LECTURE NOTES FOR THE COURSE SC 4160

Modeling and Control of Hybrid Systems

Bart De Schutter and Maurice Heemels

December 2015

TU Delft
Artikelnummer 06917690004

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
http://www.dcsc.tudelft.nl

Authors:

B. De Schutter
Delft Center of Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
email: b.deschutter@tudelft.nl
URL: http://www.dcsc.tudelft.nl

W.P.M.H. Heemels
Hybrid & Networked Systems group
Mechanical Engineering
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
email: W.P.M.H.Heemels@tue.nl
URL: http://www.dct.tue.nl/heemels

Contents

Preface

These lecture notes have their origins in the lecture notes for the DISC (Dutch Institute for Systems and Control) postgraduate course on hybrid systems, which is taught jointly by Bart De Schutter (Delft University of Technology) and Maurice Heemels (Eindhoven University of Technology).

We want to acknowledge the contribution of Alberto Bemporad, Daniel Liberzon, John Lygeros, Hans Schumacher, and Arjan van der Schaft to parts of these lecture notes, and we also want to thank them for the fruitful discussions we had with them on various topics discussed in these lecture notes.

List of abbreviations

DAE	Differential Algebraic Equation
DES	Discrete Event System(s)
ELC	Extended Linear Complementarity
GUAS	Globally Uniformly Asymptotically Stable
LC	Linear Complementarity
LCS	Linear Complementarity System
LCP	Linear Complementarity Problem
LMI	Linear Matrix Inequality
LP	Linear Programming
LPCS	Linear Passive Complementarity System
MIQP	Mixed-Integer Quadratic Programming
MLD	Mixed Logical Dynamical
MMPS	Max-Min-Plus-Scaling
MPC	Model Predictive Control
ODE	Ordinary Differential Equation
PWA	PieceWise Affine
PWM	Pulse Width Modulation
QP	Quadratic Programming
SQP	Sequential Quadratic Programming
TPBVP	Two-Point Boundary-Value Problem

List of symbols

$\mathbb{N}$	set of the nonnegative integers $(\mathbb{N} = \{0, 1, 2, \ldots\})$
$\mathbb{R}$	set of the real numbers
$\mathbb{R}^+$	set of the nonnegative real numbers $(\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geqslant 0\})$
$\mathbb{R}^-$	set of the nonpositive real numbers $(\mathbb{R}^- = \{x \in \mathbb{R} \mid x \leqslant 0\})$
$\mathbb{Z}$	set of the integers
$\#S$	cardinality (number of elements) of the set S
a^{T}	transpose of the vector a
$a \perp b$	orthogonality of the vectors a and b $(a \cdot b = 0)$
$L_2[a, b]$	Lebesgue space of square integrable functions on the interval $[a, b]$
$L_1[a, b]$	Lebesgue space of integrable functions on $[a, b]$
M_{IJ}	submatrix of M given by $(M_{ij})_{i \in I, j \in J}$
$M_{I\bullet}$	submatrix of M containing the rows indexed by I (if $M \in \mathbb{R}^{m \times n}$ then $M_{I\bullet} = M_{IJ}$ with $J = \{1, \ldots, n\}$)
$M_{\bullet J}$	submatrix of M containing the columns indexed by J (if $M \in \mathbb{R}^{m \times n}$ then $M_{\bullet J} = M_{IJ}$ with $I = \{1, \ldots, m\}$)
$\det M$	determinant of the (square) matrix M
$f : X \to Y$	function with domain X and target set Y
$f : x \mapsto f(x)$	function mapping x to $f(x)$

Chapter 1

Introduction

1.1 Multi-disciplinary design

When designing a high-tech system such as a wafer stepper, electron microscope, copier, etc. multiple disciplines need to make the overall design in close co-operation. E.g., the electronic design, mechanical design and software design together need to describe a consistent, functioning machine. The designs are often made in parallel by multiple groups of people, where the communication between these groups is hampered by lack of common understanding, organizational issues, politics and out-of-phase project evolution. A typical example of the latter is that the mechanical design often precedes the electronic design, which on its turn precedes the software design. In addition, the complexity of a copier (typically millions lines of codes, tens of thousands of mechanical components like pinches, springs, belts, motors, bolts, etc.) gives rise to many cross-disciplinary design decisions. Often choices are made which may have benefits in one discipline but disadvantages in other disciplines. To make a good decision the overall effect of such a choice needs to be evaluated, as early as possible. Therefore, a framework that supports efficient evaluation of design choices over multiple disciplines would be very beneficial.

The research field of multi-disciplinary design methods is relatively immature as opposed to the more conventional mono-disciplinary research areas like mechanical, electrical, or software engineering. The latter scientific disciplines have little experience in the field of multi-disciplinary creation method for high-tech systems. Most experience can be found in industry, but is often implicitly present in the minds of experienced system engineers.

There are only a few scientific research directions that aim at approaching the design problems of high-tech systems in a rigorous manner. One of them is hybrid system theory. Hybrid system theory studies the behavior of models of (parts of) high-tech systems that involve both continuous dynamical models (using e.g., differential equations) typically describing the physical / mechanical part of the machine and discrete models (e.g., finite state machines or automata) describing the software behavior. The hybrid field is relatively immature and many issues are at present unsolved (at least at the large-scale needed for industrial usefulness). However, the industrial need for analysis / synthesis methods for high-tech machines in which the “hybrid interaction” plays an important role, will stimulate the research in this domain over the years to come.

1.2 Hybrid systems

As already mentioned, there is a need for understanding the complex interaction between the discrete and continuous dynamics in high-tech systems. The modeling formalisms used combine models from the software discipline on one hand and control engineering on the other. Due to this mixed character, these models are called "hybrid systems[1]."

As an illustrative example of a simple hybrid system we mention the regulation of temperature in a house. In a simplified description, the heating element is assumed either to work at its maximum power or to be turned off. This is a system that can operate in two modes: "on" and "off". In each mode the evolution of the temperature $T(t)$ can be described by a differential equation (see Figure 1.1). Switching between two modes of operation is controlled by a logical device (the embedded controller) called the *thermostat*. The mode is changed from "on" to "off", when the temperature crosses a certain upper value $T_{\rm upp}$ (determined by the desired temperature). Vice versa, if the temperature drops below a minimum value $T_{\rm low}$, the heating is switched "on."

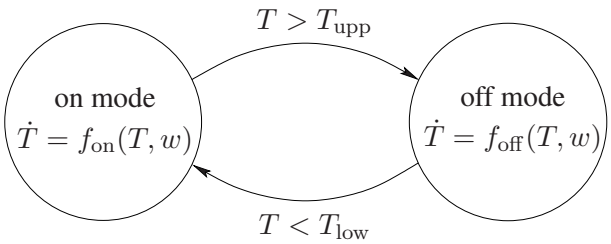

Figure 1.1: A temperature control system as an example of a hybrid system. The (vector) signal $w(t)$ contains external inputs such as, e.g., the outside temperature.

Of course, the thermostat example is trivial if compared to the complexity of a high-tech system. Let us take the example of a copier as indicated in figure 1.2. The application context is best characterized by document printing systems that are highly productive, reliable, and user-friendly. These systems can print on several sizes of media with different weights, automatically on both sides, and include stapling, booklet production, or other types of finishing. In order to be perceived as reliable devices, such copiers must be very robust with respect to variations in media. As the printing speed is rather high (typically above 1 image per second), timing requirements are tight and advanced mechatronics are indispensable. This indicates that variations in timing parameters that relate to paper and image transport must be controlled up to a high degree. This becomes even more apparent if one realizes that the positioning of images on paper has tolerances well below 1 mm.

When considering the embedded control of these systems, one should think of controlling multiple sheets that travel the paper path simultaneously and synchronizing this sheet flow with the imaging process. When the copier is in normal operation, a sheet is separated from the trays in the paper input module (PIM), after which it is sent to the paper path that transports the sheets accurately in the direction of the print engine, where the image is fused on a sheet of paper. After that, the sheet is turned for duplex printing, or transported by the paper path to the finisher. When sheets do not arrive at the scheduled times at specific positions (within certain margins) error handling is alerted and has to take care of a safe stop of the print job.

Typically, hierarchical control methods are being utilized in these type of control problems. On the lowest level, local single-input single-output controllers (e.g., PID) are implemented for controlling the

[1]Term used in this context for the first time by Witsenhausen in 1966 [220].

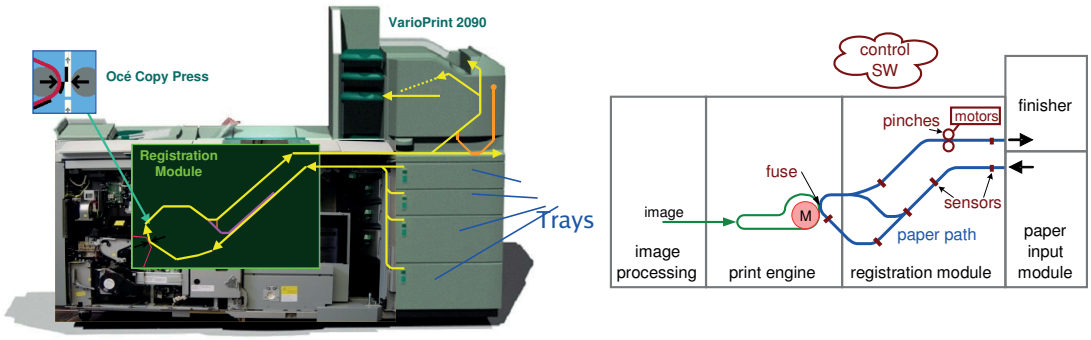

Figure 1.2: Illustration and schematic picture of a copier.

velocity of the motors that drive the rollers that transport the sheets. The set-points are generated by a scheduling algorithm that provides idealized position profiles for the sheets, that have to be tracked by the low-level controllers. When the sheets are behind or ahead of schedule minor adaptations are made to the velocity set-points. When the arrival of sheets at the discrete sensor locations is out of bounds, error handling takes care that all the sheets are transported to positions from which the sheets can be removed easily by a user. Also when temperatures get out of bounds, error handling has to take care that no hazard occurs for the users. The complete embedded controllers are implemented on processing platforms that take care of the different control layers ranging from the implementation of local digital controllers, to exception handling, safety, alarm detection, switching between operating modes, and starting and stopping procedures (the supervisory layer). The overall system consisting of the physical plant and the embedded controller will form a complex hybrid system. Similar examples include automated highways [100], coordinated submarine systems, and air-traffic management [203]. In these situations the digital and logical aspects are brought in by designers to control the physics and mechanics in the system.

The hybrid nature of hybrid systems is not necessarily caused by design in smooth systems. Although many examples originate from systems involving both digital controllers and physical processes as is often the case in high-tech systems, the switching between dynamical regimes is naturally present in a variety of systems. E.g., in mechanics, one encounters friction models that make a clear distinction between stick and slip phases. Other examples include models describing the evolution of rigid bodies. In this case the governing equations depend crucially on the fact whether a contact is active or not. The dynamics of a robot arm moving freely in space is completely different from the situation in which it is striking the surface of an object. Backlash in gears and dead zones in cog wheels also results in multi-modal descriptions. It is not difficult to come up with interesting applications in the mechanical area: control of robotic manipulators driving nails or breaking objects [47], vibration control in suspension bridges [119], reduction of rattling in gear boxes of cars, drilling machines [184], simulation of crash-tests, regulating landing maneuvers of aircraft, design of juggling robots [48], and so on.

Examples are not only found in the mechanical domain. Nowadays switches like thyristors and diodes are used in electrical networks for a great variety of applications in both power engineering and signal processing. Examples include switched-capacitor filters, modulators, analog-to-digital converters, switching power converters, choppers, etc. In the ideal case, diodes are considered as elements with two (discrete) modes: the blocking mode and the conducting mode. Mode transitions for diodes are governed by state events (sometimes also called “internally induced events”), i.e., certain system variables (current or voltage) changing sign. In duty-ratio control the duration of a switch being open and closed (or the

ratio between them in a fixed time interval) is determined by a control system and hence, the transitions are triggered by time events ("externally induced events").

Other sources of this behavior (also called multi-modal behavior) are saturation, hysteresis, and sensor and actuator failures. Actuator saturation truncates implemented control values outside the actuator range. Sensors provide reliable and accurate measurements only within a specific region, while outside the region the only available information is whether the measured signal is above the maximum or below the minimum of the sensor range. The malfunctioning of sensors or actuators has the effect that control signals or measurements are not available and as a consequence, the input-output description changes abruptly. Control design must take switching and impact phenomena into account such that a desirable behavior of the closed loop system is realized.

1.3 Hybrid systems: Major challenges

How dependent our lives are on computer technology is illustrated by the efforts taken to solve the millennium bug. The number of computer-controlled products in our homes will grow even further in the coming years. To support this evolution, new methodologies for the analysis and synthesis of hybrid systems are needed. To guarantee the safety and proper functionality, we have to improve our understanding of the interaction between physical processes, digital controllers and software, as all three parts influence the dynamic behavior of the overall process.

Nowadays, the design of such combined systems is often performed by methods exclusively tailored for either discrete-event systems (DES) or time-continuous systems. As a result, the models neglect either the continuous or the discrete characteristics of the system. As an example, consider the air-traffic management of an airport. In describing the airport accurately, the model must contain the differential equations determining the trajectories of the aircraft, as well as the human and/or organizational processes realizing the communication and assignments between the aircraft and the traffic control center. An air-traffic controller obtained from a model not incorporating one of these aspects, may fail in practice or will at least show a lower performance than a controller designed by techniques incorporating both the discrete and the continuous behavior.

Another approach for the combined design consists in separating the analysis and design of the continuous and discrete parts and merging them in the final stage. The synthesis of a digital controller (PID, H_∞, IMC, etc.) for a process in a certain operating point is backed up by the vast literature on systems and control theory. Also the tools for the design of a DES controller taking care of, e.g., mode switching and exception handling are available. However, a combined controller design of the system is currently impossible. The merging of the complete embedded controller with the physical plant is performed in a heuristic and ad hoc manner and requires often years of tuning, prototyping and trouble-shooting, which are extremely expensive and time consuming. The time-to-market and the necessary investments for new products can be decreased considerably, if techniques are available that facilitate combined synthesis of both the discrete and continuous parts.

Fortunately, it is widely recognized by the academic world that mixing different devices and concepts will play an increasingly important role in industry. Starting from their own backgrounds, control engineers [13, 169], computer scientists [186], mathematicians and simulation experts work towards systematic methods to support the development of new products. The increasing interest in this research area has become apparent from a series of workshops on hybrid systems in recent years [8, 11, 12, 104, 123, 159].

1.4 Classification of dynamical systems

In [153] a useful characterization has been made of dynamical systems. Roughly speaking a *dynamical system* describes the *evolution* of a *state* over *time*. As a well-known illustration we could consider the following differential equation:

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

The state variable at time t in this case is given by $x(t)$ (e.g., position and velocity of an aircraft) and typically takes values in $\mathcal{X} \subseteq \mathbb{R}^n$ and evolves over time $t \in \mathcal{T} \subseteq \mathbb{R}$ according to (1.1). The variable $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ at time t denotes either control inputs, that may be chosen as we like (e.g., commands of the pilot to the engines or wings), or disturbances (e.g., wind affecting the plane). Loosely speaking, one could say that the state $x(\tau)$ at time τ summarizes all the information from the past (for times $t \leqslant \tau$) of the system that is needed in order to understand the future behavior of the state $x(t)$ for $t > \tau$ except for the purely external effects due to the inputs and disturbances [200]. In principle one could say that a dynamical system is defined by a relation between the current state and an applied input or disturbance and a state at a later time instant. So, for the differential equation given in (1.1) this means that we have a map ϕ from the (initial) state $x \in \mathcal{X}$, the initial time $\tau \in \mathcal{T}$, the (final) time $\sigma \in \mathcal{T}$ with $\sigma \geqslant \tau$ and a function $u : [\tau, \sigma] \to \mathcal{U}$ to the value $x(\sigma)$ of the state at time σ. Hence,

$$x(\sigma) = \phi(\tau, \sigma, x, u) \ .$$

This is an interesting way to look at systems which yields a nice unifying way to include also computer science models. Sontag [200] formalized this concept of *system* or *machine* as follows.

Definition 1.4.1 A system or machine $\Sigma = (\mathcal{T}, \mathcal{X}, \mathcal{U}, \phi)$ consists of:

- A time set $\mathcal{T}$;
- A nonempty set $\mathcal{X}$ called the state space of Σ;
- A nonempty set $\mathcal{U}$ called the control-value or input-value space of Σ;
- A map $\phi : \mathcal{D}_\phi \to \mathcal{X}$ called the transition map of Σ, which is defined on a subset $\mathcal{D}_\phi$ of $\{(\tau, \sigma, x, \omega) \mid \tau, \sigma \in \mathcal{T}, \tau \leqslant \sigma, x \in \mathcal{X}, \omega : [\tau, \sigma) \to \mathcal{U}\}$ such that the non-triviality, restriction, semi-group and identity properties (see [200] for exact descriptions) hold.

Note that the time set should have a total ordering[2] "$\leqslant$." In this case intervals of the form $[\tau, \sigma)$ can be defined. The transition function defines just as in the example of the differential equation above the new state at time σ given the (initial) state x at (initial) time τ when the input ω is applied on the interval $[\tau, \sigma)$.

Based on the type of the state the following classification can be made of systems:

1. **continuous state:** The state takes values in a continuous set, e.g., $\mathbb{R}^n$ for some $n \geqslant 1$. We will use $x \in \mathbb{R}^n$ to denote the continuous state.
2. **discrete state:** The state takes values in a countable or finite discrete set $\{q_1, q_2, \ldots\}$. We will use q to denote the discrete state.

[2]The ordering $\leqslant$ defined on $\mathcal{T}$ is total if for any two elements $\tau, \nu \in \mathcal{T}$ we have $\tau \leqslant \nu$ or $\nu \leqslant \tau$.

Based on the set of times $\mathcal{T}$ over which the state evolves, dynamical systems can be categorized in

1. **continuous time:** $\mathcal{T}$ is a continuous set, e.g., a subset of $\mathbb{R}$. We will use $t \in \mathbb{R}$ to denote continuous time.
2. **discrete time:** $\mathcal{T}$ is a discrete set, e.g., a subset of the integers $\mathbb{Z}$. We will use $k \in \mathbb{Z}$ to denote discrete time (and t_k to denote the corresponding "real" time instant).

Finally, we make a distinction between systems based on the mechanism that drives their evolution, which can be:

1. **time-driven:** The state of the system changes as time progresses, i.e., continuously (for continuous-time systems), or at every tick of the clock (for discrete-time systems).
2. **event-driven**: The state of the system changes due to the occurrence of an event. An event corresponds to the start or the end of an activity. In general, event-driven systems are asynchronous and the event occurrence times are not equidistant. Typical examples of event-driven systems are manufacturing systems, telecommunication networks, parallel processing systems, and logistic systems. For a manufacturing system possible events are: the completion of a part on a machine, a machine breakdown, or a buffer becoming empty.

We can also have combinations of continuous and discrete states, of continuous and discrete time, or of time-driven and event-driven dynamics. The resulting systems are called hybrid. In these lecture notes a hybrid system essentially is a system with a combination of a discrete and continuous state and a combination of time-driven and event-driven evolution. Typically, phases of time-driven evolution are separated by discrete time instants when "something happens" (e.g., the occurrence of an event that results in a mode change of the system, see also Section 1.5.2; usually the mode is then characterized by a discrete state variable). The time-driven nature can be defined on continuous or discrete sets of times.

An example of a time-driven continuous-time continuous-state system is a wind mill where the rotational speed changes continuously over time depending on, e.g, the speed and the direction of the wind. A digital computer with a certain clock rate that executes a given program can then be considered as a time-driven discrete-time (discrete-state) system. An example of an event-driven continuous-time[3] discrete-state system is a waiting line at the counter of a supermarket. The state is the number of customers waiting in the queue. An event is then the arrival or departure of a customer. If we consider a water tank with the water level as the state and with adding or removing an amount of fluid to or from the tank, then we have an event-driven continuous-time continuous-state system.

1.4.1 Models for time-driven systems

The behavior of continuous-time time-driven continuous-state systems can be described by a system of differential equations of the form

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

[3] In the waiting queue and the water tank the time variable evolves continuously. However, we could also introduce a "logical time" or event counter k, which only increments when an event occurs. In that case the queuing system and the water tank can be considered as discrete-time systems as far as the "logical time" is concerned. If we keep on using the "real time", the waiting queue and the water tank can be artificially transformed into a discrete-time system by only allowing the events to take place at the ticks of a clock (e.g., by putting a computer controlled gate before or after the queue, or by letting a computer-controlled robot take care of adding or removing the fluid).

where $x(t)$ and $u(t)$ are the state and the input of the system at time t, respectively. Note that these descriptions are typically used for continuous-state systems, i.e., $x(t) \in \mathbb{R}^n$. In the linear case, often descriptions are used of the form

$$\dot{x}(t) = Ax(t) + Bu(t)$$

in which $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are real matrices. In this case the transition map ϕ in Definition 1.4.1 is equal to

$$\phi(\tau, \sigma, x, u) = e^{A(\sigma-\tau)}x + \int_{\tau}^{\sigma} e^{A(\sigma-\theta)}Bu(\theta)d\theta,$$

where $\tau, \sigma \in \mathbb{R}$, $x \in \mathbb{R}^n$ and $u : [\tau, \sigma) \to \mathbb{R}^m$ (which should be integrable).

Time-driven discrete-time (or sampled) systems can be modeled by a system of difference equations of the form

$$x(k+1) = f(x(k), u(k))$$

where $x(k)$ and $u(k)$ are the state and the input of the system at discrete time k. In the linear case, often descriptions are used of the form

$$x(k+1) = Ax(k) + Bu(k) \ .$$

In that case the transition map ϕ in Definition 1.4.1 is equal to

$$\phi(\tau, \sigma, x, u) = A^{\sigma-\tau}x + \sum_{\theta=\tau}^{\sigma-1} A^{\sigma-\theta-1}Bu(\theta)$$

for $\tau, \sigma \in \mathbb{Z}$, $x \in \mathbb{R}^n$ and $u : [\tau, \sigma) \to \mathbb{R}^m$.

1.4.2 Models for event-driven systems

Automata or *finite state machines* are the most common models for (discrete-state)[4] event-driven systems. We need some notation before we can give a formal definition of automata. For a set V we denote the collection of all subsets of V (the power set) by $P(V)$. Moreover, if we have two sets V and W a *partial function* from V to W is a mapping that is not necessarily defined for all values of V, but only for a subset D of V (its domain). Hence, if f is a partial function from V to W, then there is a subset $D(f) \subseteq V$ such that f is a function from $D(f)$ to W.

Definition 1.4.2 (Automaton) An *automaton* is defined by the triple $\Sigma = (\mathcal{Q}, \mathcal{U}, \phi)$ with

- $\mathcal{Q}$ a finite or countable set of discrete states;
- $\mathcal{U}$ a finite or countable set of discrete inputs or the input alphabet;
- $\phi : \mathcal{Q} \times \mathcal{U} \to P(\mathcal{Q})$ a partial *transition function*.

[4] We follow the convention of [58], in which discrete-event systems are defined as discrete-state event-driven systems. Note however that other authors [18, 176] do not explicitly limit this class to discrete-state systems.

In case $\mathcal{Q}$ and $\mathcal{U}$ are finite, we speak of a *finite automaton*.

The evolution of an automaton is rather simple; given a discrete state $q \in \mathcal{Q}$ and a discrete input symbol $u \in \mathcal{U}$, the transition function defines the collection of next possible states $\phi(q, u) \subseteq \mathcal{Q}$. Here, "next" state should be interpreted as the new state due to the occurrence of the event characterized by the input symbol u. Note that since ϕ is not necessarily defined for all combinations of q and u, this means that possibly from some discrete states not all input symbols can be applied. On the other hand, sometimes the set of next possible states may have more than one element. This so-called *nondeterminism* is illustrated in Example 1.4.4 below. In case this nondeterminism is absent, i.e., if $\phi(q, u)$ is undefined or has zero or one elements for each combination (q, u), we speak of a *deterministic automaton*, which fits more or less directly in the definition of a system as given in Definition 1.4.1. Indeed, $\mathcal{T}$ can be taken as $\mathbb{N}$, $\mathcal{X}$ as $\mathcal{Q}, \mathcal{U}$ as it is, and ϕ in Definition 1.4.1 can be trimmed down to the one-step function as above. In case one would like to define ϕ exactly as in Definition 1.4.1, this means successive application of the ϕ in Definition 1.4.2.

The dynamics of such a finite state machine or finite automaton can conveniently be represented by a *labeled directed graph*. This is a *graph* $\Gamma = (V, A, E)$ with V the set of *nodes or vertices*, A the set of labels, $E_a \subset V \times V$ the set of *arcs or edges* with label a, and $E = \{E_a \mid a \in A\}$. A labeled digraph $\Gamma = (V, A, E)$ is now associated to a finite state machine $\Sigma = (\mathcal{Q},\mathcal{U},\phi)$ as follows. To every discrete state $q \in \mathcal{Q}$, we assign a vertex $v_q \in V$ and to every discrete input $u \in \mathcal{U}$ a label $a_u \in A$. For every $u \in \mathcal{U}$ we link the set of edges E_{a_u} as follows:

$$(v_{q_1}, v_{q_2}) \in E_{a_u} \Leftrightarrow q_2 \in \phi(q_1, u).$$

To illustrate this consider the following example.

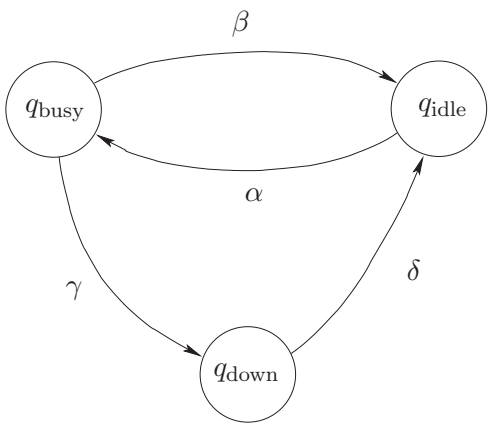

Figure 1.3: A processing unit in a manufacturing system.

Example 1.4.3 Consider a processing unit in a manufacturing system. The unit can be in one of the following three states: "idle" (q_{idle}), "busy" (q_{busy}) , and "down" (q_{down}). The automaton $(\mathcal{Q},\mathcal{U},\phi)$ that represents the evolution of the system is given in Figure 1.3. We have the set of states $\mathcal{Q} = \{q_{\text{idle}}, q_{\text{busy}}, q_{\text{down}}\}$ and the set of events $\mathcal{U} = \{\alpha, \beta, \gamma, \delta\}$ where the events are defined as follows. Suppose the system starts in state q_{idle} when a part arrives (the input event is labeled α in the graph), the system goes to the busy state q_{busy}. When the part has been processed (β), that system goes back to the idle state q_{idle}. However, if the machine breaks down during the processing (γ), it goes into the down

state q_{down}. After the processing unit has been repaired (δ), it goes back to the idle state q_{idle}. So we have

$$\begin{aligned} \phi(q_{\text{busy}}, \beta) &= \{q_{\text{idle}}\}, & \phi(q_{\text{idle}}, \alpha) &= \{q_{\text{busy}}\}, \\ \phi(q_{\text{busy}}, \gamma) &= \{q_{\text{down}}\}, & \phi(q_{\text{down}}, \delta) &= \{q_{\text{idle}}\} \ . \end{aligned}$$

Note that this automaton is deterministic as $\phi(q, u)$ — when defined — is a singleton.

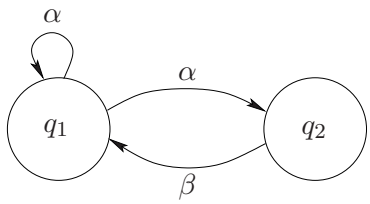

Figure 1.4: A non-deterministic automaton.

Example 1.4.4 (A non-deterministic automaton) Consider the automaton $(\mathcal{Q}, \mathcal{U}, \phi)$ of Figure 1.4 with $\mathcal{Q} = \{q_1, q_2\}, \mathcal{U} = \{\alpha, \beta\}$, and

$$\phi(q_1, \alpha) = \{q_1, q_2\}, \quad \phi(q_2, \beta) = \{q_1\} \ .$$

Suppose the system starts in state q_1. When event α occurs at state q_1, the transition can be either to state q_2 or back to state q_1. Hence, we have a nondeterministic automaton. Note that this non-determinism is usually the result of unmodeled or unspecified dynamics.

1.5 Models for hybrid systems

As models are the ultimate tools for obtaining and dealing with knowledge, not only in engineering, but also in philosophy, biology, sociology and economics, a search has been undertaken for appropriate mathematical models for hybrid systems. A whole range of possible model structures for hybrid systems has already been proposed. An overview of possible modeling techniques has been given in, e.g, [35, 80, 127]. Mentioned are, among others,

- Timed or hybrid Petri-nets, see, e.g., [75];
- Differential automata [202];
- Switched systems [147];
- Hybrid automata [41, 157];
- Jump-flow systems or hybrid inclusions [50, 101, 102];
- Brockett's model [45];
- Complementarity systems [117, 210, 211];
- Mixed logical dynamic models [27];
- Piecewise linear or piecewise affine models [198, 199];

- Duration calculus [65];
- Real-time temporal logics [5, 185];
- Switched bond graphs [196];
- Timed communicating sequential processes [76, 125].

We would like to emphasize that this list is by no means exhaustive. We are not going to treat all these classes here, but will focus on the most well-known classes. The common feature of all the modeling paradigms and in fact of hybrid systems is the interaction of different dynamics. This indicates that also the model structure should mix two modeling formalisms. Typically, one might think of the interaction of time-driven models (governed by differential or difference equations) on one hand, and event-driven systems (described by temporal logic, automata, finite state machines, etc. [10,13,123,169,206]) or logic rules[5] on the other. In some way these features should be combined in a unifying model structure. One of the nicest ways to look at hybrid systems is via hybrid automata, which can be seen as a cross production of finite state machines and differential or difference equations (depending if we use a discrete time or continuous time formalism).

1.5.1 Switched systems

Before going to a very general class of hybrid systems, we will first describe the hybrid system class consisting of *switched systems* that is close to the common models adopted in the control community. These switched systems are given by

$$\dot{x}(t) = f_{q(t)}(x(t)), \tag{1.2}$$

where $x(t) \in \mathbb{R}^n$ denotes the state at time $t \in \mathbb{R}^+$ and where $q : \mathbb{R}^+ \to \{1, \ldots, N\}$ is the switching signal that determines which vector field f_i, $i \in \{1, \ldots, N\}$ is active at time $t \in \mathbb{R}^+$. For a fixed value of $q(t)$, (1.2) describes a non-switched system, which is sometimes called a *subsystem* of the switched system. Hence, switching means to change the choice of the active subsystem.

The switching may depend on time only as above, but it can also be a function of the state $x(t)$ at time t, or of an external input, and even have memory in it.

In particular, when the switching only depends on the state variable $x(t)$ at the present time t, one speaks of *discontinuous dynamical systems* or *piecewise smooth systems*. An example is the system below that switches between two dynamics as a result of inequalities in the state variable :

$$\dot{x}(t) = f(x(t)) = \begin{cases} f_-(x(t)), & \text{if } \phi(x(t)) < 0 \\ f_+(x(t)), & \text{if } \phi(x(t)) > 0 \ . \end{cases} \tag{1.3}$$

The state space is separated into two parts by a hyper-surface defined by $\phi(x) = 0$ (see Fig. 1.5). On one side of the surface $\mathcal{C}_+ := \{x \in \mathbb{R}^n \mid \phi(x) > 0\}$ the dynamics $\dot{x} = f_+(x)$ holds, on the opposite side $\mathcal{C}_- := \{x \in \mathbb{R}^n \mid \phi(x) < 0\}$ the dynamics $\dot{x} = f_-(x)$ is valid. Hence, one can also consider this system as a *differential equation with a discontinuous right-hand side* [94].

As the choice of whether the vector field f_- or f_+ is active at time t only depends upon the state $x(t)$, no discrete state is necessary to describe these discontinuous dynamical systems. Clearly, the need

[5] Logic rules cause switches when a given rule becomes active and passive. This can also be considered as a (controller-induced) event.

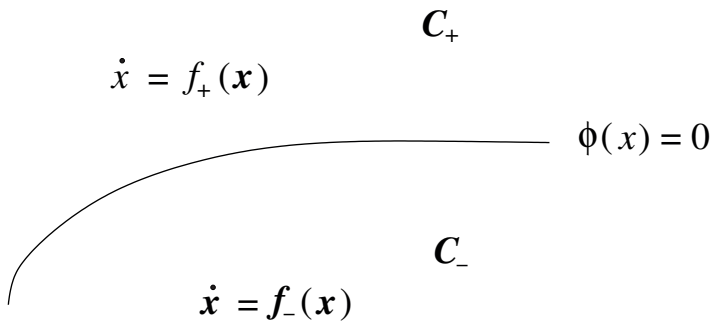

Figure 1.5: Discontinuous dynamical system with 2 subsystems.

for a discrete state in switched systems depends on the way the switching between the subsystems is orchestrated. For instance, if some memory is present in the switching (see the hysteresis example in Section 1.6.1 below) then one needs the explicit inclusion of a discrete state q in the model. Dynamical systems that switch between various subsystems and require next to a continuous state x also a discrete state q (and corresponding discrete dynamics as in the models for discrete-event systems of Section 1.4.2) can be described by *hybrid automata*, which are introduced next.

1.5.2 Hybrid automata

The hybrid automaton model as proposed in [3] may be described briefly as follows. The discrete part of the dynamics is modeled by means of a graph whose vertices are called *discrete states* or *modes*, and whose edges are *transitions*. The continuous state takes values in a vector space X (typically one can think of $\mathbb{R}^n$). For each mode there is a set of trajectories, which are called *activities* in [3], and which represent the continuous dynamics of the system. They are typically described by differential or difference equations. Interaction between the discrete dynamics and the continuous dynamics takes place through *invariants* and *transition relations*. Each mode has an invariant associated to it, which describes the conditions that the continuous state has to satisfy at this mode. Each transition has an associated transition relation, which describes the conditions on the continuous state under which that particular transition may take place and the effect that the transition will have on the continuous state. Invariants and transition relations play supplementary roles: whereas invariants describe when a transition *must* take place (namely when otherwise the motion of the continuous state as described in the set of activities would lead to violation of the conditions given by the invariant), the transition relations serve as "enabling conditions" that describe when a particular transition *may* take place[6].

Various ramifications of the hybrid automaton model have been proposed in the literature. Sometimes the notion of a transition relation is split up into two components, namely a *guard* which specifies the subset of the state space where a certain transition is enabled, and a *reset map* which is a (set-valued) function that specifies how new continuous states are related to previous continuous states for a particular transition. Often the hybrid automaton model is extended with a description format for continuous dynamics, typically systems of differential equations. Furthermore, versions of the hybrid automaton model which include external inputs have been discussed in, e.g., [6, 41, 58, 154, 157].

[6] In the model of [3], transitions are further equipped with *synchronization labels*, which express synchronization constraints between different automata. This construct allows the introduction of a notion of *parallel composition* between two automata. But we will not deal with this feature in these notes.

We opt here for the following description of (autonomous[7]) *hybrid automata*, which stems from [153].

Definition 1.5.1 A hybrid automaton H is an 8-tuple $H = (Q, X, f, \mathrm{Init}, \mathrm{Inv}, E, G, R)$ where

- $Q = \{q_1, \ldots, q_N\}$ is a finite set of *discrete states*;
- $X = \mathbb{R}^n$ is a set of *continuous states*;
- $f : Q \times X \to X$ is a *vector field*;
- $\mathrm{Init} \subseteq Q \times X$ is a set of *initial states*;
- $\mathrm{Inv} : Q \to P(X)$ describes the *invariants*;
- $E \subseteq Q \times Q$ is a set of *edges*;
- $G : E \to P(X)$ is a *guard condition*;
- $R : E \to P(X \times X)$ is a *reset map*.

Note that $P(X)$ is the power set of X, i.e., the collection of all subsets of X. Hence, note that the guard condition G gives for each mode transition or edge a subset of X. The *(hybrid) state* variable of the system H is given by $(q, x) \in Q \times X$.

