illustrative everyday example is motion planning: driving directions can generally be given in terms of left and right turns and distances, while parallel parking instructions require much more detail; it is not reasonable, however, to prescribe driving directions with the same detail because of the resulting complexity. Hierarchical control is an approach that uses increasingly abstracted

---

[5] This is a simplified version of Theorem 3.1 from [23], for the case of equal prediction and control horizons.

models for tasks which require planning further into the future, but with decreasing precision.

Supervisory control is the application of a DES controller to a dynamical system. When the 'real' plant model is a hybrid system, it may be abstracted as a Discrete-Event System for the purpose of supervisory control. A more specific case is when the plant, or the plant lumped with previously-designed controllers is modeled as a continuous system, and the supervisory controller is a discrete-event system. The plant and controller are connected via interfaces mapping the countable set of states of the DES to the non-countable set of states of the continuous system. In both cases, the closed-loop system comprised of the controller and plant is a hybrid system.

Supervisory Control

The Supervisory Control Architecture

The supervisory control system consists of a plant defined by a differential equation $\dot{x} = f(x,u)$, with, $x \in$ X, $u \in$ U, an *actuator* map β: C$\rightarrow$ U, which generates a piecewise constant signal in U from the control alphabet C, of the supervisory controller, and a *generator* map α: X$\rightarrow$ P. The function α determines a finite partition of the plant state space X with equivalence classes $A_p = \{x \in \text{X} \mid \alpha(x)=p\}$ indexed with plant events $p \in$ P. The supervisory control architecture is depicted in Figure 9. The supervisory controller is a finite state automaton, possibly non-deterministic. For the purposes of the following proposition, only the input-output relation γ: P$\rightarrow 2^{\text{C}}$ describing the supervisory controller is needed, which in general is set-valued even when the controller is deterministic.

Proposition 4.7 [16] Given a control loop L described by α, β, γ, f as in Figure 9, there exists a hybrid automaton H, such that the trajectories of H are in one-to-one correspondence with the closed-loop trajectories of L.

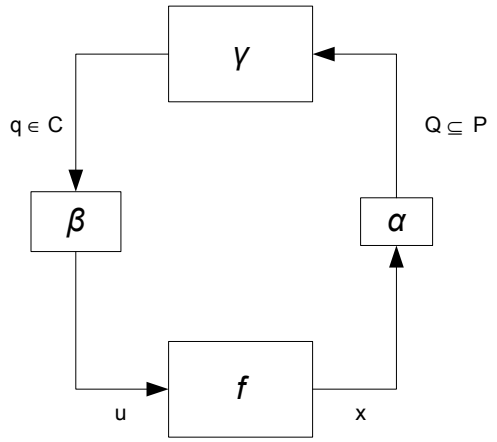

Figure 9 Supervisory Control

The system defined by this architecture specifies a hybrid automaton. The supervisory control problem is to define α, P, and γ, given C, β and f, in such a way, that the resulting hybrid automaton shall satisfy a control objective, e.g. the event sequence control objective mentioned above.

Discrete Abstractions of Continuous Systems

Consider the architecture of Figure 9 with a set P which includes a silent event ε. The function α: X→ P generates a non-silent event when boundaries in X are crossed in a specified direction [13]. The continuous system $\dot{x} = f(x,u)$ with the actuator β and generator α, form a non-deterministic finite state automaton G = (S, P, C, ψ, λ) where S is the discrete plant state, P is the set of plant symbols, C is the set of control symbols, ψ:

$S \times C \rightarrow 2^S$ is the state transition function, and λ: $S \times S \rightarrow 2^P$ the output function. The state $s \in S$ is corresponds to the most recently entered region of the plant state space.

The Discrete Event System plant model G is a non-deterministic automaton. This abstraction permits application of discrete domain methods for designing the controller to satisfy control specifications using DES control methods. The key to successful design is to choose the state space partition such that the properties of interest are preserved [21]. It should be possible to design the supervisory controller to achieve the desired closed-loop behavior despite the non-determinism that is introduced in the abstraction process.

The discrete abstraction approach, presented above for the continuous system, can be applied to hybrid systems in general. Given a hybrid system and some desired property, one extracts a finite, discrete system while preserving all properties of interest [22]. Once the discrete abstraction is obtained discrete-domain methods can be used for control design.

Hierarchical Control of PWA systems

Supervisory control is concerned with the control of continuous (or hybrid systems) by abstracting them into discrete-event systems. The continuous system is driven by a piece-wise constant input signal, which may be processed at the input to the continuous system to take a different form e.g. a ramp, but nonetheless is determined by the supervisory controller. Hierarchical control [7], in contrast, delegates some of the control task to continuous-state controllers, which form a layer between the plant and the supervisory controller. It is assumed that some given closed-loop specifications are satisfied by the combination of the plant and the continuous-state controllers. The supervisory controller

is designed with relation to the closed-loop specifications but without knowledge of the control laws governing the controllers' behavior.

Summary

This Chapter surveyed related work. The work presented provides three major building blocks which are put to use to solve the reconfiguration problem.

The theory of invariant sets in control provides the mathematical tools for calculating necessary conditions for existence of a control law in a constrained system.

Hierarchical control is provides the framework in which a supervisory controller transitions between modes, based on existence of control inputs that guarantee reachability in a constrained system.

Model-Predictive Control serves as a candidate implementation of low-level control in the hierarchical control architecture presented in the next Chapter. The interest in Model-Predictive Control is that its complexity problems can be alleviated using the methodology of this thesis and that necessary and sufficient conditions for feasibility of MPC are known, which are based on invariant sets.

CHAPTER V

ARCHITECTURE

Proposed Architecture

The control architecture shown in Figure 10 describes the proposed hierarchical fault-tolerant control architecture.

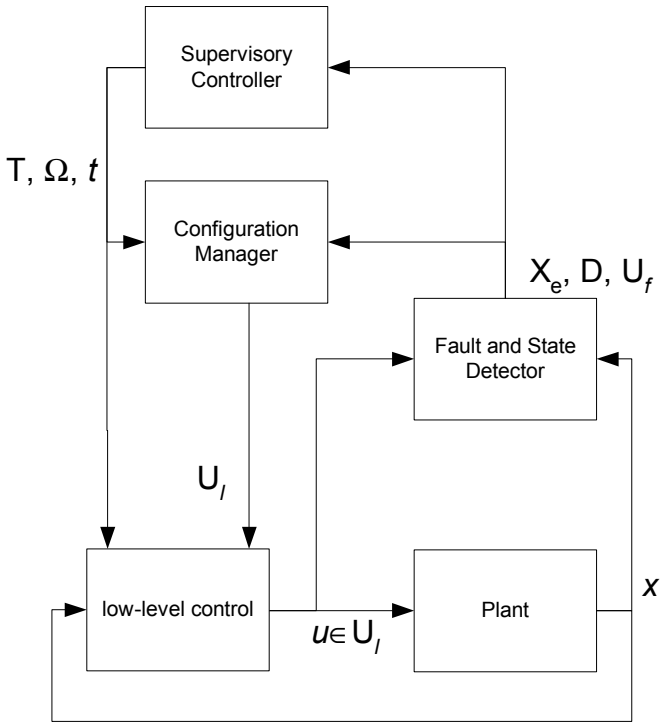

Figure 10. Architecture

The plant is modeled as a piecewise-affine system as defined in Definition 2.5 with continuous states $X \subseteq R^n$, a finite set of discrete states Q, and inputs $U \subseteq R^m$ operating in

a hybrid state space $Q \times X$. An additive state disturbance is assumed, taking values in a polyhedral region $D \subseteq R^n$. Actuator faults are manifested as limitations which constrain the input values to a reduced input set $U_f \in U$.

The fault and state detector identifies the plant state as a set $X_e \subseteq X$ which determines the possible values of the state vector $x(t)$. The state detection is based on partitioning the state space and generating an event when partitions are crossed, in a similar fashion to what was described in the previous chapter. The fault and state detector also determines the disturbance set D and the fault-induced input constraints U_f.. The sets X_e, D, U_f are assumed to be available correct conservative approximations, which are continually updated at each time step. All the sets are assumed to be convex polyhedra.

The system is designed with respect to a global control objective as defined in Definition 3.1 (Global Control Objective). Based on the global control objective, the supervisory controller determines a control objective as in Definition 3.2 (Control Objective) for the low level control and the configuration manager.

The supervisory controller specifies a set of alternate control objectives. The set of control objectives can be passed on to the lower levels as a prioritized list. If the objectives are not prioritized an arbitrary prioritization will be imposed in order to choose one control objective. The problem of achieving one of the control objectives is broken down into two sub-problems.

Problem 5.1 (Reconfiguration). Given system (1), a set of control objectives O, a state and fault detection X_e, D, U_f, and a set of possible input constraints, $\bar{U} \subseteq 2^U$ determine input constraints $U_l \in \bar{U}$ and a control objective $(T, \Omega, t) \in$

O, such that system (1) with constraints $u \in U_l$ and disturbance set D can be driven to target set T, within k time steps, with $1 \leq k \leq t$ while staying in Ω for the first k-1 time steps and that $u \in U_l \Rightarrow u \in U_f$.

Problem 5.2 (Low-Level Control). Determine inputs $u(t) \in U_l$ needed to reach T within k time steps, with $1 \leq k \leq t$ while staying in Ω for the first k-1 time steps.

The low-level control problem is solved continuously by the low-level control module. When a control objective is achieved, the supervisory controller sets a new set of control objectives. Reconfiguration occurs when either of the following happens:

- The set of control objectives specified by the supervisory controller is changed, and no longer includes the current objective.
- The fault-induced input constraints become more restrictive and violate the current configuration. (i.e. $U_l \not\subseteq U_f$.)
- The disturbance set becomes larger and violates the current configuration.

When reconfiguration occurs, the configuration selects one of the control objectives from the set specified by the supervisory controller, and selects input constraints $u \in U_l$ for reconfiguration.

Motivating Example

To understand better what is proposed for hierarchical control architecture it is useful to compare it to the state of the art. Rather than presenting the latest mathematical advances

in hybrid systems control, this section will present a real-life system in development. The system is a simplified model of an aircraft fuel system.

The aircraft fuel system consists of six of interconnected tanks of fuel shown in Figure 11

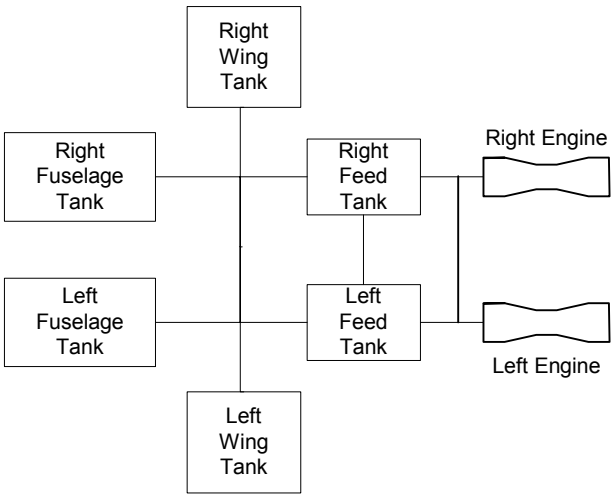

Figure 11. Simplified Aircraft Fuel System

The six tanks are in order: the left and right feed tanks which directly feed the engines, the left and right wing tanks located at the wings, and the left and right fuselage tanks located at the central body of the aircraft. The state of the system is given by a vector $x(t) = [x_1(t), x_2(t), x_3(t), x_4(t), x_5(t), x_6(t)]^T$ which is interpreted as the amount of fuel in each of the six tanks. The inputs, $u(t)$, to the control system are commands to valves and pumps, which control the flow between the interconnected tanks. The discrete state of the system $q(t)$ is mainly a function of the positions of the valves. The fuel consumption is modeled as a disturbance $d(t)$ which acts additively on the state $x(t)$. Clearly, this system falls

nicely into the model (1) if the evolution of $x(t)$ in each discrete state can be modeled as discrete-time affine function. The model will not be detailed here, but it can be readily obtained from data or from a hybrid model.

Ample fuel supply to the engines is provided by keeping the feed tanks almost full at all times. Center of gravity is maintained by following the global control objective defined as follows:

> **Definition 1 (Fuel System Global Control Objective).** The state of the fuel tank system x will start at $x_0 \in A_1$, as defined in Table 1. When the state leaves A_1 it will enter A_2, and so forth until it reaches $[0, 0, 0, 0, 0, 0]^T \in A_6$.

Obviously, the state of zero total fuel should never be reached, but this is not a fuel-system control issue, but a mission planning issue. The regions A_k are given in Table 1.

The control system operates pumps and valves (not shown in Figure 11) to move fuel from more remote fuel transfer tanks to engine feed tanks. The primary objective is to keep the engine feed tanks near full at all times, such that if failure occurs upstream, as much fuel as possible is available directly to the engines. The secondary objective is to transfer fuel in a sequence which results in proper aircraft center of gravity.

The design of the control system for controlling the pumps which transfer fuel from the transfer tanks is based on decision logic which implements the transfer sequence of Table 1. For example, when the system is in regions A_3, A_4 and A_5 fuel is pumped from the wing tanks according to the logic in Figure 12.

Table 1**.** Fuel Quantity (lb.) for each tank in the simplified Aircraft Fuel System, the configurations are symmetric.

Region	Total		Left/ Right Feed		Left/Right Wing		Left / Right Fuselage	
	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
A_1	11200	12000	1500	1500	1600	2000	2500	2500
A_2	11000	11200	1500	1500	1600	1600	2400	2500
A_3	7800	11000	1500	1500	800	1600	1600	2400
A_4	4600	7800	1500	1500	0	800	800	1600
A_5	3000	4600	1500	1500	0	0	0	800
A_6	0	3000	0	1500	0	0	0	0

```
Iff
   Fuselage Total Quantity - 800 - Wing Total Quantity < 400
Then
   Pump From Fuselage Tanks

Iff
   Wing Total Quantity + 800 - Fuselage Total Quantity < 400
Then
   Pump From Wing Tanks
```

Figure 12 Example Decision Logic for Fuel System Control

The feed tanks must provide uninterrupted fuel supply to the engines at all times to prevent engine failures. The left and right feed tanks, provide fuel to the left and right engines, respectively. The tanks are interconnected to provide redundancy in case of

failures. Fuel flows from the feed tanks to the engine, and can be boosted by a boost press system on each feed tank. The feed line between the feed tanks and the engines also has an interconnect to provide redundancy in case of a boost press failure.

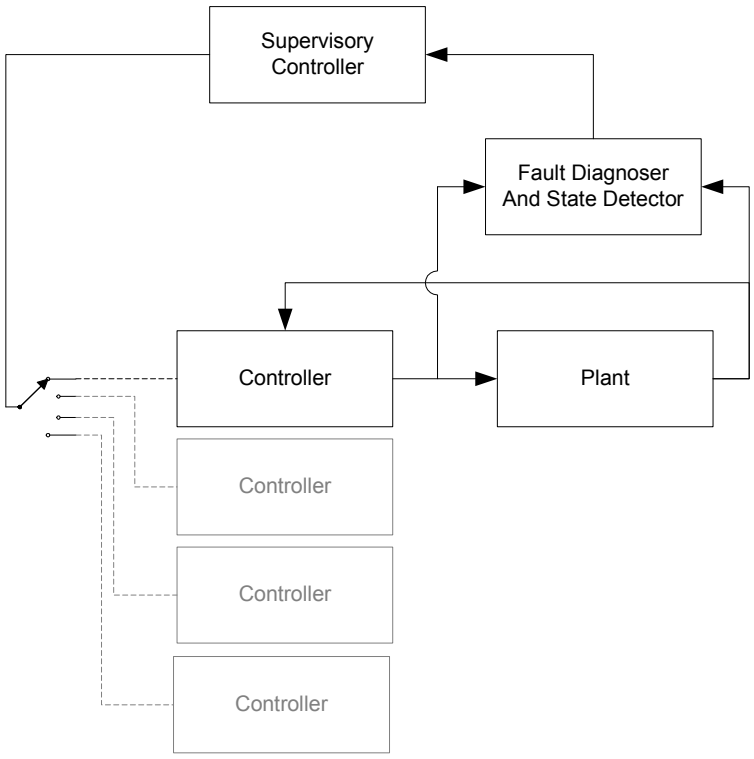

Figure 13**.** Architecture of the Aircraft Fuel Control System

Conceptually, the architecture for the control system is shown in Figure 13. The architecture shown in Figure 13 is essentially a switching control architecture similar to that of Figure 6. In this system the decision logic located at the "supervisory controller" of Figure 13 determines one of several control laws depending on the fault conditions and the region in the state space. This addresses the supervision problem (determining the set-

points for each tank), the control problem (finding a control law that achieves the set-point), and the reconfiguration problem (finding the set of actuators – pumps and valves) at the same level. All three are solved by the supervisory controller which, by applying a pre-designed decision logic, determines one of a limited set of control laws to use. The choice of actuators is predetermined for each control law.

The implementation that relies entirely on predetermined logic decisions has the advantages of being relatively straightforward to verify. However, the number of fault scenarios for which such a system can be made tolerant is limited. Complications arise especially when multiple faults occur. For example, the feed line interconnect is opened in case of a boost system failure so that fuel can flow from both feed tanks to both engines, thus providing redundancy. But a fall in feed tank pressure resulting from a leak can also trigger the same response, thus causing even more fuel to be lost through the leak.

Accounting for multiple-fault scenarios is an inherent difficulty in the design of fault-tolerant systems. In some cases, such as the flight 171 example cited in the introduction, the system does in fact posses robustness attributes that can enable it to overcome a multiple-fault event, but the in the design of the fault-tolerant control system this robustness is not exploited because it is not exposed. This is one of the issues that can be addressed by the methods developed in this research.

The next section presents a different system from the same domain of flow-control.

Benchmark Problem

The three-tank system shown in Figure 1 was developed as a benchmark problem for the European Control of Complex Systems (COSY) project [28]. Several research papers have been written, which consider control reconfiguration in the event of faults as applied to this benchmark example [11][31][35][36]. To the best of this author's knowledge the only work in the COSY project which considers hierarchical control of this benchmark problem is [33], which is summarized in [28]. The hierarchical reconfiguration method presented in [33], in contrast to the method presented here, is validated only by trial and-error. The non-hierarchical methods suffer from the problems mentioned earlier, namely poor tradeoff of the quality of control.

In the three-tank system the objective is to regulate the level of fluid in tank 3. The nonlinear continuous-time hybrid model is detailed in [43]. An approximation of this hybrid system as a mixed-logical dynamic system is given in [39]. Without going into all the details, the features of the system are these: the system is characterized by discrete modes of operation resulting from the levels of the fluid in each tank (above or below the upper pipe in each tank, $2^3 = 8$ possibilities), the relative fluid level between each two neighboring tanks (two pairs of tanks: :in each, the right can be fuller than the left or vice versa: $2^2 = 4$ possibilities) and the position of each valve (four valves, each with an open and closed position: $2^4 = 16$), in all there are 193 distinct continuous behaviors (of the 512 discrete modes, most are redundant). The flow Q of fluid out of a tank through an open valve is determined by Toricelli's law:

$$Q = a_z \cdot S \cdot \sqrt{2 \cdot g \cdot h} \tag{51}$$

where a_z is a flow correction term, S the cross-section of the valve, *g* the gravity constant and *h* is the level of water in the tank. In case both tanks have fluid above the level of the valve, the flow is directed from the fuller tank to the emptier one and *h* is the level difference between the tanks, i.e.. $h=|h_i-h_j|$. The piecewise affine model is generated by enumerating all the discrete modes which are linked to the values of the state variables (fluid levels) and inputs (valve control commands), and by linearizing and discretizing the nonlinear continuous behavior of the flow.

Control of the three-rank system is as follows. In the nominal case, Tank 1 serves as a buffer tank, and tank 3 is regulated by controlling the flow between tanks 1 and 3 using valve V_1. One of the possible faults in the system is a leak in Tank 1. The scenario for control and reconfiguration of this system is shown in Figure 15. The control objectives (Ω, T, t) and input constraints U_1 for this scenario are shown in Table 2.

The results in Figure 3 and Figure 4 were obtained using the model in [43]. The low-level control was implemented using PI controllers on the pumps, hysteresis switches on the valves, and additional simple switching elements. The scenario is comprised of four phases:

I. The system starts with all tanks empty. Tank 1 and tank 3 are filled to their nominal levels.

II. The system is regulated at the nominal levels around $h_1=0.5$, $h_2=0$, $h_3=0.1$.

III. Following the detection of a leak in tank 1, the supervisory controller sets the control objective to filling tank 2, while regulating tank 3 and emptying tank 1.

IV. The system is regulated around the set-point h_1=0, h_2=0.5, h_3=0.1, which mirrors the regulation of phase II

For the valves, a value of 1 is interpreted as the valve open, and a value of 0 as closed. Note that V_{23} is never opened in the configurations detailed in Table 1. This means that the configurations are tolerant to faults which cause V_{23} to be permanently closed. Note also that only two actuators are used in each phase.

In phase III shown in Table 2, the supervisory controller specifies a control objective of reaching the neighborhood of h_3=0.1, h_2=0.5, and for phase 4, regulation around that point. Three alternate points specified in order of descending priority are

- h_3=0.1, h_2=0.3,
- h_3=0.1, h_2=0.2,
- h_3=0.0, h_2=0.0.

The last option is a shutdown, a safe state, which covers the case where no other objective is achievable. Consider two cases where reconfiguration is necessary:

1. Valve V_2 is faulty. The configuration shown in Table 1, phase 4, is no longer valid as it requires V_2 to be manipulable. The configuration manager selects a configuration which uses V_{23} instead of V_2 to achieve the set-point of h_3=0.1, h_2=0.3
2. From time t=380 sec onwards valve V_{23} is permanently open. The configuration shown in Table 1, phase 4, is no longer valid as it requires V_{23} to be permanently closed. In this case the same target set can be achieved, with different input constraints. Figure 4 shows this scenario. The system can still be controlled using

pump Q_2 alone. The difference is that when using Q_2 alone, the disturbance that can be tolerated is smaller.

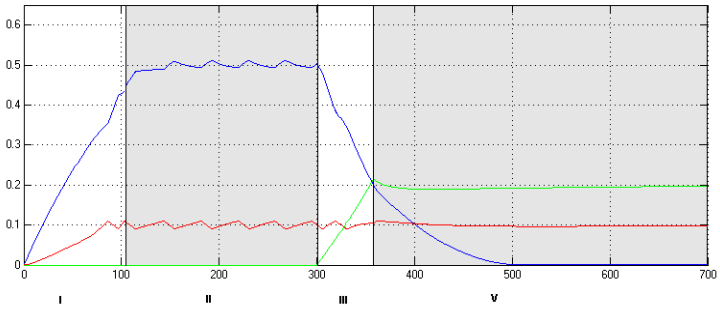

Figure 14. Alternative ending to the leak scenario

Summary

This Chapter presented the proposed architecture for fault-tolerant control of piecewise-affine systems with additive state disturbance.

The functional elements of the architecture were named as: the supervisory controller, the fault and state detector, the configuration manager (which performs reconfiguration), the low-level control and the plant.

The motivation for the architecture and justification for considering the global control objective was given by a real-life example of an aircraft fuel system and a benchmark academic example of a three-tank system.

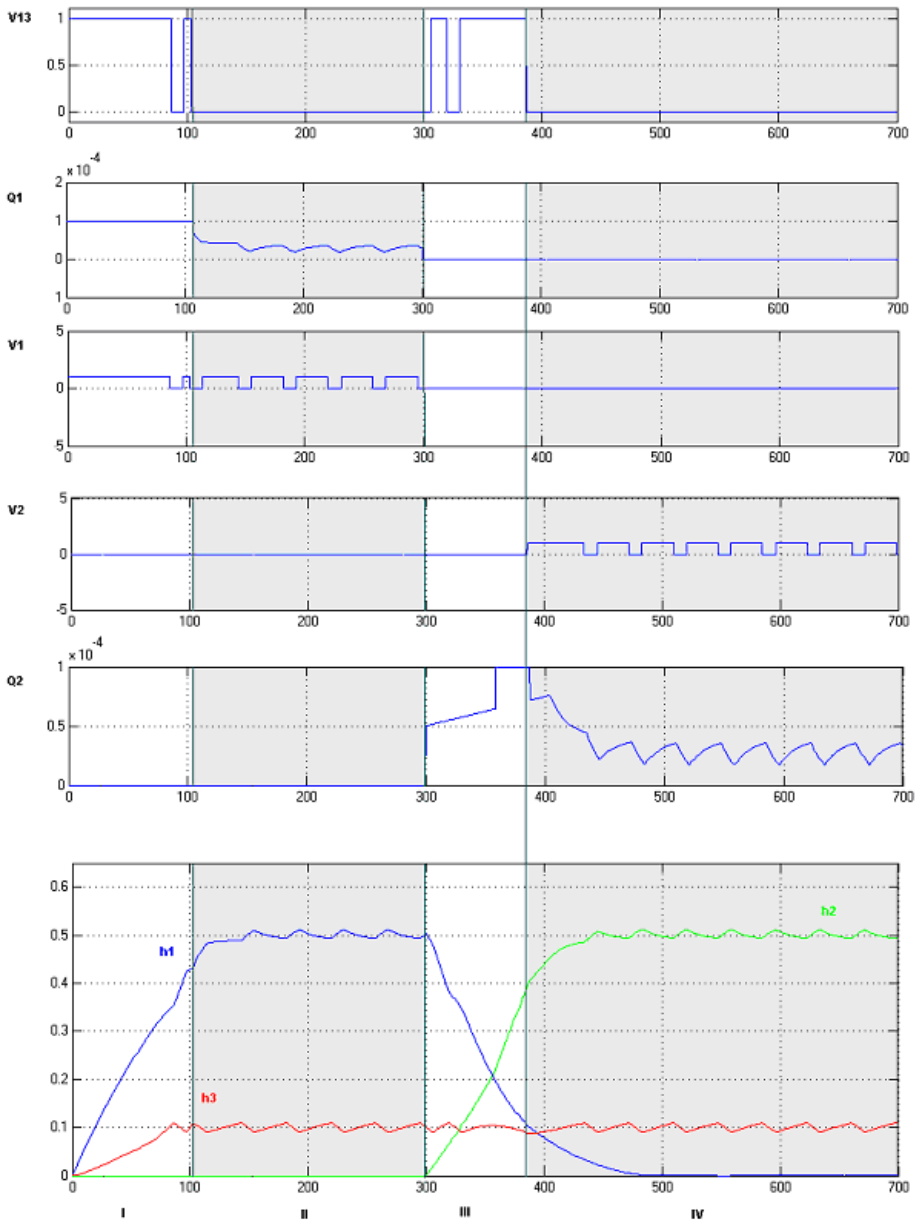

Figure 15. Reconfiguration scenario for leak in tank 1

Table 2. Control Objectives and Configurations. For each phase the target set T must be achieved within t=200 time steps