Just to give you a bit of the flavor of how the evolution of this dynamical system behaves, we give a short, rather informal description. The initial hybrid state (q_0, x_0) of "trajectories" of a hybrid automaton lies in the initial set Init. From this hybrid state the continuous state x evolves according to the differential equation

$$\dot{x} = f(q_0, x) \quad \text{with } x(0) = x_0$$

and the discrete state q remains constant $q(t) = q_0$. The continuous evolution can go on as long as x stays in $\mathrm{Inv}(q_0)$. If at some point the continuous state x reaches the guard $G(q_0, q_1)$, we say that the transition is enabled. The discrete state may then change to q_1, and the continuous state jumps from the current value x^- to a new value x^+ with $(x^-, x^+) \in R(q_0, q_1)$. After this transition, the continuous evolution resumes and the whole process is repeated.

Note that in terms of Section 1.4 one can say the time-driven part of the evolution is given by differential equations and the event-driven part by transitions and reset maps. The interaction between these evolution mechanisms is given by invariants and guards.

This framework indicates the behavior of a hybrid system: continuous phases separated by events at which (maybe multiple) discrete actions (re-initialization of the continuous state x and discrete state q) take place. It is obvious that these systems switch between many operating modes where each mode is governed by its own characteristic dynamical laws (see Figure 1.6). Mode transitions are triggered by variables crossing specific thresholds (state events) and by the elapse of certain time periods (time events) (by modeling time as an additional state variable and including differential equations of the form $\dot{t} = 1$) due to the invariants and guards. With a change of mode, discontinuities in the time-continuous variables may occur as given by the reset map. Extensions of this formalism are possible, so as to include also external inputs (input events) as triggering a mode change (see, e.g., [41]).

[7] For automata with inputs and outputs we refer to [58, 157].

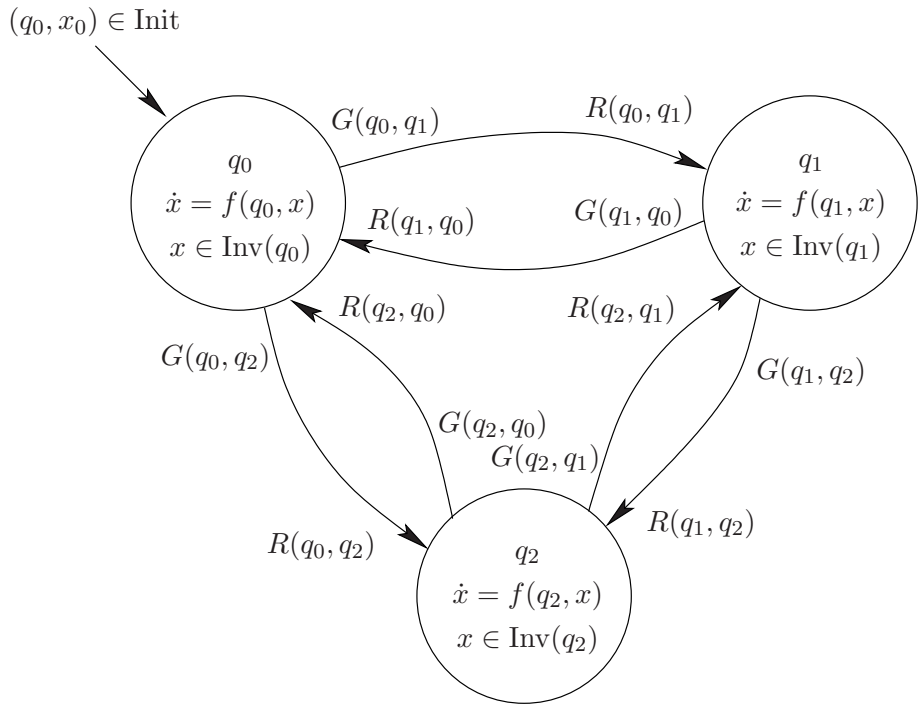

Figure 1.6: Schematic representation of a hybrid automaton with 3 discrete states. Each node of the directed graph represents a mode (operating point) given by a system of differential (or difference) equations. The arrows indicate the possible discrete transitions that correspond to a change of the mode.

We would like to stress that it can be a nontrivial task to rewrite a physical model description in terms of a hybrid automaton. Especially, the definition of the guards and the reset maps (switching and re-initialization rules) can be really difficult. Consider, e.g., the electrical network example of Section 1.6.6 below.

1.5.3 Modeling versus decisive power

The choice of a suitable framework is a trade-off between two conflicting criteria: the modeling power and the decisive power. The modeling power indicates the size of the class of systems allowing a reformulation in terms of the chosen model description. The decisive power is the ability to prove quantitative and qualitative properties of individual systems in the framework. A model structure that is too broad (like the hybrid automaton in the previous section) cannot reveal specific properties of a particular element in the model class. The size of a model class is often taken too large for analysis purposes. As indicated by [33], even for the easiest hybrid systems analysis and control problems are often undecidable[8], NP-complete or NP-hard[9], or require a high computational load.

[8] A problem is undecidable if there cannot exist generally applicable algorithms that solve the problem and for which finite termination can be guaranteed.

[9] A *decision problem* is a problem that has only two possible solutions: either the answer "yes" or the answer "no". A *search problem* is a problem for which we either have to give a solution or have to establish that the problem has no solution. A search problem (such as an optimization problem or a design problem) is called NP-hard if the corresponding decision problem (e.g., deciding whether or not there exists an optimal solution or an optimal design) is NP-complete. An NP-complete problem can only be solved in polynomial time if the class P would coincide with the class NP [98]. With the present state of knowledge it

As an example, for many classes of hybrid and timed automata the reachability problem (i.e., whether some of the trajectories of the system can attain a specific set of desired states) is *undecidable* [122].

Tsitsiklis and Blondel [33] consider the elementary hybrid system given by

$$x(k+1) = \begin{cases} A_1 x(k), & \text{when } c^{\mathrm{T}} x(k) \geqslant 0, \\ A_2 x(k), & \text{when } c^{\mathrm{T}} x(k) < 0, \end{cases} \tag{1.4}$$

where A_1, A_2 are matrices and c is a (column) vector of appropriate dimensions. To decide whether this switching system is stable is shown to be *NP-hard*. Loosely speaking, this means that there is no algorithm that answers the question of stability in polynomial time (as function of the size of A_1, A_2 and c).

The complexity of hybrid systems is also shown by a simple piecewise linear forced Van der Pol oscillator with an ideal diode studied in [128]. The system consists of a capacitor, an inductor, a linear negative resistor, a diode and a sinusoidal voltage source. For the analysis the diode is assumed to be an ideal switch. The system switches between the blocking and conducting mode and the dynamics in the individual modes are linear. For a specific region of the parameter values (which are analytically determined) this system displays chaotic behavior that has been experimentally and numerically verified in [128]. The occurrence of chaos in such a simple system is rather intriguing, but indicates that multi-modal systems are extremely complex.

As a conclusion, one can state that in certain cases it is nice to have some additional structure on the model class that you consider as the hybrid automaton model is too broad for detailed analysis. Before we go into a discussion of other hybrid model classes in Chapter 2, we will first present some examples in which different types of hybrid phenomena are illustrated.

1.6 Examples of hybrid systems

Some of the examples of this section are taken from Chapter 2 of [212].

1.6.1 Hysteresis

Consider a control system with a hysteresis element in the feedback loop (cf. [41]):

$$\dot{x} = H(x) + u \tag{1.5}$$

where the multi-valued function H is shown in Figure 1.7.

Note that this system is not just a differential equation whose right-hand side is piecewise continuous. There is "memory" in the system, which affects the right-hand side of the differential equation. Indeed, the hysteresis function H has an automaton naturally associated to it, and the system (1.5) can be formalized as the hybrid automaton depicted in Figure 1.8.

In Figure 1.8 the mode invariants are written inside the two circles representing the two modes, and the transitions (events) are labeled with their guards. Note that this is a hybrid system involving internally induced switchings, but no resets.

is still an open question whether the class P coincides with the class NP. However, since no NP-complete problem is known to be solvable in polynomial time despite the efforts of many excellent researchers, it is widely conjectured that no NP-complete problem can be solved by a polynomial time algorithm.

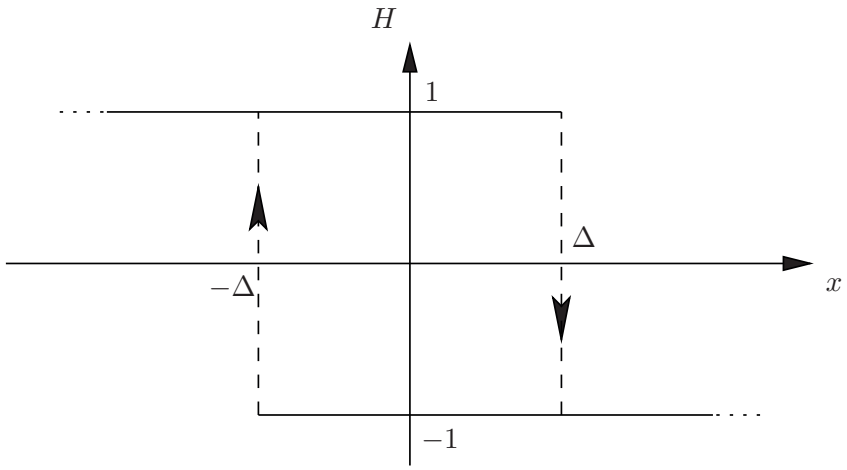

Figure 1.7: Hysteresis.

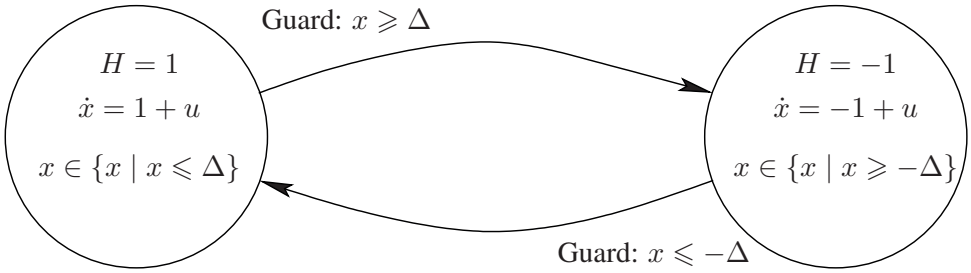

Figure 1.8: Control system with hysteresis as a hybrid automaton.

1.6.2 Manual transmission

Consider a simple model of a manual transmission [45]:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{-a\,x_2 + u}{1 + v} \end{aligned}$$

where v is the gear shift position $v \in \{1, 2, 3, 4\}$, u is the acceleration and a is a parameter of the system. Clearly, this is a hybrid system having four modes and a two-dimensional continuous state, with controlled transitions (switchings) and no resets.

1.6.3 Water-level monitor

Our next example (adapted from [3]) is somewhat more elaborate than the ones that we discussed before. The example concerns the modeling of a water-level control system. There are two continuous variables, denoted by $y(t)$ (the water level) and $x(t)$ (time elapsed since last signal was sent by the monitor). There are also two discrete variables, denoted by $P(t)$ (the status of the pump, taking values in $\{\texttt{on}, \texttt{off}\}$) and $S(t)$ (the nature of the signal last sent by the monitor, also taking values in $\{\texttt{on}, \texttt{off}\}$). The dynamics of the system is given in [3] as follows. The water level rises one unit per second when the pump is on

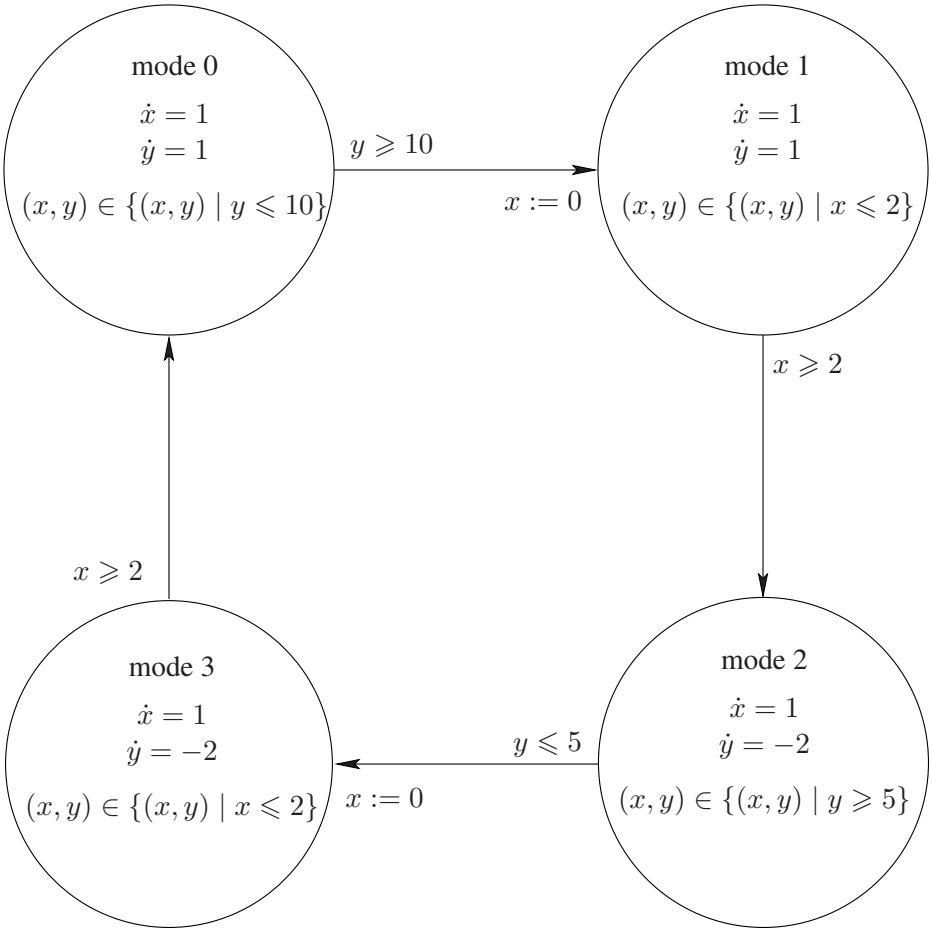

Figure 1.9: Water-level monitor.

and falls two units per second when the pump is off. When the water level rises to 10 units, the monitor sends a switch-off signal, which after a delay of two seconds results in the pump turning off. When the water level falls to 5 units, the monitor sends a switch-on signal, which after a delay of again two seconds causes the pump to switch on.

There are several ways in which one may write down equations to describe the system, which may be related to various ways in which the controller may be implemented. E.g., the monitor should send a switch-off signal when the water level reaches 10 units and is rising, but not when the level reaches 10 units on its way down. This may be implemented by the sign of the derivative of y, by looking at the status of the pump, or by looking at the signal last sent by the monitor. Under the assumptions of the model these methods are all equivalent in the sense that they produce the same behavior; however there can be differences in robustness with respect to unmodeled effects. The solution proposed in [3] is based on the signal last sent by the monitor. The hybrid automaton model of this system is given in Figure 1.9. The different modes correspond to the possible combinations of $P(t)$ and $S(t)$ as follows: mode 0 corresponds to the status of the pump being `on`, and value of the last monitor signal value being `on`; mode 1 corresponds to the `on`, `off` status/value; mode 2 to `off`, `off`; and mode 3 to `off`, `on`.

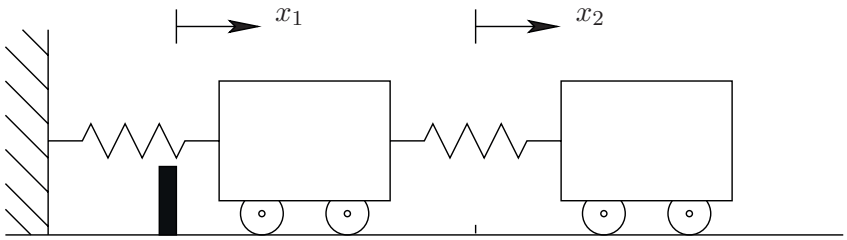

Figure 1.10: Two-carts system.

1.6.4 Two-carts system

As an introduction to the class of linear complementarity systems (see Section 2.3), we now already illustrate some of the aspects that play a role in the evolution of such systems with an example of two carts connected by a spring (used also in [117, 210]). The left cart is attached to a wall by a spring. The motion of the left cart is constrained by a completely inelastic stop. The system is depicted in Figure 1.10.

For simplicity, the masses of the carts and the spring constants are set equal to 1. The stop is placed at the equilibrium position of the left cart. By x_1, x_2 we denote the deviations of the left and right cart, respectively, from their equilibrium positions and x_3, x_4 are the velocities of the left and right cart, respectively. By z, we denote the reaction force exerted by the stop. Furthermore, the variable w is set equal to x_1. Simple mechanical laws lead to the dynamical relations

$$\begin{aligned} \dot{x}_1(t) &= x_3(t), \\ \dot{x}_2(t) &= x_4(t), \\ \dot{x}_3(t) &= -2x_1(t) + x_2(t) + z(t), \\ \dot{x}_4(t) &= x_1(t) - x_2(t), \\ w(t) &:= x_1(t). \end{aligned} \tag{1.6}$$

To model the stop in this setting, the following reasoning applies. The variable $w(t) = x_1(t)$ should be nonnegative because it is the position of the left cart with respect to the stop. The force exerted by the stop can act only in the positive direction implying that $z(t)$ should be nonnegative. If the left cart is not at the stop at time t ($w(t) > 0$), the reaction force vanishes at time t, i.e., $z(t) = 0$. Similarly, if $z(t) > 0$, the cart must necessarily be at the stop, i.e., $w(t) = 0$. This is expressed by the conditions

$$0 \leqslant w(t) \perp z(t) \geqslant 0 \ , \tag{1.7}$$

where $\perp$ denotes the orthogonality of vectors (i.e., $w(t) \perp z(t)$ means that $w^{\mathrm{T}}(t) \cdot z(t) = 0$). The system can be represented by two modes, depending on whether or not the stop is active. We distinguish between the unconstrained mode ($z(t) = 0$) and the constrained mode ($w(t) = 0$). The dynamics of these modes are given by the following differential and algebraic equations:

unconstrained	constrained
$\dot{x}_1(t) = x_3(t)$	$\dot{x}_1(t) = x_3(t)$
$\dot{x}_2(t) = x_4(t)$	$\dot{x}_2(t) = x_4(t)$
$\dot{x}_3(t) = -2x_1(t) + x_2(t)$	$\dot{x}_3(t) = -2x_1(t) + x_2(t) + z(t)$
$\dot{x}_4(t) = x_1(t) - x_2(t)$	$\dot{x}_4(t) = x_1(t) - x_2(t)$
$z(t) = 0$	$w(t) = x_1(t) = 0.$

Remark 1.6.1 Note that the unconstrained mode directly is an ordinary differential equation (ODE) in the state variable. However, the constrained mode is a differential algebraic equation (DAE) (of high index) as the variable z is not explicit. Since $w \equiv 0$ in this mode, that also means that $\dot{w} \equiv 0$, which yields $x_3 \equiv 0$. Similarly, we must have $\ddot{w} \equiv 0$, which leads to $-2x_1 + x_2 + z = 0$. Hence, we have $z = -x_2$. By substituting this in the DAE of the constrained mode, we obtain $x_1 = x_3 = 0$ and $\ddot{x}_2 = -x_2$ (note that the constrained mode has a state variable $(x_2, x_4)^{\mathrm{T}}$ of dimension 2). Hence, by differentiating the variable w we can rewrite the DAE as an ODE.

When the system is represented by either of these modes, the triple (x, w, z) is given by the corresponding dynamics as long as the following inequalities are satisfied (cf. (1.7)):

unconstrained	constrained
$z(t) = 0, \;\; w(t) > 0$	$w(t) = 0, \;\; z(t) > 0$.

A mode change is triggered by violation of one of these (in)equalities. The mode transitions that are possible for the two-carts systems are described below.

- **Unconstrained → constrained.** The inequality $w(t) > 0$ tends to be violated at a time instant $t = \tau$. The left cart hits the stop and stays there. The velocity of the left cart is reduced to zero instantaneously at the time of impact: the kinetic energy of the left cart is totally absorbed by the stop due to a purely inelastic collision. A state for which this happens is, e.g., $x(\tau^-) = (0^+, -1, -1, 0)^{\mathrm{T}}$.

- **Constrained → unconstrained.** The inequality $z(t) > 0$ tends to be violated at $t = \tau$. The right cart is located at or moving to the right of its equilibrium position, so the spring between the carts is stretched and pulls the left cart away from the stop. This happens, for example, if $x(\tau) = (0, 0, 0, 1)^{\mathrm{T}}$.

- **Unconstrained → unconstrained with re-initialization according to constrained mode.** The inequality $w(t) > 0$ tends to be violated at $t = \tau$. As an example, consider $x(\tau^-) = (0^+, 1, -1, 0)^{\mathrm{T}}$. At the time of impact, the velocity of the left cart is reduced to zero just as in the first case. Hence, a state reset (re-initialization) to $(0, 1, 0, 0)^{\mathrm{T}}$ occurs. The right cart is at the right of its equilibrium position and pulls the left cart away from the stop. Stated differently, from $(0, 1, 0, 0)^{\mathrm{T}}$ smooth continuation in the unconstrained mode is possible.

This last transition is a special one in the sense that, first, the constrained mode is active, causing the corresponding state reset. After the reset, no smooth continuation is possible in the constrained mode resulting in a second mode change back to the unconstrained mode.

From state $x(\tau^-) = (0^+, -1, -1, 0)^{\mathrm{T}}$, we can enter the constrained mode by starting with an instantaneous reset to $x(\tau^+) = (0, -1, 0, 0)^{\mathrm{T}}$. This reset can be modeled as the result of a (Dirac) pulse δ exerted by the stop. In fact, $z = \delta$ results in the state jump $x(\tau^+) - x(\tau^-) = (0, 0, 1, 0)^{\mathrm{T}}$. This motivates the use of distributional theory as a suitable mathematical framework for describing physical phenomena such as collisions with discontinuities in the state vector (see Section 3.5).

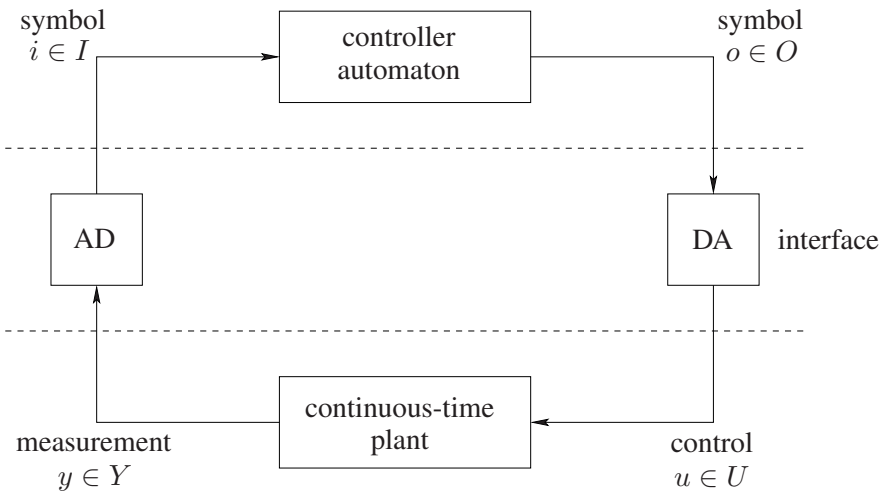

Figure 1.11: Supervisor model.

1.6.5 Supervisor model

The following model has been proposed for controlling a continuous-time input-state-output system

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned}$$

by means of an (input-output) finite automaton (see [14, 41, 171] for further discussion). The model consists of three basic parts: continuous-time plant, finite control automaton, and interface. The interface in turn consists of two parts, viz. an analog-to-digital (AD) converter and a digital-to-analog (DA) converter. The supervisor model is illustrated in Figure 1.11.

Associated to the plant are an input space U, a state space X, and an output space Y, while the controller automaton has a (finite) input space I, a state space Q and an output space O. The controller automaton may be given as an input-output automaton as follows:

$$\begin{aligned} q^{\#} &= \nu(q,i) \\ o &= \eta(q,i) \ , \end{aligned}$$

where $^{\#}$ denotes the next state operator, i.e., the new state after an event (characterized by the input i) has occurred. The AD converter is given by a map $f_{\mathrm{AD}} : Y \times Q \to I$. The values of this map (the discrete input symbols) are determined by a partition of the output space Y, which may depend on the current value of the state of the finite automaton. The DA converter is given by a map $f_{\mathrm{DA}} : O \to \mathcal{P}_U$, where $\mathcal{P}_U$ denotes the set of piecewise right-continuous input functions for the plant.

The dynamics can be described as follows. Assume that the state of the plant is evolving and that the controller automaton is in state q. Then $f_{\mathrm{AD}}(\cdot, q)$ assigns to output $y(t)$ a symbol from the input alphabet I. When this symbol changes, the controller automaton carries out the associated state transition, causing a corresponding change in the output symbol $o \in O$. Associated to this symbol is a control input $f_{\mathrm{DA}}(o)$ that is applied as input to the plant until the input symbol of the controller automaton changes again.

In the literature the design of the supervisor is often based on a *quantization* (also sometimes called *abstraction*) of the continuous plant to a discrete-event system. In this case one considers, e.g., an appropriate (fixed) partition of the state space and the output space of the continuous time plant, together with

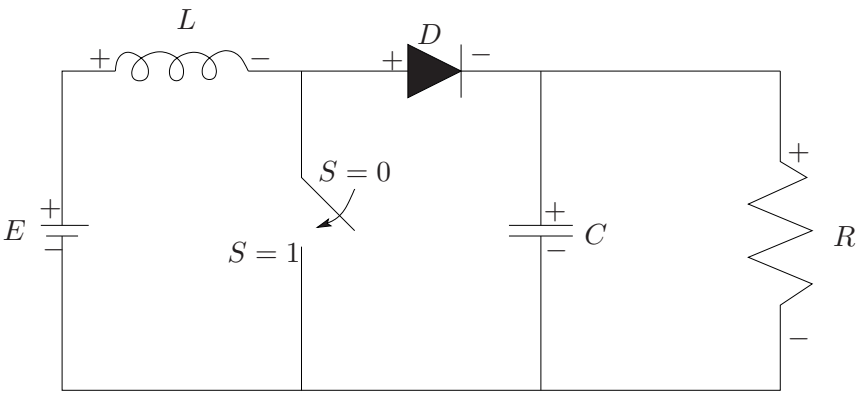

Figure 1.12: Boost converter with clamping diode.

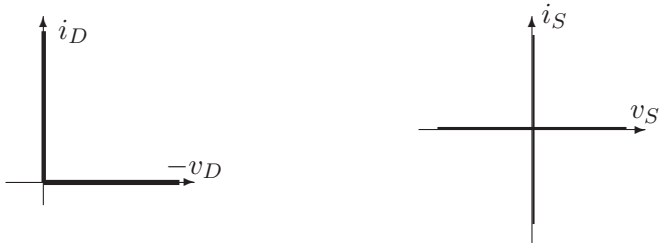

Figure 1.13: Voltage-current characteristic of an ideal diode and ideal switch.

a fixed set of input functions, and one constructs a (not necessarily deterministic) discrete-event system covering the quantized continuous-time dynamics. The events are then determined by the crossing of the boundaries defined by the partition of the state space.

1.6.6 Boost converter

The boost converter (see, e.g., [88]) consists of a capacitor C with electric charge q, an ideal diode D, a battery (voltage source) E, an inductor L with magnetic flux ϕ, a resistor R, and an ideal switch S. The voltage-current characteristics of the ideal diode and switch are depicted in Figure 1.13. These type of circuits, usually called *step-up* converters, are used to obtain a voltage at the resistance load that is higher than the voltage E of the input source.

The presence of switching elements (D and S) introduces hybrid dynamics. Depending on the (discrete) state of the diode and the switch, one can distinguish four modes: $\{(v_S = 0, v_D = 0), (v_S = 0, i_D = 0), (i_S = 0, v_D = 0), (i_S = 0, i_D = 0)\}$. A hybrid automaton representation can be obtained by analyzing these four modes. E.g., mode 1 in which the switch S is open ($i_S = 0$ or $S = 0$ – note that $S = 1$ complies with $v_S = 0$) and the diode is conducting (see Figure 1.14) is governed by the following network equations $\dot{q} = -\frac{1}{RC}q + \frac{1}{L}\phi$ and $\dot{\phi} = -\frac{1}{C}q + E$. This mode is active as long as $i_D \geqslant 0$, i.e., $\phi \geqslant 0$ since $i_D = i_L = \frac{1}{L}\phi$. Stated differently, $\phi \geqslant 0$ determines the invariant set of this mode. A mode transition occurs if either the current through the diode tends to be negative or the switch is closed. The former is an autonomous event whereas the latter is a controlled event. A similar (tedious) analysis of each mode yields a hybrid automaton model as depicted in Figure 1.15. The mode transitions and the corresponding guards and reset maps are summarized in the Table 1.1.

It is obvious that the hybrid automaton model is very involved (even its derivation) and does not

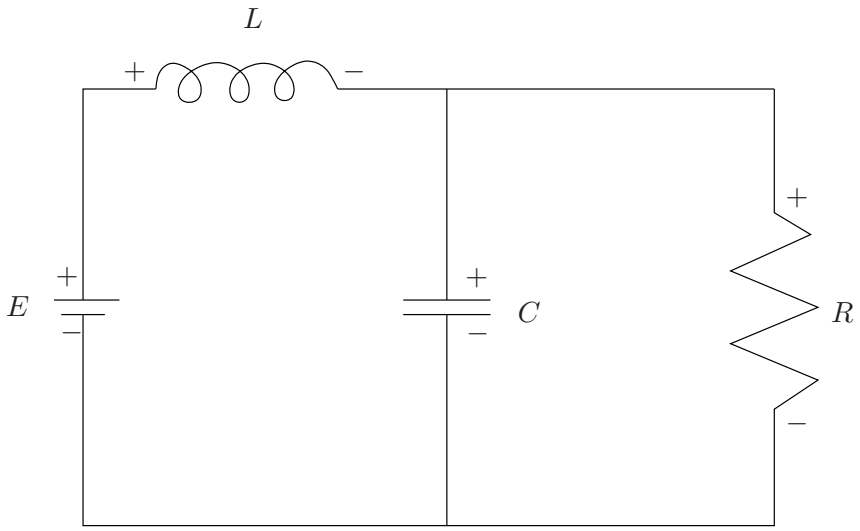

Figure 1.14: Mode 1 complying with $i_S = 0$ and $v_D = 0$.

transition	guard	reset
mode 1→mode 2	$S = 1$ and $q \geqslant 0$	
mode 1→mode 3	$\phi = 0$ and $q > CE$	
mode 1- - →mode 3	$\phi < 0$	$\phi^+ = 0$
mode 1→mode 4	$S = 1$ and $q \leqslant 0$	$q^+ = 0$
mode 2→mode 1	$S = 0$ and $\phi \geqslant 0$	
mode 2→mode 3	$S = 0$ and $\phi \leqslant 0$	$\phi^+ = 0$
mode 2→mode 4	$q = 0$	
mode 2- - →mode 4	$q < 0$	$q^+ = 0$
mode 3→mode 1	$q = CE$	
mode 3→mode 2	$S = 1$ and $q \geqslant 0$	
mode 3→mode 4	$S = 1$ and $q \leqslant 0$	$q^+ = 0$
mode 4→mode 1	$S = 0$ and $\phi \geqslant 0$	
mode 4→mode 3	$S = 0$ and $\phi \leqslant 0$	$\phi^+ = 0$
mode 4→mode 4	$q < 0$	$q^+ = 0$

Table 1.1: The mode transitions and the corresponding guards and reset maps for the boost converter.

maintain the structure of the system at hand. Therefore, alternatively, one may use the more compact model:

$$\begin{aligned}
\dot{q} &= -\tfrac{1}{RC}q + i_D && \text{(1.8a)}\\
\dot{\phi} &= v_S + E && \text{(1.8b)}\\
-v_D &= \tfrac{1}{C}q + v_S && \text{(1.8c)}\\
i_S &= \tfrac{1}{L}\phi - i_D && \text{(1.8d)}\\
0 \leqslant i_D &\perp -v_D \geqslant 0 && \text{(1.8e)}\\
v_S &\perp i_S. && \text{(1.8f)}
\end{aligned}$$

Note that just as in the two-carts system of Section 1.6.4 a complementarity relation (1.8e) shows up

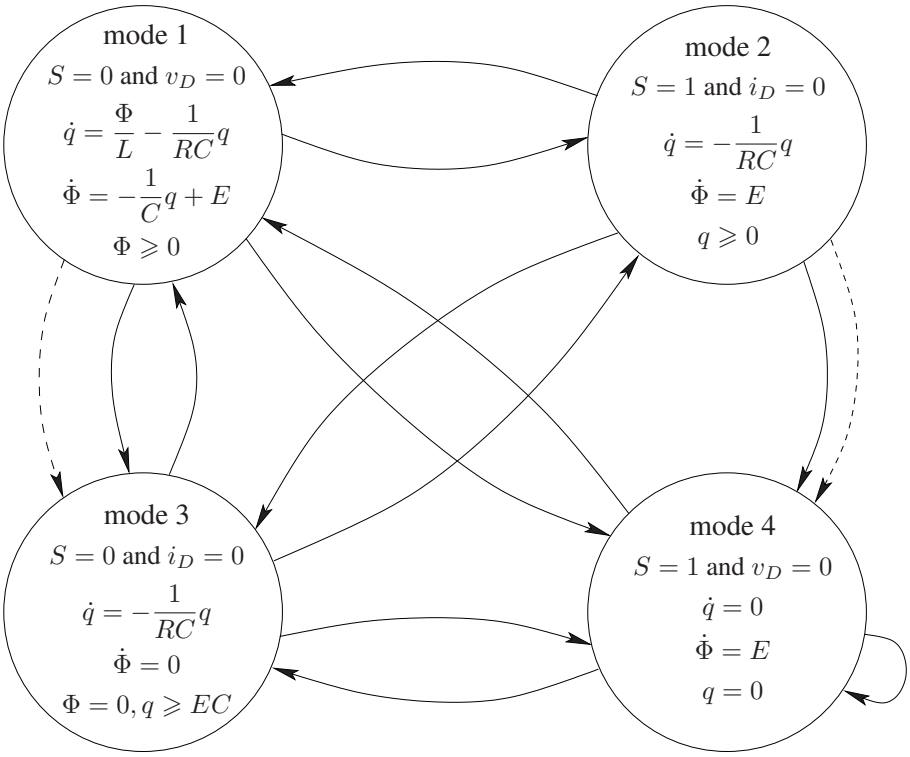

Figure 1.15: A hybrid automaton model of the boost converter.

between the diode current i_D and the diode voltage v_D.

1.7 Examples with Zeno behavior

1.7.1 Bouncing ball

A well-known example is a model of a bouncing ball (height of ball is x) with dynamics $\ddot{x} = -g$ and constraint $x \geqslant 0$. To complete the model we include Newton's restitution rule $\dot{x}(\tau+) = -e\dot{x}(\tau-)$ when $x(\tau-) = 0$ and $\dot{x}(\tau-) < 0$ ($0 < e < 1$). In case $x(\tau-) = \dot{x}(\tau-) = 0$, the dynamics are equal to $\ddot{x} = 0$ due to the constraint $x \geqslant 0$. The event times $\{\tau_i\}_{i\in\mathbb{N}}$ are related through (see [46, p.234])

$$\tau_{i+1} = \tau_i + \frac{2e^i\dot{x}(0)}{g}, \quad \text{for } i \in \mathbb{N}$$

assuming that $x(0) = 0$ and $\dot{x}(0) > 0$. Hence, $\{\tau_i\}_{i\in\mathbb{N}}$ has a finite limit equal to $\tau^* = \frac{2\dot{x}(0)}{g-ge} < \infty$. Since the continuous state $(x(t), \dot{x}(t))$ converges to $(0, 0)$ when $t \uparrow \tau^*$ a continuation beyond τ^* can be defined by $(x(t), \dot{x}(t)) = (0, 0)$ for $t > \tau^*$. The physical interpretation is that the ball is at rest within a finite time span, but after infinitely many bounces. This is a specific example of *Zeno behavior*, i.e., a infinite number of events in a finite length time interval. In this case we have an infinite number of state re-initializations and the set of event times $\mathcal{E}$ for the bouncing ball contains a so-called *right-accumulation point*.