	Ω	T	U_l
1	$\{h_1, h_2, h_3 \mid$ $0 \le h_1 \le 0.6,$ $0 \le h_2 \le 0,$ $0 \le h_3 \le 0.11\}$	$\{h_1, h_2, h_3 \mid$ $0.45 \le h_1 \le$ $0.55, 0 \le h_2 \le 0,$ $0.09 \le h_3 \le$ $0.11\}$	$\{ V_{13,} V_{1,} V_{2,} V_{23,} Q_{1,} Q_2 \mid$ $0 \le V_{13} \le 1,$ $0 \le V_1 \le 1,$ $Q_1 = 10^{-4},$ $V_{23}= V_2=Q_2=0\}$
2	$\{h_1, h_2, h_3 \mid$ $0.45 \le h_1 \le 0.55,$ $0 \le h_2 \le 0,$ $0.09 \le h_3 \le$ $0.11\}$	$\{h_1, h_2, h_3 \mid$ $0.45 \le h_1 \le$ $0.55,$ $0 \le h_2 \le 0,$ $0.0905 \le h_3 \le$ $0.105\}$	$\{ V_{13,} V_{1,} V_{2,} V_{23,} Q_{1,} Q_2 \mid$ $0 \le V_1 \le 1,$ $0 \le Q_1 \le 10^{-4},$ $V_{13}= V_{23}= V_2=Q_2=0\}$
3	$\{h_1, h_2, h_3 \mid$ $0 \le h_1 \le 0.55,$ $0 \le h_2 \le 0.6,$ $0.09 \le h_3 \le$ $0.11\}$	$\{h_1, h_2, h_3 \mid$ $0 \le h_1 \le 0.2,$ $0.4 \le h_2 \le 0.6,$ $0.09 \le h_3 \le$ $0.11\}$	$\{ V_{13,} V_{1,} V_{2,} V_{23,} Q_{1,} Q_2 \mid$ $0 \le V_1 \le 1,$ $0 \le Q_1 \le 10^{-4},$ $V_{13}= V_{23}= V_2=Q_2=0\}$
4	$\{h_1, h_2, h_3 \mid$ $0 \le h_1 \le 0.2,$ $0.4 \le h_2 \le 0.6$ $0.09 \le h_3 \le$ $0.11\}$	$\{h_1, h_2, h_3 \mid$ $0 \le h_1 \le 0,$ $0.45 \le h_2 \le$ $0.55, 0.09 \le h_3$ $\le 0.11 \}$	$\{ V_{13,} V_{1,} V_{2,} V_{23,} Q_{1,} Q_2 \mid$ $0 \le V_2 \le 1,$ $0 \le Q_2 \le 10^{-4},$ $V_1= V_{13}= V_{23}=Q_1=0\}$

CHAPTER VII

RECONFIGURATION

A fault-tolerant control architecture includes redundant actuators, which are placed in order to provide the necessary control functionality in the event of faults. In the hierarchical control architecture, the complexity for low-level control is reduced by limiting three factors:

- The prediction horizon
- The number of manipulable input variables
- The number of discrete states

The prediction horizon needed for feed-forward control is limited by (i) the fact that low-level control is aimed at a control objective of reaching a target set in a limited time, and prediction is not needed beyond that time, and (ii) the fact that reaching a target set is a relatively simple objective, which can often be translated to an objective of shortening the distance to the target set, thereby using an even smaller prediction horizon.

The number of manipulated variables is limited, as will be detailed in this chapter by imposing constraints on the input variables. By staying within a limited state-set as required by Definition 3.2 (Control Objective) while the inputs also remain in a limited set, the state and input vector $[x(t)\ u(t)]^T$ is limited to a region, thus simplifying the PWA model of the system by removing many of the discrete states from consideration. The simplified model reduces the complexity of the control problem

The purpose of imposing constraints on the inputs is to reduce the number of manipulatable input variables for the low-level control. If the low-level control is

implemented by model-predictive control (MPC) – which is possible for piecewise-affine systems – the manipulable input variables are decision variables for the MPC optimization problem and reducing their number reduces the computational complexity [6]. Clearly, the reconfiguration task is required to have less computational cost than what is saved by not allowing all input variables to be manipulable by the low-level control. For this reason, the approach taken here is to perform the reachability calculations required for reconfiguration at design-time.

The computational complexity of model-predictive control is taken as a case-in-point because model-predictive control is a method for synthesizing the control law online which utilizes all the design space available. In MPC, at each time step a mixed-integer quadratic program with mixed-integer linear constraints is solved.

Definition 7.1 (MIQP) a Mixed-Integer Quadratic Program is an optimization program of the following from:

$$\begin{aligned} \min \quad & x^T Q x + b^T x \\ s.t \quad & Cx + d \le 0 \\ & x = \begin{bmatrix} x_c \\ x_d \end{bmatrix}, \quad x_c \in R^{n_c} \quad x_d \in \{0,1\}^{n_d} \end{aligned} \tag{52}$$

In the worst case, the time-complexity depends exponentially on the number of integer variables and the number of variables involved depends linearly on the prediction horizon [10].

The discrete modes of the hybrid system are defined as polyhedral regions in the state and input space. Each polyhedral region in n dimensions is defined by a finite

number of facets which are polyhedra in n-1 dimensions (e.g. n+1 facets for a n-simplex, $2n$ facets for a n-hypercube). The number of integer variables (including auxiliary variables) is related to the number of facets of each polyhedra. Therefore it is assumed that the number of integer variables in the optimization problem depends linearly on the number of manipulated variables.

Besides reducing the number of manipulated inputs, another potentially significant reduction in complexity is gained by the fact that the control problem is defined only in a limited region Ω of the state space, where only a subset of the discrete modes of the hybrid system are active and therefore the control problem is further simplified. Typically, in a practical application, the system states that the supervisory controller prescribes follow the behavioral modes of the system and thus the reduction in complexity is significant indeed. For example, in the three-tank system the three state variables define eight regions (above or below upper connecting pipe for each tank). But for the scenario depicted in Figure 15 four of the eight modes are never active, since under no circumstances is the level in Tank 2 above h_2=0.3. Moreover, in all but one of the reconfiguration phases, only two modes are active. And even this phase can be split into two supervisory control modes if needed.

Based on the above discussion it is conjectured that the complexity of low-level control is exponential in the prediction horizon, the number of inputs and the number of states.

Conjecture 7.1. The complexity of low level control is $O(2^{K+M+N})$ where N is the prediction and control horizon, M and K are the number of input and state variables, respectively, that appear in the inequalities defining the boundaries

of the polyhedral regions χ_q in the system that appear as variables in (4) with constraints $x \in \Omega$ and $u \in U_l$.

The significance of this assertion is that by reducing each one these parameters the complexity of low-level control is reduced.

Reduction in the prediction horizon is made possible by the presence of a supervisory controller. By assigning the supervisory controller all control tasks that require significant prediction, the low-level control need only concern itself with minimizing the distance to a set-point defined by the supervisory controller within a short prediction horizon. By Theorem 4.2 (Feasibility of Model Predictive Control) the prediction horizon in the MPC optimization problem to guarantee optimization needs to be set to the number of time steps in which the target set is required to be reached.

The configuration manager's task is to select input constraints which will guarantee reachability from the current state $x(t)$ to a target state set $T \subseteq X$ in t time steps without leaving $\Omega \subseteq X$ for the first t-1 time-steps. In this paper this process is called *reconfiguration*. The reconfiguration problem was formally defined in Problem 5.1 (Reconfiguration). It is proposed to perform the necessary reachability calculations at design time and store the results of these calculations in a reconfiguration database. A necessary condition for reachability is that a sequence of input vectors which satisfies the input constraints and the control objective exist.

The reconfiguration problem can be solved by calculating reachability in the presence of disturbance. For a discrete-time system, reachability in a finite number of steps can be computed, based on reachability in single steps.

Recall Definition 4.6 (One-Step Set) and Definition 4.7 (Robust One-Step Set). For a PWA with inputs $u(t) \subseteq U_l$ the One-Step set is given by

$$Q(\Omega) = \{ x \in X \mid \exists u \in U_l \, \exists q \in Q : (x, u) \in \chi_q, A_q x + B_q u + f_q) \in \Omega \}. \quad (53)$$

And the robust one-step set is given by

$$\tilde{Q}(\Omega) = \{ x \in X \mid \exists u \in U \, \exists q \in Q : (x, u) \in \chi_q, \forall d \in D \ (A_q x + B_q u + f_q + d) \in \Omega \}. \quad (54)$$

For systems with additive state disturbance the robust one-step set can be calculated as a one-step set to a target set, eroded by the disturbance set as follows.

Definition 7.3 (Pontryagin Difference). Given the sets $\Omega \subseteq R^n$ and $\Theta \subseteq R^n$ the Pontryagin difference between Ω and Θ is defined as

$$\Omega \sim \Theta = \{ \omega \in R \mid \omega + \theta \in \Omega, \forall \theta \in \Theta \}. \quad (55)$$

Proposition 7.1 [23] For a PWA system with additive state disturbance

$$\tilde{Q}(\Omega) = Q(\Omega \sim D), \text{ if } \Omega \sim D \neq \varnothing \quad (56)$$

Proof. It follows immediately from the definitions that

$$Q(\Omega \sim D) = \{ x \in X \mid \exists u \in U_l \, \exists q \in Q : (x, u) \in \chi_q, (A_q x + B_q u + f_q) \in \Omega \sim D \}. \quad (57)$$

Which is equal to

$$\{ x \in X \mid \exists u \in U_l \, \exists q \in Q : (x, u) \in \chi_q, \ (A_q x + B_q u + f_q) + d \in \Omega, \forall d \in D \} \quad (58)$$

$$= \tilde{Q}(\Omega)..$$

The following propositions regarding robust controllable sets for systems with additive disturbance will be instrumental in calculating the sets needed for reconfiguration.

Proposition 7.2 For a system with additive state disturbance, if Ω is given by the union

$$\Omega = \bigcup_i \Omega_i \tag{59}$$

then

$$\Omega \sim D \supseteq \bigcup_i (\Omega_i \sim D) \tag{60}$$

Proposition 7.3. For a system with additive state disturbance, if Ω is given by the union (60) then

$$\widetilde{Q}(\Omega) \supseteq \bigcup_i \widetilde{Q}(\Omega_i) \tag{61}$$

Proof.

$$\widetilde{Q}(\Omega) = \mathrm{Q}(\Omega \sim D) \supseteq \mathrm{Q}(\bigcup_i (\Omega_i \sim D)) = \bigcup_i Q(\Omega_i \sim D) = \bigcup_i \widetilde{Q}(\Omega_i) \tag{62}$$

Definition 7.4 (Robust Controllable [*i,j*]-step Set). For the system (1), with inputs $\mathrm{u}(t) \in \mathrm{U}_l$ and disturbances $d(t) \in \mathrm{D}$, the *robust controllable [i,j]-step set* $\widetilde{K}_i^j(\Omega, T)$ is the largest set of states in Ω for which an integer $i \le k \le j$ exists for which there exists an admissible control input which will drive the system to T in exactly *k* steps, while keeping the evolution of the state inside Ω for the first *k*-1 steps, for any time-varying disturbance $d(t) \in \mathrm{D}$, i.e.

$$\widetilde{K}_i^j(\Omega, T) = \left\{ \begin{array}{l} x_0 \in R^n \mid \exists i \le k \le j \quad \exists \{u(t) \in U\}_0^{k-1} : \\ \{x(t) \in \Omega\}_0^{k-1}, x(k) \in T, \forall \{d(t) \in D\}_0^{k-1} \end{array} \right\}. \tag{63}$$

Proposition 7.4.

$$\widetilde{K}_{i+1}^{j+1}(\Omega,T)=\widetilde{Q}(\widetilde{K}_{\mathrm{i}}^{\mathrm{j}}(\Omega,T))\cap\Omega. \tag{64}$$

Proof.

$$\widetilde{K}_{i+1}^{j+1}(\Omega,T)=\bigcup_{i+1\le k\le j+1}\widetilde{K}_{k}^{k}(\Omega,T)=\bigcup_{i\le k\le j}\widetilde{Q}(\widetilde{K}_{k}^{k}(\Omega,T))\cap\Omega\subseteq\widetilde{Q}(\widetilde{K}_{i}^{j}(\Omega,T))\cap\Omega \tag{65}$$

So that

$$\widetilde{K}_{\mathrm{i+1}}^{\mathrm{j+1}}(\Omega,T)\subseteq\widetilde{Q}(\widetilde{K}_{\mathrm{i}}^{\mathrm{j}}(\Omega,T))\cap\Omega \tag{66}$$

If

$$x\in\widetilde{Q}(\widetilde{K}_{\mathrm{i}}^{\mathrm{j}}(\Omega,T))\cap\Omega \tag{67}$$

then it is in Ω and can be driven to $K_{\mathrm{i}}^{\mathrm{j}}(\Omega,T)$ in one step and subsequently to T in k steps with $i\le k\le j$. Therefore

$$x\in\widetilde{Q}(\widetilde{K}_{\mathrm{i}}^{\mathrm{j}}(\Omega,T))\cap\Omega\Rightarrow x\in\widetilde{K}_{\mathrm{i+1}}^{\mathrm{j+1}}(\Omega,T) \tag{68}$$

and

$$\widetilde{K}_{\mathrm{i+1}}^{\mathrm{j+1}}(\Omega,T)\supseteq\widetilde{Q}(\widetilde{K}_{\mathrm{i}}^{\mathrm{j}}(\Omega,T))\cap\Omega\ \square \tag{69}$$

Theorem 7.1. The robust controllable [*i*,*j*]-step set can be computed by the following recursive formula:

$$\widetilde{K}_{\mathrm{i}}^{\mathrm{j}}(\Omega,T)=\begin{cases}T & i=j=0\\ \widetilde{Q}\left(\widetilde{K}_{\mathrm{j\text{-}1}}^{\mathrm{j\text{-}1}}(\Omega,T)\right)\cap\Omega & 0<i=j\\ \widetilde{Q}(\widetilde{K}_{\mathrm{i}}^{\mathrm{j\text{-}1}}(\Omega,T))\cap\Omega\cup\widetilde{K}_{\mathrm{i}}^{\mathrm{j\text{-}1}}(\Omega,T) & i<j\end{cases} \tag{**70**}$$

Proof. For *i=j,* the algorithm and proof is shown in [9, section 2.6]. For $i<j$, by definition

$$K_{\mathrm{i}}^{\mathrm{j}}(\Omega,T) = K_i^{j-1}(\Omega,T) \cup K_{i+1}^{j}(\Omega,T)\,.$$

From Proposition 7.4,

$$K_{i+1}^{j}(\Omega,T) = \widetilde{Q}(K_{\mathrm{i}}^{\mathrm{j-1}}(\Omega,T)\,) \cap \Omega.$$

therefore

$$K_{\mathrm{i}}^{\mathrm{j}}(\Omega,T) = K_i^{j-1}(\Omega,T) \cup \widetilde{Q}(K_{\mathrm{i}}^{\mathrm{j-1}}(\Omega,T)\,) \cap \Omega. \square$$

Affine Constrained Systems

Algorithms and theoretical results developed for constrained linear systems often assume the origin is in the interior of the admissible set of states and/or inputs [24][4][42] . For this reason it is useful first to clarify the relation between such systems and constrained affine system with admissible state and input sets which do not necessarily include the origin.

A constrained affine system is given by

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + f \\ Pu(t) &\le p \\ Qx(t) &\le q \end{aligned} \tag{71}$$

It is desired to form an equivalent LTI system

$$\begin{aligned} \bar{x}(t+1) &= \bar{A}\bar{x}(t) + \bar{B}\bar{u}(t) \\ \bar{P}\bar{u}(t) &\le \bar{p} \\ \bar{Q}\bar{x}(t) &\le \bar{q} \end{aligned} \tag{**72**}$$

Where the origin is an interior point of the set of admissible states

$$\bar{q} > 0 . \tag{73}$$

First, an interior point φ is found for the polyhedron $Qx(t) \le q$. This can be done by standard linear programming methods. For the simple case where none of the inequalities are forced to be tight, the interior point can be found as the point x at which *max e,* s.t. $Q x + e \le q$ is attained.

The coordinates of the state space are translated by φ,

$$\bar{x}(t) = x(t) + \varphi \tag{74}$$

After transformation the system becomes

$$\begin{aligned} &\bar{x}(t+1) = x(t+1) + \varphi = A\left(\bar{x}(t) - \varphi\right) + Bu(t) + f + \varphi = \\ &= A\bar{x}(t) + Bu(t) - A\varphi + f + \varphi \end{aligned} \tag{75}$$

By augmenting vector u with n inputs fixed at a constant value of $u' = f + \varphi - A\varphi$, an LTI system is obtained

$$\bar{x}(t+1) = A\bar{x}(t) + \begin{bmatrix} B \\ I \end{bmatrix} \bar{u}(t) \tag{76}$$

with constraints

$$\begin{aligned} &\begin{bmatrix} P & 0 \\ 0 & I \\ 0 & -I \end{bmatrix} \bar{u}(t) \le \begin{bmatrix} p \\ \mathrm{f} + \varphi - A\varphi \\ -\mathrm{f} - \varphi + A\varphi \end{bmatrix} \\ &Q\bar{x}(t) \le q + Q\varphi \end{aligned} \tag{77}$$

Robust Controllable Sets for PWA systems

Recall from Proposition 7.3 that for a system with additive state disturbance the robust one-step set $\tilde{Q}(\Omega)$ is equal to the nominal one-step set $Q(\Omega \sim D)$. This implies that in order to compute the $[i,j]$-robust controllable set by the formula presented in Theorem 7.1 it is

sufficient to develop procedures to compute the Pontryagin difference Ω~D, the nominal one-step set Q(Ω~D), the intersection Q(Ω~D) ∩ Ω, and the set union $\widetilde{K}_i^{j-1}(\Omega,T) \cup \widetilde{K}_{i+1}^{j}(\Omega,T)$. The first three issues are addressed in [23]. In computing the set union, the problem which arises is that the union of two convex sets is not necessarily convex, or even connected. This is true also for the computation of $\tilde{Q}(\Omega)$ for a PWA system, since it consists of a union of all discrete modes $q \in Q$ of the system. If the robust controllable set of a polyhedral region T is to be computed exactly this would necessitate enumeration or introduction of discrete variables to encode the set. The complexity of such a computation of $\widetilde{K}_i^{j}(\Omega,T)$ is exponential in the number of steps *j*. However, for practical purposes, the exact computation of $\widetilde{K}_i^{j}(\Omega,T)$ is not necessary. An inner approximation $\hat{K}_i^{j}(\Omega,T)$, such that $\hat{K}_i^{j}(\Omega,T) \subseteq \widetilde{K}_i^{j}(\Omega,T)$ is sufficient. If an inner approximation that is convex can be found wherever a union of polyhedra is calculated, it can be used to reduce the computational complexity of the off-line computations which generate the reconfiguration database. In any case, such an approximation is needed for the final iteration of Theorem 7.1 which produces the set which appears in the database. A method for convexity recognition of a union of polyhedra was shown in [44]. This approach can be extended for approximating a union of polyhedra as a convex polyhedron.

Reconfiguration and Fault-Tolerance

Reconfiguration provides fault tolerance by choosing input constraints, which are compatible with fault conditions. For example, if valve V_1 in the three-tank system is

fixed in position V_1=0, then any configuration constraint which is satisfied by V_1=0 is compatible with this fault.

This method of representing configurations requires some enumeration, although the enumeration is only of configurations – not faults. This is still more efficient than approaches that require enumeration of faults, since the fault space is typically much larger than the configuration space.

Assume all configurations $u \in U_l$ are given by rectangular constraints

$$u_{\min} \le u \le u_{\max} \tag{78}$$

Or in standard form

$$\begin{bmatrix} I \\ -I \end{bmatrix}[u] \le \begin{bmatrix} u_{\max} \\ -u_{\min} \end{bmatrix} \qquad u_{\min} \le u_{\max} \tag{79}$$

Likewise, the configurations encoding the fault conditions $v \in U_f$ are given by

$$\begin{bmatrix} I \\ -I \end{bmatrix}[u] \le \begin{bmatrix} v_{\max} \\ -v_{\min} \end{bmatrix} \qquad v_{\min} \le v_{\max} \tag{80}$$

This is a reasonable limitation, because the objective of configuration selection is to minimize the number of manipulated variables, given by the number of inputs for which $u_{i\max} - u_{i\min} > 0$. The dimension of the polyhedron of admissible values of u is all that matters, not its shape. It is also assumed, with out loss of generality, that

$$u_{\min}, u_{\max} \in [0,1] \tag{81}$$

$$v_{\min}, v_{\max} \in [0,1] \tag{82}$$

Constraints (**78**), (**81**) on $u_{\min}$, $u_{\max}$ define an actuator that can be either fixed at

$$u=u_{\min}=u_{\max}=0 \tag{83}$$

fixed at

$$u=u_{\min}=u_{\max}=1 \tag{84}$$

or manipulated in the range

$$0 \le u_{\min} \le u \le u_{\max} \le 1 \tag{85}$$

This representation is very useful since the constraints (**78**), (**81**) can be taken into account implicitly, and the database need only include extra constraints for each manipulated actuator

$$u_{i\max} - u_{i\min} \ge 1 \tag{86}$$

A simple constraint such as $\sum_{i=1}^{k} u_{i\max} - u_{i\min} \ge n$ can be used to represent the case where n out of k available actuators can be used to achieve the control objective.

Proposition 7.5. For the sets

$$U_l = \{u \mid \ u_{\min} \le u \le u_{\max} \ \} \tag{87}$$

and

$$U_f = \{v \mid \ v_{\min} \le v \le v_{\max} \ \} \tag{88}$$

$$U_l \subseteq U_f \Leftrightarrow \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} u_{\max} \\ u_{\min} \end{bmatrix} \le \begin{bmatrix} v_{\max} \\ -v_{\min} \end{bmatrix} \tag{89}$$

The Reconfiguration Database

The reconfiguration database consists of six-tuples ($\tilde{X}$, $\tilde{D}$, $\tilde{U}$, $\tilde{T}$,$\tilde{\Omega}$, $\tilde{t}$) for which it has been determined that $\tilde{X}$ is a robust controllable [1,t]-step set $K_1^t(\tilde{\Omega},\tilde{T})$ for the system with disturbance $\tilde{D}$ and input constraints $\tilde{U}$. At runtime, the configuration manager's task is to find a six-tuple from the database, for which $X_e \subseteq \tilde{X}$, $D \subseteq \tilde{D}$, $\tilde{U} \subseteq U_f$, $\tilde{T} \subseteq T$, $t \leq \tilde{t}$, $\tilde{\Omega} \subseteq \Omega$, based on X_e, D, U_f supplied by the fault and state detector and Ω, T, t supplied by the supervisory controller. The sets are all assumed to be convex polyhedral sets, so the computation of the set inclusions amount to the solution of linear programs. In general the robust controllable set for a piecewise affine system is not convex; however it is sufficient for the purpose of reconfiguration to use an inner approximation of the robust controllable set, which is convex, for the value of $\tilde{X}$ in the database.

By removing all reconfiguration options which do not satisfy the necessary conditions for reachability, the search space for the low-level control is reduced, while ensuring the existence of appropriate control inputs to satisfy a control objective. The problem of designing the low-level control to select the optimal control inputs is beyond the scope of this paper. One possibility is to apply model-predictive control for which necessary and sufficient conditions for robust feasibility are known [24].

The reconfiguration database lists six-tuples ($\tilde{X}$, $\tilde{D}$, $\tilde{U}$, $\tilde{T}$,$\tilde{\Omega}$, $\tilde{t}$) for possible combinations of state and fault identification and control objectives given by the state and fault detector and the supervisory controller, respectively. The task of partitioning the state and input sets to determine these sets is the subject of the next section.

Summary

In this chapter it was shown how the introduction of a reconfiguration layer between the supervisory controller and the low-level control of a PWA system can reduce complexity of low level control by limiting the prediction horizon, number of inputs and number of discrete states that need to be considered by the low-level control in each supervisory control mode.

The theory of invariant sets was shown to provide the tools for determining the initial set from which a specified set can be reached within a specified time for all allowable disturbances. The robust controllable sets can be computed recursively from the robust one-step sets using **Theorem 7.1**. Robust one-step sets can be computed using already existing tools for LTI systems [23]. Using the transformations introduced in this chapter these calculations can be applied to each mode of a piecewise-affine system.