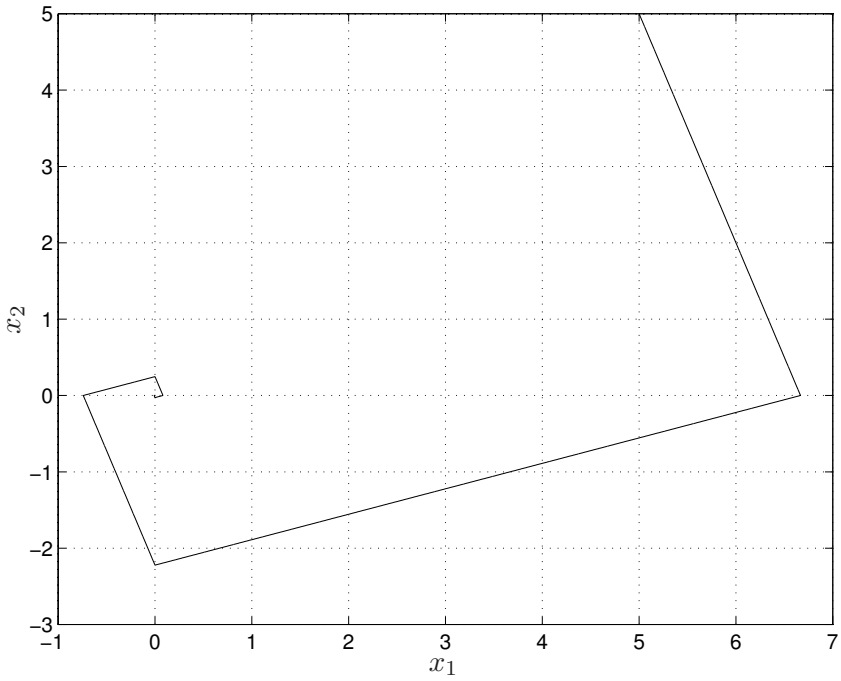

Figure 1.16: Trajectory in the phase plane with initial state $(5, 5)^{\mathrm{T}}$.

1.7.2 Time-reversed Filippov's example

Consider a time-reversed version of a system studied by Filippov [94, p. 116] (mentioned also in [152]), i.e.,

$$\dot{x}_1 = -\text{sgn}(x_1) + 2\text{sgn}(x_2) \tag{1.9a}$$
$$\dot{x}_2 = -2\text{sgn}(x_1) - \text{sgn}(x_2), \tag{1.9b}$$

where "sgn" denotes the sign function given by

$$\text{sgn}(x) := \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{if } x < 0, \\ [-1, 1] & \text{if } x = 0, \end{cases} \tag{1.10}$$

which is strictly speaking a set-valued function. We get back to this issue of differential equations that can have multiple values in their right-hand side (called *differential inclusions*) in Chapter 3. Solutions of this piecewise constant system are spiraling towards the origin, which is an equilibrium. Since $\frac{\mathrm{d}}{\mathrm{d}t}(|x_1(t)| + |x_2(t)|) = -2$, when $x(t) \neq 0$, solutions reach the origin in finite time. See Figure 1.16 for a trajectory. However, solutions cannot arrive at the origin without going through an infinite number of mode transitions (relay switches). Since these mode switches occur in a finite time interval, the event times contain a right-accumulation point (i.e., the time that the solution reaches the origin) after which the solution stays at zero.

Hence, again a type of Zeno behavior shows up (an infinite number of mode switches in a finite length time interval).

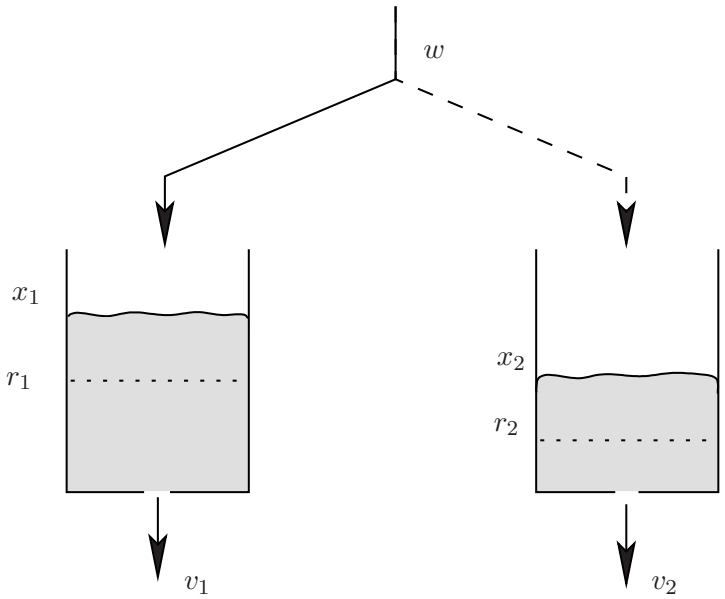

Figure 1.17: The two-tank system.

1.7.3 Two-tank system

Consider the two-tank system shown in Figure 1.17, which consists of two tanks containing water. Both tanks are leaking at a constant rate. Water is added to the system at a constant rate through a hose, which at any point in time is dedicated to either one tank or the other. It is assumed that the hose can switch between the tanks instantaneously. Let x_i denote the volume of water in tank i for $i = 1, 2$, and let $v_i > 0$ denote the constant flow of water out of tank i. Let w denote the constant flow of water into the system. The objective is to keep the water volumes above r_1 and r_2, respectively, assuming that the water volumes are above r_1 and r_2 initially. This is to be achieved by a controller that switches the inflow to tank 1 whenever $x_1 \leqslant r_1$ and to tank 2 whenever $x_2 \leqslant r_2$.

Let us now define the hybrid automaton that describes this process. We have two discrete states or modes depending on whether the hose is filling tank 1 (mode q_1) or tank 2 (mode q_2). The continuous state x has two components x_1 and x_2, and their evolution is given by

$$\begin{cases} \dot{x}_1 = w - v_1 \\ \dot{x}_2 = -v_2 \end{cases} \text{ in mode } q_1, \qquad \begin{cases} \dot{x}_1 = -v_1 \\ \dot{x}_2 = w - v_2 \end{cases} \text{ in mode } q_2.$$

Furthermore, as we assume that initially the water level in each of the tanks is above the low level mark, we have

$$\text{Init} = \{q_1, q_2\} \times \{(x_1, x_2) \mid x_1 \geqslant r_1 \text{ and } x_2 \geqslant r_2\} \ .$$

As we put water in the current tank as long as the level in the other tank is above the low level mark, we have the following invariant

$$\begin{aligned} \text{Inv}(q_1) &= \{x \in \mathbb{R}^2 \mid x_2 \geqslant r_2\} \\ \text{Inv}(q_2) &= \{x \in \mathbb{R}^2 \mid x_1 \geqslant r_1\} \ . \end{aligned}$$

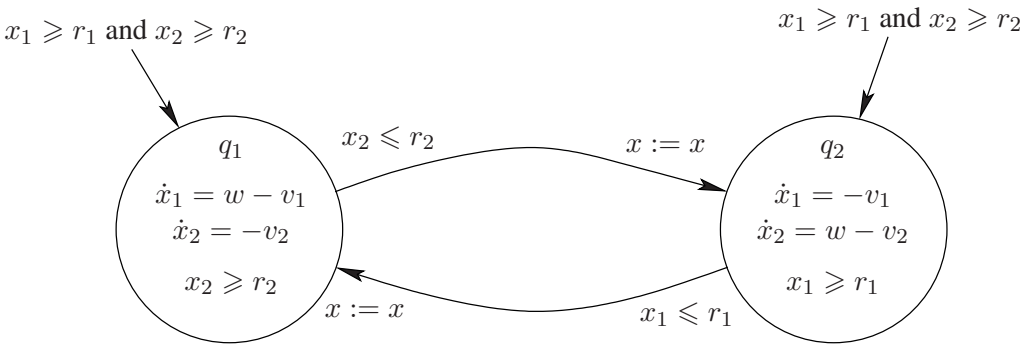

Figure 1.18: The hybrid automaton for the two-tank system.

The guards are given by

$$G(q_1, q_2) = \{x \in \mathbb{R}^2 \mid x_2 \leqslant r_2\}$$
$$G(q_2, q_1) = \{x \in \mathbb{R}^2 \mid x_1 \leqslant r_1\} \ ,$$

since we switch the inflow to the other tank as soon as the water there reaches the low level mark. Since the continuous state does not change as a result of switching the inflow, we have

$$R(q_1, q_2) = R(q_2, q_1) = \{(x^-, x^+) \mid x^-, x^+ \in \mathbb{R}^2 \text{ and } x^- = x^+\} \ .$$

The directed graph corresponding to this hybrid automaton is shown in Figure 1.18.

A motivation for looking at Zeno solutions may be derived from this water tank example as follows [5]. Recall that v_1, v_2 and w are constants. Assume that the total outflow $v_1 + v_2$ of the system is larger than the inflow w. In that case we cannot satisfy our objective of keeping the levels in both tanks above the low level mark. Indeed, at some time the water level in each of the tanks will then reach the low level mark, resulting in the hose continuously being switched from one tank to another. So we get a finite accumulation point of the switching times. Nevertheless, one can show that the goal is achieved along all non-Zeno trajectories as they are only defined on finite time intervals. We will briefly come back to this example later on (see Chapter 3).

1.7.4 Three balls example

Consider a system consisting of three balls in which perfectly inelastic impacts[10] are modeled by successions of simple impacts (Figure 1.19). Suppose the balls all have unit mass and are touching at time 0. The initial velocity $v_1(0)$ of ball 1 is equal to 1 and for balls 2 and 3 $v_2(0) = v_3(0) = 0$. By modeling all impacts separately, first an inelastic collision occurs between ball 1 and 2 resulting in $v_1(0+) = v_2(0+) = \frac{1}{2}$, $v_3(0+) = 0$. Next, ball 2 hits ball 3 resulting in $v_1(0++) = \frac{1}{2}$, $v_2(0++) = v_3(0++) = \frac{1}{4}$ after which ball 1 hits ball 2 again. In this way, a sequence of resets is generated

$$\begin{array}{lccccccc} v_1: & 1 & \frac{1}{2} & \frac{1}{2} & \frac{3}{8} & \frac{3}{8} & \frac{11}{32} & \cdots \\ v_2: & 0 & \frac{1}{2} & \frac{1}{4} & \frac{3}{8} & \frac{5}{16} & \frac{11}{32} & \cdots \\ v_3: & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{5}{16} & \frac{5}{16} & \cdots \end{array}$$

[10] An inelastic collision of two point masses is characterized by the conservation of momentum, but *not* of kinetic energy. This implies that the speeds of the masses after the collision is determined either by specifying one of the two speeds, or by assuming the masses stick together (i.e., a perfectly inelastic collision). On the other hand, for an elastic collision we have both conservation of momentum and conservation of kinetic energy.

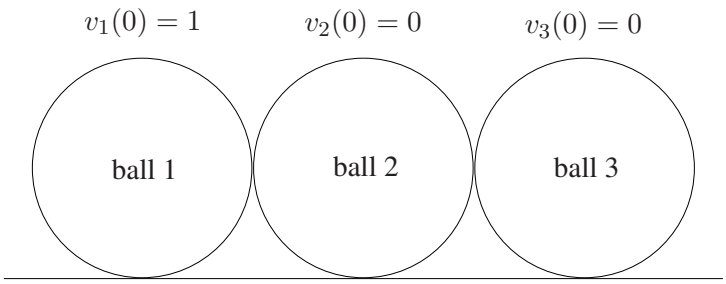

Figure 1.19: Three balls example.

which converges to $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^{\mathrm{T}}$ after which a smooth continuation is possible with constant and equal velocity for all balls.

In this example we even have an infinite number of events (resets) at one time instant, which is sometimes called *live-lock*, which is another special case of Zeno behavior.

1.8 Summary

This chapter has motivated the need for understanding the behavior of systems with a combination of event-driven and time-driven dynamics called hybrid systems. Typically, high-tech systems that involve both physical processes and digital embedded controllers constitute a broad application area of hybrid systems. However, also many physical systems exhibit hybrid behavior: switched electrical circuits or mechanical systems with friction and impacts.

We have introduced hybrid automata as one of the most general formalisms to model hybrid systems and we have also discussed the evolution of this type of system. This gave a brief glance of what we mean by hybrid dynamics. The exact notion of a solution trajectory will be the topic of interest in Chapter 3. Of course, the more general a model is (the larger its modeling power) the harder it is to analyze specific instances in the model class (low decisive power). Hence, the hybrid automaton model has a very large modeling power, but a low decisive power. Considering the complexity of hybrid systems, it is of interest to have some structure in the modeling formalism that can be exploited for analysis and synthesis purposes. Chapter 2 will propose several other modeling formalisms that might be used with various levels of modeling or decisive power.

This chapter was complemented with various examples to illustrate the presence of hybrid systems in different domains. Also several strange phenomena like Zenoness were observed in these systems, which might complicate the analysis and synthesis of controllers for these systems. Besides formalizing the concept of a solution trajectory in Chapter 3, we will study well-posedness: the property of existence and uniqueness of solution trajectories given an initial condition. In Chapter 4 we will study the stability for hybrid systems, which is — among other approaches — used in Chapter 5 to synthesize stabilizing hybrid controllers. The synthesis of hybrid controllers via optimization-based approaches is discussed in Chapter 6. Finally, Chapter 7 will give a brief introduction to a computer science approach to analysis of discrete-event and hybrid systems.

Chapter 2

Modeling frameworks

In Chapter 1 we have already discussed some general models for hybrid systems such as hybrid automata. Although hybrid automata can be considered as one of the most descriptive and general models for hybrid systems, analysis and control design based on these models often result in computationally hard problems (cf. Section 1.5.3). Therefore, we will discuss some special classes of hybrid systems for which tractable analysis and control design techniques are available as will be demonstrated in the next chapters. In particular, we discuss the following models:

- mixed logical dynamical (MLD) systems [25, 27],
- piecewise-affine (PWA) systems [198],
- linear complementarity (LC) systems [117, 211],
- extended linear complementarity (ELC) systems,
- max-min-plus-scaling (MMPS) systems [78],
- hybrid inclusions [50, 101, 102],
- timed automata,
- timed Petri nets.

Furthermore, we will also show that some of these classes (in particular MLD, PWA, LC, ELC and MMPS systems) are equivalent, possibly under mild additional assumptions related to well-posedness and boundedness of input, state, output or auxiliary variables. Each subclass has its own advantages over the others. E.g., stability criteria were proposed for PWA systems [131], control and verification techniques for MLD hybrid models [25, 27, 30] and for MMPS systems [78, 81, 84], and conditions of existence and uniqueness of solution trajectories (well-posedness) for LC systems [117, 211]. So it really depends on the application, which of these classes is best suited.

Also note that in practice the most widely used technique for hybrid systems is computer simulation, via a combination of discrete-event simulation and DAE[1] solvers. Some computer simulation and verification tools that are (also) used for hybrid systems are BaSiP, Modelica, HyTech, KRONOS, Chi,

[1] DAE: Differential Algebraic Equation.

20-sim, and UPPAAL. Simulation models can represent the plant with a high degree of detail, providing a close correspondence between simulated behavior and real plant behavior. This approach is, for any large system, computationally very demanding, and moreover it is difficult to understand from a simulation how the behavior depends on model parameters. This difficulty is even more pronounced in the case of large hybrid systems which consist of many interacting modules. Fast simulation techniques based on variance reduction, and perturbation analysis techniques [57] have been developed in order to partially overcome these limitations.

In the next sections we will discuss some tractable classes of hybrid systems, for which efficient analysis and control design methods are available as we will see in the next chapters.

2.1 Piecewise affine (PWA) systems

Piecewise affine (PWA) systems [198] are described by

$$\begin{array}{rcll} x(k+1) & = & A_i x(k) + B_i u(k) + f_i & \text{for} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \Omega_i, \\ y(k) & = & C_i x(k) + D_i u(k) + g_i & \end{array} \tag{2.1}$$

for $i = 1, \ldots, N$ where $\Omega_1, \ldots, \Omega_N$ are convex polyhedra (i.e., given by a finite number of linear inequalities) in the input/state space with non-overlapping interiors. The variables $u(k) \in \mathbb{R}^m$, $x(k) \in \mathbb{R}^n$ and $y(k) \in \mathbb{R}^l$ denote the input, state and output, respectively, at time k. PWA systems have been studied by several authors (see [25, 70, 131, 136, 145, 198, 207, 213] and the references therein) as they form the "simplest" extension of linear systems that can still model non-linear and non-smooth processes with arbitrary accuracy and are capable of handling hybrid phenomena.

Many authors also study the *continuous-time* variant of the above model, which is given by

$$\begin{array}{rcll} \dot{x}(t) & = & A_i x(t) + B_i u(t) + f_i & \text{for} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \Omega_i, \\ y(t) & = & C_i x(t) + D_i u(t) + g_i & \end{array} \tag{2.2}$$

for $i = 1, \ldots, N$ where $\Omega_1, \ldots, \Omega_N$ are convex polyhedra in the combined state-input space $\mathbb{R}^n \times \mathbb{R}^m$ and time t lies now on the real line $\mathbb{R}$. If the switching is only dependent on the state we obtain a description

$$\begin{array}{rcll} \dot{x}(t) & = & A_i x(t) + B_i u(t) + f_i & \text{for } x(t) \in \Omega_i, \\ y(t) & = & C_i x(t) + D_i u(t) + g_i & \end{array} \tag{2.3}$$

where the regions Ω_i are now convex polyhedra in the state space $\mathbb{R}^n$. Now (2.3) form special cases of the discontinuous dynamical systems as introduced in Section 1.5.1 if one adds control inputs u to the latter. In this chapter we will focus mostly on the discrete-time version as we can establish some relations to other well-known hybrid model classes, such as mixed-logical dynamical (MLD) models and linear complementarity (LC) models. In the next chapters the continuous-time PWA models will play a more prominent role.

As a very simple example of a discrete-time PWA model we can consider an integrator with upper saturation:

$$\begin{aligned} x(k+1) &= \begin{cases} x(k) + u(k) & \text{if } x(k) + u(k) \leqslant 1 \\ 1 & \text{if } x(k) + u(k) \geqslant 1\,. \end{cases} \\ y(k) &= x(k) \end{aligned} \tag{2.4}$$

X_1	X_2	$X_1 \wedge X_2$	$X_1 \vee X_2$	$\sim X_1$	$X_1 \Rightarrow X_2$	$X_1 \Leftrightarrow X_2$	$X_1 \oplus X_2$
T	T	T	T	F	T	T	F
T	F	F	T	F	F	F	T
F	T	F	T	T	T	F	T
F	F	F	F	T	T	T	F

Table 2.1: Truth table.

If we rewrite (2.4) as in (2.1) then we have

$$\begin{aligned}
&\Omega_1 = \left\{(x(k), u(k)) \in \mathbb{R}^2 \;\middle|\; x(k) + u(k) \leqslant 1\right\} \\
&\Omega_2 = \left\{(x(k), u(k)) \in \mathbb{R}^2 \;\middle|\; x(k) + u(k) \geqslant 1\right\} \\
&A_1 = 1, \quad A_2 = 0, \quad B_1 = 1, \quad B_2 = 0 \\
&f_1 = 0, \quad f_2 = 1, \quad C_1 = C_2 = 1 \\
&D_1 = D_2 = 0, \quad g_1 = g_2 = 0 \ .
\end{aligned}$$

2.2 Mixed Logical Dynamical (MLD) systems

2.2.1 Preliminaries

First, we will provide some tools to transform logical statements involving continuous variables into mixed-integer linear inequalities. This section is based on [27].

We will use capital letters X_i to represent statements, e.g., "$x \leqslant 0$" or "Temperature is hot". X_i is commonly referred to as a literal, and has a truth value of either "T" (true) or "F" (false). Boolean algebra enables statements to be combined in compound statements by means of connectives: "$\wedge$" (and), "$\vee$" (or), "$\sim$" (not), "$\Rightarrow$" (implies), "$\Leftrightarrow$" (if and only if), "$\oplus$" (exclusive or). These connectives are defined by means of the truth table given in Table 2.1. The connectives satisfy several properties (see, e.g., [69]), which can be used to transform compound statements into equivalent statements involving different connectives, and simplify complex statements. The following properties will be used in the sequel:

$$X_1 \Rightarrow X_2 \quad \text{is the same as} \quad \sim X_1 \vee X_2 \tag{2.5}$$
$$X_1 \Rightarrow X_2 \quad \text{is the same as} \quad \sim X_2 \Rightarrow \sim X_1 \tag{2.6}$$
$$X_1 \Leftrightarrow X_2 \quad \text{is the same as} \quad (X_1 \Rightarrow X_2) \wedge (X_2 \Rightarrow X_1) \ . \tag{2.7}$$

One can associate with a literal X_i a logical variable $\delta_i \in \{0, 1\}$, which has a value of either 1 if $X_i = \mathrm{T}$, or 0 if $X_i = \mathrm{F}$. A propositional logic problem, where a statement X must be proved to be true given a set of (compound) statements involving literals $X_1, \ldots, X_n$, can thus be solved by means of a linear integer program, by suitably translating the original compound statements into linear inequalities involving logical variables $\delta_1, \ldots, \delta_n$. In fact, the following propositions and linear constraints can easily be seen to be equivalent [219, p. 176]:

$$X_1 \wedge X_2 \quad \text{is equivalent to} \quad \delta_1 = \delta_2 = 1 \tag{2.8}$$
$$X_1 \vee X_2 \quad \text{is equivalent to} \quad \delta_1 + \delta_2 \geqslant 1 \tag{2.9}$$
$$\sim X_1 \quad \text{is equivalent to} \quad \delta_1 = 0 \tag{2.10}$$
$$X_1 \Rightarrow X_2 \quad \text{is equivalent to} \quad \delta_1 - \delta_2 \leqslant 0 \tag{2.11}$$

$$X_1 \Leftrightarrow X_2 \quad \text{is equivalent to} \quad \delta_1 - \delta_2 = 0 \tag{2.12}$$
$$X_1 \oplus X_2 \quad \text{is equivalent to} \quad \delta_1 + \delta_2 = 1 \ . \tag{2.13}$$

We can use this computational inference technique to model logical parts of processes (on/off switches, discrete mechanisms, combinational and sequential networks) and heuristic knowledge about plant operation as integer linear inequalities. In this way we can construct models of hybrid systems.

As we are interested in systems which contain both logic and continuous dynamics, we wish to establish a link between the two worlds. As will be shown next, we end up with mixed-integer linear inequalities, i.e., linear inequalities involving both continuous variables $x \in \mathbb{R}^n$ and logical variables $\delta \in \{0,1\}^{n_\delta}$.

Consider the function $f : \mathbb{R}^n \to \mathbb{R}$. Assume that $x \in \mathcal{X}$, where $\mathcal{X}$ is a given bounded set, and define

$$M = \max_{x \in \mathcal{X}} f(x), \qquad m = \min_{x \in \mathcal{X}} f(x) \ . \tag{2.14}$$

Theoretically, an over-estimate (under-estimate) of M (m) suffices for our purpose. However, more realistic estimates provide computational benefits [219, p. 171]. Now it is easy to verify that

$$[f(x) \leqslant 0] \wedge [\delta = 1] \quad \text{is true if and only if} \quad f(x) - \delta \leqslant -1 + m\,(1 - \delta) \tag{2.15}$$
$$[f(x) \leqslant 0] \vee [\delta = 1] \quad \text{is true if and only if} \quad f(x) \leqslant M\delta \tag{2.16}$$
$$\sim[f(x) \leqslant 0] \quad \text{is true if and only if} \quad f(x) \geqslant \varepsilon \ , \tag{2.17}$$

where ε is a small tolerance (typically the machine precision), beyond which the constraint is regarded as violated.

Remark 2.2.1 The reason for introducing ε in (2.17) is that an equation like $f(x) > 0$ does not fit the MLD framework, in which only nonstrict inequalities are allowed. Therefore, the equation $f(x) > 0$ is replaced by the equation $f(x) \geqslant \varepsilon$ with ε a small tolerance, typically the machine precision, where we assume that in practice the case $0 < f(x) < \varepsilon$ cannot occur due to the finite number of bits used for representing real numbers on a computer.

By (2.5) and (2.15)–(2.17), it also follows that

$$[f(x) \leqslant 0] \Rightarrow [\delta = 1] \quad \text{is true if and only if} \quad f(x) \geqslant \varepsilon + (m - \varepsilon)\delta \tag{2.18}$$
$$[f(x) \leqslant 0] \Leftrightarrow [\delta = 1] \quad \text{is true if and only if} \quad \begin{cases} f(x) \leqslant M(1 - \delta) \\ f(x) \geqslant \varepsilon + (m - \varepsilon)\delta \,. \end{cases} \tag{2.19}$$

Finally, we report procedures to transform products of logical variables, and of continuous and logical variables, in terms of linear inequalities, which however require the introduction of auxiliary variables [219, p. 178]. The product term $\delta_1\delta_2$ can be replaced by an auxiliary logical variable $\delta_3 = \delta_1\delta_2$. Then, $[\delta_3 = 1] \Leftrightarrow [\delta_1 = 1] \wedge [\delta_2 = 1]$, and therefore

$$\delta_3 = \delta_1\,\delta_2 \quad \text{is equivalent to} \quad \begin{cases} -\delta_1 + \delta_3 \leqslant 0 \\ -\delta_2 + \delta_3 \leqslant 0 \\ \delta_1 + \delta_2 - \delta_3 \leqslant 1 \,. \end{cases}$$

Moreover, the term $\delta f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}$ and $\delta \in \{0, 1\}$, can be replaced by an auxiliary real variable $y = \delta f(x)$ that satisfies $[\delta = 0] \Rightarrow [y = 0]$, $[\delta = 1] \Rightarrow [y = f(x)]$. Therefore, by defining M and m as in (2.14), $y = \delta f(x)$ is equivalent to

$$\begin{aligned} &y \leqslant M\delta \\ &y \geqslant m\delta \\ &y \leqslant f(x) - m(1-\delta) \\ &y \geqslant f(x) - M(1-\delta) \ . \end{aligned} \tag{2.20}$$

2.2.2 MLD systems

The results of the previous section will now be used to express relations describing the evolution of systems where physical laws, logic rules, and operating constraints are interdependent. Before giving a general definition of such a class of systems, consider the following (PWA) system:

$$x(k+1) = \begin{cases} 0.8\,x(k) + u(k) & \text{if } x(k) \geqslant 0 \\ -0.8\,x(k) + u(k) & \text{if } x(k) < 0 \end{cases} \tag{2.21}$$

where $x(k) \in [-10, 10]$, and $u(k) \in [-1, 1]$. The condition $x(k) \geqslant 0$ can be associated to a binary variable $\delta(k)$ such that

$$[\delta(k) = 1] \Leftrightarrow [x(k) \geqslant 0] \ .$$

By using the transformation (2.19), this equation can be expressed by the inequalities

$$10\,\delta(k) \leqslant x(k) + 10 \tag{2.22}$$

$$(-10 - \varepsilon)\,\delta(k) \leqslant -x(k) - \varepsilon \ , \tag{2.23}$$

where $M = -m = 10$, and ε is a small positive scalar (typically the machine precision, cf. Remark 2.2.1). Then (2.21) can be rewritten as

$$x(k+1) = 1.6\,\delta(k)x(k) - 0.8\,x(k) + u(k) \ .$$

By defining a new variable $z(k) = \delta(k)\,x(k)$ which, by (2.20), can be expressed as

$$z(k) \leqslant M\delta(k) \tag{2.24}$$

$$z(k) \geqslant m\delta(k) \tag{2.25}$$

$$z(k) \leqslant x(k) - m(1 - \delta(k)) \tag{2.26}$$

$$z(k) \geqslant x(k) - M(1 - \delta(k)) \ , \tag{2.27}$$

the evolution of system (2.21) is ruled by the linear equation

$$x(k+1) = 1.6z(k) - 0.8x(k) + u(k)$$

subject to the linear constraints (2.22), (2.23), and (2.24)–(2.27). This example can be generalized by describing mixed logical dynamical (MLD) systems through the following linear relations:

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \tag{2.28a}$$

$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \tag{2.28b}$$

$$E_1 x(k) + E_2 u(k) + E_3 \delta(k) + E_4 z(k) \leqslant g, \tag{2.28c}$$

where $x(k) = [{x_{\rm r}}^{\rm T}(k)\ {x_{\rm b}}^{\rm T}(k)]^{\rm T}$ with $x_{\rm r}(k) \in \mathbb{R}^{n_{\rm r}}$ and $x_{\rm b}(k) \in \{0,1\}^{n_{\rm b}}$ ($y(k)$ and $u(k)$ have a similar structure), and where $z(k) \in \mathbb{R}^{r_{\rm r}}$ and $\delta(k) \in \{0,1\}^{r_{\rm b}}$ are auxiliary variables. The inequalities (2.28c) have to be interpreted componentwise. Systems of the form (2.28) are called *mixed logical dynamical* (MLD) systems.

The MLD formalism allows specifying the evolution of continuous variables through linear dynamic equations, of discrete variables through propositional logic statements and automata, and the mutual interaction between the two. As explained above the key idea of the approach consists of embedding the logic part in the state equations by transforming Boolean variables into 0-1 integers, and by expressing the relations as mixed-integer linear inequalities. MLD systems are therefore capable of modeling a broad class of systems, in particular those systems that can be modeled through the hybrid system description language HYSDEL [204].

Remark 2.2.2 It is assumed that for all $x(k)$ with $x_{\rm b}(k) \in \{0,1\}^{n_{\rm b}}$, all $u(k)$ with $u_{\rm b}(k) \in \{0,1\}^{m_{\rm b}}$, all $z(k) \in \mathbb{R}^{r_{\rm r}}$ and all $\delta(k) \in \{0,1\}^{r_{\rm b}}$ satisfying (2.28c) it holds that $x(k+1)$ and $y(k)$ determined from (2.28a)-(2.28b) are such that $x_{\rm b}(k+1) \in \{0,1\}^{n_{\rm b}}$ and $y_{\rm b}(k) \in \{0,1\}^{l_{\rm b}}$. This is without loss of generality, as we can take binary components of states and outputs (if any) to be auxiliary variables as well (see the proof of [25, Prop. 1]). Indeed, if, e.g., $y_{\rm b}(k) \in \{0,1\}^{l_{\rm b}}$ is not directly implied by the (in)equalities, we introduce an additional binary variable $\delta_y(k) \in \{0,1\}^{l_{\rm b}}$ and the inequalities

$$[Cx(k) + D_1 u(k) + D_2 \delta(k) + D_3 z(k)]_{\rm b} - \delta_y(k) \leqslant 0 \tag{2.29a}$$
$$[-Cx(k) - D_1 u(k) - D_2 \delta(k) - D_3 z(k)]_{\rm b} + \delta_y(k) \leqslant 0, \tag{2.29b}$$

which sets $\delta_y(k)$ equal to $y_{\rm b}(k)$. The notation $[\,]_{\rm b}$ is used to select the rows of the expression (2.28b) that correspond to the binary part of $y(k)$. Hence, $y_{\rm b}(k) = \delta_y(k) \in \{0,1\}^{l_{\rm b}}$. Similarly, we can deal with $u_{\rm b}(k)$ and $x_{\rm b}(k+1)$.

The following example is a corrected version of an example from [27][2].

Example 2.2.3 Consider the following system :

$$x(k+1) = 0.8 \begin{bmatrix} \cos\alpha(k) & -\sin\alpha(k) \\ \sin\alpha(k) & \cos\alpha(k) \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \tag{2.30}$$
$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \tag{2.31}$$
$$\alpha(k) = \begin{cases} \frac{\pi}{3} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \geqslant 0 \\ -\frac{\pi}{3} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) < 0 \end{cases} \tag{2.32}$$
$$x(k) \in [-10, 10] \times [-10, 10] \tag{2.33}$$
$$u(k) \in [-1, 1] \tag{2.34}$$

Define

$$A_1 = 0.8 \begin{bmatrix} \cos\frac{\pi}{3} & -\sin\frac{\pi}{3} \\ \sin\frac{\pi}{3} & \cos\frac{\pi}{3} \end{bmatrix},\ A_2 = 0.8 \begin{bmatrix} \cos\left(-\frac{\pi}{3}\right) & -\sin\left(-\frac{\pi}{3}\right) \\ \sin\left(-\frac{\pi}{3}\right) & \cos\left(-\frac{\pi}{3}\right) \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

[2] In particular, (2.36) has been corrected w.r.t. the equation given in [27].

If we define $\delta(k) \in \{0,1\}$ such that $[\delta(k)=1] \Leftrightarrow \big[[1\ \ 0]x(k) \geqslant 0\big]$, then combining (2.30) and (2.32) results in

$$x(k+1) = z_{(1)}(k) + z_{(2)}(k)$$

with

$$\begin{aligned} z_{(1)}(k) &= \big(A_1 x(k) + Bu(k)\big)\delta(k) \\ z_{(2)}(k) &= \big(A_2 x(k) + Bu(k)\big)(1-\delta(k)) \end{aligned}$$

Note that the latter equations are nonlinear. However, using the results of Section 2.2.1, they can be rewritten as in (2.24)–(2.27), which ultimately results in

$$x(k+1) = \begin{bmatrix} I & I \end{bmatrix} z(k) \tag{2.35}$$

$$\begin{bmatrix} 10 \\ -10-\varepsilon \\ -M \\ -M \\ M \\ M \\ M \\ M \\ -M \\ -M \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta(k) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ I & 0 \\ -I & 0 \\ 0 & I \\ 0 & -I \\ I & 0 \\ -I & 0 \\ 0 & I \\ 0 & -I \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} z(k) \leqslant \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ B \\ -B \\ B \\ -B \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} u(k) + \begin{bmatrix} [1\ \ 0] \\ [-1\ \ 0] \\ 0 \\ 0 \\ 0 \\ 0 \\ A_1 \\ -A_1 \\ A_2 \\ -A_2 \\ I \\ -I \\ 0 \\ 0 \end{bmatrix} x(k) + \begin{bmatrix} 10 \\ -\varepsilon \\ 0 \\ 0 \\ M \\ M \\ M \\ M \\ 0 \\ 0 \\ N \\ N \\ 1 \\ 1 \end{bmatrix}, \tag{2.36}$$

with I the (2-by-2) identity matrix, and $M = 4(1+\sqrt{3})\big[1\ \ 1\big]^{\mathrm{T}} + B$, $N = 10\big[1\ \ 1\big]^{\mathrm{T}}$, $z(k) = \big[z_{(1)}^{\mathrm{T}}(k)\ \ z_{(2)}^{\mathrm{T}}(k)\big]^{\mathrm{T}} \in \mathbb{R}^4$, and ε is a properly small positive scalar (typically the machine precision, see also Remark 2.2.1).

2.3 Linear Complementarity (LC) systems

Linear complementarity (LC) systems are studied in e.g., [117, 211]. In discrete time these systems are given by the equations

$$\begin{aligned} x(k+1) &= Ax(k) + B_1 u(k) + B_2 w(k) && \text{(2.37a)} \\ y(k) &= Cx(k) + D_1 u(k) + D_2 w(k) && \text{(2.37b)} \\ v(k) &= E_1 x(k) + E_2 u(k) + E_3 w(k) + g && \text{(2.37c)} \\ 0 \leqslant v(k) &\perp w(k) \geqslant 0 && \text{(2.37d)} \end{aligned}$$

with $v(k), w(k) \in \mathbb{R}^s$ and where $\perp$ denotes the orthogonality of vectors (i.e., $v(k) \perp w(k)$ means that $v^{\mathrm{T}}(k)w(k) = 0$). We call $v(k)$ and $w(k)$ the complementarity variables.

In [52,53,55,110,111,117,211,212] (linear) complementarity systems in *continuous* time have been studied. Applications include constrained mechanical systems, electrical networks with ideal diodes or

other dynamical systems with piecewise affine relations, variable structure systems, constrained optimal control problems, projected dynamical systems, and so on [110, Ch. 2]. Actually, the continuous-time setting is the natural habitat for complementarity systems as is also evidenced by the literature on this topic [52, 53, 55, 110, 111, 117, 211, 212].

For an example of a (continuous-time) LC model we refer to the two-carts example of Section 1.6.4, and to the boost converter example of Section 1.6.6.

2.4 Extended Linear Complementarity (ELC) systems

In [78, 79, 84] it has been shown that several types of hybrid systems can be modeled as extended linear complementarity (ELC) systems:

$$x(k+1) = Ax(k) + B_1u(k) + B_2d(k) \tag{2.38a}$$
$$y(k) = Cx(k) + D_1u(k) + D_2d(k) \tag{2.38b}$$
$$E_1x(k) + E_2u(k) + E_3d(k) \leqslant g \tag{2.38c}$$
$$\sum_{i=1}^{p} \prod_{j \in \phi_i} \big(g - E_1x(k) - E_2u(k) - E_3d(k)\big)_j \;=\; 0, \tag{2.38d}$$

where $d(k) \in \mathbb{R}^r$ is an auxiliary variable. Condition (2.38d) is equivalent to

$$\prod_{j \in \phi_i} \big(g - E_1x(k) - E_2u(k) - E_3d(k)\big)_j = 0 \quad \text{for each } i \in \{1, 2, \ldots, p\} \tag{2.39}$$

due to the inequality conditions (2.38c). This implies that (2.38c)–(2.38d) can be considered as a system of linear inequalities (i.e., (2.38c)), where there are p groups of linear inequalities (one group for each index set ϕ_i) such that in each group at least one inequality should hold with equality.

2.5 Max-Min-Plus-Scaling (MMPS) systems

In [84] a class of discrete-event systems has been introduced that can be modeled using the operations maximization, minimization, addition and scalar multiplication. Expressions that are built using these operations are called max-min-plus-scaling (MMPS) expressions.

Definition 2.5.1 (Max-min-plus-scaling expression) A max-min-plus-scaling expression f of the variables $x_1, \ldots, x_n$ is defined by the grammar[3]

$$f := x_i | \alpha | \max(f_k, f_l) | \min(f_k, f_l) | f_k + f_l | \beta f_k \tag{2.40}$$

with $i \in \{1, 2, \ldots, n\}$, $\alpha, \beta \in \mathbb{R}$, and where f_k, f_l are again MMPS expressions.

An MMPS expression is, e.g., $5x_1 - 3x_2 + 7 + \max(\min(2x_1, -8x_2), x_2 - 3x_3)$.

Consider now systems that can be described by

$$x(k+1) = \mathcal{M}_x(x(k), u(k), d(k)) \tag{2.41a}$$

[3]The symbol | stands for OR and the definition is recursive.

$$y(k) = \mathcal{M}_y(x(k), u(k), d(k)) \tag{2.41b}$$
$$\mathcal{M}_{\mathrm{c}}(x(k), u(k), d(k)) \leqslant c, \tag{2.41c}$$

where $\mathcal{M}_x$, $\mathcal{M}_y$ and $\mathcal{M}_{\mathrm{c}}$ are MMPS expressions in terms of the components of $x(k)$, the input $u(k)$ and the auxiliary variables $d(k)$, which are all real-valued. Such systems will be called MMPS systems.

A typical example of an (unconstrained) MMPS is a system consisting of traffic-signal controlled intersections [78].

It is easy to verify that if we recast the PWA model (2.4) into an MMPS model we obtain

$$x(k+1) = \min(x(k) + u(k), 1)$$
$$y(k) = x(k) \ .$$

Remark 2.5.2 MMPS systems are an extension of max-plus-linear (discrete-event) systems, for which many analytical analysis methods are available [18], and for which MPC-like controllers can be designed very efficiently [82, 209].

2.6 Equivalence of MLD, LC, ELC, PWA and MMPS systems

In this section we prove that MLD, LC, ELC, PWA and MMPS systems[4] are equivalent (although in some cases additional assumptions are required). Here, equivalence between two model classes $\mathcal{A}$ and $\mathcal{B}$ should be interpreted as follows: for a given model A in $\mathcal{A}$ there exists a model B in $\mathcal{B}$ (and vice versa) such that the input-output behavior of both models coincides. The relations between the models are depicted in Figure 2.1.

2.6.1 MLD and LC systems

Proposition 2.6.1 *Every MLD system can be written as an LC system.*

Proof. Consider the MLD system (2.28). To rephrase the condition $\delta(k) \in \{0,1\}^{r_{\mathrm{b}}}$ in complementarity terms, we note that $\delta_i(k) \in \{0,1\}$ is equivalent to $0 \leqslant \delta_i(k) \perp 1 - \delta_i(k) \geqslant 0$. By introducing the auxiliary variable $v_1(k)$ this gives in vector notation $v_1(k) = e - \delta(k)$ together with $0 \leqslant \delta(k) \perp v_1(k) \geqslant 0$, where e denotes the vector for which all entries are equal to one. Note that the binary constraints over $u_{\mathrm{b}}(k)$, $y_{\mathrm{b}}(k)$, and $x_{\mathrm{b}}(k+1)$ are included in these complementarity conditions as indicated in Remark 2.2.2.
Next the inequality constraints in (2.28c) are modeled by introducing the auxiliary variables $w_2(k)$ and $v_2(k)$. Define $v_2(k) = g - E_1 x(k) - E_2 u(k) - E_3 \delta(k) - E_4 z(k)$. It is clear that $v_2(k) \geqslant 0$ implies the existence of a $w_2(k)$ (take $w_2(k) = 0$) such that

$$0 \leqslant v_2(k) \perp w_2(k) \geqslant 0. \tag{2.42}$$

Vice versa, if (2.42) is satisfied, it is obvious that $v_2(k) \geqslant 0$. Since $w_2(k)$ does not influence any other relation, it follows that $v_2(k) \geqslant 0$ can be replaced by (2.42).

[4]Note that we consider discrete-time systems in this chapter. In the continuous-time case the equivalence between the various model classes is very complicated. Moreover, the formulation of the dynamics in the continuous-time case is also more involved (See also Chapter 3.)

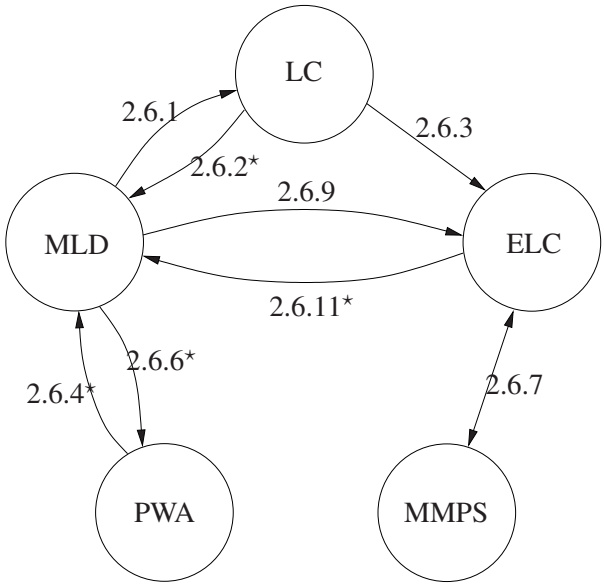

Figure 2.1: Graphical representation of the links between the classes of hybrid systems considered in this chapter. An arrow going from class A to class B means that A is a subset of B. The number next to each arrow corresponds to the proposition that states this relation. Moreover, arrows with a star ($\star$) require conditions to establish the indicated inclusion.

The special structure of LC systems does not directly allow auxiliary variables $z(k)$ in the right-hand side of (2.37a)–(2.37b) (only nonnegative complementarity variables are possible). Therefore, we split $z(k)$ in its "positive" and "negative part" as $z(k) := z^+(k) - z^-(k)$ with $z^+(k) = \max(0, z(k))$ and $z^-(k) = \max(0, -z(k))$. In complementarity terms this can be written as $z(k) = z^+(k) - z^-(k)$ with $0 \leqslant z^+(k) \perp z^-(k) \geqslant 0$. By collecting all equations, and introducing two extra auxiliary vectors $v_3(k)$ and $v_4(k)$ (which will in fact be equal to $z^-(k)$ and $z^+(k)$, respectively), we obtain the LC system

$$
\begin{aligned}
x(k+1) &= Ax(k) + B_1 u(k) + [B_2 \;\; 0 \;\; B_3 \;\; -B_3] w(k) && \text{(2.43a)}\\
y(k) &= Cx(k) + D_1 u(k) + [D_2 \;\; 0 \;\; D_3 \;\; -D_3] w(k) && \text{(2.43b)}\\
\underbrace{\begin{pmatrix} v_1(k) \\ v_2(k) \\ v_3(k) \\ v_4(k) \end{pmatrix}}_{=:v(k)} &= \begin{pmatrix} e \\ g - E_1 x(k) - E_2 u(k) \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -I & 0 & 0 & 0 \\ -E_3 & 0 & -E_4 & E_4 \\ 0 & 0 & 0 & I \\ 0 & 0 & I & 0 \end{pmatrix} \underbrace{\begin{pmatrix} \delta(k) \\ w_2(k) \\ z^+(k) \\ z^-(k) \end{pmatrix}}_{=:w(k)} && \text{(2.43c)}\\
0 &\leqslant v(k) \perp w(k) \geqslant 0. && \text{(2.43d)}
\end{aligned}
$$

where I denotes the identity matrix.

Proposition 2.6.2 *Every LC system can be written as an MLD system, provided that the variables $w(k)$ and $v(k)$ are (componentwise) bounded.*

Proof. Note that the complementarity condition (2.37d) implies that for each $i \in \{1, \ldots, s\}$ we have $v_i(k) = 0$, $w_i(k) \geqslant 0$ or $v_i(k) \geqslant 0$, $w_i(k) = 0$. The idea is now to introduce a vector of binary variables $\delta(k) \in \{0,1\}^s$ and represent $v_i(k) = 0$, $w_i(k) \geqslant 0$ with $\delta_i(k) = 1$, and $v_i(k) \geqslant 0$, $w_i(k) = 0$ with

$\delta_i(k) = 0$. This can be achieved by introducing the constraints

$$w(k) \leqslant M_w \delta(k); \quad v(k) \leqslant M_v(e - \delta(k)); \quad w(k) \geqslant 0; \quad v(k) \geqslant 0,$$

where M_w and M_v are diagonal matrices containing upper-bounds on $w(k)$ and $v(k)$, respectively, on the diagonal, and e denotes (once more) the vector for which all entries are equal to one. By setting $z(k) = w(k)$ and replacing $v(k)$ in the inequalities above by $E_1 x(k) + E_2 u(k) + E_3 w(k) + g$ it is easy to rewrite the LC system (2.37) as the following MLD model

$$\begin{aligned} x(k+1) &= Ax(k) + B_1 u(k) + B_2 z(k) \\ y(k) &= Cx(k) + D_1 u(k) + D_2 z(k) \\ \begin{bmatrix} 0 \\ E_1 \\ 0 \\ -E_1 \end{bmatrix} x(k) &+ \begin{bmatrix} 0 \\ E_2 \\ 0 \\ -E_2 \end{bmatrix} u(k) + \begin{bmatrix} -M_w \\ M_v \\ 0 \\ 0 \end{bmatrix} \delta(k) + \begin{bmatrix} I \\ E_3 \\ -I \\ -E_3 \end{bmatrix} z(k) \leqslant \begin{bmatrix} 0 \\ M_v e - g \\ 0 \\ g \end{bmatrix} . \end{aligned}$$

Proposition 2.6.2 assumes that upper bounds on w, v are known. This hypothesis is not restrictive in practice, as these quantities are related to continuous inputs and states of the system, which are usually bounded for physical reasons.

2.6.2 LC and ELC systems

Proposition 2.6.3 *Every LC system can be written as an ELC system.*

Proof. It can easily be verified that (2.37) can be rewritten as

$$x(k+1) = Ax(k) + B_1 u(k) + B_2 \underbrace{w(k)}_{=d(k)} \tag{2.44a}$$

$$y(k) = Cx(k) + D_1 u(k) + D_2 w(k) \tag{2.44b}$$

$$-E_1 x(k) - E_2 u(k) - E_3 w(k) \leqslant g \tag{2.44c}$$

$$-w(k) \leqslant 0 \tag{2.44d}$$

$$\sum_{i=1}^{p} \prod_{j \in \phi_i} (g + E_1 x(k) + E_2 u(k) + E_3 w(k))_j (w(k))_j = 0, \tag{2.44e}$$

where the sets ϕ_i contain typically two elements and are given by $\phi_i = \{i, i+s\}$ for $i = 1, 2, \ldots, p$, where s is the dimension of $w(k)$ in (2.37). Note that the system of inequalities (2.44c)–(2.44d) corresponds to (2.38c).

2.6.3 PWA and MLD systems

A PWA system of the form (2.1) is called well-posed, if (2.1) is uniquely solvable in $x(k+1)$ and $y(k)$, once $x(k)$ and $u(k)$ are specified. The following proposition has been stated in [25] and is an easy extension of the corresponding result in [27] for piecewise linear (PWL) systems (i.e., PWA systems with $f_i = g_i = 0$).

Proposition 2.6.4 *Every well-posed PWA system can be rewritten as an MLD system assuming that the set of feasible states and inputs is bounded.*

Remark 2.6.5 As already indicated in Remark 2.2.1 MLD models only allow for nonstrict inequalities in (2.28c). Hence, in rewriting a discontinuous PWA system as an MLD model strict inequalities like $x(k) < 0$ must be approximated by $x(k) \leqslant -\varepsilon$ for some $\varepsilon > 0$ (typically the machine precision), with the assumption that $-\varepsilon < x(k) < 0$ cannot occur due to the finite number of bits used for representing real numbers (no problem exists when the PWA system is continuous, where the strict inequality can be equivalently rewritten as nonstrict, or $\varepsilon = 0$). See [27] for more details and Section 2.6.6 for an example. From a strictly theoretical point of view, the inclusion stated in Proposition 2.6.4 is therefore not exact for discontinuous PWA systems, and the same clearly holds for an LC, ELC or MMPS reformulation of a discontinuous PWA system when the route via MLD is taken. One way to circumvent such an inexactness is to allow part of the inequalities in (2.28c) to be strict. On the other hand, from a numerical point of view this issue is not relevant. The equivalence of LC and MLD systems as in Section 2.6.1 implies that all continuous PWA systems can be exactly written as LC systems as well. A similar result for continuous PWA systems can be derived from [87].

The reverse statement of Proposition 2.6.4 has been established in [25] under the condition that the MLD system is completely well-posed. The MLD system (2.28a) is called completely well-posed, if $x(k+1)$, $y(k)$, $\delta(k)$ and $z(k)$ are uniquely defined in their domain, once $x(k)$ and $u(k)$ are assigned [27].

Proposition 2.6.6 *A completely well-posed MLD system can be rewritten as a PWA system.*

2.6.4 MMPS and ELC systems

Proposition 2.6.7 *The classes of MMPS and ELC systems coincide.*

Proof. First we prove that the MMPS system (2.41) can be recast as an ELC system by showing that each of the six basic constructions for MMPS expressions fits in the ELC framework:

- Expressions of the form $f = x_i$, $f = \alpha$, $f = f_k + f_l$ and $f = \beta f_k$ result in linear equations of the form (2.38a)–(2.38b).
- An expression of the form $f = \max(f_k, f_l) = -\min(-f_k, -f_l)$ can be rewritten as

$$f - f_k \geqslant 0,\ f - f_l \geqslant 0,\ (f - f_k)(f - f_l) = 0,$$

  which is an expression of the form (2.38c)–(2.38d).

Furthermore, it is easy to verify that two or more ELC systems can be combined into one large ELC system. As a consequence, every MMPS system can be rewritten as an ELC system.

Now we show that the ELC system (2.38) can be written in the form (2.41). Clearly, (2.38a) and (2.38b) are MMPS expressions (albeit without $\max$ or $\min$) of the form (2.41a) and (2.41b), respectively. Note that by (2.38c) we have

$$\big(g - E_1 x(k) - E_2 u(k) - E_3 d(k)\big)_j \geqslant 0 \qquad \text{for each } j\ . \tag{2.45}$$

Furthermore, the complementarity condition (2.38d) can be rewritten as (2.39), or equivalently:

$$\forall i \in \{1,2,\ldots,p\} : \exists j \in \phi_i \text{ such that } \big(g - E_1x(k) - E_2u(k) - E_3d(k)\big)_j = 0 \ .$$

If we combine this with (2.45) we obtain

$$\min_{j\in\phi_i}\big(g - E_1x(k) - E_2u(k) - E_3d(k)\big)_j = 0 \quad \text{for } i = 1,2,\ldots,p, \tag{2.46}$$

which are all MMPS constraints of the form (2.41c). The conditions in (2.45) for which j does not belong to some ϕ_i can be bundled as the MMPS constraint

$$\min_{j\in\Psi}\big(g - E_1x(k) - E_2u(k) - E_3d(k)\big)_j \geqslant 0, \tag{2.47}$$

where $\Psi = \big\{j \in \{1,2,\ldots,q\} \,|\, \forall i \in \{1,2,\ldots,p\} : j \notin \phi_i\big\}$ and where q is the dimension of the vector g. So, the constraints (2.38c)–(2.38d) are equivalent to the MMPS constraints (2.46)–(2.47).

Remark 2.6.8 Note that LC systems (2.37) can be written as a MMPS on the basis of the above proposition and Proposition 2.6.3. A more direct reformulation of LC systems is given by (2.37a)–(2.37b) and $\min[(E_1x(k) + E_2u(k) + E_3w(k) + g)_j, w_j(k)] = 0$ for all j. This was pointed out by Aganagic [1].

2.6.5 MLD and ELC systems

Proposition 2.6.9 *Every MLD system can be rewritten as an ELC system.*

Proof. If we make an abstraction of the range of the variables then (2.28a)–(2.28c) coincide with (2.38a)–(2.38c) with $d(k) = [\ \delta^{\mathrm{T}}(k)\ \ z^{\mathrm{T}}(k)\ ]^{\mathrm{T}}$. Furthermore, a condition of the form $\delta_i(k) \in \{0,1\}$ is equivalent to the ELC conditions $-\delta_i(k) \leqslant 0$, $\delta_i(k) \leqslant 1$, $\delta_i(k)(1-\delta_i(k)) = 0$. So every MLD system can be rewritten as an ELC system.

Remark 2.6.10 Note that the condition $\delta_i(k) \in \{0,1\}$ is also equivalent to the MMPS constraint $\max(-\delta_i(k), \delta_i(k) - 1) = 0$ or $\min(\delta_i(k), 1 - \delta_i(k)) = 0$.

Proposition 2.6.11 *Every ELC system can be written as an MLD system, provided that the quantity $g - E_1x(k) - E_2u(k) - E_3d(k)$ is (componentwise) bounded.*

Proof. Introduce the following two inequalities

$$g_j - (E_1x(k) + E_2u(k) + E_3d(k))_j \leqslant M_j\delta_j(k) \text{ for each } j \in \phi_i \tag{2.48a}$$

$$\sum_{j\in\phi_i}\delta_j(k) \leqslant \#\phi_i - 1, \tag{2.48b}$$

where $\#\phi_i$ is the cardinality (i.e., the number of elements) of the set ϕ_i and where $\delta_j(k) \in \{0,1\}$ are auxiliary variables, and M_j is an upper bound for $(g - E_1x(k) - E_2u(k) - E_3d(k))_j$. As by the last condition at least one $\delta_h(k)$ is zero for some $h \in \phi_i$, the first inequality and the ELC inequality $g_j - (E_1x(k) + E_2u(k) + E_3d(k))_j \geqslant 0$ degenerate to an equality condition for $j = h$. Hence, the system of equations (2.48) in combination with (2.38c) is of the form (2.38c)–(2.38d). So by defining $z(k) = d(k)$ and collecting all the inequalities, it is immediate to rewrite the ELC representation (2.38) into an MLD form.

Note that (just as for Proposition 2.6.2) the boundedness hypothesis in Proposition 2.6.11 is not restrictive in practice, since the inputs and states of the system are usually bounded for physical reasons.

2.6.6 Example

To demonstrate the equivalences proven above, we consider the example [27]

$$x(k+1) = \begin{cases} 0.8x(k) + u(k) & \text{if } x(k) \geqslant 0 \\ -0.8x(k) + u(k) & \text{if } x(k) < 0 \end{cases} \tag{2.49}$$

with $m \leqslant x(k) \leqslant M$. In [27] it is shown that (2.49) can be written as[5]

$$\begin{array}{rcl@{\qquad}rcl} x(k+1) & = & -0.8x(k) + u(k) + 1.6z(k) & & & \\ -m\delta(k) & \leqslant & x(k) - m; & x(k) & \leqslant & (M+\varepsilon)\delta(k) - \varepsilon \\ z(k) & \leqslant & M\delta(k); & z(k) & \geqslant & m\delta(k) \\ z(k) & \leqslant & x(k) - m(1-\delta(k)); & z(k) & \geqslant & x(k) - M(1-\delta(k)) \end{array} \tag{2.50}$$

and the condition $\delta(k) \in \{0,1\}$. Note that the strict inequality $x(k) < 0$ has been replaced by $x(k) \leqslant -\varepsilon$, where $\varepsilon > 0$ is a small number (typically the machine precision). In view of Remark 2.6.5 observe that $\varepsilon = 0$ results in a mathematically exact MLD model. In this case the model is well-posed[6], but not completely well-posed as $x(k) = 0$ allows both $\delta(k) = 0$ and $\delta(k) = 1$.

One can verify that (2.49) can be rewritten as the MMPS model

$$x(k+1) = -0.8x(k) + 1.6\max(0, x(k)) + u(k) \ , \tag{2.51}$$

as the LC formulation

$$x(k+1) = -0.8x(k) + u(k) + 1.6z(k); \tag{2.52a}$$
$$0 \leqslant w(k) = -x(k) + z(k) \perp z(k) \geqslant 0, \tag{2.52b}$$

and as the ELC representation

$$x(k+1) = -0.8x(k) + u(k) + 1.6d(k) \tag{2.53a}$$
$$-d(k) \leqslant 0; \qquad x(k) - d(k) \leqslant 0; \qquad 0 = \big(x(k) - d(k)\big)\big(-d(k)\big). \tag{2.53b}$$

While the MLD representation (2.50) requires bounds on $x(k)$, $u(k)$ to be specified (although such bounds can be arbitrarily large), the PWA, MMPS, LC, and ELC expressions do not require such a specification.

Note that we only need one max-operator in (2.51) and one complementarity pair in (2.52). If we would transform the MLD system (2.50) into, e.g., the LC model as indicated by the equivalence proof, this would require nine complementarity pairs. Hence, it is clear that the proofs only show the conceptual equivalence, but do not result in the most compact models.

[5] For more complex examples, it is easier to introduce one dummy variable $\delta_i(k)$ for each region Ω_i of the PWA system. However, this will result in more binary variables in the final MLD system, increasing the complexity of the model. In this example a more efficient approach can be used in which only one dummy variable is required (since for this example Ω_1 is the complement of Ω_2). See also [22] for additional information.

[6] An MLD model is called well-posed, if $x(k+1)$ and $y(k)$ are uniquely determined, once $x(k)$ and $u(k)$ are given. Note that there are no requirements on $\delta(k)$ and $z(k)$.

2.7 Jump-flow systems and hybrid inclusions

Jump-flow systems (sometimes also called hybrid inclusions) [50, 101, 102] form a quite natural extension of differential equations $\dot{x} = f(x)$ in the sense that invariants, guards, and resets are added as are present in the hybrid automata description as well. Hybrid inclusions are given by the data of two subsets $\mathcal{C}$ (the flow set) and $\mathcal{D}$ (the jump set) of $\mathbb{R}^n$, and two mappings $f : \mathcal{C} \to \mathbb{R}^n$ and $g : \mathcal{D} \to \mathbb{R}^n$. The hybrid inclusion is then written as

$$\dot{x}(t) = f(x(t)) \quad \text{if} \quad x(t) \in \mathcal{C} \tag{2.54a}$$
$$x(t^+) = g(x(t)) \quad \text{if} \quad x(t) \in \mathcal{D}. \tag{2.54b}$$

The latter equation indicates that when the state $x(t)$ reaches $\mathcal{D}$ a reset of the state can take place at time t to $x(t^+) = g(x(t))$. The variable $x(t) \in \mathbb{R}^n$ is the state variable with $t \in \mathbb{R}$, but certain components of x sometimes only take discrete values to model Boolean and logic variables in the system. In general, the mappings f and g may also be set-valued and then the description becomes

$$\dot{x}(t) \in f(x(t)) \quad \text{if} \quad x(t) \in \mathcal{C} \tag{2.55a}$$
$$x(t^+) \in g(x(t)) \quad \text{if} \quad x(t) \in \mathcal{D}, \tag{2.55b}$$

which explains the term *hybrid inclusions* as it is a natural extension of differential inclusions.

Clearly, this description provides compact models with a clear structure, which encompasses many hybrid phenomena (although sometimes describing specific hybrid models, e.g. a hybrid automaton, in this framework might be quite cumbersome). Hybrid inclusions apply naturally in areas such as reset and networked control systems, see e.g., [101, 118, 173].

2.8 Timed automata

Timed automata are a class of hybrid systems that involve particularly simple continuous dynamics: all differential equations are of the form $\dot{x} = 1$, and all the invariants, guards, etc. involve comparison of the real-valued states with constants (e.g., $x < 2$, $x \geqslant 0$, $x = 3$, etc.). Clearly, timed automata are somewhat limited when it comes to modeling physical systems. They are very good however for encoding timing constraints (such as "event A must take place at least 2 seconds after event B and not more than 5 seconds before event C", etc.). For some applications, such as multimedia, Internet and audio protocol verification, etc. this type of description is sufficient for both the dynamics of the system and the properties that we want the system to satisfy. We will only give a brief introduction to timed automata. Readers interested in the details are referred to the (rather technical but classic) paper [4].

Consider $x \in \mathbb{R}^n$. A subset of $\mathbb{R}^n$ set is called rectangular if it can be written as a finite boolean combination of constraints of the form $x_i \leqslant a$, $x_i < b$, $x_i = c$, $x_i \geqslant d$, or $x_i > e$, where a, b, c, d, e are rational numbers. Roughly speaking, rectangular sets are "rectangles" in $\mathbb{R}^n$ whose sides are aligned with the axes, or unions of such rectangles. For example, the set

$$\{(x_1, x_2) \in \mathbb{R}^2 \mid (x_1 \geqslant 0) \wedge (x_1 \leqslant 2) \wedge (x_2 \geqslant 1) \wedge (x_2 \leqslant 2)\}$$

is rectangular, and so is the set

$$\{(x_1, x_2) \in \mathbb{R}^2 \mid ((x_1 \geqslant 0) \wedge (x_2 = 0)) \vee ((x_1 = 0) \wedge (x_2 \geqslant 0))\} \ .$$

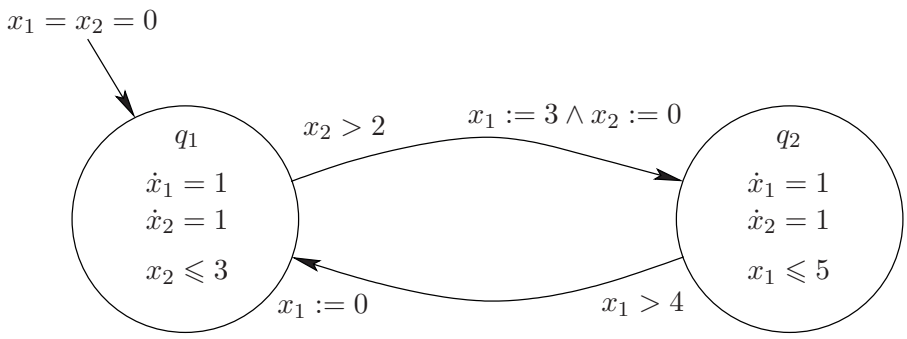

Figure 2.2: Example of a timed automaton.

The empty set is also a rectangular set (e.g., $\varnothing = \{(x_1, x_2) \in \mathbb{R}^2 \mid (x_1 > 1) \wedge (x_1 \leqslant 0))\}$. However the set $\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 = 2x_2\}$ is not rectangular.

Note that rectangular sets are easy to encode in a computer. Instead of storing the set itself (which is impossible since the set is infinite) we can store and manipulate the list of constraints used to generate the set (which is finite).

Roughly speaking, a timed automaton is a hybrid automaton with the following characteristics:

- the automaton involves differential equations of the form $\dot{x}_i = 1$. Continuous variables governed by this differential equation are known as clocks.
- the sets involved in the definition of the initial states, guards and invariants are rectangular sets,
- the reset maps involve either a rectangular set, or may leave certain states unchanged.

Example 2.8.1 (Timed automaton) An example of a timed automaton is given in Figure 2.2. We have

- $\mathcal{Q} = \{q_1, q_2\}$,
- $\mathcal{X} = \mathbb{R}^2$,
- $f(q_1, x) = f(q_2, x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$,
- $\text{Init} = \{(q_1, (0, 0))\}$,
- $\text{Inv}(q_1) = \{x \in \mathbb{R}^2 \mid x_2 \leqslant 3\}$, $\text{Inv}(q_2) = \{x \in \mathbb{R}^2 \mid x_1 \leqslant 5\}$,
- $E = \{(q_1, q_2), (q_2, q_1)\}$,
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 \mid x_2 > 2\}$, $G(q_2, q_1) = \{x \in \mathbb{R}^2 \mid x_1 > 4\}$,
- $R(q_1, q_2) = \big\{\big((x_1, x_2),\, (3, 0)\big) \,\big|\, x_1, x_2 \in \mathbb{R}\big\}$, $R(q_2, q_1) = \big\{\big((x_1, x_2),\, (0, x_2)\big) \,\big|\, x_1, x_2 \in \mathbb{R}\big\}$.

2.9 Timed Petri nets

The well-known Petri net formalism — an untimed, logical representation of the dynamics of a discrete-event system — can be extended by requiring that a transition be executed within a certain time interval

after the transition becomes enabled. This results in a hybrid system that contains both discrete state variables (the marking of the places) and continuous state variables (the timers).

The graphical representation of a classical Petri net [170,181,182] efficiently represents the ordering of events in a plant (an event corresponds to the execution of a transition). The interaction between several modules of the plant is represented by ensuring that transitions which are common to several Petri net components can only be executed when their execution is allowed in each of the Petri net components. In this section we extend the Petri net model by including in the description the real time at which events take place. A model of a transportation system illustrates how timed Petri nets can represent fairly large plants. We also indicate how classical analysis techniques for Petri nets can be extended to study properties of timed Petri nets.

2.9.1 Semantics of timed Petri net models

Consider as an example an integrated multi-modal transport system where different goods, following different routes, are moved from origin to destination, using different types of vehicles; transshipment stations allow transfer of goods from one type of vehicle to another type. One set of modules represents the different routes along which goods can be transported, using a timed Petri net component for each route. These components all look very similar, only the names of places and transitions, and perhaps the length of the paths, differing (compare this to classes in object oriented programming). Another class of modules represents the location and availability of all vehicles of a particular type. The overall model contains as many components of this class as there are different types of vehicles. Finally a third class of components represents the state of the resources in the different transshipment stations. Each of these modules will be represented by a timed Petri net. Figure 2.3 provides a very simple instantiation for each of the three classes of Petri net components necessary to represent a multi-modal transportation system.

An untimed Petri net consists of a finite set of places P, a finite set of transitions T, and a set of directed arcs connecting some places to some transitions, and connecting some transitions to some places. Graphically places are represented by circles, transitions by bars, and directed arcs by lines with an arrow. The state of the module represented by the Petri net, is described by the distribution of tokens over the places of the Petri net. In other words the number of tokens $m(p) \geqslant 0$ in each place $p \in P$ (i.e., the number of dots in the circle representing place p) specifies the possible future behavior of the module. Transition t is called state-enabled when each upstream place of t contains at least one token. The set of upstream places of t is denoted by ${}^\bullet t$, while output places of t are denoted by $t^\bullet$. The event modeled by transition t can be executed only if t is state-enabled. Executing transition t at time θ removes a token from each upstream place of t and at the same time adds one token to each downstream place of t.

An untimed Petri net model specifies the order in which events can occur during the operation of the plant. In applications one is not only interested in the order in which events happen, but also in the time instants at which these events can occur. One might for example want to determine the range of feasible throughputs of the transportation system, or the maximal time delay for perishable items. In order to represent these time constraints the model has to describe exactly when transitions can actually fire, after they have become state-enabled. This requires an extension of the Petri net model.

The state of the net at time θ for a timed Petri net not only remembers the number of tokens in each place, but also their arrival times. This implies that the state M_θ of the timed, marked Petri net at time θ lists, for each place p, the set $M_\theta(p) := \{\theta_1, \ldots, \theta_{m_\theta(p)}\}$ of arrival times $\theta_1 \leqslant \theta_2 \leqslant \ldots \leqslant \theta_{m_\theta(p)}$ $(\leqslant \theta)$ of the $m_\theta(p)$ tokens in place p.

With each transition t an interval $[L(t), U(t)]$ is associated (with $0 \leqslant L(t) \leqslant U(t) \leqslant \infty$). Transition

t becomes enabled at the time

$$\max_{p \in {}^\bullet t} \min M_\theta(p) \quad .$$

Then the transition t must fire at some time

$$\theta \in [\max_{p \in {}^\bullet t} \min M_\theta(p) + L(t), \max_{p \in {}^\bullet t} \min M_\theta(p) + U(t)] \tag{2.56}$$

provided the condition enabling t is maintained during the whole interval. In particular if the enabling condition is still valid at the final time

$$\max_{p \in {}^\bullet t} \min M_\theta(p) + U(t)$$

of the firing interval, then the transition is forced to fire at this exact time.

Given the state M_0 of the net at initial time 0, one can describe the evolution of the timed Petri net as follows. One of the transitions t_1, which is enabled at time 0 will fire at some time $\theta \geqslant 0$ within the firing interval

$$[\max_{p \in {}^\bullet t_1} \min M_\theta(p) + L(t_1), \max_{p \in {}^\bullet t_1} \min M_\theta(p) + U(t_1)]$$

of t_1 as specified above. The firing of transition t_1 changes the state of the net as follows: from each place $p \in {}^\bullet\, t_1$ the token $\min M_\theta(p)$ is removed; to each place in $t_1^\bullet$ a token with value the firing time θ is added.

Let us now return to the example of multi-modal transport. The Petri net model of Figure 2.3(a) represents the route taken by a load of type $\alpha \in A$ (each element of A represents one instantiation of the class of Petri net components representing a route), while Figure 2.3(b) is one instantiation of a Petri net model representing availability and location of a vehicle of type x, and Figure 2.3(c) is an instantiation of a Petri net component representing a transshipment station where an item of type $\alpha \in A$ can be unloaded by a vehicle of type $a \in V$ and later picked up by a vehicle of type $b \in V$. Note that if K_x vehicles of type $x \in V$ are present in the system, then the Petri net for the xth instantiation of Petri net 2.3(b) contains K_x tokens. It is also possible that more than one item of type $\alpha \in A$ is present in the system at the same time. In a typical system, there will be constraints on how many items can be present at the same time. For example, the storage capacity of the transshipment station will typically be bounded, leading to a constraint on the set of allowable markings of the form $\sum_{\xi \in A} m(p_{2,\xi}) \leqslant B$.

A particular item to be transported is represented by a token in one place of the Petri net model of the route of that particular type of goods. This place represents the present location or stage reached by the item along its route. The item, represented by the token, can only move to the next place in the net if a token is also available in the corresponding place(s) of the Petri net model of the vehicle to be used, and possibly in the transshipment station model, and if all these enabling tokens arrived at least $L(t)$ time units ago. This synchronization assumption models the interaction between different components in the plant model.

Consider now in detail the evolution over time of the multi-modal transport example. A new item which will follow route α has arrived at time $\theta_{0,\alpha}$ (the token in place $p_{0,\alpha}$) while another item — which arrived earlier — following the same route α has already reached the input buffer $p_{1,\alpha}$ of the transshipment station at some other time $\theta_{1,\alpha}$. Moreover, a vehicle of type a has been available since $\theta_{a,0}$, as indicated by the token in $p_{a,0}$ while the transport system in the transshipment station is available since θ_2. All these arrival times are prior to the present time, but such that no transition has yet been forced to occur. Hence, the following transitions can be executed:

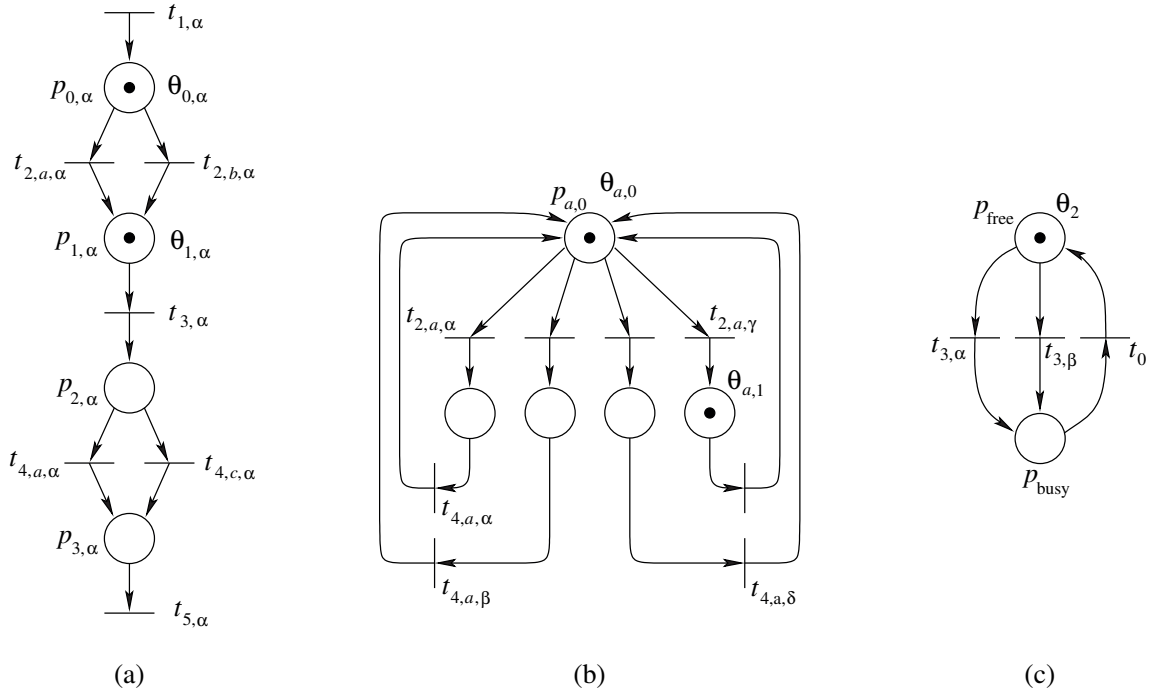

Figure 2.3: Petri net model of a multi-modal transport system.