Appendix B includes an example of calculating a robust controllable set using Theorem 7.1, for phase III of the three-tank reconfiguration process shown in Figure 15.

CHAPTER VII

SUPERVISORY CONTROL

The supervisory control of a hybrid system can be approached as a discrete-event control problem, by abstracting the plant into a discrete event system preserving all properties of interest. In hierarchical control, this is done by forming a partition of the state space, for which it can be guaranteed that the system can be forced to reach a desired region by choosing appropriate controls. In this section the subject of partitioning the continuous state space will be considered. The property of interest is maintaining the control system's ability to satisfy the control objectives, which in this case are formulated as reachability specifications.

For the notations and definitions of mathematical terms used in this chapter see Appendix A.

Bisimulation

One method for generating discrete abstractions of a hybrid system is bisimulation. A bisimulation is a reachability preserving quotient system in the sense that checking a property on the quotient system is equivalent to checking the property on the original system [34]. If an equivalence relation ~ is a bisimulation, then given two systems P and Q each with an initial and final set, the states in the initial sets of each system are mutually equivalent, the states in the final set of each system are mutually equivalent, and if a state $p \in P$ and a state $q \in Q$ are equivalent, and p has a next state,

then q has a next state which is equivalent to the next state of p. (See Appendix A for formal definitions).

Bisimulation is a strong property as it provides:

1. Equivalence of states for all the state space of the equivalent systems
2. Preservation of the one-step predecessor (successor) operator.

The first requirement is strong because it means that the abstraction applies to the entire state space, including regions that are not meant to be reachable. The second requirement is strong because it applies by induction to all time and it also means that equivalent trajectories have the same number of time steps.

Quasideterminism

Another method for generating such discrete abstractions is Quasideterminism [7]. In contrast to bisimulation, Quasideterminism requires equivalence only to the degree that set membership of a state in equivalence classes of a primary partition be preserved. If the primary partition is given by an equivalence relation E_π and the predecessor operator defines an equivalence relation E_{pred} then the final partition is the meet of E_π and and E_{pred}, $E_{\pi f} = E_\pi \bullet E_{pred}$ [7].

Quasideterminism is a weaker property than bisimulation, but still a stronger property than what is needed with respect to Definition 3.1 (Global Control Objective). Several observations can be made:

- Quasideterminism with respect to a primary partition defined by Definition 3.1 (Global Control Objective) may be hard to achieve because safety cannot be guaranteed for some of the state space in these regions.

However, it may be achievable for a region $\psi \subseteq X$ with $\psi \cap Bad = \varnothing$, when ψ / E_π are taken as the regions of the primary partition rather than X / E_π. If it can be guaranteed that the system starting in A_0 (from Definition 3.1 (Global Control Objective)) will never leave ψ then it will also not reach *Bad*.

- Quasideterminism refers to reachability in one step, whereas Definition 3.1 (Global Control Objective) refers to reachability within a time window of between one and *time*(*k*,*l*) time steps.
- Quasideterminism enables open-loop control of the system without knowledge of its exact state. In contrast, assuming an architecture like the one shown in Figure 10, the state detector can determine to which region of the final partition the system was driven.

State Space Partition

Given these observations, this chapter seeks to characterize the state-space partition with respect to Definition 3.1 (Global Control Objective).

First note that all the regions A_k are assumed to be disjoint sets. If this is a problem, it can sometime be helpful to use the fact that the following definition of a is equivalent to Definition 3.1 (Global Control Objective)..

Definition 8.1 (Equivalent Global Control Objective) Given a set $Bad \subseteq X$ and a finite collection of sets $A_k \subseteq X$, $k \in K$, that includes an initial set $A_0 \subseteq X$, with $A_k \cap Bad = \varnothing$, a set valued map *next*: $K \rightarrow 2^K$ and a function time *time*:

$K \times K \rightarrow Z^+$ the global control objective is that for the system with initial conditions $x \in A_0$ the continuous state will remain in any set A_k and subsequently cross into set $A_{k'}$ after t time steps for some $k' \in \text{next}(k)$ for which $t \leq time(k,k')$, while remaining in $A_k \cup A_{k'}$ for t-1 time steps.

$$x(t_0) \in A_k \Rightarrow \exists k' \in \text{next}(k): \exists t \leq time(k,k') : \\ x(t_0+t) \in A_{k'} \wedge \forall 0 \leq t' < t, \; x(t_0+t') \in A_k \cup A_{k'} \tag{90}$$

Proposition 8.1 Definition **8.1** and Definition 3.1 (Global Control Objective) are equivalent.

Proof.

(Definition 3.1 $\Rightarrow$ Definition **8.1**) follows from

$$x(t_0+t') \in A_k \Rightarrow x(t_0+t') \in A_k \cup A_{k'} \tag{91}$$

(Definition 3.1 $\Leftarrow$ Definition **8.1**) Let $t \leq time(k,k')$, $x(t_0+t) \in A_{k'}$. If t=1, then for t'=0,, $x(t_0+t') \in A_k$. If t>1, $x(t_0+t') \in A_k \cup A_{k'}$ and $x(t_0+t') \notin A_k$, then $x(t_0+t') \in A_{k'}$. In particular for t'=1, $t' \leq time(k,k')$, $x(t_0+t') \in A_{k'}$ and $+x(t_{0+}t'')$, $\forall 0 \leq t'' < t' \in$

Example 8.1. The global control objective based on Definition 8.1 is to bring the state of the three tank system from $A_{init}=\{h_1, h_2, h_3 \mid 0 \leq h_1 \leq 0.6, \; 0 \leq h_2 \leq 0, \; 0 \leq h_3 \leq 0.11\}$ to $A_1=\{h_1, h_2, h_3 \mid 0.45 \leq h_1 \leq 0.55, 0 \leq h_2 \leq 0, 0.09 \leq h_3 \leq 0.11\}$ within 200 time steps while remaining in $A_{init} = A_{init} \cup A_1$ for the first 199 time steps and subsequently to regulate it in A_1. Since $A_1 \subseteq A_{init}$, by replacing A_{init}

with $A_0 = A_{init} - A_1$ the objective can be restated as staying in A_0 and then crossing into A_1.

Assumption 8.1. The sets A_k are disjoint.

Definition 8.2. (Primary Partition) Given a global control objective, defining a collection of sets $A_k \subseteq X$, $k \in K$, the partition of $A = \bigcup_{i=0...N} A_k$ into sets forming the collection $\{A_k\}$is called the *primary partition*.

As with bisimulation [34] and quasi-determinism [7], the required partition is generated by $l \in next(k)$ a process of partition refinement resulting in a final partition. The primary partition π, forms an equivalence relation E_π. The set of all the partitions of the state space X with polyhedral equivalence classes is characterized as a lattice, and partition refinement is defined with respect to the partial order of the lattice [7].

The process of partition refinement results in the partitioning of the state space into regions which are subsets of the regions $\{A_k\}$. The primary partition is refined by applying a finite number of meet operations, resulting in a *final partition*. The requirement from the final partition is that the induced system operating in the quotient space $X / E_{\pi f,}$ can be controlled by the supervisory controller in such a way that the trajectory in the original system will satisfy the global control objective. The state identification given by the state detector, in the form of a set $K \subseteq X$ as well as the sets $\Omega \subseteq X$ and $T \subseteq X$ in the control objective are regions defined in terms of the final partition. The partition must be such that when the supervisory controller is given a state and fault

identification (K, U_f, D), it can determine the a control objective (Ω,T, t), for which the system can be forced on a trajectory consistent with the global control objective by choosing appropriate controls.

Assume a given disturbance set D_k for each region A_k. Let $U_L \subseteq 2^U$ be the set of admissible input sets. The choice of input constraints is based on two considerations: fault-tolerance and reducing the number of manipulated variables. A configuration $U_l \in U$ will be tolerant to a fault if the configuration admits only input vectors which are not precluded by the fault. The additive state disturbance can also be used to model certain input faults (e.g. a leak in the tank, which is an additive state disturbance because the leak reduces the volume of liquid in the tank – a state variable – by a certain amount at each time step).

Let Ψ be a collection of sets $\Omega_{k,}$ $k \in K$, which appear as invariant sets Ω in the reconfiguration database, and let $\Omega_0 = A_{0;}$ $\Omega_k \subseteq A_k$ $\forall k \in K$. It is required that at any trajectory starting in Ω_0 can be driven to follow the global control objective. This can be assured if

$$\forall x \in \Omega_{k,} \exists l \in \text{next}(k), \exists u \in U_L : x \in K_1^{\text{time}(k,l)}(\Omega_k, \Omega_l). \quad (92)$$

Where $K_1^{\text{time}(k,l)}(\Omega_k, \Omega_l)$ is the [1, time(k,l)]-step robust controllable set. In the nominal case reconfiguration occurs when the system crosses into a target set from which reachability to the next target set is assured within the required time. When a fault occurs, the time constraint is not necessarily satisfied; however, the condition of (92) ensures that the next state is reachable when reconfiguration occurs at any point along the trajectory. The collection Ψ can be calculated recursively by Algorithm 8.1.

Algorithm 8.1. (Compute Collection of Invariant Sets Ψ)

INPUT:

partition π defining regions A_k, $k \in K$

input constraints U_L

disturbance set D_k for each A_k

BEGIN

FOR each $k \in K$

$\Omega'_k = A_k$.

REPEAT

FOR each $k \in K$,

$\Omega_k = \Omega'_k$

FOR each $k \in K$,

$$\Omega'_k = \bigcup_{U_l \in U_L} \bigcup_{l \in next(k)} K_1^{time(k,l)}(\Omega_k, \Omega_l);$$

UNTIL $\Omega_k = \Omega'_k, \forall k \in K$

END

OUTPUT

Collection $\Psi = \{\Omega_k\}_{k \in K}$ of invariant sets.

The algorithm succeeds if it terminates and $\Omega_0 = A_0$. If the algorithm terminates successfully (92) is satisfied. For practical purposes the number of iterations must be limited to check for successful termination. What remains is to partition the sets $\Psi = \{\Omega_k\}_{k \in K}$ into regions such that from each region, it can be determined which next target set can be reached and by what configuration. This is performed by Algorithm **8.2**.

Algorithm 8.2. (Partition sets Ψ, with configurations U_L).

INPUT:

State space X

Global control objective: K, $\{A_k\}$, *time*, *next*

input constraints U_L

disturbance set D_k for each A_k

BEGIN

$\pi_f := (X \setminus \bigcup_{k \in K} A_k , A_0, \Omega_1, A_1 \setminus \Omega_1, \Omega_2, A_2 \setminus \Omega_2, \ldots)$

FOR each $k \in$ K, $U_l \in U_L$, $l \in$ *next*(k),

compute partition: $\pi = (X \cap K_1^{time(k,l)}(\Omega_k, \Omega_l) , X \setminus K_1^{time(k,l)}(\Omega_k, \Omega_l))$

Refine: $\pi_f := \pi_f \cdot \pi$

END

END

OUTPUT:

Refined partition π_f.

The final partition resulting from application of **Algorithm 8.1** and **Algorithm 8.2** for a set of configurations U_L divides the state space into regions that are not necessarily convex, or even connected, because the robust controllable sets computed for different $U_l \in U_L$ and $l \in$ *next*(k) can overlap. Therefore, to arrive at a collection of convex and compact regions, it may be necessary to further divide the regions along the hyperplanes that define their boundaries. Figure 16 illustrates this process. It would also be possible to simply partition the state space X along all the hyperplanes, which define the partition π', but this would result in a larger number of regions and unnecessarily increase the size of the reconfiguration database.

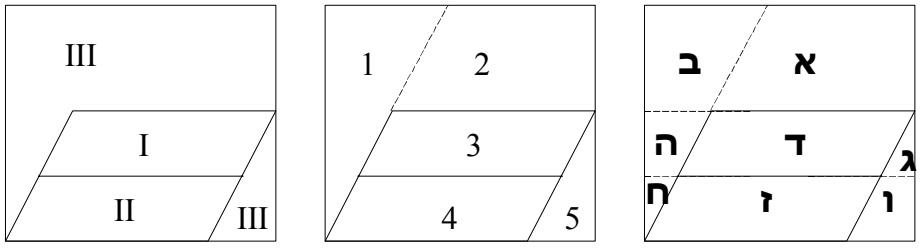

Figure 16. Refining the final partition (a) into convex and compact regions (b). (c) Shows the partition along hyperplanes. Note that regions in (b) overlap at the boundaries.

When the system is in region $A_{k,}$ the supervisory controller sets the control objective for the configuration manager and low level control as all the 3-tuples (Ω, T, t) with $\Omega = \Omega_k$, $T = \Omega_{l,}$, $l \in$ *next*(k), t = time(k,l). The state detector must detect partition crossing in the final partition so that when the system crosses into $\Omega_{l,}$ it can be determined in which region of the final partition the current state is, so that reconfiguration can proceed. The reconfiguration database is also constructed based on the final partition and the possible control objectives. Throughout this section the disturbance set D_k was assumed to be given for each region A_k of the global control objective. The disturbance set provides another design parameter, which can be relaxed or tightened to enable the global control objective to be achieved or to increase robustness.

After finding the final partition, the supervisory controller and the state detector can be designed to specify their outputs in terms of the refined partition. Let the final partition be given by $\pi' = \{P_1, P_2, \ldots, P_N\}$. Where each P_i is a convex and compact subset of a region A_k of the primary partition. The supervisory controller can be designed to specify its outputs in terms of the final partition. If the system crosses into region $A_{k,}$ at time t_k then the set of control objectives is constructed as follows.

Algorithm 8.3. (Construct Control Objective Set).

INPUT:

Final partition π' defining regions ($P_{1, \ldots} P_M$),

Global Control Objective

Collection of regions $\psi = \{\Omega_{0, \ldots} \Omega_N\}$,

Set valued map *next*: $K \rightarrow 2^K$

A function time *time*: $K \times K \rightarrow Z^+$,

A region A_k in the final partition, which has been crossed into.

BEGIN

S = {};

FOR each $l \in next(k)$,

FOR each $P_i \subseteq A_l$,

Add a control objective (Ω, T, t) to S, with $\Omega = \Omega_k$ and T = P_i and t = *time*(k, l).

END

END

OUTPUT:

Set of control objectives, S.

The set of control objectives remains the same as long as the system remains in A_k. The reconfiguration manager takes note of the time t_k at which the control objectives were given. All times in the control objective are relative to this time, so that if a different control objective is selected, the reconfiguration manager must take the time offset into account.

Partition Crossing Detection

The state detector must detect partition crossing in the primary partition for the supervisory controller to generate the control objectives, and in the final partition for the

configuration manager to select a configuration, so that when the system crosses into A_l, it can be determined in which region P_j of the final partition the current state is. The state detection is performed using a threshold function.

The *n*-dimensional state space X is partitioned by (n-1)-dimensional hyperplanes, which form the boundaries of the regions P_i. The requirement from the state detector is to detect when the system crosses each one of these hyperplanes. Following [7], let h_i be a collection $\{h_i\}_{i=12,\ldots Nf}$ $h_i : R^n \to R$ of real valued functions of the form $h_i(x) = g_i^T x - c_i$ where $g_i \in R^n$ and $c_i \in R$. Let H_i be the (n-1)-dimensional hyperplanes given by

$$H_i = \ker(h_i) = \left\{x \in R^n \mid h_i = g_i^T x - c_i = 0\right\} \tag{93}$$

Let $\{\hat{h}_i\}$ be a collection of threshold functions defined [7] as

$$\hat{h}_i(x) = \begin{cases} -1 & if \quad g_i^T x - c_i < 0 \\ 0 & if \quad g_i^T x - c_i = 0 \\ +1 & if \quad g_i^T x - c_i > 0 \end{cases} \tag{94}$$

The state detector is composed of two stages. The vector function $\lambda(x) = [\hat{h}_1(x), \ldots, \hat{h}_l(x)]^T$ is a mapping $\lambda : R^n \to \{-1,0,1\}^l$ where there is a bijection between $\{-1,0,1\}^l$ and the set of all region resulting from partitioning *X* along the hyperplanes which define π' [7]. The next stage is mapping the elements of $\{-1,0,1\}^l$ to the sets in Ψ.

Summary

In this chapter it was shown how a state space partition can be generated that assures that the global control objective can be achieved in the nominal case and degraded performance in the case of a fault.

Bisimulation and Quasideterminism were introduced in the beginning of the chapter. In contrast to these approaches, the state space partition refinement method presented in this chapter is always a finite process. If a partition can be found, the low-level control can drive the system from one partition to the next within the time constraints specified. The collection of sets ψ includes the initial state set A_0, from which other sets are reachable. The control system can keep the plant from leaving the sets in ψ and therefore the rest of the state space is not considered when partitioning the sets.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

In this research it was shown that the theory of invariant sets can be used to provide robustness in control of hybrid systems in a practical setting. A hierarchical architecture was shown to reduce the complexity of low-level control by limiting the number of inputs used in each supervisory control mode, and thus simplifying the model of the hybrid system.

To further develop these ideas more work will be needed to implement the algorithms and software for computing invariant sets for piecewise-affine systems. This chapter summarizes in detail the contributions and conclusions of previous chapters and the work that lies ahead.

Contributions

Following is a list of specific contributions presented in this thesis:

- An architecture for reconfigurable hierarchical control Figure 10
- The reconfiguration problem defined in the context of hierarchical control: Problem 5.1 (Reconfiguration)
- Relation of hierarchy to reduction of complexity for low-level control Conjecture 7.1
- Robust reachability of PWA systems within a time window: Proposition 7.2, Proposition 7.3, Proposition 7.4, Definition 7.4, Theorem 7.1
- Transformation of a constrained affine system to a constrained linear system with the origin in the interior of the set of admissible states: Equations (76), (77).

- Representation of configuration as inequality and integrality constraints on its bounds: Equations (78)-(89), Proposition 7.5.
- State space partition for supervisory control for Definition 3.1 (Global Control Objective): Definition 8.1, Proposition 8.1, Equation (92), Algorithm 8.1, Algorithm 8.2, Algorithm 8.3

Conclusions

One of the attributes of the proposed architecture, which makes it useful in a practical setting, is that from a design perspective the reconfiguration database provides a decoupling between the mathematical computations, which are based on a PWA model, with various assumptions about disturbance etc. and the reconfiguration strategies actually deployed. That is to say, that while this thesis provided the theoretical justification for a three-layer architecture which uses a "reconfiguration database" in practice, the entries in the reconfiguration database can be decided upon based upon simulation, testing or other methods, which require fewer assumptions about the system. For example, as a design methodology, it is suggested, that the initial values be generated by calculating the invariant sets based on a discrete-time PWA model with all the necessary assumptions; subsequently the database entries can be refined by testing them on more accurate non-linear simulation models and later validated experimentally on a real system. In contrast, many of the control and reconfiguration methods proposed by other researchers, though mathematically valid, are not transparent in the sense that the details of the reconfiguration and control strategies are hidden by the synthesis algorithms.

Minimization of the Number of Manipulable Inputs

The method described in Chapter VII for generating the reconfiguration database, starts with a given set of configurations $U_L \subset 2^U$, each of which has a sufficiently small number of variables for which the values are not fixed.

In order to find the smallest number of variables, that need to be manipulated, it would be desirable to solve the following optimization problem for all control objectives.

Problem 9.1 (Minimal Configuration). Given a piecewise-affine system with inputs $u(t) \in U$, a control objective (Reach T in t time steps, while staying in Ω for t-1 time steps), and a disturbance set $D \subset R^n$, solve

$$min\left\{ \ dim(U_l) \ : \ U_l \subseteq U, \ \ \Omega \subseteq \widetilde{K}_1^t(\Omega, T) \ \right\}$$

Complexity of MPC for PWA systems

A comprehensive study of complexity of model-predictive control for PWA systems in MLD form is not available. Currently there are a number of unknowns that may strongly affect the complexity. In particular it is not known exactly how the number of constraints and auxiliary variables in the MLD model grows with the addition of additional discrete modes to the PWA system. Such a study is needed to establish the scalability of the MLD modeling approach.

It was conjectured in Chapter VII that the number of integer variables in the MIQP problem of MPC is proportional to the number of inputs and states of the PWA system. This is only a coarse estimate. The nature of the transformations needed to generate the MLD model is complicated and systems that arise in practical applications

may have special structures (such as self-similarity). A complexity analysis that ties the performance of algorithms such as [25] to attributes of the system in some canonical form, that can be linked to physical attributes such as the number of energy storage elements, number of switches etc.

Approximate Calculation of Invariant Sets for Hybrid Systems

Invariant sets for piecewise-affine systems can be calculated by the method outlined in [23]. For the case where there is disturbance but no control input, by the method of [48]. These methods calculate the invariant sets exactly. But for practical applications, approximations of the invariant sets are sufficient. The work of [47] on reachability computations, which applies to hybrid systems, might be extended to the calculation of invariant sets.

Although one of the main arguments in this thesis was that design-time calculations can be afforded to be computationally expensive, complexity of these calculations is nonetheless an issue. The reachability calculations are in general exponential in the number of discrete modes, and thus do not scale up well. One method described in Appendix B is an attempt to eliminate the complexity arising from the discrete modes by performing a convex approximation at each iteration of the algorithm to compute the robust controllable set. It is not clear, though, what conditions would make this method work.

Tools for Computational Geometry in High Dimensions

Computational geometry has traditionally focused on two- and three-dimensional problems because applications were in areas such as Computer-Aided Design (CAD). Clearly, for control-theoretic applications higher dimensions must be considered.

In [51] the idea of "griddy polyhedra" is presented as a means to alleviate the computational issues arising from the set calculations required to control theory. By the methods presented in [49][50] this can probably be used to calculate the robust controllable sets as required for the architecture presented in this thesis.

Piecewise-Affine-Systems Toolbox

Piecewise-affine systems have proven to be convenient approximations of hybrid systems that lend themselves to relatively simple methods of analysis, verification and control synthesis. It would be useful to develop a MATLAB® toolbox that provides for defining piecewise-affine systems, analyzing them and using them as building blocks for composing systems.

For piecewise-linear hybrid dynamical system a toolbox has been recently developed [55] that calculates backwards and forwards reachability and checks properties relative to a control objective.

Reconfiguration for Constrained Systems with Polytopic Uncertainty

In this thesis, the faults that were considered are those that can be modeled as input constraints. These are structural faults that limit the control system's ability to influence the plant, typically actuator faults.

Faults resulting from changes in physical properties of a system, are often best modeled as parameter deviations. For LTI systems a bounded uncertainty in parameter identification can be defined as polytopic uncertainty in the system matrices (A,B). The system matrices are assumed to lie within a convex hull of a finite number of (A, B) tuples. Such uncertainty was suggested in [45] as a method for modeling a non-linear system. It is argued therein that analysis and design methods developed for an experimentally-obtained model with polytopic uncertainty can be applied to the real non-linear system. For continuous-state systems with polytopic uncertainty, invariant sets [52] [23] and MPC [45][53] have been studied.

If combined with diagnosis methods [46] the reconfiguration method presented here can be applied to LTI systems with polytopic parametric uncertainty. However, the problem of PWA systems with polytopic uncertainty appears to be much more difficult. In [56] hierarchical control of piecewise-linear hybrid dynamical systems with polytopic uncertainty in the matrices associated with each discrete mode is considered. However, it is assumed in [56] that the uncertainty is given separately for each mode, and the question of of relating the uncertainty in the system (e.g. uncertainty in value of each physical parameter) to uncertainty in each entry of the system matrices associated with each discrete mode is not addressed. To enable reconfiguration of PWA systems with parametric uncertainty this question must be addressed or a diagnosis method must be applied that can circumvent this problem by supplying sufficient information for reconfiguration without explicitly mapping parameter variations to variations in the matrices of each discrete mode.

APPENDIX A

MATHEMATICAL CONCEPTS

Definition A.6. (Partition) A *partition* of a set $A \subseteq X$ is a collection of non-empty pair-wise disjoint subsets of A whose union is A. The subsets are called *blocks*.

Definition A.7. (Refinement) A partition π_1 is said to refine another π_2, denoted $\pi_1 \leq \pi_2$ if every block of π_1 is contained in some block of π_2

Definition A.8. (Meet) Given two partitions, π_1 and π_2, their *meet* $\pi_1 \cdot \pi_2$ is the largest partition which refines both π_1 and π_2.

Definition A.9. (Join) Given two partitions, π_1 and π_2, their *join* $\pi_1 + \pi_2$ is the smallest partition which refines both π_1 and π_2.

The refinement relation is a partial ordering of the poset consisting of all partitions of A. The blocks of the meet are blocks from all nonempty intersections of a block from π_1 and a block from π_2. The blocks of the join are the smallest subsets, which are exactly the union of blocks from both π_1 and π_2. The poset Π is a lattice under the meet and join operations.

Definition A.10. (Equivalence Relation) A relation is an equivalence relation if it is reflexive, transitive and symmetric.

Definition A.11. (Equivalence relation of a partition) Given a set A, and a partition π of A., the relation E_π is an equivalence relation defined as $a\ E_\pi\ b \Leftrightarrow \exists\ B \in \pi : a, b \in B$.

Definition A.12. (Equivalence Class) Given a set A, an element $a \in A$, and an equivalence relation E, the equivalence class containing a is $\{x \in A \mid x\ E\ a\}$.

Definition A.13. (Quotient Space) Given an equivalence relation E, **t**he set consisting of all equivalence classes of A, is called the quotient space and denoted A/E.

Definition A.14. (Region) A region is a subset $P \subseteq A$. For a region P and equivalence class E. P/E denotes all the equivalence classes that intersect P.

Definition A.15. (E-block) Given an equivalence relation E on a set A, a set is called an *E-block* if it is a union of equivalence classes.