- either $t_{2,a,\alpha}$ — corresponding to the event "*move item of type α from arrival position to transshipment station*" — will be executed in the interval

$$[\max(\theta_{0,\alpha}, \theta_{a,0}) + L(t_{2,a,\alpha}), \min(\max(\theta_{0,\alpha}, \theta_{a,0}) + U(t_{2,a,\alpha}), \max(\theta_{1,\alpha}, \theta_2) + U(t_{3,\alpha}))],$$

- or $t_{3,\alpha}$ — corresponding to the event "*move item from input buffer of transshipment station to output buffer*" — will be executed in the interval $[\max(\theta_{1,\alpha}, \theta_2) + L(t_{3,\alpha}), \min(\max(\theta_{0,\alpha}, \theta_{a,0}) + U(t_{2,a,\alpha}), \max(\theta_{1,\alpha}, \theta_2) + U(t_{3,\alpha}))]$.

If $t_{2,a,\alpha}$ is executed first, then the next state will be such that $p_{1,\alpha}$ contains 2 tokens (the new token having the value θ at which the event occurred), $p_{0,\alpha}$ and $p_{a,0}$ have no tokens left, and $p_{1,a,\alpha}$ also has one token. If on the other hand $t_{3,\alpha}$ is executed first, then the next state has tokens in places $p_{0,\alpha}$, $p_{2,\alpha}$, $p_{1,a,\alpha}$, and p_{busy} with the token in $p_{2,\alpha}$ having the value θ at which the transition occurred. Both sequences of execution are allowed, unless the upper bound $\min(\max(\theta_{0,\alpha}, \theta_{a,0}) + U(t_{2,a,\alpha}), \max(\theta_{1,\alpha}, \theta_2) + U(t_{3,\alpha}))$ is less than one of the lower bounds. External control actions may dictate the choice between these two options.

For untimed Petri nets there is a vast literature with tools for verifying whether the Petri net model satisfies certain properties and to design suitable controllers [85, 163, 167]. Many of these techniques can be extended to timed Petri nets [34, 66, 126, 201].

2.10 Summary

In this chapter we have discussed several tractable classes of hybrid systems, including PWA, MLD, LC, ELC, MMPS systems, hybrid inclusions, timed automata, and timed Petri nets. We have also discussed

the equivalence between the PWA, MLD, LC, ELC, MMPS systems. In the next chapters we will consider analysis and control of several of these classes of hybrid systems.

Chapter 3

Dynamics and well-posedness

Engineers represent dynamical systems using certain modeling formalisms or description formats (think of differential equations used by control engineers or finite state machines by computer scientists). Such a description format only specifies a collection of trajectories if one provides a notion of solution. Actually the term "solution" already more or less suggests an implicit description format. In computer science terms, one may also say that a definition should be given of what is understood by a *run* (or an *execution*). Formally speaking, description formats are a matter of syntax: they specify what is a well-formed expression. The notion of solution provides semantics: to each well-formed expression it associates a collection of functions of time (called the behavior of the system in [218]).

3.1 Smooth systems: Differential equations

We first start by recalling the solution concept for smooth systems and briefly discuss well-posedness (existence and uniqueness of solutions) in this case. Consider the differential equations

$$\dot{x} = f(t, x) \tag{3.1}$$

with $t \in \mathbb{R}$ and $x \in \mathbb{R}^n$ for the initial time t_0 at the initial state x_0, i.e., $x(t_0) = x_0$.

Definition 3.1.1 A function $x : [t_0, t_1] \to \mathbb{R}^n$ is called a solution to the differential equation (3.1) with initial state x_0 at time t_0, if x is continuous, x is differentiable on (t_0, t_1) (i.e., $\dot{x}(t)$ exists for all $t \in (t_0, t_1)$), $\dot{x}(t) = f(t, x(t))$ for all $t \in (t_0, t_1)$ and $x(t_0) = x_0$.

Note that if f is continuous in t and x, any solution x will be continuously differentiable.

It would be convenient if for each initial time and state, we can find a unique solution trajectory. This would be called *well-posedness* of the system. Since we only consider conditions at the initial time instant, this is called an *initial value problem*. The following theorem gives a sufficient condition for *local* existence and uniqueness of solutions given an initial condition.

Theorem 3.1.2 [137, Theorem 2.2] *Let $f(t, x)$ be piecewise continuous in t and satisfy the following Lipschitz condition: there exist an $L > 0$ and $r > 0$ such that*

$$\|f(t, x) - f(t, y)\| \leqslant L\|x - y\|$$

for all x and y in a neighborhood $B := \{x \in \mathbb{R}^n \mid \|x - x_0\| < r\}$ of x_0 and for all $t \in [t_0, t_1]$. Then there exists a $\delta > 0$ such that the equation

$$\dot{x} = f(t, x) \text{ with } x(t_0) = x_0$$

has a unique solution over $[t_0, t_0 + \delta]$.

Note that if $f : [a, b] \times X \to X$ is continuous in t and continuously differentiable with respect to x for some set $X \subseteq \mathbb{R}^n$ and moreover, assume that for a convex subset $\tilde{X} \subseteq X$ $\|\frac{\partial f}{\partial x}(t, x)\| \leqslant L$ on $[a, b] \times \tilde{X}$, then $\|f(t, x) - f(t, y)\| \leqslant L\|x - y\|$ for all $t \in [a, b]$ and $x, y \in \tilde{X}$. Hence, roughly speaking, continuous differentiability (with respect to x) with bounded derivative implies Lipschitz continuity. See Lemma 2.2 in [137] for a proof of this statement.

Two examples are now in order. The first example will show that if this local Lipschitz condition is not satisfied, the well-posedness (and in particular the uniqueness) will be destroyed.

Example 3.1.3 Consider

$$\dot{x} = 2\sqrt{x} \text{ with } x(0) = 0.$$

It can easily be verified that $x(t) = 0$ and $x(t) = t^2$ are two solutions of this initial value problem. This means that the conditions of Theorem 3.1.2 must be violated. Indeed, the vector field $f(x) = 2\sqrt{x}$ is not locally Lipschitz at the origin. The infinitely large derivative in zero makes the occurrence of a second trajectory next to the zero trajectory possible.

Hence, dropping the local Lipschitz condition might cause problems. Another issue is the fact that we have only *local* existence of trajectories instead of *global* existence on an interval of the type $[t_0, \infty)$. The reason for this is the occurrence of so-called *finite escape times* as demonstrated by the following example.

Example 3.1.4 The system

$$\dot{x} = x^2 + 1 \text{ with } x(0) = 0$$

has as solution $x(t) = \tan t$ which is only locally defined on the interval $[0, \pi/2)$ for its argument t. Note that we have $\lim_{t \uparrow \pi/2} x(t) = \infty$ and hence, there is a finite time at which the trajectory escapes to infinity! The reason for this is the fast (superlinear) growth of $f(x)$.

Actually, if we have some kind of *global Lipschitz continuity*, finite escape times cannot occur.

Theorem 3.1.5 [137, Theorem 2.3] *Suppose $f(t, x)$ is piecewise continuous in t and satisfies*

$$\|f(t, x) - f(t, y)\| \leqslant L\|x - y\|$$

for all x and y in $\mathbb{R}^n$ and for all $t \in [t_0, t_1]$. Then, the equation

$$\dot{x} = f(t, x) \text{ with } x(t_0) = x_0$$

has a unique solution over $[t_0, t_1]$.

Note that linear systems with $f(t, x) = Ax + Bu(t)$ always satisfy the conditions of Theorem 3.1.5 for all arbitrary piecewise continuous inputs u. Another remark is that the global Lipschitz conditions

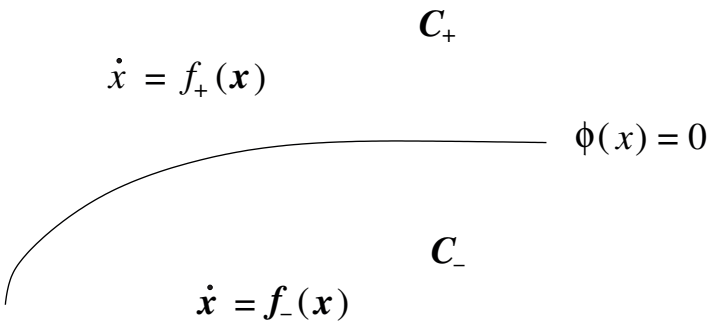

Figure 3.1: Discontinuous dynamical system with 2 subsystems.

are not necessary for global existence and uniqueness of trajectories. Indeed, the example given in [137] with $\dot{x} = -x^3$ is not globally Lipschitz, but has for each initial condition a unique global solution.

Of course, these "smooth phenomena" can also show up in hybrid systems, but additionally it can be even worse as we have already seen some "funny phenomena" that can happen in hybrid model descriptions like the various kinds of Zeno behavior in the examples of Chapter 1. These phenomena arise typically due to the interaction of continuous and discrete dynamics. To define the trajectories of these mixed systems, we have to take this into account in some way or another. Before we precisely define solution concepts we first say something about another phenomenon that is regularly encountered in hybrid systems for which the terms "infinitely fast switching", "chattering" and "sliding modes" pop up in the literature.

3.2 Discontinuous differential equations: A class of switched systems

In this section, we recall the discontinuous dynamical systems as mentioned already in Section 1.5.1.

Consider a system that switches between two dynamics as a result of inequalities:

$$\dot{x} = f(x) = \begin{cases} f_-(x), & \text{when } \phi(x) < 0 \\ f_+(x), & \text{when } \phi(x) > 0, \end{cases} \tag{3.2}$$

where $x(t) \in \mathbb{R}^n$ denotes the state variable at time $t \in \mathbb{R}$. This system has already been graphically illustrated in Figure 1.5, but is repeated here in Figure 3.1. The state space is separated into two parts by a hyper-surface defined by $\phi(x) = 0$. On one side of the surface $C_+ := \{x \in \mathbb{R}^n \mid \phi(x) > 0\}$ the dynamics $\dot{x} = f_+(x)$ holds, on the opposite side $C_- := \{x \in \mathbb{R}^n \mid \phi(x) < 0\}$ the dynamics $\dot{x} = f_-(x)$ is valid. Hence, one can also see this system as a *differential equation with a discontinuous right-hand side*. Note that this implies that Lipschitz continuity of the vector field f is lost and consequently, the definition of solutions and well-posedness results of the previous section do not hold. Let us study the behavior of such a system in more detail.

As long as you are in the interior of either C_- or C_+ there is no ambiguity in how the system's state is evolving. If the state x is at the switching plane (i.e., $\phi(x) = 0$) and both $f_-(x)$ and $f_+(x)$ point in the same direction with respect to the switching plane (either both towards C_- or both towards C_+), then the state follows this direction and gets you into the interior of C_- or C_+ and the evolution is clear again. This case can be mathematically characterized as follows. Let $n(x)$ be the normal of ϕ at x that is

pointing in the direction of C_+, i.e., take $n(x) = \frac{\nabla\phi(x)}{\|\nabla\phi(x)\|}$, where $\nabla\phi = \frac{\partial\phi(x)}{\partial x}$ is the gradient of ϕ at point x. The function ϕ is assumed to be continuously differentiable. The above situation is characterized by $(n(x)^{\mathrm{T}} f_-(x)) \cdot (n(x)^{\mathrm{T}} f_+(x)) > 0$. In case $(n(x)^{\mathrm{T}} f_-(x)) \cdot (n(x)^{\mathrm{T}} f_+(x)) < 0$, there are two options:

- $n(x)^{\mathrm{T}} f_+(x) > 0$ ($f_+(x)$ points towards C_+) and $n(x)^{\mathrm{T}} f_-(x) < 0$ ($f_-(x)$ points towards C_-): for an x at the switching plane satisfying these conditions, there are (at least) two trajectories to follow. One going into C_- according to dynamics $\dot{x} = f_-(x)$ and one into C_+ following $\dot{x} = f_+(x)$.
- $n(x)^{\mathrm{T}} f_+(x) < 0$ ($f_+(x)$ points towards C_- and hence, is steering the state away from C_+) and $n(x)^{\mathrm{T}} f_-(x) > 0$ ($f_-(x)$ points towards C_+ and consequently, is steering the state away from C_-). See Figure 3.2. Hence, from the state x it is impossible to go to C_- or C_+, because the dynamics immediately steers you back to the hyper-surface given by $\phi(x) = 0$. Hence, there is no solution according to one of the dynamics $\dot{x} = f_-(x)$ or $\dot{x} = f_+(x)$.

Hence, if one would allow that the state evolves only according to one of the dynamics $\dot{x} = f_-(x)$ or $\dot{x} = f_+(x)$, then in the first case one would have two solutions starting at the initial state x (non-uniqueness of solutions) and in the second case one would have non-existence of solutions.

However, a generalization of the solution concept is possible, which can be inspired by some kind of *relaxation* of the system. In case the switching between the two dynamics is not instantaneous, but there is a small time delay or a type of hysteresis switching with a small Δ as, e.g., in the example of hysteresis in Chapter 1, you would observe fast switching in the second case above. The motion of the system would be close to the switching plane. If the time delay goes to zero, or the Δ to zero this would become infinitely fast switching (chattering) and in the limit case one obtains a sliding motion along the separating surface, for which several formalizations exist. Depending on the relaxation one observes a different behavior on the sliding surface.

One of the limit cases is due to Filippov [93] and is called the *convex definition*. In brief, it states that the sliding mode is given by taking a convex combination of both dynamics $\dot{x} = \lambda f_+(x) + (1-\lambda) f_-(x)$, $0 \leqslant \lambda \leqslant 1$ such that x moves ("slides") along $\phi(x) = 0$.

Since we assigned multiple values to the right-hand side of (3.2) for $\phi(x) = 0$, we arrive in the realm of *differential inclusions* [16], which are systems of the form $\dot{x} \in F(x)$, where $F(x)$ is a set-valued mapping. Indeed, if we define $F(x)$ as

$$F(x) = \begin{cases} \{f_+(x)\}, & \text{for } \phi(x) > 0, \\ \{\lambda f_+(x) + (1-\lambda) f_-(x) \mid \lambda \in [0,1]\}, & \text{for } \phi(x) = 0, \\ \{f_-(x)\}, & \text{for } \phi(x) < 0. \end{cases}$$

we obtain the extended system for (3.2). We assume that f_- and f_+ are sufficiently smooth functions defined on $\{x \mid \phi(x) \geqslant 0\}$ and $\{x \mid \phi(x) \leqslant 0\}$, respectively. As now a *differential inclusion* has been obtained, the corresponding solution concepts can be applied [16].

Definition 3.2.1 A function $x : [a,b] \to \mathbb{R}^n$ is a *solution* of $\dot{x} \in F(x)$, if x is absolutely continuous[1] and satisfies $\dot{x}(t) \in F(x(t))$ for almost all $t \in [a,b]$.

[1] A function $x : [a,b] \to \mathbb{R}^n$ is called absolutely continuous, if x is continuous and can be written as the integral of its derivative, i.e., $x(t) = x(a) + \int_a^t \dot{x}(\tau) d\tau$ for all $t \in [a,b]$ for a function $\dot{x}$ (its derivative) in $L_{1,\mathrm{loc}}[a,b]$, the set of locally integrable functions. This implies that x is almost everywhere differentiable.

Note that with this definition, we can follow in the case that $(n(x)^{\mathrm{T}} f_-(x)) \cdot (n(x)^{\mathrm{T}} f_+(x)) < 0$ either 1 (when $n(x)^{\mathrm{T}} f_-(x) > 0$ and $n(x)^{\mathrm{T}} f_+(x) < 0$) or 3 modes (when $n(x)^{\mathrm{T}} f_-(x) < 0$ and $n(x)^{\mathrm{T}} f_+(x) > 0$), which all produce a solution. So, we have either one or multiple solutions. Note that in the latter case the state can leave the switching surface any time (just follow the switching surface for a while and the trajectory can follow either $\dot{x} = f_+(x)$ or $\dot{x} = f_-(x)$ at an arbitrary point). Consider Example 3.4.1 below for a linear relay system, where this situation occurs and consequently, multiple solutions originate from one initial condition.

Instead of the Filippov view on this system, one might also consider a more "modern" view and take a hybrid automaton point of view (see Section 1.5.2). In the sense of hybrid automata, we have added an additional mode (the sliding mode) to the original system (3.2). As a consequence, the system has now *three* modes instead of the original two.

mode 1 Dynamics: $\dot{x} = f_+(x)$ and $\lambda = 1$. Invariant: $\phi(x) \geqslant 0$

mode 2 Dynamics: $\dot{x} = f_-(x)$ and $\lambda = 0$. Invariant: $\phi(x) \leqslant 0$

mode 3 Dynamics: $\dot{x} = \lambda f_+(x) + (1-\lambda) f_-(x)$ and $\phi(x) = 0$. Invariant: $0 \leqslant \lambda \leqslant 1$

Note that the dynamics do not only consist of differential equations, but contain also algebraic equations (e.g., $\phi(x) = 0$). So, the mode dynamics are sometimes specified by Differential and Algebraic Equations (DAEs) [44]. One can solve these DAEs explicitly by assuming that if the evolution of the system will be in the sliding mode on some non-trivial interval $[t_0, t_1]$, then we must have $\phi(x(t)) = 0$ for $t \in [t_0, t_1]$ and consequently, $\frac{d\phi(x(t))}{dt} = 0$ for $t \in (t_0, t_1)$. In case ϕ is differentiable this yields $\frac{\partial \phi(x(t))}{\partial x} \dot{x}(t) = \frac{\partial \phi(x(t))}{\partial x} [\lambda f_+(x) + (1-\lambda) f_-(x)] = 0$ from which λ can be solved. Substituting this expression for λ as a function of x in $\dot{x} = \lambda f_+(x) + (1-\lambda) f_-(x)$ and $\phi(x) = 0$ gives the sliding mode dynamics. The example below will illustrate this computation.

Let us first consider an example with sliding behavior.

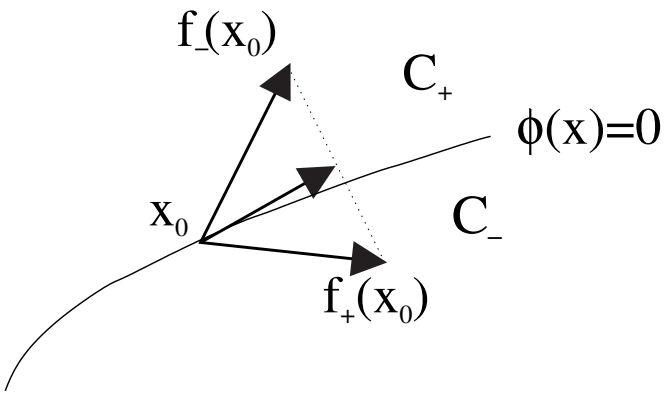

Figure 3.2: Sliding mode.

Example 3.2.2 Let the state space $\mathcal{X}$ be $\mathbb{R}^2$ and $\phi(x) = x_2$, $f_+(x) = (x_1^2, -x_1 + \frac{1}{2}x_1^2)^{\mathrm{T}}$ and $f_-(x) = (1, x_1^2)^{\mathrm{T}}$. If we consider initial state $x_0 = (1,0)^{\mathrm{T}}$, we observe that $f_+(x_0) = (1, -\frac{1}{2})^{\mathrm{T}}$ and $f_-(x_0) = (1,1)^{\mathrm{T}}$. Hence, we have to find a convex combination of both dynamics as described above such that $\phi(x) = 0$. Since $\phi(x_0) = 0$, it is sufficient to keep the time derivative of $\phi(x)$ equal to zero along the

trajectory: $\frac{d\phi}{dt}(x(t)) = \left[\frac{d\phi}{dx}(x(t))\right]^{\mathrm{T}} \dot{x}(t) = \dot{x}_2(t) = \lambda(-x_1 + \frac{1}{2}x_1^2) + (1-\lambda)x_1^2 = 0$. Hence, we have to take $\lambda(x) = \frac{x_1}{\frac{1}{2}x_1+1}$. The sliding mode is valid as long as the corresponding $\lambda(x)$ lies in the interval $[0,1]$. Hence, in terms of hybrid automata, the "invariant" of the sliding mode is $0 \leqslant x_1 \leqslant 2$. So, when x_1 reaches the value 2 the mode switches again and the trajectory will move into C_+. Note that the system behaves at the sliding mode as

$$\dot{x}_1 = \frac{2x_1^3 - x_1 + 2}{x_1 + 2} \tag{3.3}$$

with $x_2 = 0$ and which is valid as long as $0 \leqslant x_1 \leqslant 2$. A solution to this system from initial state $(0, -\frac{1}{3})$ is depicted in Figure 3.3 where at time instant $t = 1$ the surface $x_2 = 0$ is hit in $(1,0)^{\mathrm{T}}$. It slides along this surface till time $t_{\mathrm{leave}} \approx 1.531$ ($x_1(t_{\mathrm{leave}}) = 2$) and then leaves the surface again to C_+.

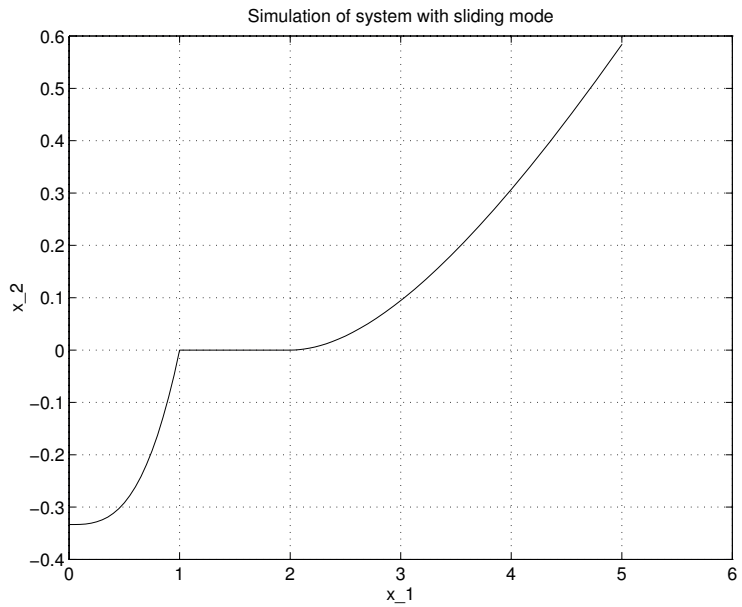

Figure 3.3: Simulation of a system with sliding modes.

An alternative definition for the sliding behavior mentioned in [94] is based on the *equivalent control definition* of sliding modes, which is essentially due to Utkin [205]. This definition is related to "switching control systems." The system given by

$$\dot{x} = f(x,u) \tag{3.4}$$

with x the state variable is controlled by the discontinuous feedback

$$u = \begin{cases} g_+(x), & \xi(x) > 0 \\ g_-(x), & \xi(x) < 0 \end{cases} \tag{3.5}$$

with the function $\xi : \mathbb{R}^n \to \mathbb{R}$ modeling the switching surface. Similar to the previous subsection, a sliding mode occurs when the dynamics $f_+(x) := f(x, g_+(x))$ and $f_-(x) := f(x, g_-(x))$ both point outside C_+ and C_-, respectively. The equivalent control definition of a sliding mode picks a convex

combination of the control laws (!) instead of a convex combination of $f_+(x)$ and $f_-(x)$ (note that the definitions are equivalent when $f(x,u)$ is affine in u, i.e., $f(x,u)$ is of the form $k(x)+l(x)u$). We define the set of convex combinations of $g_+(x)$ and $g_-(x)$ by $U(x) := \{\lambda g_+(x) + (1-\lambda)g_-(x) \mid \lambda \in [0,1]\}$. The sliding mode is given by the differential and algebraic equations $\dot{x} = f(x, \lambda g_+(x) + (1-\lambda)g_-(x))$, $\xi(x) = 0$ with "equivalent control" $u_{\text{equiv}} = \lambda g_+(x) + (1-\lambda)g_-(x)$ and valid as long as $\lambda \in [0,1]$ is satisfied. One can solve for λ as in the convex definition case. Again a differential inclusion can be obtained as

$$\dot{x} \in F(x)$$

with

$$F(x) = \begin{cases} \{f(x, g_+(x))\}, & \text{for } \xi(x) > 0, \\ f(x, U(x)) = \{f(x,u) \mid u \in U(x)\}, & \text{for } \xi(x) = 0, \\ \{f(x, g_-(x))\}, & \text{for } \xi(x) < 0. \end{cases} \tag{3.6}$$

The definition of solutions for differential inclusions (3.2.1) can be applied to (3.6) to obtain solutions for (3.4)-(3.5).

We restricted ourselves above to the simple case of two original dynamics (f_- and f_+) or a controller switching originally between two control laws (g_- and g_+). Filippov [94] presents a much more general setting (including multiple > 2 "modes"). E.g., he considers equations of the form $\dot{x} = f(x)$ in which f is a piecewise continuous function. A function f is called piecewise continuous on a bounded domain G, if G consists of a finite number of domains G_i on which f is continuous up to the boundary and a set M of measure zero which consists of the boundary points of these domains. In other words, $G_i \cap G_j = \varnothing$ and $M \cup \bigcup_i G_i = G$ and f is continuous up to its domain when restricted to G_i. Note that for (3.2) we have $G_1 = C_+ := \{x \mid \phi(x) > 0\}$, $G_2 = C_- := \{x \mid \phi(x) < 0\}$ and $M = \{x \mid \phi(x) = 0\}$. We call a function f continuous up to the boundary of G_i, when all x in the closure of G_i (i.e., including the boundary) there exists a value $\alpha_i(x)$ such that any converging sequence $\{x_k\}_{k=0}^{\infty}$, with $x_k \in G_i$ for all k and $x_k \to x$, it holds that $\lim_{k\to+\infty} f(x_k) = \alpha_i(x)$. Typically, for x in the interior of G_i $\alpha_i(x) = f(x)$. On the boundary of G_i, the value can be different from $f(x)$. Hence, this limit may be different depending on the region G_i from which x is approached. If G is unbounded, then f is called piecewise continuous on G, if for any bounded subset $\tilde{G}$ of G, it holds that f is piecewise continuous on $\tilde{G}$. For this setting, the differential inclusion that replaces $\dot{x} = f(x)$ by using Filippov's convex definition is given by $\dot{x} \in F(x)$ with $F(x)$ equal to the smallest convex closed set containing all limit values of the function f in the point x (all existing limit values $\lim_{k\to+\infty} f(x_k)$ for arbitrary sequences $\{x_k\}_{k\in\mathbb{N}}$ with $x_k \to x$, when $k \to +\infty$). Hence, the solution concept can then be inherited again from the field of differential inclusions. For more details (e.g., including time-varying systems $\dot{x} = f(t,x)$), see [94].

The Filippov or Utkin solutions do not only have interpretations in the sense of the idealized limit cases of fast switching, but also have a physical interpretation as in the mechanical example with friction below.

Example 3.2.3 The control equivalent solution concepts have strong links to the Coulomb type of dry friction. Indeed, consider the simple example, where a block with mass M is driven by a force $F_{\text{d}}(t)$ over a surface. The position of the block at time t is $q(t)$ and its velocity is $v(t) = \dot{q}(t)$. The friction $F(t)$ between the block and the surface is given by Figure 3.4, i.e., $F(t) \in F_{\text{c}} \operatorname{sgn}(-\dot{q}(t))$. The equations of motion are then given as

$$M\ddot{q}(t) \quad = \quad F_{\text{d}}(t) + F(t) \tag{3.7a}$$

$$F(t) \quad \in \quad F_{\mathrm{c}}\,\mathrm{sgn}(-\dot{q}(t)), \tag{3.7b}$$

where sgn is defined as in (1.10). Here, we see that the "sliding mode" coincides with the control equivalent definition of the other mode dynamics corresponding to $\dot{q} > 0$ and $\dot{q} < 0$. For this system, the sliding mode has a physical interpretation as the stick phase of the friction: as long as the block is at rest (i.e., $\dot{q} = 0$) and the driving force $F_{\mathrm{d}}(t)$ does not exceed the maximal friction force F_{c} the block stays at rest. The friction force $F(t)$ is the "equivalent control" that keeps the block at rest ($\dot{q} = 0$).

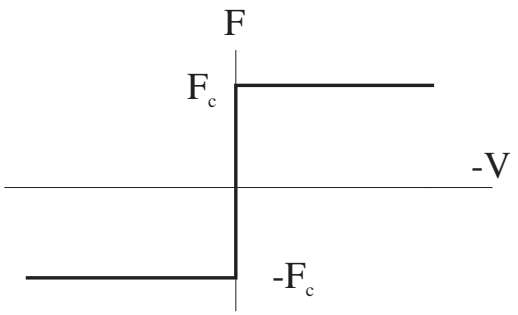

Figure 3.4: Coulomb friction.

3.3 Event times

Let us first return to the example of the two cart system of Section 1.6.4, which can be described by the complementarity model

$$\begin{aligned}
\dot{x}_1(t) &= x_3(t),\\
\dot{x}_2(t) &= x_4(t),\\
\dot{x}_3(t) &= -2x_1(t) + x_2(t) + z(t),\\
\dot{x}_4(t) &= x_1(t) - x_2(t),\\
w(t) &:= x_1(t).
\end{aligned} \tag{3.8}$$

with

$$0 \leqslant z(t) \perp w(t) \geqslant 0. \tag{3.9}$$

An informal description has already been given in Chapter 1 and a possible trajectory is given in Figure 3.5 for initial state

$$x_0 = e^{-A}(0\ -1\ -1\ 0)^{\mathrm{T}} \approx (0.3202, -0.4335, 0.3716, -1.0915)^{\mathrm{T}}$$

on the interval $[0, 3.5]$

Note the complementarity between z (denoted by "u" in the figure) and $w = x_1$ and the discontinuity in the derivative of x_1 at time $t = 1$.

Discrete actions (mode transitions and resets) occur at the *event times* 0 (as the initial time), 1 and $1 + \frac{\pi}{2}$. Hence, the *event times set* $\mathcal{E}$ is $\{0, 1, 1 + \frac{\pi}{2}\}$. To give a formal definition about which event times sets are allowed, we state the following:

Definition 3.3.1 A set $\mathcal{E} \subset \mathbb{R}^+$ is called an *admissible event times set*, if it is closed and countable, and $0 \in \mathcal{E}$.

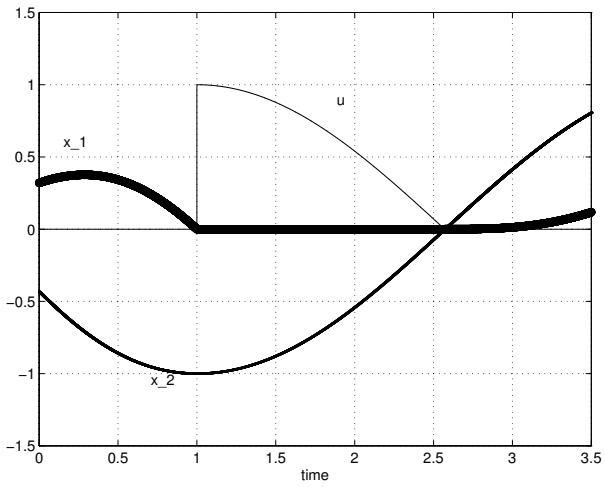

Figure 3.5: Solution trajectory of two-carts system.

Note that both so-called left and right accumulations of event times are allowed by the above definition.

Definition 3.3.2 An element t of a set $\mathcal{E}$ is said to be a *left accumulation point* of $\mathcal{E}$, if for all $t' > t$ $(t, t') \cap \mathcal{E}$ is not empty. It is called a *right accumulation point*, if for all $t' < t$ $(t', t) \cap \mathcal{E}$ is not empty.

We categorize the event times set as follows:

Definition 3.3.3 An admissible event times set $\mathcal{E}$ (or the corresponding solution) is said to be *left (right) Zeno free*, if it does not contain any left (right) accumulation points.

Consider, e.g., the bouncing ball example of Section 1.7.1, which has the event times set $\mathcal{E} = \{\tau_i \mid i \in \mathbb{N}\} \cup \{\tau^*\}$ with $\tau_0 = 0$ and

$$\tau_i = \frac{2e^i}{g}, i = 1, 2, \ldots$$

and $\tau^* = \frac{2}{g-ge}$ for $x(0) = 0$, $\dot{x}(0) = 1$ and the trajectory staying at rest after τ^* (i.e., $x(t) = 0$ for $t \geqslant \tau^*$). Here $\mathcal{E}$ has a right accumulation point. Hence, this one is not right Zeno free.

There exist also models with left accumulation points. One example can be "artificially" obtained by time-reversing the bouncing ball model, which leads to trajectories of the type as depicted in Figure 3.6.

Another model with solutions having left accumulation points is the following example due to Filippov, who used it to show that this phenomenon might cause non-uniqueness of trajectories (from the origin).

Example 3.3.4 The time-reverse of (1.9) in Chapter 1 (which is the original example in [94]) given by

$$\dot{x}_1 = \text{sgn}(x_1) - 2\text{sgn}(x_2) \tag{3.10a}$$
$$\dot{x}_2 = 2\text{sgn}(x_1) + \text{sgn}(x_2), \tag{3.10b}$$

has the zero trajectory starting in the origin. For a trajectory in the phase plane, see Figure 1.16. From this it can be seen that there are (infinitely many) solutions corresponding to initial state $x_0 = 0$ starting up with a left accumulation point. A sample trajectory is given in Figure 3.7. Hence, this example

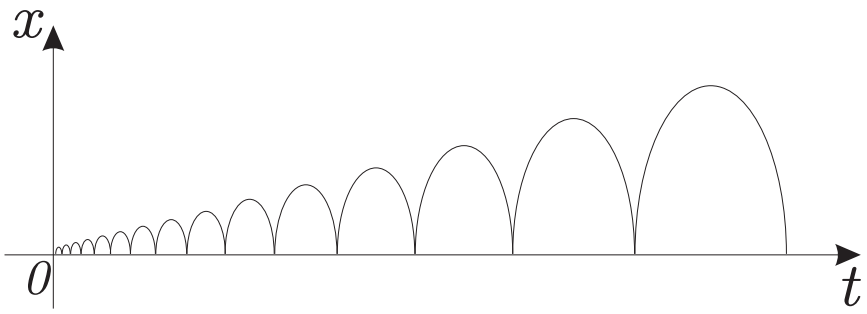

Figure 3.6: A time reversed "bouncing ball."

demonstrates nicely that if we consider a solution concept that only allows left Zeno free solutions, then we have existence and uniqueness of solutions given an initial condition (i.e., well-posedness). However, if left Zeno behavior is included (e.g., if we use the solution concept corresponding to differential inclusions as given in Def. 3.2.1), then uniqueness is lost. Stated otherwise, uniqueness is lost due to generalizing the solution concept.