Definition A.16. (Bisimulation) Given a predecessor operator *Pred* : $2^A \rightarrow 2^A$ (interpreted as a region of predecessor states of a region in a state space, and sets A_i, A_f, (interpreted as initial and final regions in a state space), an equivalence relation E is a bisimulation iff A_i, and A_f are E-blocks, and the predecessor of every E-blocks is an E-block.

APPENDIX B

CONVEX APPROXIMATION OF THE UNION OF CONVEX POLYHEDRA

In order to compute the robust controllable set using **Theorem 7.1** it is necessary to compute the union of polyhedral sets. Since the union of polyhedral sets is not guaranteed to be convex, this can lead to a need to compute and represent the robust controllable set as a union of an exponentially large number of polyhedral sets.

Given that only an under-approximation of the initial set is needed for backwards reachability, it is suggested to compute the approximation of a union of convex polyhedra as a convex polyhedron which is included in the union.

This method was applied to the computation of the robust controllable set of for the three-tank benchmark with some success. However, it suffers from several problems as will be detailed later.

Convexity Recognition

Bemporad et. al. [54] describe a method for recognizing the convexity of a union of polyhedra. The basics of this method are as follows:

Given two Polyhedra $P = \{x \in \mathrm{R}^d : Ax \le a\}$ and $Q = \{x \in \mathrm{R}^d : Bx \le b\}$ the envelope is defined as

$$env(P,Q) = \left\{x \in R^d : \overline{A}x \le \overline{a}, \overline{B}x \le \overline{b}\right\} \tag{95}$$

where $\overline{A}x \le \overline{a}$ is obtained from $Ax \le a$ by removing all the inequalities not valid for the other polyhedron Q, and $\overline{B}x \le \overline{b}$ are obtained from $Bx \le b$ in a similar way with respect to

P. In [54] it is proven that env(P,Q) = P $\cup$ Q iff P $\cup$ Q is convex. The algorithm for implementing this result is given in [54]. Figure 17 shows an example of two polyhedra an their envelope.

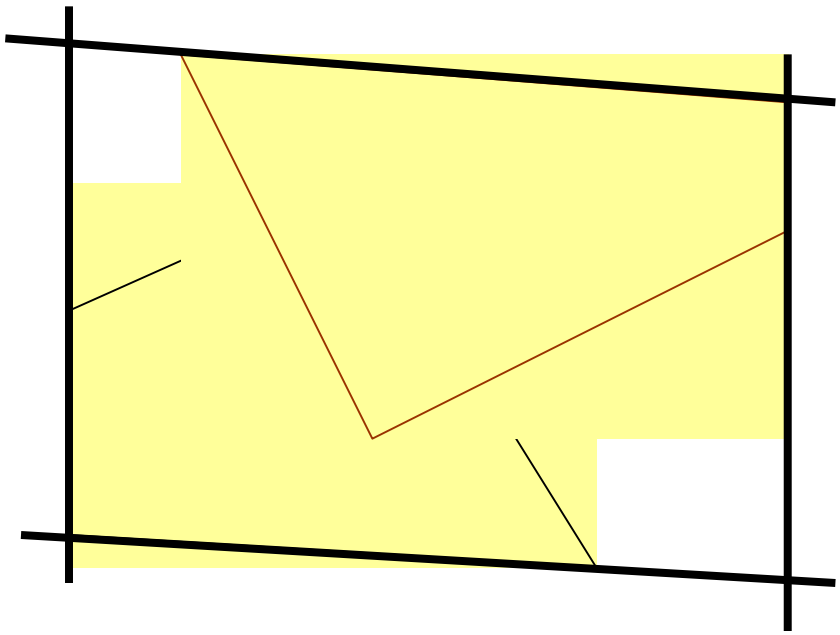

Figure 17. Envelope of Two Polyhedra

When P $\cup$ Q is not convex, it is desired to find a polyhedron P $\cap$ Q $\subseteq$ R $\subseteq$ P $\cup$ Q, such that R is convex. The method which was implemented was is as follows: for each pair of intersecting hyperplanes, a cutting plane is passed through the hyperplane of intersection, which bisects the angle between them as shown in Figure 18.

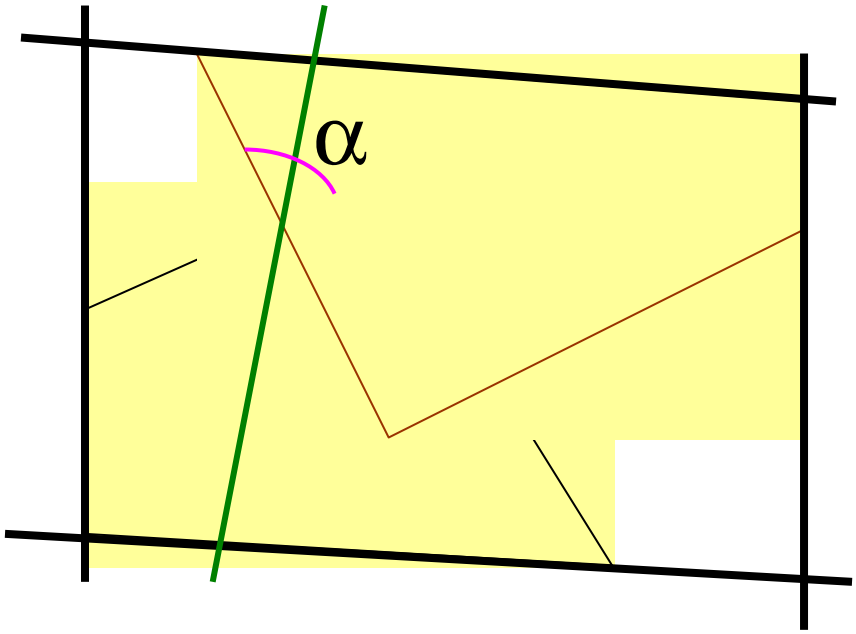

Figure 18 Bisecting Cutting Plane

If the angle is very obtuse (e.g. cos $\alpha > 0.99$) then the two intersecting hyperplanes are included in the convex approximation, and the cutting plane is not. Otherwise the two intersecting hyperplanes are excluded from the convex approximation and the cutting plane is included. Also included are all the non-redundant hyperplanes in the envelope. The result is shown in Figure 19. Note that $P \cap Q \subseteq R \subseteq P \cup Q$.

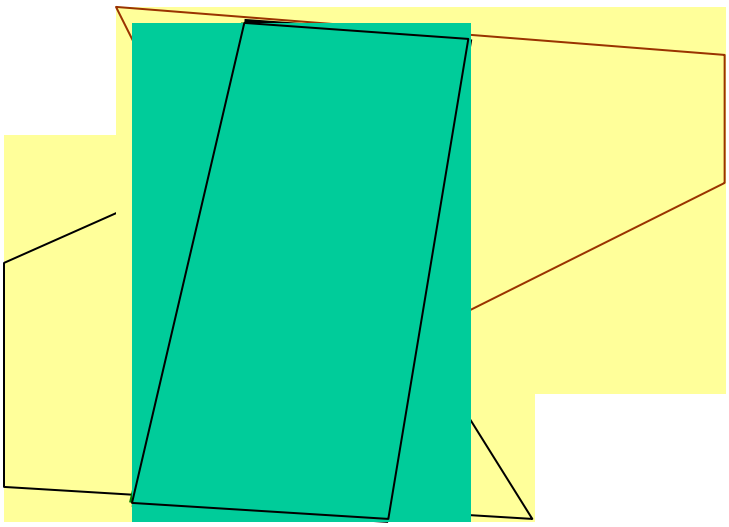

Figure 19 Convex Approximation

This method was implemented and used to compute the robust controllable set for the switchover phase, phase III, in the three-tank reconfiguration scenario of Figure 15.

The linearized and discretized model of the system, includes two discrete modes: with valve V_{13} open (96)and closed (97) respectively.

$$\begin{bmatrix} h_1(t+1) \\ h_2(t+1) \\ h_3(t+1) \end{bmatrix} = \begin{bmatrix} 0.9948 & 0 & 0.0052 \\ 0 & 1 & 0 \\ 0.0052 & 0 & 0.9858 \end{bmatrix} \begin{bmatrix} h_1(t) \\ h_2(t) \\ h_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 64.93 \\ 0 \end{bmatrix} u_{pump2}(t) + \begin{bmatrix} -0.0016 \\ 0 \\ 0.0007 \end{bmatrix} \tag{96}$$

$$\begin{bmatrix} h_1(t+1) \\ h_2(t+1) \\ h_3(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.9909 \end{bmatrix} \begin{bmatrix} h_1(t) \\ h_2(t) \\ h_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 64.93 \\ 0 \end{bmatrix} u_{pump2}(t) + \begin{bmatrix} 0 \\ 0 \\ 0.0009 \end{bmatrix} \tag{97}$$

with constraints

$$\Omega = \{h \mid 0 \le h_1 \le 0.55,\ 0 \le h_2 \le 0.6,\ 0.09 \le h_3 \le 0.1\} \tag{98}$$

$$\chi_{1,2} = \{h, u \mid h_3 \le h_1,\ 0 \le h_1 \le 0.6,\ 0 \le h_2 \le 0.6,\ 0 \le h_3 \le 0.6\} \tag{99}$$

$$U = \{u \mid 0 \le u_{pump} \le 0.0001\} \tag{100}$$

$$T = \{h \mid 0 \le h_1 \le 0.2,\ 0.4 \le h_2 \le 0.6,\ 0.09 \le h_3 \le 0.11\} \quad (101)$$

The set $K_{1,200}(\Omega,T)$ is shown in Figure 20.

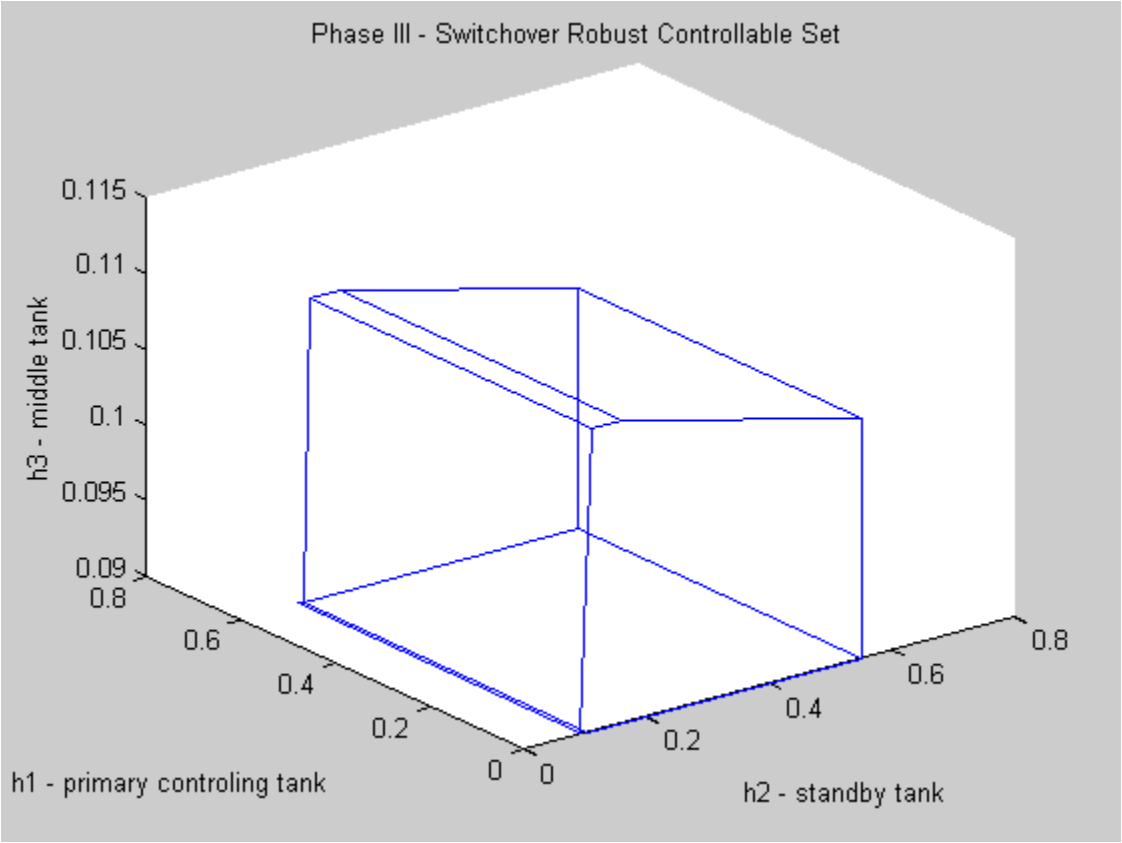

Figure 20 $K_{1,200}(\Omega,T)$ for the three-tank switchover

REFERENCES

[1] National Transportation Safety Board, "American Airlines, Inc., DC-10, N110AA, Chicago International Airport, Chicago, IL May 25, 1979", NTSB-AAR-79-17, 12/21/1979