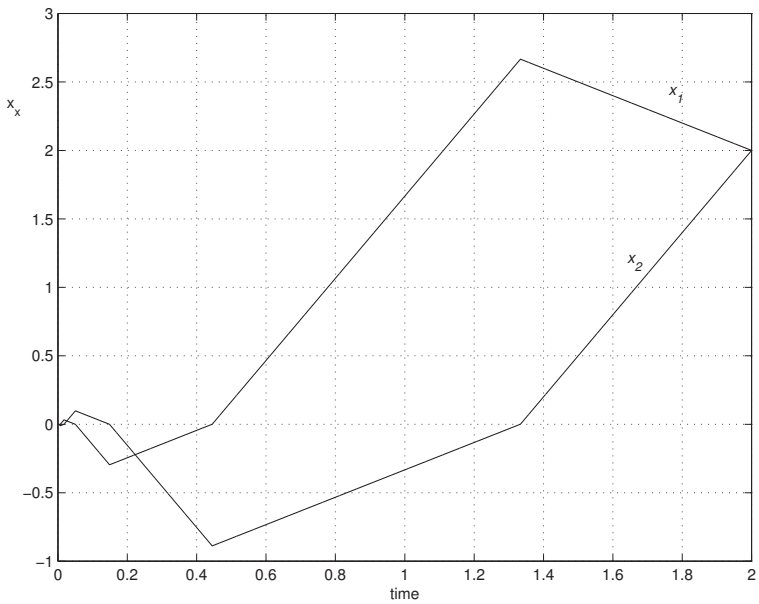

Figure 3.7: A sample trajectory of Filippov's example.

3.3.1 Relevance of choice of solution concept

The system (3.10) has made clear that the choice of allowable event times set has a major impact on the well-posedness issue. Moreover, in case of verification of certain system property, it is crucial which type of trajectories you include in the model of the plant (and hence, in the analysis). Definitely the mathematical behavior of the model should be "rich enough" to reflect the real plant or system. What one often sees, is that, e.g., Zeno solutions are excluded in the analysis of system's property. In this case one has to realize that the property holds only for "non-Zeno" trajectories, while the actual system

might have solutions beyond Zeno points and the system might fail in practice. A clear example of this is the water tank example of Chapter 1. Suppose that this system was the outcome of the design problem to realize a controller for the filling strategy in such a way that the levels of both tanks stay above certain thresholds. If one considers only non-Zeno solutions in the analysis, both tanks stay full and then it seems that one has found a "feasible solution" to the problem, which is of course nonsense as both tanks will drain after the accumulation point. Hence, one has to be careful with excluding these type of phenomena or stated differently, with the choice of solution concept.

A major problem is played by the Zeno phenomenon in hybrid systems and it is one of the challenges that the system and control theory for hybrid systems has to face. Defining solutions in a systematic way related to reality for, e.g., hybrid automata is a hard and maybe impossible task. Depending on the underlying real system of the model, the "limit case" or "continuation" might differ just as for the formalization of sliding modes.

3.3.2 Evolutions of hybrid automata

We will start by presenting a solution concept for hybrid automata, that is based on the concept of "hybrid time trajectories."

Definition 3.3.5 [155] A hybrid time trajectory $\tau = \{I_i\}_{i=0}^{N}$ is a finite ($N < \infty$) or infinite ($N = \infty$) sequence of intervals of the real line, such that

- $I_i = [\tau_i, \tau_i']$ with $\tau_i \leqslant \tau_i' = \tau_{i+1}$ for $0 \leqslant i < N$;
- if $N < \infty$, either $I_N = [\tau_N, \tau_N']$ with $\tau_N \leqslant \tau_N' \neq \infty$, or $I_N = [\tau_N, \tau_N')$ with $\tau_N \leqslant \tau_N' \leqslant \infty$.

The event times set corresponding to a hybrid time trajectory as defined above is given by $\{\tau_i \mid i = 0, \ldots, N-1\}$ in case N is finite and $\{\tau_i \mid i = 0, 1, \ldots\}$ in case $N = \infty$.

Definition 3.3.6 An execution χ of a hybrid automaton is a collection $\chi = (\tau, q, x)$ with

- τ a hybrid time trajectory;
- $q = \{q_i\}_{i=0}^{N}$ with $q_i : I_i \to Q$; and
- $x = \{x_i\}_{i=0}^{N}$ with $x_i : I_i \to X$

satisfying

Initial condition $(q_0(\tau_0), x_0(\tau_0)) \in \mathrm{Init}$;

Continuous evolution for all i

- q_i is constant, i.e., $q_i(t) = q_i(\tau_i)$ for all $t \in I_i$;
- x_i is the solution to the differential equation $\dot{x} = f(q_i(t), x(t))$ on the interval I_i with initial condition $x_i(\tau_i)$ at τ_i;
- for all $t \in [\tau_i, \tau_i')$ it holds that $x_i(t) \in \mathrm{Inv}(q_i(t))$.

Discrete evolution for all i, $e = (q_i(\tau_i'), q_{i+1}(\tau_{i+1})) \in E$, $x_i(\tau_i') \in G(e)$ and $(x_i(\tau_i'), x_{i+1}(\tau_{i+1})) \in R(e)$.

Note that a hybrid time trajectory does not allow left accumulation points in its event times set. Hence, for Filippov's example (3.10) one would only get the zero trajectory starting in the origin. The hybrid time trajectory corresponds to a well-ordered event set with respect to the usual (increasing) ordering of the real line.

For general hybrid automata it is in principle impossible to define a solution concept beyond a right accumulation point of event times or a point where live-lock occurs (an infinite number of events at one time instant). In [129] one mentions three possibilities of extension of Zeno executions:

- regularization: modifying the original Zeno automaton by adding temporal (i.e., time delay) or spatial regularization (i.e., a kind of hysteresis) parameters
- averaging: averaging the vector field close to the Zeno time
- Filippov relaxation, which only applies in particular cases of differential equations with discontinuous right-hand sides, as we have seen earlier.

In principle, in the latter case we are dealing with a situation in which the mode (discrete state) is determined fully by the continuous state. Actually, this is also true for solution concepts with no explicit mention of the discrete state or an application to classes like discontinuous differential equations, complementarity systems, or multi-modal linear systems as one can continue by taking the left limit of the state – provided it exists – and define a continuation from the left limit state (as if it was a new initial state). Note that we did this for the bouncing ball, Filippov's example and the three balls example of Chapter 1. In this manner, a 'natural relaxation' can be obtained.

3.3.3 Complementarity systems

Systems as depicted in Figure 3.1 are sometimes known as *variable-structure systems*; they describe a type of mode-switching. A similar mode-switching behavior is obtained from a class of systems known as *complementarity systems* [117,211]. Equations for a complementarity system may be written in terms of a state variable x and auxiliary variables w and z, which must be vectors of the same length, as:

$$\dot{x} \;=\; f(x,z) \tag{3.11a}$$
$$w \;=\; h(x,z) \tag{3.11b}$$
$$0 \leqslant w \perp z \geqslant 0, \tag{3.11c}$$

where the last line means that the components of the auxiliary variables w and z should be nonnegative, and that for each index i and for each time t at least one of the two variables $w_i(t)$ and $z_i(t)$ should be equal to 0. The description (3.11) is in principle implicit in the discrete variables or modes and this has advantages since it gives the possibility to define solutions beyond accumulation points by taking suitable limits of the continuous state variable. However, the system (3.11) can be considered to consist of a number of different dynamical systems or "modes" that are glued together. The modes can be thought of as discrete states. They correspond to a fixed choice, for each of the indices i, between the two possibilities $w_i \geqslant 0,\ z_i = 0$ and $w_i = 0,\ z_i \geqslant 0$, so that a complementarity system in which the vectors w and z have length m has 2^m different modes (all parameterized by an index set $I \subseteq \{1,\ldots,m\}$ such that $w_i = 0,\ z_i \geqslant 0$ holds if $i \in I$ and $w_i \geqslant 0,\ z_i = 0$ as $i \notin I$. The specification (3.11) is in general not complete yet; one has to add a rule that describes possible jumps of the state variable x when a transition from one mode to another takes place (think of mechanical systems with impacts). In the

example of the two carts we have chosen inelastic impacts, while the bouncing ball had an elastic impact with restitution coefficient e.

A complementarity system can in principle be rewritten in the explicit hybrid automaton format, but the representation that is obtained may be awkward (see [116] or the example of the Boost circuit in Chapter 1). However, the fact that they form a subclass of hybrid automata means that we can borrow their solution concept as was formalized in the previous section. The only additional feature is that in principle the "hybrid state" only consists of the continuous state x and the discrete state is determined by x implicitly. This gives the advantage that at a right accumulation point τ^*, we can define the trajectory beyond τ^*, if $x^* := \lim_{t\uparrow\tau^*} x(t)$ exists and from initial state x^* a continuation is possible (like in Filippov's example) as we do not need information about the discrete mode, since this follows from x^*. For a hybrid automaton, one would have to select a mode in some unclear way. Similarly, we can define trajectories beyond live-lock points, by taking also a limit of the sequence of states at this time instant. Since the executions of hybrid automata do not include left accumulations of event times, and the trajectories for complementarity systems are inherited from the executions, we call them *left Zeno free solutions or forward solutions*. A fully worked out definition for linear complementarity systems can be found in [117].

Alternatively, one might define solutions of complementarity systems as follows.

Definition 3.3.7 A triple $(z, x, w) \in L_2^{m+n+m}$ is said to be an L_2*-solution* of (3.11) on the interval $[0, T]$ with initial condition x_0 if for almost all $t \in [0, T]$ the following conditions hold:

$$\begin{aligned} x(t) &= x_0 + \int_0^t f(x(s), z(s))\, ds \\ w(t) &= h(x(t), z(t)) \\ 0 &\leqslant z(t) \perp w(t) \geqslant 0. \end{aligned}$$

This definition is in the spirit of the one given above for differential inclusions like Filippov's solutions ("convex definition") or Utkin's solutions ("control equivalent definition").

3.4 Well-posedness

Consider the following example.

Example 3.4.1 Let the linear system

$$\begin{aligned} \dot{x}_1 &= -z \\ w &= x_1 \end{aligned}$$

be coupled to a negative relay or negative sign characteristic

$$z = -\operatorname{sgn} w. \tag{3.12}$$

It can be verified that with initial condition $x_0 = 0$ we have the following three left Zeno free solutions (all corresponding to one of the three modes):

- $x_1(t) = w(t) = t$ and $z(t) = -1$;
- $x_1(t) = w(t) = -t$ and $z(t) = 1$;
- $x_1(t) = w(t) = z(t) = 0$

In fact, this linear relay system has from the initial state $x_0 = 0$ an infinite number of solutions. Indeed, any trajectory defined as

$$x_1(t) = w(t) = z(t) = 0 \text{ for } t \in [0, t_0], \text{ and}$$
$$x_1(t) = w(t) = t,\ z(t) = -1 \text{ for } t \in [t_0, \infty)$$

is a solution of the linear relay system for initial condition $x(0) = 0$ as well.

Hence, non-uniqueness can easily occur, just as non-existence of solutions of hybrid systems given an initial condition. As seen in the system (3.10) the Zeno effect can even make the question of well-posedness more involved.

Several forms of well-posedness can be distinguished. In Section 3.3.1 we have already discussed how the existence and uniqueness issue is influenced by the choice of the solution concept. Allowing left Zeno solutions might cause non-uniqueness as was demonstrated by Filippov's example. Similarly, excluding right Zeno solutions might yield problems with *global* existence of solutions as we cannot define the trajectory beyond the finite accumulation point. Of course, the broader the solution class, the earlier the existence of trajectories is true and the faster the uniqueness is destroyed.

One may also distinguish between various notions of well-posedness on the basis of the time interval that is involved. E.g., in the context of hybrid automata, one may say that a given automaton is *non-blocking* [129] if for each initial condition either at least one transition is enabled or the continuous dynamics $f(\cdot, q)$ corresponding to some mode q can be followed during an interval of positive length. If the continuation exists and is unique (the automaton is *deterministic* [129]), one may say that the automaton is *initially well-posed*. However, the definition of initial well-posedness allows a situation in which a transition from mode 1 to mode 2 is immediately followed by a transition back to mode 1 and so on in an infinite loop, so that all $\tau_i' - \tau_i$ in the definition of a run of the hybrid automaton are equal to zero (livelock). A stronger notion is obtained by requiring that a solution exists at least on an interval $[0, \varepsilon)$ with $\varepsilon > 0$; system descriptions for which such solutions exist and are unique are called *locally well-posed*. In computer science terminology, such systems "allow time to progress". Finally, if solutions exist and are unique on the whole half-line $[0, \infty)$, then one speaks of *global well-posedness*.

The concepts of non-blocking and deterministic for hybrid automata are formalized in Section 3.4.2.

In the order we presented the solution concepts for the various hybrid model classes, we will also present the well-posedness results.

3.4.1 Well-posedness for switched systems

For systems of the type (3.2) (or the more general systems discussed briefly at the end of Section 3.2) the existence of "Filippov" or "Utkin" solutions is guaranteed. Indeed, Filippov [94] proved the following existence result.

Theorem 3.4.2 *Consider systems of the form* (3.2) *using Filippov's convex definition or of the form* (3.4)-(3.5) *using Utkin's equivalent control definition to obtain the differential inclusion* $\dot{x} \in F(x)$.

Then for any initial state x_0 there exists a solution x to $\dot{x} \in F(x)$ with $x(0) = x_0$. If F is defined on a ball around x_0 with radius b (i.e., on $\mathcal{B} := \{x \in \mathbb{R}^n \mid \|x - x_0\| \leqslant b\}$), then the solution is guaranteed to exist on the interval $[0, d]$ with $d = \frac{b}{\sup_{z\in\mathcal{B}} \|F(z)\|}$.

This result can be proven via Theorem 1 on page 77 using Lemma 3 on page 67 of [94]. Hence, existence of solution trajectories is generally guaranteed. However, the uniqueness is often an issue as was already demonstrated by Example 3.4.1. In this setting Filippov [94, Section 2.10] proved the following result, which considers also solutions in the sense of Definition 3.2.1 with the convex definition for *discontinuous differential equations* as treated in Section 3.2.

Theorem 3.4.3 *Consider the differential equation with discontinuous right-hand side given by* (3.2) *and depicted in Figure 3.1 and assume that f_- and f_+ are continuously differentiable in their domains (C^1), ϕ is twice continuously differentiable C^2 and the discontinuity vector $h(x) := f_+(x) - f_-(x)$ is continuously differentiable (C^1). If for each point x with $\phi(x) = 0$ at least one of the two inequalities $n(x)^{\mathrm{T}} f_+(x) < 0$ or $n(x)^{\mathrm{T}} f_-(x) > 0$ holds (for different points a different inequality may hold), then the Filippov solutions are unique.*

This amounts to a point-by-point check along the switching surface to see if at least one of the inequalities $n(x)^{\mathrm{T}} f_+(x) < 0$ or $n(x)^{\mathrm{T}} f_-(x) > 0$ holds. Using a different solution concept (left Zeno free or forward solutions as introduced for complementarity systems), another test for existence and uniqueness of trajectories for a subclass of switched systems is presented in Theorem 3.4.15 below.

3.4.2 Well-posedness of hybrid automata

We first need some definitions.

We say that the hybrid time trajectory $\tau = \{I_i\}_{i=0}^N$ is a prefix of $\tau' = \{J_i\}_{i=0}^M$ and write $\tau \leqslant \tau'$, if they are identical or τ is finite, $M \geqslant N$, $I_i = J_i$ for $i = 0, 1, \ldots, N-1$, and $I_N \subseteq J_N$. In case τ is a prefix of τ' and they are not identical, τ is a *strict* prefix of τ'.

An execution $\chi = (\tau, q, x)$ is called *finite*, if τ is a finite sequence ending with a closed interval, *infinite*, if τ is an infinite sequence or if $\sum_i (\tau_i' - \tau_i) = \infty$, and *maximal* if τ is not a strict prefix of any other hybrid trajectory of the hybrid automaton. We denote the set of all maximal and infinite executions of the automaton with initial state $(q_0, x_0) \in \mathrm{Init}$ by $\mathcal{H}^M_{(q_0,x_0)}$ and $\mathcal{H}^\infty_{(q_0,x_0)}$, respectively.

Definition 3.4.4 A hybrid automaton is called *non-blocking*, if $\mathcal{H}^\infty_{(q_0,x_0)}$ is non-empty for all $(q_0, x_0) \in \mathrm{Init}$. It is called *deterministic*, if $\mathcal{H}^M_{(q_0,x_0)}$ contains at most one element for all $(q_0, x_0) \in \mathrm{Init}$.

A hybrid automaton is called *initially well-posed*, if it is non-blocking and deterministic. These concepts say nothing about a possible continuation beyond livelock or accumulation points of event times.

To simplify the characterization of non-blocking and deterministic automata, the following assumption has been introduced in [155].

Assumption 3.4.5 The vector field $f(q, \cdot)$ is globally Lipschitz continuous for all $q \in Q$. The edge $e = (q, q')$ is contained in E if and only if $G(e) \neq \varnothing$ and $x \in G(e)$ if and only if there is an $x' \in X$ such that $(x, x') \in R(e)$.

The first part of the assumption is standard to guarantee global existence and uniqueness of solutions within each location given a continuous initial state, as we have seen in Section 3.1. The latter part is without loss of generality as can easily be seen [155].

A state $(\hat{q}, \hat{x})$ is called *reachable*, if there exists a finite execution (τ, q, x) with $\tau = \{[\tau_i, \tau_i']\}_{i=0}^{N}$ and $(q(\tau_N'), x(\tau_N')) = (\hat{q}, \hat{x})$. The set $\mathrm{Reach} \subseteq Q \times X$ denotes the collection of reachable states of the automaton.

The set of states from which continuous evolution is impossible is defined as

$$\mathrm{Out} = \{(q_0, x_0) \in Q \times X \mid \forall \varepsilon > 0, \exists t \in [0, \varepsilon) : x_{q_0,x_0}(t) \notin \mathrm{Inv}(q_0)\}$$

in which $x_{q_0,x_0}(\cdot)$ denotes the unique solution to $\dot{x} = f(q_0, x)$ with $x(0) = x_0$.

Theorem 3.4.6 [155] *A hybrid automaton is non-blocking, if for all $(q, x) \in \mathrm{Reach} \cap \mathrm{Out}$, there exists $e = (q, q') \in E$ with $x \in G(e)$. In case the automaton is deterministic, this condition is also necessary.*

Theorem 3.4.7 *A hybrid automaton is deterministic, if and only if for all $(q, x) \in \mathrm{Reach}$*

- *if $x \in G((q, q'))$ for some $(q, q') \in E$, then $(q, x) \in \mathrm{Out}$;*
- *if $(q, q') \in E$ and $(q, q'') \in E$ with $q' \neq q''$, then $x \notin G((q, q')) \cap G((q, q''))$; and*
- *if $(q, q') \in E$ and $x \in G((q, q'))$, then there is at most one $x' \in X$ with $(x, x') \in R((q, q'))$.*

As a consequence of the broad class of systems covered by the results in this section, the conditions are rather implicit in the sense that for a particular example the conditions cannot be verified by direct calculations (i.e., are not in an algorithmic form). Especially, if the model description itself is implicit (e.g., variable structure systems or complementarity models) these results are only a start of the well-posedness analysis as the hybrid automaton model and the corresponding sets Reach and Out have to be determined first. However, some explicit characterizations of the set Out as can be found in [155] might be convenient in this respect. In the next sections, we will present results that can be checked by direct computations.

The extension of the initial well-posedness results for hybrid automata to local or global existence of executions are awkward as Zeno behavior is hard to characterize or exclude, and continuation beyond Zeno times is not easy to show, as mentioned before. However, in case the location can be described as a function of the continuous state (like for complementarity systems or differential equations with discontinuous right-hand sides) you are able to define an evolution beyond the Zeno time by proving that the (left) limit of the continuous state exists at the Zeno point. Continuation from this limit follows then again by initial or local existence, which makes it possible to say more about local and global well-posedness for this type of systems. In the next sections, we will give some illustrating theorems that have been obtained for complementarity systems and their subclasses (like linear relay systems). See [112] and the references therein for an overview.

3.4.3 Well-posedness of linear complementarity systems

As the interconnection of a continuous, time-invariant, linear system and complementarity conditions, a continuous-time *linear complementarity system* (LCS) can be given by

$$\dot{x}(t) = Ax(t) + Bz(t) \tag{3.13a}$$

$$w(t) = Cx(t) + Dz(t) \tag{3.13b}$$
$$0 \leqslant w(t) \perp z(t) \geqslant 0. \tag{3.13c}$$

where $x(t) \in \mathbb{R}^n$, $w(t) \in \mathbb{R}^m$, $z(t) \in \mathbb{R}^m$, and A, B, C and D are matrices with appropriate sizes. We denote (3.13a)-(3.13b) by $\Sigma(A, B, C, D)$ and (3.13) by LCS(A, B, C, D). Note that the two-carts system in Chapter 1 fits within this class.

One may look at LCS as a dynamical extension of the linear complementarity problem (LCP) of mathematical programming. See [73] for an excellent survey on the LCP.

Problem 3.4.8 The LCP problem LCP(q, M) is defined as follows: Given an m-vector q and $m \times m$ matrix M find an m-vector z such that

$$0 \leqslant w := q + Mz \perp z \geqslant 0. \tag{3.14}$$

We say z *solves* (or is a *solution* of) LCP(q, M), if z satisfies (3.14). The set of solutions of LCP(q, M) is denoted by SOL(q, M).

Several generalizations of the LCP have been defined among which the Extended Linear Complementarity Problem [79], which is directly connected to the ELC systems discussed in Chapter 2. Indeed, in principle an ELCP is given by the equations (2.38c)-(2.38d) (without the time dependence k). For a further discussion on this problem the interested reader is referred to [79].

Definition 3.4.9 A matrix $M \in \mathbb{R}^{m\times m}$ is called

- *nondegenerate* if its principal minors $\det M_{II}$ for $I \subseteq \{1, \ldots, m\}$ are nonzero.
- a *P-matrix* if all its principal minors are positive, i.e., $\det M_{II} > 0$ for all $I \subseteq \{1, \ldots, m\}$.
- *positive (nonnegative) definite*[2] if $x^{\mathrm{T}}Mx > 0$ $(\geqslant 0)$ for all $0 \neq x \in \mathbb{R}^m$.

It can be shown that every (symmetric) positive definite matrix is a P-matrix; but the converse is not true. However, every *symmetric* P-matrix is also positive definite.

Definition 3.4.10 The dual cone of a given nonempty set $\mathcal{S} \subset \mathbb{R}^m$, denoted by $\mathcal{S}^*$, is given by $\{v \in \mathbb{R}^m \mid v^{\mathrm{T}}w \geqslant 0 \text{ for all } w \in \mathcal{S}\}$.

The final ingredient of our preparation is the "*index*" of a rational matrix.

Definition 3.4.11 A rational matrix $H(s) \in \mathbb{R}^{l\times l}(s)$ is said to be *of index k* if it is invertible as a rational matrix and $s^{-k}H^{-1}(s)$ is proper. It is said to be *totally of index k* if all its principal submatrices are of index k.

With a slight abuse of terminology, we say that a linear system $\Sigma(A, B, C, D)$ is (totally) of index k, if its transfer function is (totally) of index k.

[2]Note that the matrix is not assumed to be symmetric.

Linear complementarity systems with index 1

We will start investigating well-posedness of LCS for which the underlying linear system is totally of index 1.

The following theorem provides sufficient conditions for well-posedness in the sense of existence and uniqueness of LCS with index 1.

Theorem 3.4.12 [63] *Consider an* LCS(A, B, C, D) *with* $\Sigma(A, B, C, D)$ *totally of index* 1. *Suppose that* $D + C(\sigma I - A)^{-1}B$ *is a P-matrix for all sufficiently large* $\sigma \in \mathbb{R}$. *There exists a left Zeno free solution of* LCS(A, B, C, D) *with* continuous *state trajectory for the initial state* x_0 *if and only if* LCP(Cx_0, D) *is solvable. Moreover, if such a solution exists it is left Zeno free unique, i.e., there is no other left Zeno free solution.*

Linear passive complementarity systems

When the underlying system $\Sigma(A, B, C, D)$ is passive (in the sense of [217]) we call the overall system (3.13) a *linear passive complementarity system* (LPCS). For a detailed study on LPCS, the reader may refer to [62]. As shown in [63, Lemma 3.8.5], the passivity of the system (under some extra assumptions) implies that it is of index 1. Hence, Theorem 3.4.12 is applicable to LPCS. Additionally, it can be shown that there are no left Zeno solutions for LPCS as formulated in the following theorem.

Theorem 3.4.13 [63] *Consider an* LCS(A, B, C, D) *with* $\Sigma(A, B, C, D)$ *being passive,* (A, B, C) *being minimal, and* $\operatorname{col}(B, D + D^{\mathrm{T}}) := \begin{pmatrix} B \\ D + D^{\mathrm{T}} \end{pmatrix}$ *of full column rank. Let* $\mathcal{Q}_D = \{z \mid z \text{ solves } \mathrm{LCP}(0, D)\}$. *There exists a left Zeno free solution of* LCS(A, B, C, D) *with* continuous *state trajectory for the initial state* x_0 *if and only if* $Cx_0 \in \mathcal{Q}_D^*$. *Moreover, if a solution exists it is unique*[3].

3.4.4 Piecewise linear systems

As is well-known (see, e.g., [87]), piecewise linear relations may be described in terms of the linear complementarity problem. Similarly, piecewise linear systems can be rewritten as complementarity systems in various cases (cf. Chapter 2 for the discrete-time case). For the sake of simplicity, we will focus on a specific type of piecewise linear systems, namely linear saturation systems, i.e., linear systems coupled to saturation/relay characteristics. They are of the form

$$\dot{x}(t) = Ax(t) + Bz(t) \tag{3.15a}$$

$$w(t) = Cx(t) + Dz(t) \tag{3.15b}$$

$$(z(t), w(t)) \in \mathrm{saturation}_i \tag{3.15c}$$

where $x(t) \in \mathbb{R}^n$, $z(t) \in \mathbb{R}^m$, $w(t) \in \mathbb{R}^m$, A, B, C and D are matrices of appropriate sizes, and $\mathrm{saturation}_i$ is the curve depicted in Figure 3.8 with $e_2^i - e_1^i > 0$ and $f_1^i \geqslant f_2^i$. This curve is formally described by the set

$$\{(u, v) \in \mathbb{R}^2 \mid (v = e_2^i \text{ and } u \leqslant f_2^i) \text{ or } (v = e_1^i \text{ and } u \geqslant f_1^i) \text{ or}$$

[3] It can also be shown that this solution is unique in L_2.

$$(e_1^i \leqslant v \leqslant e_2^i \text{ and } (f_1^i - f_2^i)(v - e_2^i) = (e_1^i - e_2^i)(u - f_2^i))\}. \quad (3.16)$$

We denote the overall system (3.15) by $\mathrm{SAT}(A, B, C, D)$. Note that relay characteristics can be obtained from saturation characteristics by setting $f_1^i = f_2^i$.

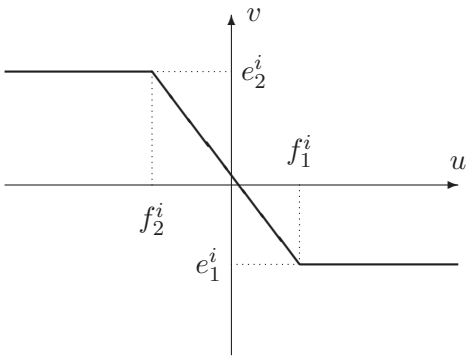

Figure 3.8: Saturation characteristic.

One may argue that the saturation characteristic is a Lipschitz continuous function (provided that $f_1^i - f_2^i > 0$) and hence existence and uniqueness of solutions follow from the theory of ordinary differential equations. The following example shows that this is not correct in general if the feedthrough term D is nonzero.

Example 3.4.14 Consider the single-input single-output system

$$\dot{x} = z \quad (3.17)$$

$$w = x - 2z \quad (3.18)$$

where z and w are restricted by a saturation characteristic as shown in Figure 3.8 with $e_1 = -f_1 = -e_2 = f_2 = \frac{1}{2}$. Let the periodic function $\tilde{z} : \mathbb{R}^+ \to \mathbb{R}$ be defined by

$$\tilde{z}(t) = \begin{cases} 1/2 & \text{if } 0 \leqslant t < 1 \\ -1/2 & \text{if } 1 \leqslant t < 3 \\ 1/2 & \text{if } 3 \leqslant t < 4 \end{cases}$$

and $\tilde{z}(t-4) = \tilde{z}(t)$ whenever $t \geqslant 4$. By using this function define $\tilde{x} : \mathbb{R}^+ \to \mathbb{R}$ as

$$\tilde{x}(t) = \int_0^t \tilde{z}(s)\, ds,$$

and $\tilde{w} : \mathbb{R}^+ \to \mathbb{R}$ as

$$\tilde{w} = \tilde{x} - 2\tilde{z}.$$

It can be verified that $(-\tilde{z}, -\tilde{x}, -\tilde{w})$, $(0, 0, 0)$ and $(\tilde{z}, \tilde{x}, \tilde{w})$ are all solutions of SAT$(0, 1, 1, -2)$ with the zero initial state.

As illustrated in the example, the Lipschitz continuity argument does not work in general when $f_1^i > f_2^i$. Also in the case, where $f_1^i = f_2^i$ this reasoning does not apply. The following theorem gives a sufficient condition for the well-posedness of linear systems with saturation characteristics.

Theorem 3.4.15 [63] *Consider* SAT(A, B, C, D) *and define its transfer function as* $G(s) := C(sI - A)^{-1}B + D$. *Let* $R = \mathrm{diag}(e_2^i - e_1^i)$ *and* $S = \mathrm{diag}(f_2^i - f_1^i)$. *Suppose that* $G(\sigma)R - S$ *is a* P*-matrix for all sufficiently large* $\sigma \in \mathbb{R}$. *Then, there exists a unique left Zeno free solution of* SAT(A, B, C, D) *with a* continuous *state trajectory for all initial states.*

This theorem generalizes the result in [152] that considered linear relay systems only.

Example 3.4.16 Let us reconsider Example 3.4.1. Note that we have connected a negative relay element (i.e., $S = 0$ and $R = 2$ in the theorem above) to the linear system

$$\begin{aligned} \dot{x}_1 &= -z \\ w &= x_1. \end{aligned}$$

Since the transfer function of the system is $\frac{-1}{s}$, we indeed see that the system does not satisfy the conditions of Theorem 3.4.15! If we would couple the negative relay to

$$\begin{aligned} \dot{x}_1 &= z \\ w &= x_1, \end{aligned}$$

left Zeno free well-posedness is guaranteed as the transfer function $\frac{1}{s}$ is positive for sufficiently large values of s.

It is interesting to note that the conditions in Theorem 3.4.15 are sufficient for existence and uniqueness of *left Zeno free solutions* given arbitrary initial states. However, if one uses Filippov's solution concept based on differential inclusion and thus including left accumulation points in the event times set, this is no longer true. If $G(s) = \frac{1}{s^3}$ (a triple integrator) is coupled to a negative relay, one has multiple solutions starting in the origin in spite of the positiveness of the transfer function (see [187]). In [187] it has been shown that for the case where $D = 0, CB > 0$ or where $D = CB = 0, CAB > 0$ there is a unique Filippov solution from the origin in the sense of Definition 3.2.1. If one would like to claim uniqueness of solutions in the sense of Filippov, then the result of Theorem 3.4.3 can be applied. Note that Theorem 3.4.3 proves uniqueness in a broader solution space and for general nonlinear systems. However, it has to be verified on a point-by-point basis, while Theorem 3.4.15 can be checked by computing the signs of the principal minors of a rational matrix. However, Theorem 3.4.15 deals with linear relay systems that uses a left Zeno solution concept.

3.5 Generalizations including jumps

Up to this point, we have presented well-posedness results for complementarity systems in which the x-part of the solutions is continuous. In this subsection, the available generalizations will be mentioned including the possibility of re-initializations (state jumps). In such studies the issue of irregular initial states had to be tackled, i.e., the initial states for which there is no solution with a continuous state trajectory (e.g., in case of the systems and solution concept considered in Theorem 3.4.12 all initial states x_0 for which LCP(Cx_0, D) is not solvable). A distributional framework was used to obtain a new solution concept for LCS [117]. In principle, this framework is based on so-called *impulsive-smooth distributions* [106] of the form $z(t) = \sum_{i=0}^{l} z^{-i}\delta^{(i)} + z_{reg}(t)$, where δ is the delta or Dirac distribution (supported at 0), $\delta^{(i)}$ is the ith derivative of δ and z_{reg} is a smooth function. Sometimes we consider Bohl distributions instead of impulsive-smooth ones, which are characterized by the fact that they have

rational Laplace transforms. An impulsive-smooth distribution (z, x, w) is called an *initial solution* for initial state x_0, if it satisfies $\dot{x} = Ax + Bz + x_0\delta$; $w = Cx + Dz$ as equalities of distributions, there exists an $I \subseteq \{1, \ldots, m\}$ with $w_i = 0$, $i \in I$ and $z_i = 0, i \notin I$ and finally, the Laplace transforms satisfy $\hat{z}(\sigma) \geqslant 0$ and $\hat{w}(\sigma) \geqslant 0$ for all sufficiently large σ. In case $(z(t), x(t), w(t))$ is an ordinary function these conditions mean that the system's equations (3.13) are satisfied on an interval of the form $[0, \varepsilon)$ for some $\varepsilon > 0$. In case the initial solution is not a function, the impulsive part of $u(t)$ will result in a state jump from x_0 to $x^+ := x_0 + \sum_i A^i B z^{-i}$ (see [106]). Compare this with the two-carts example and the discussion at the end of Section 1.6.4 in which the z-variable (reaction force) contains Dirac impulses at impact times. Particularly, in [117] it is shown that the above re-initialization procedure corresponds for linear mechanical systems with unilateral constraints to the *inelastic* impact case. Moreover, in some cases the jump of the state variable can be made more explicit in terms of the linear projection operator onto the consistent subspace of the new mode along a jump space [117].

For the LCS(A, B, C, D) the rational matrices $G(s)$ and $Q(s)$ are defined by $G(s) = C(s\mathcal{I} - A)^{-1}B + D$ and $Q(s) = C(s\mathcal{I} - A)^{-1}$.

Theorem 3.5.1 [115] *An* LCS(A, B, C, D) *is initially well-posed if and only if for all* x_0 *the problem* LCP$(Q(\sigma)x_0, G(\sigma))$ *is uniquely solvable for sufficiently large* $\sigma \in \mathbb{R}$.

The strength of this theorem is that dynamical properties of an LCS are coupled to properties of families of static LCPs, for which a wealth of existence and uniqueness are available [73]. E.g., a sufficient condition for initial well-posedness is $G(\sigma)$ being a P-matrix for sufficiently large σ.

Clearly, initial well-posedness does not imply local existence of solutions as in principle, an infinite number of re-initializations (jumps) may occur on one time-instance without "time progressing" (live-lock). However, sufficient conditions for *local* well-posedness have been provided for LCS [117,210], as presented next. Consider the LCS(A, B, C, D) with Markov parameters $H^0 = D$ and $H^i = CA^{i-1}B$, $i = 1, 2, \ldots$ and define the leading row and column indices by

$$\eta_j = \inf\{i \mid H^i_{\bullet j} \neq 0\},\ \rho_j = \inf\{i \mid H^i_{j\bullet} \neq 0\}, \tag{3.19}$$

where $j \in \{1, \ldots, k\}$ and $\inf \varnothing := \infty$. The *leading row coefficient matrix* $\mathcal{M}$ and *leading column coefficient matrix* $\mathcal{N}$ are then given for *finite* leading row and column indices by

$$\mathcal{M} := \begin{pmatrix} H^{\rho_1}_{1\bullet} \\ \vdots \\ H^{\rho_k}_{k\bullet} \end{pmatrix} \text{ and } \mathcal{N} := (H^{\eta_1}_{\bullet 1} \ldots H^{\eta_k}_{\bullet k})$$

Theorem 3.5.2 [117] *If the leading column coefficient matrix* $\mathcal{N}$ *and the leading row coefficient matrix* $\mathcal{M}$ *are both defined and P-matrices, then* LCS(A, B, C, D) *has a unique* local *left Zeno free solution on an interval of the form* $[0, \varepsilon)$ *for some* $\varepsilon > 0$. *Moreover, live-lock does not occur; after at most one jump a smooth continuation exists.*

Theorem 3.5.3 *Consider a bimodal LCS (3.13) with*[4] $C \neq 0$. *The following statements are equivalent.*

1. *The leading Markov parameter* $\mathcal{M} = \mathcal{N}$ *is defined (i.e.,* $\rho_1 = \eta_1 < \infty$*) and positive.*

[4]Note that $C = 0$ is a degenerate and uninteresting case, since the complementarity conditions do not involve the state vector x. Any quadruple $(\mathcal{E}, u_c, x_c, y_c)$ with $u(t)$ a solution to LCP$(0, D)$ for all $t \notin \mathcal{E}$ and satisfying (3.13a)-(3.13b) is a solution to (3.13). It can easily be seen that for a scalar D, LCP$(0, D)$ has a unique solution if and only if $D \neq 0$.

2. *The linear complementarity system (3.13) is locally well-posed.*

3. *The linear complementarity system (3.13) is globally well-posed.*

First steps in the direction of getting global well-posedness results for LCS *with external inputs* are due to [62] for LPCS and [64], where the underlying linear system is of index 1.

3.6 Summary

For smooth systems the definition of the solution concept is quite straightforward and the study of well-posedness is well know. Lipschitz conditions on the vector field play a crucial role. Only in absence of *local Lipschitz continuity* non-uniqueness of solutions can occur and only in absence of *global Lipschitz continuity* finite escape time causing the lack of global existence of solutions can happen.

In the hybrid case these issues become more involved as additional complications related to Zenoness (infinitely fast switching, livelock and accumulation points of event times, etc) arise. First of all we introduced the concept of the sliding mode by giving Filippov's convex definition and Utkin's control equivalent definition. After the introduction of these concepts, we discussed the impact of the choice of the solution concept on well-posedness and analysis issues. E.g., the water tank system demonstrated that the tanks stay full along right Zeno free solution trajectories, although in 'reality' they will be empty in finite time.

A distinction of well-posedness can be made on the basis of the solution concept that is used, but also on the time interval that is considered. Initial, local and global well-posedness were introduced and the relationships were discussed. An initially well-posed system might not be locally well-posed due to livelock (infinite number of "jumps" or events on one time instant), while a locally well-posed system might not be globally well-posed as a consequence of right accumulation points of event times. In the latter case also finite escape times might play a role.

Some illustrating well-posedness results for hybrid automata, complementarity systems, piecewise affine and discontinuous differential equations have been discussed. For an overview of other well-posedness results for various classes of hybrid systems, we refer the reader to [112] and the references therein.

Chapter 4

Stability of hybrid systems

This chapter was inspired by the excellent surveys of Daniel Liberzon [146, 149]. Recently, also a good survey on stability for hybrid systems was given in [150].

4.1 Switched systems

In this chapter we will mainly focus on so-called *switched systems*, that are described by equations of the form

$$\dot{x} = f_\sigma(x) \ , \tag{4.1}$$

where $\{f_p \mid p = 1, \ldots, N\}$ is a family of sufficiently smooth vector fields from $\mathbb{R}^n$ to $\mathbb{R}^n$ and σ is a piecewise constant function of time, called the *switching signal.* The switching signal σ might depend only on time t, only on the state $x(t)$, or both, or may even be generated by more complicated techniques involving memory (like hysteresis). We assume that the system has an equilibrium in the origin for all switching sequences, i.e., all the vector fields have an equilibrium in zero ($f_i(0) = 0$ for all i). A difference with the general hybrid automata set-up is that the state variable x does not jump during switching instants. Hence, the trajectory x is continuous everywhere. At this point we do not consider any generalized solution concepts (see Chapter 3) due to chattering (infinitely fast switching). In fact, this would mean adding additional vector fields corresponding to sliding behavior.

In the particular case where all the individual subsystems given by $\dot{x} = f_i(x)$ are linear, i.e., $\dot{x} = A_i x$, we speak of a *switched linear system*. More specifically, in case the switching of a switched linear system is determined fully by the state variable x in the sense that $\dot{x} = A_i x$ is active when $x \in \mathcal{C}_i$, we speak of a *piecewise or multi-modal linear system* [131, 198, 199]. Typically, the sets $\mathcal{C}_i$ are rather regular. E.g., in the literature these sets are often polytopes (i.e., given by a finite number of inequalities). Moreover, they are often taken to form a partitioning of the state space $\mathcal{C}$, i.e.,

$$\cup_{i=1}^{n} \mathcal{C}_i = \mathcal{C} \ ,$$

and the intersection (boundary) of the *interiors* of two cells $\mathcal{C}_i$ and $\mathcal{C}_j$ for $i \neq j$ is empty or stated differently, the intersection of two cells $\mathcal{C}_i$ and $\mathcal{C}_j$ for $i \neq j$ is a lower-dimensional object, i.e., the face of one or more of the polytopes. Typically, for piecewise affine (PWA) systems it might look as

$$\dot{x} = A_i x + a_i, \text{ when } E_i x \geqslant e_i, \ i \in I \tag{4.2}$$

for certain matrices E_i and vectors e_i of appropriate dimensions. The inequality is assumed to hold componentwise.

Of course, all these descriptions can be similarly used for discrete-time systems (cf. Chapter 2) in a straightforward way.

Several interesting questions with respect to stability might be raised [149]:

Problem A : Find conditions for which the switched system (4.1) is asymptotically stable for any switching signal (without infinitely fast switching).

Problem B : Show that the switched system (4.1) is asymptotically stable for a given switching strategy or a class of switching strategies.

Problem C : Construct a switching signal that makes the switched system (4.1) asymptotically stable (i.e., a stabilization problem).

The first two problems are related to analysis of stability, while Problem C is a stabilization problem. Problems A and B will be studied in this chapter on stability *analysis*. As we consider Problem C a hybrid *control* problem , this will be treated in Chapter 5 on switched control.

We start by revealing some of the peculiarities in these type of analysis problems.

4.2 Stability of subsystems implying stability of switched system?

One might hope that the stability of the submodels says something about the stability of the switched system. Unfortunately, this hope is scattered into pieces by the following counterexample.

Consider the switched system taken from [40] given by

$$\dot{x} = \begin{cases} A_1 x, & \text{if } x_1 x_2 \leqslant 0 \\ A_2 x, & \text{if } x_1 x_2 > 0 \end{cases} \tag{4.3}$$

with

$$A_1 = \begin{pmatrix} -1 & 10 \\ -100 & -1 \end{pmatrix};\ A_2 = \begin{pmatrix} -1 & 100 \\ -10 & -1 \end{pmatrix}.$$

By inspection of the eigenvalues of A_1 and A_2 one can see that both of the dynamics are stable. See Figure 4.1, where the phase portraits are given.

The switched system activates dynamics 1, if the state lies in the second and fourth quadrant and dynamics 2, when the state lies in the first or third quadrant (note that there is no problem at the boundaries of the quadrants as the directions of the vector fields point towards the same quadrants). As one can see from a possible trajectory in Figure 4.2 starting close to the origin, the switched system is unstable.

As a consequence, it is not possible to study stability properties of the subsystems only, the switching structure has to be taken into account as well.

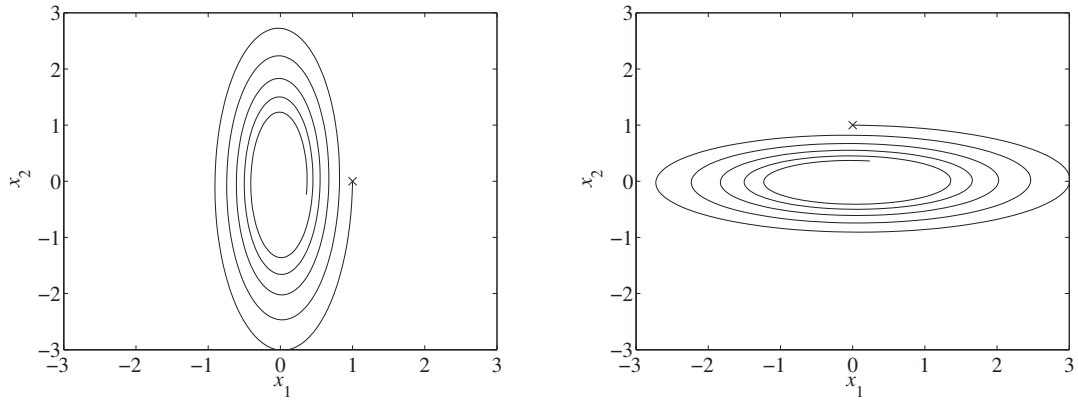

Figure 4.1: Constituting stable linear submodels.

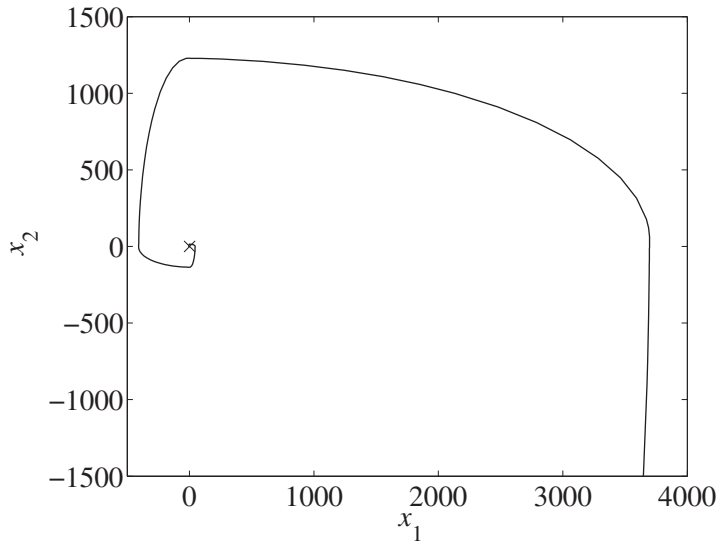

Figure 4.2: Unstable switched system.

4.3 Recapitulation of Lyapunov theory for smooth systems

Before discussing the stability results for switched systems, we will first recall the standard Lyapunov theory for smooth systems of the form

$$\dot{x} = f(x). \tag{4.4}$$

We assume that $x = 0$ is an equilibrium (i.e., $f(0) = 0$) for the system (4.4).

Definition 4.3.1 [137] A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$, if it is strictly increasing and $\alpha(0) = 0$. It is said to be class $\mathcal{K}_\infty$ if $a = \infty$ and $\alpha(r) \to \infty$ if $r \to \infty$.

Definition 4.3.2 [137] A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ is said to be of class $\mathcal{KL}$, if for each fixed s, the mapping $r \mapsto \beta(r, s)$ belongs to class $\mathcal{K}$ and for each fixed r, the mapping $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$ as $s \to \infty$.

Definition 4.3.3 [137] The equilibrium $x = 0$ of $\dot{x} = f(x)$ is

- *stable*, if for each $\varepsilon > 0$, there is a $\delta > 0$ such that $\|x(0)\| < \delta$ implies that $\|x(t)\| < \varepsilon$ for all $t \geqslant 0$;
- *globally asymptotically stable*, if there exists a class $\mathcal{KL}$ function $\beta : [0, \infty) \times [0, \infty) \to [0, \infty)$ such that $\|x(t)\| \leqslant \beta(\|x(0)\|, t)$ for all initial states $x(0)$ and all times t. Or stated differently, if the origin is stable and for any $\varepsilon > 0$ and any $c > 0$ there is a $T = T(\varepsilon, c) \geqslant 0$ such that $\|x(t)\| < \varepsilon$ for all $t \geqslant T$ and for any $\|x(0)\| \leqslant c$.

Note that if β takes the form $\beta(r, s) = cre^{-\lambda s}$ for some $\lambda > 0$ and $c > 0$, the system is called exponentially stable.

The following theorem (called the Barbashin-Krasovksii theorem (see [137])) is a nice illustrating example of the pioneering work by Lyapunov for normal smooth systems of the form $\dot{x} = f(x)$. To have no problems with the issue of well-posedness, we assume that f is locally Lipschitz. Moreover, we say that a function V is *radially unbounded*, if $V(x) \to \infty$ as $\|x\| \to \infty$.

Theorem 4.3.4 *Let $x = 0$ be an equilibrium of $\dot{x} = f(x)$ (i.e., $f(0) = 0$) and let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable radially unbounded function such that*

- *$V(0) = 0$ and $V(x) > 0$, if $x \neq 0$ (i.e., V is positive definite), and*
- *$\dot{V}(x) = L_f V(x) = \left[\frac{\partial V}{\partial x}\right]^{\mathrm{T}} f(x) < 0$ for all $x \neq 0$.*

Then $x = 0$ is globally asymptotically stable, and we say that V is a Lyapunov function.

Particularly interesting is the case where the Lyapunov function is a quadratic one $V(x) = x^{\mathrm{T}} P x$ for some positive definite matrix P. The reason is that asymptotic stability for a linear system $\dot{x} = Ax$ is equivalent to the existence of a quadratic Lyapunov function. To be precise, for every matrix A that is Hurwitz (all eigenvalues in the open left half plane) and for any $Q > 0$ there exists a $P > 0$ such that the Lyapunov equality

$$A^{\mathrm{T}} P + PA = -Q \tag{4.5}$$

holds. Actually, one can take

$$P = \int_0^\infty e^{A^{\mathrm{T}} t} Q e^{At} dt \tag{4.6}$$

as the solution.

Note that Theorem 4.3.4 expresses the existence of a Lyapunov function as a *sufficient* condition for asymptotic stability. In the linear case, as just mentioned, it is a necessary condition as well (and the Lyapunov function even has a special form as it is quadratic). Actually, the necessity of the existence of a Lyapunov function can be proven in a more general setting, which is known as a converse theorem: If a smooth system of the form (4.4) is asymptotically stable, then there exists a continuously differentiable radially unbounded function (cf. Section 3.6 in [137]).

4.4 Solving problem A: Common Lyapunov approach

4.4.1 General switched systems

One of the first ideas that appeared in the literature for the analysis of the switched system (4.1) is the *simultaneous or common Lyapunov function approach*.

To formulate the extension of this theorem to switched systems, we first introduce some nomenclature. We will say that the system (4.1) is *globally uniformly asymptotically stable* (GUAS) (a whole mouth full), if there exists a class $\mathcal{KL}$ function β such that for all switching signals σ the solutions of (4.1) for any initial state $x(0)$ satisfy the inequality

$$\|x(t)\| \leqslant \beta(\|x(0)\|, t) \text{ for all } t \geqslant 0. \tag{4.7}$$

Note that this implies that a switched system that is GUAS, has the property $\lim_{t\to\infty} x(t) = 0$ for all initial conditions and all switching signals. The terminology *uniform* is employed here to indicate the uniformity with respect to the switching signals: (4.7) is independent of σ. The term *global* refers to the fact that it holds for all initial states.

Given a continuously differentiable (C^1) function $V : \mathbb{R}^n \to \mathbb{R}$, we will say that it is a *common Lyapunov function* for the family of the switched system (4.1), if V is positive definite and the Lie derivatives

$$L_{f_i}V(x) = \frac{\partial V}{\partial x} f_i(x) < 0, \text{ when } x \neq 0 \text{ and for all } i = 1, \ldots, N. \tag{4.8}$$

Theorem 4.4.1 *If all the smooth subsystems in the switched system* (4.1) *share a positive definite radially unbounded common Lyapunov function, then the switched system is globally uniformly asymptotically stable.*

We already discussed in Section 4.3 converse Lyapunov results for smooth systems and the question arises whether similar results also hold for switched systems. The answer to this question is affirmative as will be discussed next.

We assume that the functions f_i in (4.1) are at least locally Lipschitz continuous.

Theorem 4.4.2 [161] *If the switched system (4.1) is GUAS, then all f_i share a positive definite radially unbounded common Lyapunov function.*

Note that this theorem also implies that all dynamics obtained by taking convex combinations (like in sliding modes) of the original dynamics $\{f_1, \ldots, f_N\}$ are asymptotically stable as well. Indeed, take $f_{\alpha,i,j}(x) := \alpha f_i(x) + (1-\alpha) f_j(x)$ for $\alpha \in [0, 1]$. Then, it is obvious that

$$\frac{\partial V}{\partial x} f_{\alpha,i,j}(x) = \alpha \frac{\partial V}{\partial x} f_i(x) + (1-\alpha) \frac{\partial V}{\partial x} f_j(x) < 0$$

and one can use this to prove that this yields asymptotically stable dynamics. Note that this also holds for any vector field obtained by taking a positive combination of the original dynamics, i.e., any f obtained as $f(x) = \sum_{i=1}^{N} \beta_i f_i(x)$ with $\beta_i \geqslant 0$. Hence, the "enlarged" switched system including all possible positive combinations is globally uniformly asymptotically stable as well. This might help in case possible sliding motions might occur in the system (e.g., using Filippov solutions). Note that

continuous differentiability (C^1) of V is of importance here. When the smoothness properties of V are lost, one has to be careful in the reasoning above (see also [94]).

In the next section we will use the above ideas for the case of switched linear systems. We will use particular forms of Lyapunov functions, that yield computational ways to find them.

4.4.2 Switched linear systems

Stability of switched linear systems of the form

$$\dot{x} = A_\sigma x \tag{4.9}$$

can also be approached by the common Lyapunov approach and in imitation of linear systems we will aim to use a common quadratic Lyapunov function of the form $V(x) = x^{\mathrm{T}}Px$ with P a positive definite matrix (denoted by $P > 0$).

If we apply the condition (4.8) for a switched linear system using a quadratic Lyapunov function $V(x) = x^{\mathrm{T}}Px$, we can prove the following result.

Theorem 4.4.3 *Suppose the set of linear matrix inequalities*

$$A_i^{\mathrm{T}}P + PA_i < 0 \text{ for all } i = 1, \ldots, N \tag{4.10}$$

has a solution $P > 0$. Then the switched linear system (4.9) *is GUAS.*

Actually, in the setting of the above theorem we have so-called *quadratic stability*, i.e., there exists a quadratic Lyapunov function, i.e., $V(x) = x^{\mathrm{T}}Px$ with a quadratic bound on the derivative of the Lyapunov function with respect to time of the form $\dot{V}(x) \leqslant -\varepsilon\|x\|^2$ for some $\varepsilon > 0$. Moreover, we obtained conditions in terms of *linear matrix inequalities* (LMI) [37, 193] for which efficient methods exist that aim at solving them numerically. Another nice feature is that through dual theorems we can show the infeasibility of the LMI (4.10). The set of LMIs (4.10) is infeasible if and only if there exists positive definite matrices R_i, $i = 1, \ldots, N$ such that

$$\sum_{i=1}^{N}(A_i^{\mathrm{T}}R_i + R_iA_i) > 0. \tag{4.11}$$

See [37] for a proof.

The equations of the type (4.10) are in a somewhat implicit way as they are formulated as a problem to which a solution must exist. It might be nice to have more explicit and algebraically verifiable conditions that are sufficient for the existence of a common quadratic Lyapunov function. One such condition is formulated below.

Theorem 4.4.4 *If the matrices $\{A_1, \ldots, A_N\}$ commute pairwise*[1] *and are all stable, then the corresponding switched linear system (4.9) is GUAS and there exists a common quadratic Lyapunov function $P = P_N$, that can be found from solving the following set of Lyapunov equalities successively:*

$$A_1^{\mathrm{T}}P_1 + P_1A_1 \;=\; -I$$

[1] This means that for all i, j, it holds that $A_iA_j = A_jA_i$.

$$\begin{aligned} A_2^{\mathrm{T}}P_2 + P_2A_2 &= -P_1 \\ A_3^{\mathrm{T}}P_3 + P_3A_3 &= -P_2 \\ \vdots \qquad &\quad \vdots \\ A_N^{\mathrm{T}}P_N + P_NA_N &= -P_{N-1}. \end{aligned}$$

Exercise 4.4.5 *Prove this by using the explicit formula (4.6) and the fact that $e^Ae^B = e^Be^A$ for commutative matrices A and B.*

Nonlinear extensions of this result and more general conditions based on the Lie algebras of $\{A_1,\ldots,A_N\}$ are known and can be found in, e.g., [148].

In Section 4.3 it was mentioned that for a linear system $\dot{x} = Ax$ the converse theorem holds with a special form of the Lyapunov function. Indeed, if the linear system $\dot{x} = Ax$ is asymptotically stable, there exists a quadratic Lyapunov function. The general converse theorem for switched (nonlinear) systems given in Theorem 4.4.2 expresses that for GUAS implies the existence of a common Lyapunov function. So, also for switched linear systems (4.9) that are GUAS such a common Lyapunov function exists. It is quite natural to ask the question, if this common Lyapunov function can be taken quadratic for switched *linear* systems. The answer is negative as can be shown by the example below.

Example 4.4.6 The linear switched system with

$$A_1 = \begin{pmatrix} -1 & -1 \\ 1 & -1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -1 & -10 \\ 0.1 & -1 \end{pmatrix},$$

is GUAS, but there does not exist a common quadratic Lyapunov function, which can be verified by using (4.11) with

$$R_1 = \begin{pmatrix} 0.2996 & 0.7048 \\ 0.7048 & 2.4704 \end{pmatrix}, \quad R_2 = \begin{pmatrix} 0.2123 & -0.5532 \\ -0.5532 & 1.9719 \end{pmatrix}.$$

However, one can always show that there is a common Lyapunov function that is homogeneous of degree 2. Specifically, one can always find a common Lyapunov function in the form $V(x) = \max_{i=1,2,\ldots,k}(l_i^{\mathrm{T}}x)^2$ for constant vectors l_i, $i = 1,\ldots,k$ [164, 166].

4.5 Solving Problem B: Piecewise affine systems

The previous results considered problem A: the stability of a switched system under arbitrary switching. However, in various situations the switching strategy is already given meaning that the switching is not arbitrary, but belongs to a specific class. Of course, one can still apply the results based on a common Lyapunov function as it shows stability for any switching signal, but one has to realize that such a test is conservative. In this section we will derive less conservative results for a class of switched linear systems with state-dependent switching: piecewise affine systems as in (4.2).

We will start this section by some examples showing the conservatism of the common Lyapunov approach in case of piecewise affine systems. This conservatism gives hints upon possible relaxations.

4.5.1 Some examples

Example 4.5.1 (Single Lyapunov function) Consider the switched system

$$\dot{x} = \begin{cases} A_1 x & \text{if } x_1 x_2 \leqslant 0 \\ A_2 x & \text{if } x_1 x_2 > 0, \end{cases} \tag{4.12}$$

where

$$A_1 = \begin{pmatrix} -1 & -1 \\ 1 & -1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -1 & -10 \\ 0.1 & -1 \end{pmatrix} .$$

As seen before this system does not have a common quadratic Lyapunov function. However, it is easily seen that for the function $V(x) = x_1^2 + x_2^2$ it holds that $\dot{V} < 0$ along the nonzero solutions of the switched system, which implies global asymptotic stability.

The relaxation that can be made is that the decreasing of V does not necessarily have to hold for each individual submodel in the whole state space (as for a common Lyapunov function), but *only* in the regions where the corresponding dynamics is active. Indeed, we only need to have $L_{A_1 x} V(x) < 0$ if $x_1 x_2 \leqslant 0$ and $L_{A_2 x} V(x) < 0$ if $x_1 x_2 > 0$. Hence, one way of relaxing the common Lyapunov conditions for GUAS is to take one function V for which $L_{f_i} V(x)$ is only negative, when the dynamics f_i can be active.

Even if the stability analysis based on a single Lyapunov function as presented before does not work out, one can still use *multiple* Lyapunov functions, one for each subsystem in the PWA system. An illustration is given in the example taken from [131].

Example 4.5.2 (Multiple Lyapunov functions) Consider the switched system

$$\dot{x} = \begin{cases} A_1 x & \text{if } x_1 \leqslant 0 \\ A_2 x & \text{if } x_1 > 0, \end{cases} \tag{4.13}$$

where

$$A_1 = \begin{pmatrix} -5 & -4 \\ -1 & -2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -2 & -4 \\ 20 & -2 \end{pmatrix}.$$

This piecewise linear system does not have a common Lyapunov function for all individual submodels (this can be verified by using the infeasibility condition (4.11)) and one even cannot find one quadratic function as in the previous piecewise linear example that decreases in the regions where the submodels are active. Indeed, as the satisfaction of a Lyapunov inequality on a half space implies also satisfaction of the inequality at the other half space (by symmetry) it would actually be a common Lyapunov function, which was concluded to be impossible. However, an alternative is to use two quadratic Lyapunov functions $V_i(x) = x^{\mathrm{T}} P_i x$ with

$$P_1 = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 10 & 0 \\ 0 & 3 \end{pmatrix}.$$

Then V_i is a Lyapunov function for $\dot{x} = A_i x$. A nice feature of those particular instances of V_i is that they are continuous over the switching plane, i.e., $V_1(x) = V_2(x)$ when $x_1 = 0$. Hence, V_σ with $\sigma = 1$ if $x_1 \leqslant 0$ and $\sigma = 2$ when $x_1 > 0$ is continuous and strictly decreasing along non-zero solutions

(except at the axes) and consequently, we have global asymptotic stability (one might have to use La Salle's invariance principle [137]). The level sets of the Lyapunov function in this case are piecewise quadratic. In Figure 4.3 some trajectories are drawn with a level set of the Lyapunov function of the form $\{x \mid V(x) = c\}$.

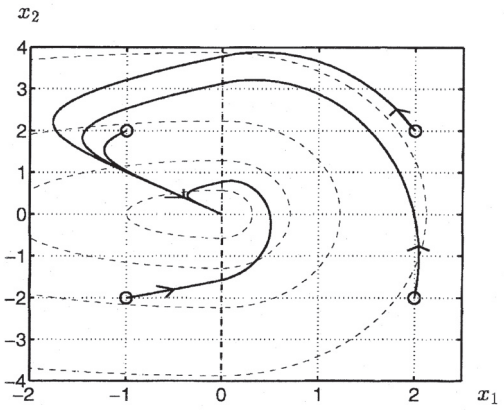

Figure 4.3: Trajectories of the piecewise affine system with level sets.

4.5.2 A more general set-up

The above examples show that there is no need to have a common Lyapunov function for each smooth subsystem to show asymptotic stability. There are several relaxations possible to the construction of a common quadratic Lyapunov function:

- One can require that the derivative $L_{f_i(x)}V(x)$ of $V(x) = x^{\mathrm{T}}Px$ is only negative in the region where the subsystem is active.
- One can use multiple Lyapunov functions, say $V_i(x) = x^{\mathrm{T}}P_ix$, for each submodel as in Example 4.5.2.
- One can require that the Lyapunov function $V_i(x) = x^{\mathrm{T}}P_ix$ is only positive definite in its active region.

How this works in a computationally easy manner is explained next. If the ith subsystem can only be active when $x \in \mathcal{C}_i$, then the Lyapunov function V_i should decrease only along solutions of the dynamics $\dot{x} = A_ix$ in the region $\mathcal{C}_i$. For switched linear systems and using (quadratic) Lyapunov functions [131] one can use the so-called S-procedure to get mathematically and computationally tractable forms.

Suppose we are looking for a $V(x) = x^{\mathrm{T}}Px$ such that $x^{\mathrm{T}}[A_i^{\mathrm{T}}P + PA_i]x < 0$ for $x \in \mathcal{C}_i$ and $x \neq 0$. A sufficient condition for this is to find a function $S_i(x)$ such that $S_i(x) \geqslant 0$ when $x \in \mathcal{C}_i$ such that there is a $\beta \geqslant 0$ satisfying

$$x^{\mathrm{T}}A_i^{\mathrm{T}}Px + x^{\mathrm{T}}PA_ix + \beta S_i(x) < 0 \tag{4.14}$$

for all $x \in \mathbb{R}^n$. Since $S_i(x)$ might be negative outside $\mathcal{C}_i$, (4.14) is less conservative than $x^{\mathrm{T}}[A_i^{\mathrm{T}}P + PA_i]x < 0$ for $x \notin \mathcal{C}_i$. In particular, if we can find a function $S_i(x)$ of the form $x^{\mathrm{T}}S_ix$ for some matrix

S_i with the property $S_i(x) \geqslant 0$ for $x \in \mathcal{C}_i$, then we get a nice LMI formulation: find $\beta_i \geqslant 0$ and $P > 0$ such that

$$A_i^{\mathrm{T}} P + P A_i + \beta_i S_i < 0. \tag{4.15}$$

Note that we should aim at finding S_i such that $\mathcal{C}_i \subseteq \Omega_{S_i} := \{x \in \mathbb{R}^n \mid x^{\mathrm{T}} S_i x > 0\}$, but we want to keep the set Ω_{S_i} as small as possible to reduce conservatism of the analysis. As the matrices S_i depend only on the region $\mathcal{C}_i$, these are usually fixed before (4.15) is solved in β and P. So, we fix the S_i and then try to solve the LMI (4.15).

The following lemma makes this S-procedure more formal.

Lemma 4.5.3 [37] *Let T_0 and T_1 be two symmetric matrices and consider the following two conditions*

$$x^{\mathrm{T}} T_0 x > 0 \text{ whenever } x^{\mathrm{T}} T_1 x \geqslant 0 \text{ and } x \neq 0$$

and

$$\text{there is a } \beta \geqslant 0 \text{ such that } T_0 - \beta T_1 > 0.$$

The second condition always implies the first condition. Moreover, if there is some x_0 such that $x_0^{\mathrm{T}} T_1 x_0 > 0$, then the first also implies the second.

The second relaxation mentioned above is the use of multiple Lyapunov function as was illustrated by Example 4.5.2. There we designed $V_i(x) = x^{\mathrm{T}} P_i x$ for each dynamics $\dot{x} = A_i x$ such that V_i decreases in the active region $\mathcal{C}_i$ and V_σ is continuous over the boundary. In [131] this has been worked out for piecewise affine systems of the form

$$\dot{x} = A_i x + a_i, \text{ when } E_i x \geqslant e_i, \ i \in I. \tag{4.16}$$

Note that $\mathcal{C}_i$ is given by $\{x \in \mathbb{R}^n \mid E_i x \geqslant e_i\}$. Here we only consider the piecewise linear case which means that all the $a_i = 0$ (note that this makes the origin an equilibrium of the system) and moreover, we assume that $e_i = 0$, which means that all cells contain the origin at the boundary. In this case, we consider so-called *conewise linear systems* as all cells are convex cones and all dynamics in the modes are linear. We refer to [131] for the extension. This means we study

$$\dot{x} = A_i x, \text{ when } E_i x \geqslant 0, \ i \in I. \tag{4.17}$$

We assume that the cells form a partitioning of the state space and have pairwise disjoint interior. Trajectories are assumed not to have sliding modes (otherwise one has to adapt the reasoning by including the modes or using a convexity based argument like at the end of Section 4.4.1). Hence, we do only consider solutions to the system that satisfy for all $t > 0$ $\dot{x}(t) = A_i x(t)$ for some i for which $x(t) \in \mathcal{C}_i$, $i \in I$. Moreover, we assume that there exist matrices F_i, F_j such that $F_i x = F_j x$ for all $x \in \mathcal{C}_i \cap \mathcal{C}_j$. In [130] there is some discussion on how to find the matrices F_i.

Note that we can take the matrices S_i in the S-procedure by suitably selecting a matrix U_i with nonnegative entries and then setting $S_i = E_i^{\mathrm{T}} U_i E_i$. Note that U_i is the "multi-dimensional" equivalent of the β_i in Lemma 4.5.3. Indeed, since $E_i x \geqslant 0$ when $x \in \mathcal{C}_i$ it is obvious that $x^{\mathrm{T}} S_i x \geqslant 0$ when $x \in \mathcal{C}_i$. That solves one problem and gives rise to finding $P_i > 0$ and U_i with nonnegative entries such that

$$A_i^{\mathrm{T}} P_i + P_i A_i + E_i^{\mathrm{T}} U_i E_i < 0 \tag{4.18}$$

to guarantee decrease of $V_i(x) = x^{\mathrm{T}} P_i x$ when system $\dot{x} = A_i x$ is active (i.e., when $x \in \mathcal{C}_i$). To guarantee also continuity of the *piecewise quadratic Lyapunov function* $V : \mathbb{R}^n \to \mathbb{R}$ defined as

$$V(x) = V_i(x) = x^{\mathrm{T}} P_i x, \text{ when } x \in \mathcal{C}_i, \tag{4.19}$$

we must have $x^{\mathrm{T}} P_i x = x^{\mathrm{T}} P_j x$ for all $x \in \mathcal{C}_i \cap \mathcal{C}_j$. Hence, there must be a relationship between P_i and P_j. Suppose that $x \in \mathcal{C}_i \cap \mathcal{C}_j$ implies that $Fx = 0$ (note that we can take $F_{ij} := F_i - F_j$), one choice of this relationship is (note that $P_i = P_j$ is also a choice, but leading to a quadratic Lyapunov function), $P_i = P_j + F_{ij}^{\mathrm{T}} T_{ij} + T_{ij} F_{ij}$ as can easily be verified. A more general parameterization is taking $P_i = F_i^{\mathrm{T}} T F_i$, $i \in I$ for some symmetric matrix T.

Finally, we can reduce conservatism by noting that $x^{\mathrm{T}} P_i x$ should only be positive when $x \in \mathcal{C}_i$ and $x \neq 0$. Applying the S-procedure in the usual way, leads to the following theorem:

Theorem 4.5.4 *If one can find symmetric matrices T, W_i and U_i with U_i and W_i having nonnegative entries and such that $P_i := F_i^{\mathrm{T}} T F_i$ satisfy*

$$A_i^{\mathrm{T}} P_i + P_i A_i + E_i^{\mathrm{T}} U_i E_i < 0, \ i = 1, \ldots, N \tag{4.20}$$

and

$$P_i - E_i^{\mathrm{T}} W_i E_i > 0, \ i = 1, \ldots, N \tag{4.21}$$

then every continuous piecewise continuously differentiable trajectory (without sliding modes) of (4.17) *tends to zero exponentially.*

Note that since these equations are linear in the matrices that we are looking for and the nonnegativity conditions on the entries of the matrices U_i and W_i are (miniature) linear matrix inequalities, one can use efficient LMI solvers to search for solutions. Note that one does not have to take T, W_i and U_i full matrices. One could reduce the solution space by only allowing diagonal matrices.

Remark 4.5.5 Another approach to guarantee continuity of V (which does not involve the construction of the matrices F_i, $i = 1, \ldots, N$) is based upon the selection of matrices Z_{ij} of full column rank (meaning that the columns are independent) with $\mathcal{C}_i \cap \mathcal{C}_j \subseteq \mathrm{im} Z_{ij}$ for $(i, j) \in \mathcal{S}$, where

$$\mathcal{S} := \{(i, j) \in \{1, \ldots, N\} \times \{1, \ldots, N\} \mid i \neq j \text{ and } \mathcal{C}_i \cap \mathcal{C}_j \neq \{0\}\}.$$

By $\mathrm{im} Z_{ij}$ we mean the image or span of the matrix Z_{ij}, i.e., $\mathrm{im} Z_{ij} = \{v \mid v = Z_{ij} w \text{ for some vector } w\}$. Hence, the matrices Z_{ij} model the "switching planes" ($\mathcal{C}_i \cap \mathcal{C}_j$) between cells in an image representation. The theorem above remains to be valid if one searches directly for matrices P_i, $i = 1, \ldots, N$ (without using the parameterization involving the matrices F_i and the T-matrix) and imposing additionally the condition

$$Z_{ij}^{\mathrm{T}} [P_i - P_j] Z_{ij} = 0, \qquad (i, j) \in \mathcal{S}$$

as this guarantees continuity of V as well (why?).

Example 4.5.6 This example (sometimes called the "flower system") is taken from [131] and considers the piecewise linear system

$$A_1 = A_3 = \begin{pmatrix} -0.1 & 1 \\ -5 & -0.1 \end{pmatrix}, \ A_2 = A_4 = \begin{pmatrix} -0.1 & 5 \\ -1 & -0.1 \end{pmatrix}.$$

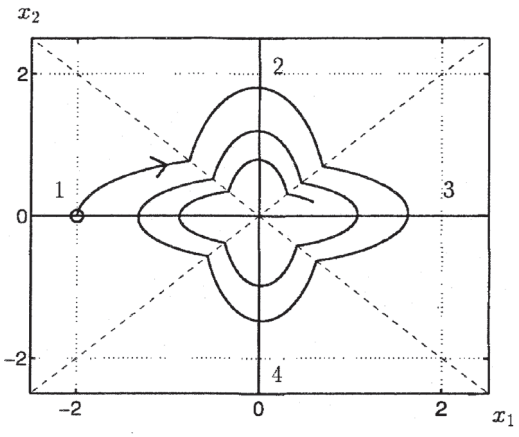

Figure 4.4: Partitioning and a trajectories of the "flower system."

See Figure 4.4 for the partitioning of the state space. The cell boundaries are dashed and a trajectory of the system starting in $(-2, 0)^{\mathrm{T}}$ is indicated by the solid line.

Looking for matrices E_i and F_i as outlined above yields

$$E_1 = -E_3 = \begin{pmatrix} -1 & 1 \\ -1 & -1 \end{pmatrix}, \quad E_2 = -E_4 = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \quad ,$$

and F_i can be taken as $[E_i^{\mathrm{T}} I]^{\mathrm{T}}$, where I denotes the two by two identity matrix. Looking for a solution to the set of LMIs in Theorem 4.5.4 yields

$$P_1 = P_3 = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}, \quad P_2 = P_4 = \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}.$$

Note that relaxation (4.21) is not used.