[2] Merav Halperin and Aharon Lapidot, editors, "Pressure Suit", Israel Defense Ministry Press, 1987, pp. 148-153

[3] Blanke M. , Frei C. , Kraus F. , Patton R.J. , and Staroswiecki M "What is Fault-tolerant Control.", Plenary address, IFAC Symposium SAFEPROCESS 2000, Budapest, 14-16 June, pp 40-51, 2000

[4] E.D. Sontag "Interconnected automata and linear systems: A theoretical framework in discrete time", in R. Alur, T.A. Henzinger, and E.D. Sontag editors, Hybrid Systems III- Verification and Control, LNCS 1066, pp. 436-448, Springer-Verlag 1996

[5] Heemels W.P.M.H., B. De Schutter and A. Bemporad, "Equivalence of Hybrid Dynamical Models", Automatica 37(7), July 2001

[6] Domenico Mignone, Alberto Bemporad, Manfred Morari "A Framework for Control, Fault Detection, State Estimation, and Verification of Hybrid Systems", Proceedings of the American Control Conference, San Diego, 1999, pp. 134-138

[7] X.D. Koutsoukos and P.J. Antsaklis. "Hierarchical Control of Piecewise Linear Hybrid Dynamical Systems Based on Discrete Abstractions", Interdisciplinary Studies of Intelligent Systems, Notre Dame University, Technical Report ISIS-2001-001, February 2001

[8] James B. Rawlings "Tutorial Overview of Model Predictive Control" IEEE Control systems Magazine, June 2000, pp. 38-52

[9] D.P. Bertsekas and I.B. Rhodes "On the minmax reachability of target sets and target tubes",Automatica 7:233-247, 1971

[10] Bemporad A. and Morari M. "Control of systems integrating logic, dynamics, and constraints", Autmotatica Special issue on hybrid systems, Vol 35, No. 3, pp. 407-427, July 2000

[11] Lunze, J. and Steffen, T. "Reconfigurable Control of Quantised Systems", proceedings of SAFEPROCESS 2000: the 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes, Budapest, June 2000

[12] Harel, D. "Statecharts: a visual formalism for complex systems." in Science of Computer Programming 8,3 pages, 231-274, June 1987.

[13] X. Koutsoukos, P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon. “Supervisory control of hybrid systems”, Proceedings of the IEEE, 88:1026--1049, July 2000.

[14] R. J. Patton, "Fault-tolerant control: The 1997 situation," in proceeding of IFAC Safeprocess 1997 pp. 1033-1055 in Fault Detection, Supervision and Safety for Technical Processes, Kingston Upon Hull, UK, pp. 1029--1051, 1997

[15] E.C. Kerrigan and J.M. Maciejowski, "Invariant Sets for Constrained Nonlinear Discrete-time Systems with Application to Feasibility in Model Predictive Control", Proceedings of the 39th Conference on Decision and Control, Sydney, Australia, December 2000

[16] Davoren, J. M., and Nerode, A. “Logics for Hybrid systems”, *Proceedings of the IEEE* special isssue on Hybrid Systems Vol. 88, No. 7, 2000

[17] Alur, R., Courcoubetis, C. Henzinger, T.and Ho, P.H. “Hybrid Automata: An Algorithmic Approach to the specification and verification of Hybrid Systems”, in Hybrid Systems, Lecture Notes in computer Science 736, 1993.

[18] T.A. Henzinger, J. Raskin: “Robust Undecidability of Timed and Hybrid Systems,” in Hybrid Systems: Computation and Control, Springer Verlag LNCS Vol. 1790, edited N. Lynch and B. Krogh., pp. 145-159.

[19] E. Asarin, O. Bournez, T. Dang, O. Maler, A. Pnueli, “Effective Synthesis of Switching Controllers for Linear Systems”, *Proceedings of the IEEE* 88, No. 7, 2000, 1011-1025

[20] B.J. Kuipers and K. Astrom, 1994, “The composition and validation of heterogeneous control laws” Automatica 30(2):233-249

[21] Raisch, J.:” Discrete Abstractions of Continuous Systems - an Input/Output Point of View. “ Mathematical and Computer Modelling of Dynamical Systems 6(1), 2000, special issue on Discrete Event Models of Continuous Systems. pp. 6-29.

[22] Alur, R. Henzinger, T., Lefferriere, G. and Pappas, G.”Discrete Abstractions of Hybrid Systems”, *Proceedings of the IEEE* special isssue on Hybrid Systems Vol. 88, No. 7, 2000

[23] E.C. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, University of Cambridge, UK, November 2000

[24] E.C. Kerrigan and J.M. Maciejowski. Robust Feasibility in Model Predictive Control: Necessary and Sufficient Conditions. In *Proceedings of the 40th Conference on Decision and Control*, Orlando, Florida, USA, December 2001

[25] A.Bemporad, D. Mignone, M.Morari, “An Efficient Branch and Bound Algorithm for State Estimation and Control of Hybrid Systems”, Proceedings of the European Control Conference, Karlsruhe, Germany, August 1999

[26] McIlraith, S., Biswas, G., Clancy, D., Gupta, V. “Hybrid Systems Diagnosis”, in Hybrid Systems: Computation and Control, Springer Verlag LNCS Vol. 1790, edited N. Lynch and B. Krogh., pp. 283-295.

[27] Stumptner M., Wotawa F. “Reconfiguration using Model-based Diagnosis”, *Proceedings of the Tenth International Workshop on Principles of Diagnosis*, Loch Awe, Scotland, June, 1999.

[28] Astrom, K.P., Albertos, P. , Blanke, M. Isidori, A, Schaufelberger W. and Sanz, R. ”Control of Complex Systems” Springer, London, 2001

[29] A. Morse, “Supervisory control of families of linear set-point con-trollers Part 1: Exact matching,” *IEEE Trans. Automat. Contr*., vol. 41, pp. 1271–1281, 1996

[30] Roozbeh Izadi-Zamanabadi,,“Fault Tolerant Supervisory Control – system Analaysis and Logic Design”, *PhD Thesis* , Department of Control Engineering, Aalborg University, Denmark, 1999

[31] Tsuda K., Mignone D., Ferrari-Trecate G. and Morari M., "Reconfiguration Strategies for Hybrid Systems", in proceedings of the American Control Conference 2001.

[32] Staroswiecki, M. & Gehin, A-L. “From Control to Supervision”, proceedings of SAFEPROCESS 2000: the 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes, Budapest, June 2000, pp. 312-323

[33] Kjaerj, J., "A Hybrid System Approach to Fault-Tolerant Control of a 3-Tank System" Final Engineering Study Project, ESIEE/France, 1998

[34] Lafferriere, G. Pappas, and S. Sastry. “Hybrid systems with finite bisimulations.” In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, Hybrid Systems V, volume 1567 of Lecture Notes in Computer Science, pages 186-203. Springer, 1999.

[35] Noura, H. Theilliol, D Sauter, D. “Actuator fault-tolerant control design: demonstration on a three-tank-system”, International Journal of Systems Science, Vol. 31, No. 9 pp. 1143-1155, 2000

[36] A.L. Gehin and Starosweicki, M. “A formal approach to reconfigurability analysis application to the three tank benchmark”, Proceedings of the European Control Conference, Karlsruhe, Germany, August 1999.

[37] Tyler and Morari “Propositional logic in control and monitoring problems”, Automatica 35 (1999) 565-582

[38] F. Blanchini, "Set invariance in control - a survey", *Automatica,* Vol 35, no. 11, pp.1747-1768, November 1999

[39] Domenico Mignone, “Moving Horizon Estimation and Fault Detection of Mixed Logic Dynamical Systems”, *Postdiploma Thesis*, Automatic Control Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland, August 1999

[40] F. Blanchini “Ultimate Boundedness Control for Discrete-Time Uncertain System via Set Induced Lyapunov Functions”, IEEE Transactions on Automatic Control, Vol. 39, no. 2, p. 428-433, February 1994

[41] C.E.T. Doréa and J.C. Hennet “(A,B)-invariant polyhedral sets of linear discrete-time systems” Journal of Optimization Theory and Applications, 103(3), pages 521-541, December 1999.

[42] F. Blanchini and S Miani, "Any Domain of Attraction for a Linear Constrained System is a Tracking Domain of Attraction" in SIAM Journal on Control and Optimization, Vol 38, no. 3. pp. 971-944, March 2000

[43] J. Lunze. Laboratory Three Tanks System Benchmark for the Reconfiguration Problem. Technical report, Tech. Univ.of Hamburg-Harburg, Inst. of Control. Eng., Germany, 1998

[44] A. Bemporad, K. Fukuda, and F. D. Torrisi “Convexity recognition of the union of polyhedra” Computational; Geometry 18 (3) pages 141-154, April 2001.

[45] M. Kothare, V. Balakrishnan and M. Morari “An LMI approach to robust constrained model predictive control”, Automatica, vol. 32, no. 10, pages 1361-1379, November 1996

[46] E. Alcorta Garcia and P.M. Frank, “Multiplicative fault isolation in linear systems”, Proceedings of the 38th Conference on Decision and Control, Phoenix, Arizona USA, December 1999.

[47] A. Chutinan and B.H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In The 37th IEEE Conference on Decision and Control: Session on Synthesis and Verification of Hybrid Control Laws (TM-01), 1998

[48] A. Bemporad, F.D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In B. Krogh and N. Lynch, editors, Hybrid Systems: Computation and Control, volume 1790 of Lecture Notes in Computer Science, pages 45-58. Springer Verlag, 2000

[49] E. Asarin, O. Bournez, T. Dang, O. Maler and A. Pnueli, “Effective Synthesis of Switching Controllers for Linear Systems”, Proceedings of the IEEE, Vol. 88, No. 7, July 2000

[50] E. Asarin, O. Bournez, T. Dang, and O. Maler, “Reachability analysis of piecewise-linear dynamical systems,” in Hybrid Systems: Computation and Control, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, no. 1790, Lecture Notes in Computer Science.

[51] O. Bournez, O. Maler, and A. Pnueli, “Orthogonal polyhedra: Representation and computation,” in Hybrid Systems: Computation and Control, F. Vaandrager and J. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1569, Lecture Notes in Computer Science, pp. 20–31.

[52] Elena de Santis “On Invariant Sets for Constrained Discrete time Linear systems with Disturbances and Parametric Uncertainties.” Automatica, Vol 33. No 11. pp. 2033-2039, 1997

[53] L. Chisci, R Falugi and G. Zappa, “Predictive control for constrained systems with polytopic uncertainty” Proceedings of the American Control Conference, Arlington, VA June 25-27, 2001

[54] Bemporad, K. Fukuda, and F. D. Torrisi. “Convexity recognition of the union of polyhedra”, Computational Geometry, Vol. 18, No. 3, pp. 141-154, April 2001

[55] H. Lin, X.D. Koutsoukos, P.J. Antsaklis “HyStar: a Toolbox for hierarchical control of Piecewise Linear Hybrid Dynamical Systems”, proceedings of the 2002 American Control Conference, Anchorage Alaska, to appear.

[56] H. Lin, X.D. Koutsoukos, P.J. Antsaklis “Hierarchical control for a class of uncertain piecewise linear hybrid dynamical systems”, in proceedings of the 15th IFAC World Congress on Automatic Control, Barcelona, Spain, 2002, to appear.

ELECTRICAL ENGINEERING

HIERARCHICAL CONTROL RECONFIGURATION FOR A CLASS OF HYBRID SYSTEMS

TAL PASTERNAK

Dissertation under the direction of Professor Janos Sztipanovits

Active approaches to Fault-Tolerant Control seek to maintain safety and availability of a plant by reconfiguring its control system when a fault occurs. The fundamental problem in this approach is complexity: closing the diagnosis-reconfiguration-control loop online poses a major challenge. This is particularly hard for hybrid Systems, which are as yet not well understood.

In this dissertation a hierarchical approach is considered aimed at lowering the complexity of low-level control. The hierarchy consist of a supervisory controller that guides the system through a sequence of regions in the state space, and a configuration manager that selects the plant inputs for a low-level controller to use, and the low-level controller. The theory of invariant sets provides the necessary tools for the reachability calculations, which are performed at design time. Using model-predictive control as a case in point, it is shown that the hierarchical architecture facilitates the reduction of the complexity of the low-level control in three ways: reducing the number of discrete modes of the plant that need to be considered, reducing the prediction horizon needed for control and reducing the number of plant inputs manipulated by the controller. A three-tank

system is used to illustrate the approach. The system model which is considered is a constrained discrete-time piecewise-affine system with additive state disturbance.

The architecture requires enumeration of configurations but not enumeration of faults. A feature of the hierarchical architecture, is that controller configurations can be designed for each supervisory control mode separately and control laws can be designed separately for each configuration. The significance from a practical point of view, is that design of fault-accommodation logic can be simplified by adopting the hierarchical approach even if it is designed by conventional simulation and testing methods, rather than formal verification.

Approved_____ Date_____