The level curves of the obtained Lyapunov function are shown in Figure 4.5.

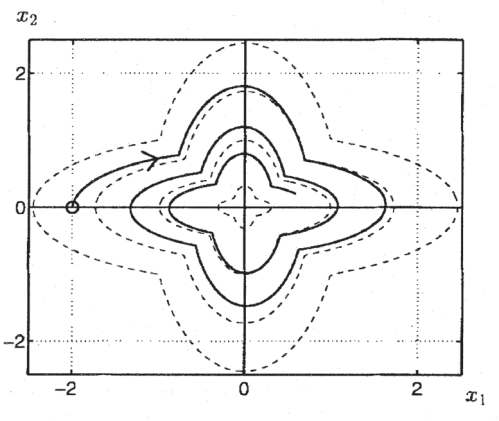

Figure 4.5: The level curves of the computed Lyapunov function.

Remark 4.5.7 There exists a connection between the above stability theory and the absolute stability theory for Lur'e systems containing a piecewise linearity in the feedback loop (or any other static nonlinearity with sector bounds). For Lur'e systems one has the famous circle and Popov criteria for stability (see, e.g., [137]). The proof of the circle criterion is actually based on the use of a common Lyapunov function of the form $V(x) = x^{\mathrm{T}}Px$ for all nonlinearities in the sector. The Popov criterion uses a more involved Lyapunov function, which turns out to be piecewise quadratic Lyapunov functions in certain cases. These piecewise quadratic Lyapunov functions can easily be found via the LMI formulations above, which extends for PWA systems the original Popov criterion.

4.6 Solving problem B: General approach based on multiple Lyapunov functions

Another relaxation beyond continuity of the multiple Lyapunov functions at the switching moments is proposed by Branicky [40]. The idea hinges on the fact that the underlying subsystems given by $f_1(x)$ and $f_2(x)$ (assume there are two of them for the moment) are (globally) asymptotically stable (note that this could be relaxed by only requiring decrease of the Lyapunov functions when the corresponding dynamics can be active). Let $V_1(x)$ and $V_2(x)$ be their respective Lyapunov functions. Since we assume that no common Lyapunov function exists (cf. Theorem 4.4.2), this means that the switched system is not GUAS and (asymptotic) stability only holds for particular classes of switching signals σ. Let the switching times be given by t_k, $k = 0, 1, 2, \ldots$. The question remains for which classes we can prove asymptotic stability of the switched system.

The reasoning in the previous sections was mainly based on the case when the values of V_1 and V_2 coincide at all switching times, i.e., $V_{\sigma(t_{k-1})}(x(t_k)) = V_{\sigma(t_k)}(x(t_k))$ for all $k = 1, 2, \ldots$. Then V_σ is a continuous Lyapunov function for the switched system (with the switching restriction as given) and asymptotic stability follows (cf. Example 4.5.2). This case will be considered later when Lyapunov functions for the individual submodels are used to design switching signals that make sure that during the switch the values of the Lyapunov functions coincide (see Section 5.2.2 below).

In general the function V_σ will be discontinuous. Since V_i is a Lyapunov function when submodel i is active, it may increase when it is not active (See Figure 4.6). However, if we add conditions on the values of the ith Lyapunov function at the beginning of each time interval on which the ith submodel becomes active, we can obtain asymptotic stability anyway.

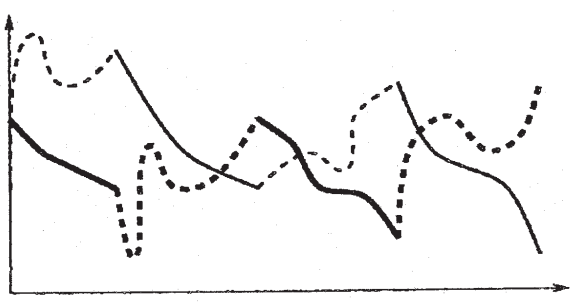

Figure 4.6: Multiple Lyapunov functions.

Theorem 4.6.1 *Consider the switched system (4.1) with all submodels $\dot{x} = f_i(x)$, $i = 1, \ldots, N$ being globally asymptotically stable and let V_i be the corresponding Lyapunov functions. Suppose that there is a class $\mathcal{K}$ function ρ with the property that for every pair of switching times (t_k, t_l), $k < l$ such that $\sigma(t_k) = \sigma(t_l) = i$ and $\sigma(t_m) \neq i$ for $t_k < t_m < t_l$, we have $V_i(x(t_l)) - V_i(x(t_k)) \leqslant -\rho(\|x(t_k)\|)$, then the switched system is globally asymptotically stable.*

See [40] for a proof. Note that in the formulation of the theorem we assume the σ to be right continuous, i.e., $\lim_{t \downarrow t_0} \sigma(t) = \sigma(t_0)$ for all t_0.

In general this theorem is difficult to apply as we need to have information of the trajectories of the system at switching time instants. In the classical Lyapunov theory in principle we do not need trajectory information. However, in the special case that the switching is constrained in a certain way, things can become easier. In the previous sections we have considered the case where the switching was determined on the basis of the state variable only and continuity was imposed of the Lyapunov functions across the switching surfaces. Theorem 4.6.1 can be exploited in other settings as well by using, e.g., knowledge of the (minimal or average) dwell times in certain submodels (the duration that a certain dynamics remains active) [124, 197].

4.7 Solving problem B: Switching signals with restricted dwell times

For reasons of completeness, we briefly discuss another method for solving problem B. This particular method relies on so-called *dwell time* restrictions of the switching signal σ in (4.1). The dwell time restrictions impose limitations on how long the switched system will be in a particular mode. For instance, one can adopt the *minimal dwell time* condition that imposes that the times of two subsequent mode switchings are separated by at least $\tau_{\mathrm{d}} > 0$ time units. Note that the times of mode switches are given by the times at which the switching signal σ is discontinuous. Of course, if you are switching between asymptotically stable subsystems and τ_{d} is sufficiently large (for instance, to guarantee that the state norm at a switching time is strictly smaller than the state norm at the previous switching time), global asymptotic stability of (4.1) is obtained. In other words, if the switching between asymptotically stable subsystems is sufficiently slow, then the switched system is asymptotically stable as well. Here, we will actually consider a more general result based on *average dwell time* restrictions.

The concept of *average dwell time* was introduced in [124]. To introduce this concept formally, let a switching signal $\sigma : \mathbb{R}^+ \to \{1, \ldots, N\}$ be given. The number of discontinuities of σ on an interval (t, T) is denoted by $N_\sigma(t, T)$. We say that σ has the average dwell time τ_{a} if there exists a positive number N_0 such that

$$N_\sigma(t, T) \leqslant N_0 + \frac{T - t}{\tau_a} \text{ for all } T \geqslant t \geqslant 0. \tag{4.22}$$

Note that when $N_0 = 0$ this implies that σ has no discontinuities at all (why?) and hence, there is no mode switching. When $N_0 = 1$ we have that σ cannot switch twice on interval of length smaller than τ_{a}. Hence, in the latter case we recover the condition of minimal dwell time τ_{a}. Generally speaking, if we discard the first N_0 switchings, then the average time between consecutive switches is at least τ_{a}.

Based on average dwell time restrictions, other stability formulations of switched systems (4.1) are available:

Theorem 4.7.1 *Consider the switched system* (4.1)*. Suppose that there exist continuously differentiable ($\mathcal{C}^1$) functions $V_i : \mathbb{R}^n \to \mathbb{R}^+$, $i = 1, \ldots, N$ and constants $0 < a < b$, $p \in \mathbb{N}$, $\mu \geqslant 1$ and $\lambda_0 > 0$ such*

that for all $i = 1, \ldots, N$

- $a\|x\|^p \leqslant V_i(x) \leqslant b\|x\|^p$ *for all* $x \in \mathbb{R}^n$*;*
- $\frac{\partial V_i}{\partial x} f_i(x) \leqslant -2\lambda_0 V_i(x)$ *for all* $x \in \mathbb{R}^n$*;*
- $V_i(x) \leqslant \mu V_j(x)$ *for all* $i, j \in \{1, \ldots, N\}$ *and all* $x \in \mathbb{R}^n$*.*

Then the switched system (4.1) *is globally asymptotically stable for every switching signal* σ *with average dwell time*

$$\tau_{\mathrm{a}} > \frac{\ln \mu}{2\lambda_0}$$

(and N_0 *arbitrary.)*

See the book [147] for the proof and more discussion on this (interesting) topic. The second condition in the above theorem states that V_i is a Lyapunov function for the i-the subsystem $\dot{x} = f_i(x)$ and hence, this subsystem is necessarily globally asymptotically stable.

4.8 Summary

This chapter treated stability of switched systems. After recalling the basic Lyapunov theory for smooth systems, we have shown that globally uniformly asymptotic stability (GUAS) for switched systems under arbitrary switching is equivalent to the existence of a common Lyapunov function. This result could be used for switched linear systems via linear matrix inequalities (LMIs) – which can be solved efficiently – by using a common quadratic Lyapunov function. However, as shown by an example, conservatism is introduced by restricting the search for a quadratic one. Moreover, in case of switched linear systems, we presented a result that indicated that if all linear subsystems are stable and the corresponding matrices commute pairwise, then there exists a common quadratic Lyapunov function that proves GUAS.

In case the switching functions are not arbitrary, but have a specific structure, the existence of a common Lyapunov function might be too conservative. Two examples illustrated that this is indeed the case and hinted upon relaxations. Especially, in the case of state-dependent switching several relaxations were proposed: only requiring that a Lyapunov function decreases in the region when the dynamics is active and using multiple Lyapunov functions. For piecewise affine systems and piecewise quadratic Lyapunov functions, these conditions could be formulated in LMIs again by using the S-procedure. These approaches insisted on the continuity of the multiple Lyapunov functions across the switching planes. One relaxation based on quite general theory was presented to relax this condition. These results are harder to transform into easily verifiable conditions like LMIs.

At the end of the chapter, we also introduced stability results for switched systems that have dwell time restrictions. In case sufficiently large lower bounds can be guaranteed on the minimal or average durations that the system is in a mode, stability of the switched system can still be guaranteed (under certain additional technical conditions).

The results of this chapter will be used in the next chapter for the synthesis of stabilizing controllers.

Chapter 5

Switched control

Some parts of this chapter are based on [212].

5.1 Introduction

In some sense, the use of hybrid controllers for continuous-time systems is classical. Indeed, we can look at *variable structure control*, *sliding mode control*, *relay control*, *gain scheduling*, *bang-bang control in time optimal control laws with a bounded control constraint set*, and even *fuzzy control* as examples of hybrid control schemes. The common characteristic of all these control schemes is their *switching* nature; on the basis of the evolution of the plant (the to-be-controlled system) and/or the progress of time the hybrid controller switches from one control regime to another.

5.1.1 Brockett's necessary condition: A motivation for switched controllers

As a motivation of hybrid control schemes, we start off by a very convincing example of the *non-holonomic*[1] *integrator*

$$\begin{aligned} \dot{x} &= u \\ \dot{y} &= v \\ \dot{z} &= xv - yu \end{aligned} \tag{5.1}$$

where u and v denote the controls.

The non-holonomic integrator is the prototype of a nonlinear system, which is *controllable*, but nevertheless cannot be asymptotically stabilized using *continuous* state feedback (static or dynamic). The reason is that the non-holonomic integrator violates what is called *Brockett's necessary condition* [45], which is formulated in the following theorem.

Theorem 5.1.1 *Consider the control system*

$$\dot{x} = f(x,u), \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad f(0,0) = 0, \tag{5.2}$$

[1] A non-holonomic system is a system for which the number of instantaneous degrees of freedom is different from the number of configuration parameters or states. Consider, e.g., a wheel, which has 2 degrees of freedom (changing the angle of the wheel, and moving forward/backward), but 3 configuration parameters or states (the angle, and the x and the y position of the wheel).

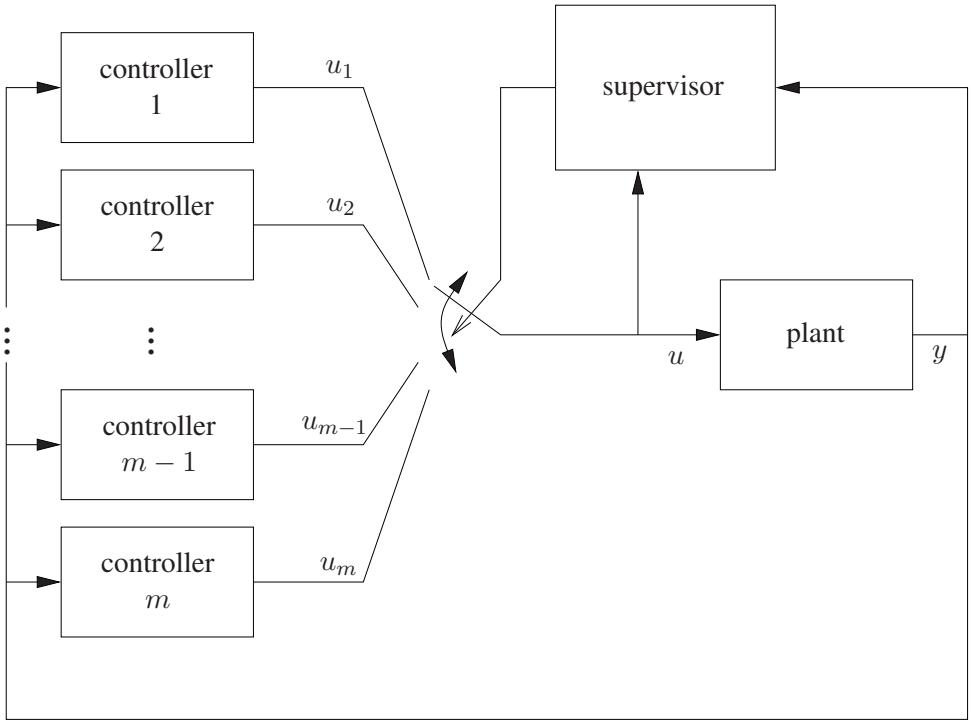

Figure 5.1: Switching control.

where f is a C^1 function. If (5.2) is asymptotically stabilizable (around $x = 0$) using a continuous feedback law $u = \alpha(x)$, then the image of every open neighborhood of $(x, u) = (0, 0)$ under f contains an open neighborhood of $x = 0$.

Remark 5.1.2 Since we allow for *continuous* feedback laws (instead of C^1 or locally Lipschitz feedback laws) some care should be taken in connection with the *existence and uniqueness* of solutions of the closed-loop system.

It is readily seen that the non-holonomic integrator does not satisfy the condition mentioned in Theorem 5.1.1 (Brockett's necessary condition), despite the fact that it is controllable (as can be rather easily seen (see also Section 5.6)). Indeed, $(0, 0, \varepsilon)$ cannot belong to the image of f for any $\varepsilon \neq 0$ (as $f_1(\tilde{x}, \tilde{u}) = 0$ and $f_2(\tilde{x}, \tilde{u}) = 0$ for some $\tilde{x}$, $\tilde{u}$ imply that $f_3(\tilde{x}, \tilde{u}) = 0$), and so the non-holonomic integrator cannot be stabilized by a time-invariant *continuous* feedback. In Section 5.6 we will give one way how to stabilize (5.1) by *switched* control.

In fact, the non-holonomic integrator is an example of a whole class of systems, sharing the same property. For example, actuated mechanical systems subject to *non-holonomic kinematic constraints* (like rolling without slipping) do not satisfy Brockett's necessary condition, but are often controllable. Hence, for this type of system we have to resort to switched or hybrid control schemes to stabilize them!

5.1.2 Switching logic

One way of formalizing *switching control* for a continuous-time input-state-output system is by means of Figure 5.1. Here, the supervisor has to decide on the basis of the input and output signals of the system,

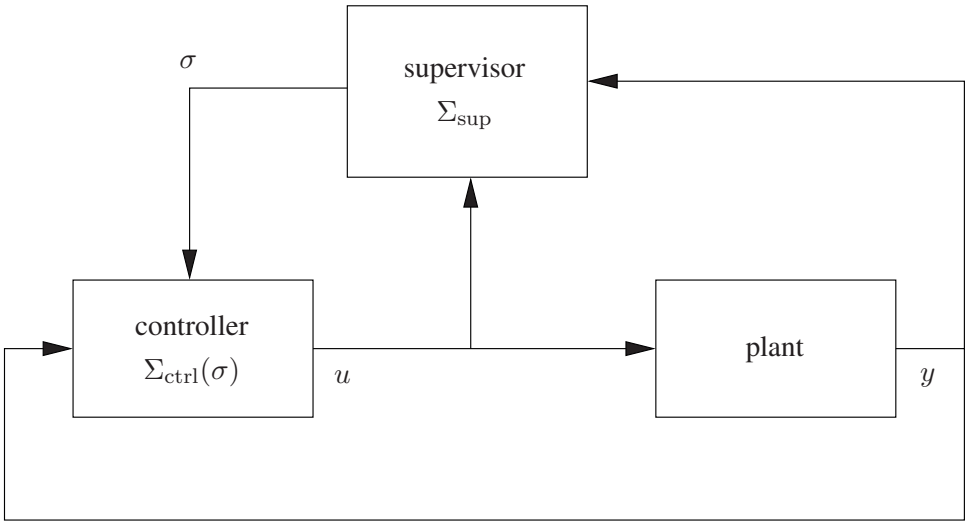

Figure 5.2: Switching control with shared controller state.

and possibly external (reference) signals or the progress of time, *which* of the, ordinary, continuous-time controllers is applied to the system. The figure indicates a finite number of controllers; however, we could also have a countable set of controllers.

In many cases the different controllers can all be given the same state space (shared state variables), which leads to the simpler switching control structure given in Figure 5.2. In this case the supervisor generates a discrete symbol σ corresponding to a controller $\Sigma_{\text{ctrl}}(\sigma)$ producing the control signal $u = u_\sigma$. An example of a switching control scheme was already encountered in Section 1.6.5 (the supervisor model), in which case the supervisor is a finite automaton, which produces control signals on the basis of the measurement of the output of the plant and the discrete state of the automaton.

The supervisor in Figures 5.1 and 5.2 is sometimes called a *switching logic*. One of the main problems in the design of a switching logic is that usually it is not desirable to have "*chattering*", that is, very fast switching. There are basically two ways to suppress chattering: one is sometimes called *hysteresis switching logic* and the other *dwell-time switching logic*.

- **Hysteresis switching logic**

  Choose $h > 0$, and assume that π_σ denotes some performance criterion depending on σ and some signal derived from the input and the output signals of the continuous-time plant, which is used in the switching logic. We assume that the rate of variation of π_σ with respect to time is bounded (this implies that π_σ is a continuous function of time.) Suppose at some time t_0, the supervisor Σ_{sup} has just changed the value of σ to q. Then σ is held *fixed* at the value q unless and until there exists some $t_1 > t_0$ at which $\pi_p + h < \pi_q$ (or $(1 + h)\pi_p < \pi_q$, if a scale-invariant criterion is needed) for some p. If this occurs then σ is set equal to p. Clearly, because of the threshold parameter h no infinitely fast switching will occur. The idea is similar to the use of a *boundary layer* around the switching surface in order to avoid chattering in sliding mode control (see Section 5.5).

- **Dwell-time switching logic**

  The basic idea is to have some fixed time $\tau > 0$, called the dwell-time, such that, once a symbol σ is chosen by the supervisor it will remain constant for at least a time equal to τ. There are many

versions of dwell-time switching logics.

We refer to [168] for further developments and other switching logic schemes. From a general hybrid system point of view a continuous-time system (plant) controlled by switching logic is a hybrid system with continuous state corresponding to the continuous state of the plant and the continuous state of the controller, together with discrete states residing in the controller. Usually, the transition from one discrete state to another will *not* give rise to a jump in the continuous state of the plant, although it could give rise to a jump in the continuous state of the controller ("resetting" the controller). Seen from this point of view it is clear that switching control, although being a very general concept, certainly does not cover all possible hybrid control strategies for continuous-time systems.

One should realize that also in the case that the plant itself displays discrete behavior and is described by some hybrid model, the coupling of plant and switched controller gives rise to a hybrid closed-loop system. In the next sections we will give some design methods to various settings of hybrid control problems.

5.2 Stabilization of switched systems

In this section we will give two approaches to the stabilization problem as referred to as Problem C in Chapter 4, i.e., how to construct a switching signal σ that stabilizes the switched system

$$\dot{x} = f_\sigma(x),$$

where $\{f_p \mid p = 1, \ldots, N\}$ is a family of sufficiently smooth vector fields from $\mathbb{R}^n$ to $\mathbb{R}^n$.

5.2.1 Quadratic stabilization of switched systems via a single Lyapunov function

We will start by considering the problem of *quadratic stabilization* of a multi-modal linear system

$$\dot{x} = A_i x, \quad i \in I, \quad x \in \mathbb{R}^n, \tag{5.3}$$

where $I = \{1, \ldots, N\}$ by selecting a suitable switching law σ that selects at every time one of the subsystems (as a function of time, state or other variables). To be more precise, the problem is to find such a switching rule such that the controlled system has a single quadratic Lyapunov function $x^{\mathrm{T}}Px$. It is not difficult to show that this problem can be solved if there exists a convex combination of the matrices A_i, $i \in I$ that is stable. To verify this claim, assume that the matrix

$$A := \sum_{i=1}^{N} \alpha_i A_i \quad (\alpha_i \geqslant 0, \ \textstyle\sum_{i=1}^{N} \alpha_i = 1) \tag{5.4}$$

is stable. Take a positive definite matrix Q, and let P be the solution of the Lyapunov equation $A^{\mathrm{T}}P + PA = -Q$. Since A is stable, P is positive definite. Let now x be an arbitrary nonzero vector. From $x^{\mathrm{T}}(A^{\mathrm{T}}P + PA)x < 0$ it follows that

$$\sum_i \alpha_i [x^{\mathrm{T}}(A_i^{\mathrm{T}}P + PA_i)x] \ < \ 0. \tag{5.5}$$

Because all the α_i are nonnegative, it follows that at least one of the numbers $x^{\mathrm{T}}(A_i^{\mathrm{T}}P + PA_i)x$ must be negative, and in fact we obtain the stronger statement

$$\bigcup_{i\in I}\{x \mid x^{\mathrm{T}}(A_i^{\mathrm{T}}P + PA_i)x \leqslant -\frac{1}{N}x^{\mathrm{T}}Qx\} \;=\; \mathbb{R}^n \tag{5.6}$$

where N denotes the number of modes.

It is clear how a switching rule for (5.3) may be chosen such that asymptotic stability is achieved; e.g., the rule

$$i(x) \;:=\; \arg\min_i \; x^{\mathrm{T}}(A_i^{\mathrm{T}}P + PA_i)x \tag{5.7}$$

is an obvious choice.

The minimum rule (5.7) indeed leads to an asymptotically stable system, which is well-defined if one extends the number of discrete states so as to include possible sliding modes. To avoid sliding modes, the minimum rule may be adapted so that the regions in which different modes are active will overlap rather than just touch. E.g., a modified switching rule (based on hysteresis switching logic) may be chosen which is such that the system will stay in mode i as long as the continuous state x satisfies

$$x^{\mathrm{T}}(A_i^{\mathrm{T}}P + PA_i)x \leqslant -\frac{1}{2N}x^{\mathrm{T}}Qx. \tag{5.8}$$

When the bound in (5.8) is reached, a switch will take place to a new mode j that may, e.g., be determined by the minimum rule. At the time at which the new mode j is entered, the number $x^{\mathrm{T}}(A_j^{\mathrm{T}}P + PA_j)x$ must be less than or equal to $-\frac{1}{N}x^{\mathrm{T}}Qx$. Suppose that the switch to mode j occurs at continuous state x_0. The time until the next mode switch is given by

$$\tau_j(x_0) \;=\; \min\{t \geqslant 0 \mid x_0^{\mathrm{T}}e^{tA_j^{\mathrm{T}}}[A_j^{\mathrm{T}}P + PA_j + \frac{1}{2N}Q]e^{tA_j}x_0 \geqslant 0\} \tag{5.9}$$

(taken to be infinity if the set over which the minimum is taken is empty). Note that $\tau_j(x_0)$ is positive and satisfies $\tau_j(\alpha x_0) = \tau_j(x_0)$ for all nonzero real α. It can be shown (see, e.g., [212, Sec. 6.2.4]) that there is a positive lower bound to the time the system will stay in mode j, namely

$$T_j \;:=\; \min\{\tau_j(x_0) \mid \|x_0\| = 1,\;\; x_0^{\mathrm{T}}(A_i^{\mathrm{T}}P + PA_i)x_0 = -\frac{1}{2N}x_0^{\mathrm{T}}Qx_0\}. \tag{5.10}$$

So under this switching strategy the system is asymptotically stable and there will be no chattering.

Above we have shown that in case one can find a convex combination of the linear dynamics being stable, then we can design a quadratically stabilizing state-dependent switching law. In case we have only two linear submodels ($N = 2$), then this condition is necessary as well, as proven by Feron [89].

Theorem 5.2.1 [89] *If there exists a quadratically stabilizing state-dependent switching law for the switched linear system (5.3) with $N = 2$, then the matrices A_1 and A_2 have a stable convex combination.*

5.2.2 Stabilization of switched systems via multiple Lyapunov functions

Stabilization has been done in the previous subsections by using a Lyapunov function $V(x) = x^{\mathrm{T}}Px$. In [216] a state-dependent stabilizing control law has been determined by switching suitably between $\dot{x} = A_1x$ and $\dot{x} = A_2x$ (actually in [216] even a more general situation has been considered). The idea

is to find a function $V_1(x) = x^{\mathrm{T}} P_1 x$ that decreases along solution trajectories of the systems $\dot{x} = A_1 x$ in a region $\mathcal{X}_1$. Similarly, we try to find $V_2(x) = x^{\mathrm{T}} P_2 x$ for the second dynamics in a region $\mathcal{X}_2$. In case $\mathcal{X}_1 \cup \mathcal{X}_2$ is the whole state space, one can try to switch between the two linear submodels to satisfy the conditions of Theorem 4.6.1 and thus guarantee asymptotic stability.

One way of doing that is to find P_1 and P_2 such that they satisfy the coupled conditions:

$$x^{\mathrm{T}}(P_1 A_1 + A_1^{\mathrm{T}} P_1)x < 0 \text{ when } x^{\mathrm{T}}(P_1 - P_2)x \geqslant 0,\ x \neq 0 \tag{5.11}$$

and

$$x^{\mathrm{T}}(P_2 A_2 + A_2^{\mathrm{T}} P_2)x < 0 \text{ when } x^{\mathrm{T}}(P_2 - P_1)x \geqslant 0,\ x \neq 0. \tag{5.12}$$

If these conditions hold, then we can use the switching law

$$\sigma(t) = \arg\max_i \{V_i(x(t)) \mid i = 1, 2\} \tag{5.13}$$

as the function V_σ will then be continuous and decrease along solutions of the closed-loop switched systems. Consequently, asymptotic stability is guaranteed.

By applying the S-procedure (Lemma 4.5.3), we get the following sufficient conditions that solve the stabilization problem: Find $\beta_1 \geqslant 0$, $\beta_2 \geqslant 0$, $P_1 > 0$ and $P_2 > 0$ such that

$$-P_1 A_1 - A_1^{\mathrm{T}} P_1 + \beta_1(P_2 - P_1) > 0 \tag{5.14}$$

$$-P_2 A_2 - A_2^{\mathrm{T}} P_2 + \beta_2(P_1 - P_2) > 0. \tag{5.15}$$

Hence, stabilization can be done by solving a set of coupled matrix inequalities.

5.3 Stabilization of piecewise and switched linear systems with continuous inputs

In switched linear systems the design problem was given a collection of linear systems find a switching sequence between them such that the closed-loop systems is stable. The mode dynamics itself could not be influenced. Hence, one could say that only "discrete inputs" (determining only the switching and taking only a finite number of values $\{1, \ldots, N\}$) are present and no "continuous inputs" (taking values in a subset of some set $\mathbb{R}^m$). However, often we encounter, e.g., a switched linear system with inputs of the form

$$\dot{x} = A_i x + B_i u,\ i \in I = \{1, \ldots, N\}$$

Suppose we would like to construct both a switching strategy σ and feedback controllers $u = K_i x$ that are activated when submodel i is active. Hence, this means that we have to find $K_1, \ldots, K_N$ such that

$$\dot{x} = (A_i + B_i K_i)x,\ i \in I \tag{5.16}$$

can be stabilized via suitable switching.

5.3.1 Stabilization of a switched linear system under arbitrary switching

If arbitrary switching is required and we *know* the mode that is currently active, a sufficient condition is to find a common *quadratic* Lyapunov function $V(x) = x^{\mathrm{T}}Px$ for some positive definite matrix P. Hence, this means that we would have to find $K_1, \ldots, K_N$ and P such that the conditions of Section 4.4.2 are satisfied, i.e.,

$$(A_i + B_iK_i)^{\mathrm{T}}P + P(A_i + B_iK_i) < 0 \text{ for all } i = 1, \ldots, N \text{ and } P > 0 \tag{5.17}$$

These equations can be transformed into Linear Matrix Inequalities (LMIs) by pre-multiplying and post-multiplying by P^{-1} yielding

$$P^{-1}(A_i + B_iK_i)^{\mathrm{T}} + (A_i + B_iK_i)P^{-1} < 0 \text{ for all } i = 1, \ldots, N \text{ and } P^{-1} > 0$$

which gives rise to the LMIs

$$ZA_i^{\mathrm{T}} + A_iZ + Y_i^{\mathrm{T}}B_i^{\mathrm{T}} + B_iY_i < 0 \text{ for all } i = 1, \ldots, N \text{ and } Z > 0,$$

where we have replaced P^{-1} by Z and K_iP^{-1} by Y_i. Once we have solved these LMIs, P can be recovered as Z^{-1} and $K_i = Y_iZ^{-1}$.

Hence, if these LMIs are feasible, we have constructed a collection of feedback control laws $u = K_ix$, $i = 1, \ldots, N$ that make the switched linear system GUAS (Globally Uniformly Asymptotically Stable) under arbitrary switching (note that we have to know the mode as we have to use feedback law $u = K_ix$ when subsystem i is active).

5.3.2 Design of switched feedback and switching sequence

Another situation arises when we both have to find the feedback control laws $u = K_ix$, $i = 1, \ldots, N$ *and* we have to construct a switching sequence such that the resulting system is asymptotically stable. To do so, we could apply the results in Section 5.2. Then we have to find feedback gains $K_1, \ldots, K_N$ such that one of the conditions of Section 5.2 is satisfied. For simplicity, we take the case where $N = 2$. Hence, one condition is to find K_1, K_2 and $\alpha \in [0, 1]$ such that $\alpha(A_1 + B_1K_1) + (1 - \alpha)(A_2 + B_2K_2)$ is stable. Note that this can be written as find $\alpha \in [0, 1]$, P positive definite and gains K_1 and K_2 such that

$$[\alpha(A_1 + B_1K_1) + (1-\alpha)(A_2 + B_2K_2)]^{\mathrm{T}}P + P[\alpha(A_1 + B_1K_1) + (1-\alpha)(A_2 + B_2K_2)] < 0, \tag{5.18}$$

which is a matrix inequality.

Another condition can be obtained using the results of Section 5.2.2 leading to another matrix inequality: Find $\beta_1 \geqslant 0$, $\beta_2 \geqslant 0$, P_1 and P_2 positive definite matrices and gains K_1 and K_2 such that

$$-P_1(A_1 + B_1K_1) - (A_1 + B_1K_1)^{\mathrm{T}}P_1 + \beta_1(P_2 - P_1) > 0 \tag{5.19}$$

$$-P_2(A_2 + B_2K_2) - (A_2 + B_2K_2)^{\mathrm{T}}P_2 + \beta_2(P_1 - P_2) > 0. \tag{5.20}$$

It is interesting to transform these inequalities into linear matrix inequalities by certain transformations and change of variables to make this approach computationally feasible. This might be non-trivial or even impossible.

5.3.3 Design of switched feedback

Sometimes, the switching structure has already been given, e.g., if the system is specified by a piecewise linear model of the form

$$\dot{x} = A_i x + B_i u, \text{ when } x \in \mathcal{X}_i, \tag{5.21}$$

where the collection $\mathcal{X}_i$, $i = 1, \ldots, N$ is such that $\bigcup_{i=1}^{N} \mathcal{X}_i = \mathbb{R}^n$ and the intersection of two cells $X_i \cap X_j$ for $i \neq j$ is a (lower-dimensional) boundary.

Now the switching structure is given, but we observe that we still have an input u to influence the dynamics. Hence, we can try to construct a state feedback controller $u = g(x)$ to make the closed-loop system stable. It is most natural to take a state feedback $u = g(x)$ that complies with the switching structure of the piecewise linear model (5.21)

$$u = K_i x \text{ when } x \in \mathcal{X}_i. \tag{5.22}$$

This yields the closed-loop dynamics

$$\dot{x} = (A_i + B_i K_i)x, \text{ when } x \in \mathcal{X}_i. \tag{5.23}$$

One way of guaranteeing closed loop stability is to apply the techniques for quadratic stabilization using a single or multiple Lyapunov function presented in Chapter 4 (see also [130, 165]). We will work out the case where we use a single Lyapunov function $V(x) = x^{\mathrm{T}} P x$ (hence, we would like to make the closed-loop systems quadratically stable) and the cells $\mathcal{X}_i$ are given by sets of the form $\{x \in \mathbb{R}^n \mid E_i x \geqslant 0\}$. In particular, we can use the stability results of Theorem 4.5.4, which lead to finding $K_1, \ldots, K_N$, P positive definite and symmetric U_i with nonnegative entries such that the nonlinear matrix inequalities

$$(A_i + B_i K_i)^{\mathrm{T}} P + P(A_i + B_i K_i) + E_i^{\mathrm{T}} U_i E_i < 0, \; i = 1, \ldots, N \tag{5.24}$$

are satisfied. Note that one has to take care of the possible occurrence of sliding modes, which might require to also use convex combinations of the controller at the switching boundaries (cf. the discussion at the end of Section 4.4.1).

5.4 Time-controlled switching and pulse width modulation (PWM) control of switched systems

In the previous section the switching between (controlled) subsystems was performed on the basis of the state variable. In many situations the switching can also be determined by the time or a clock variable. In this section we will consider such time-based switching strategies of which pulse width modulation is one of the classical ones often appearing in many electronic circuits.

We already know that if a dynamical system is switched between several subsystems, the stability properties of the system as a whole may be quite different from those of the subsystems. Also with time-based switching this is the case as can be illustrated by the following calculation.

Consider a system that follows the dynamics $\dot{x} = A_1 x$ for a period $\frac{1}{2}\varepsilon$, then switches to $\dot{x} = A_2 x$ for again a period $\frac{1}{2}\varepsilon$, then switches back, and so on. The hybrid automaton that describes this system is

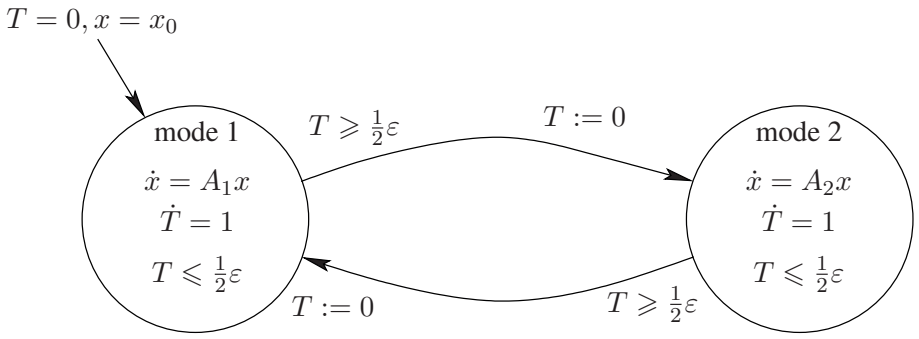

Figure 5.3: A hybrid automaton that periodically switches between two linear dynamics.

given in Figure 5.3. Let us consider a time point t_0 at which the system begins a period in mode 1, with continuous initial state x_0. At time $t_0 + \frac{1}{2}\varepsilon$, the state variable has evolved to

$$x(t_0 + \tfrac{1}{2}\varepsilon) \;=\; \exp(\tfrac{1}{2}\varepsilon A_1)x_0 \;=\; x_0 + \frac{\varepsilon}{2}A_1 x_0 + \frac{\varepsilon^2}{8}A_1^2 x_0 + \cdots$$

(this is a consequence of the general rule $\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$). At time $t_0 + \varepsilon$, we get

$$\begin{aligned} x(t_0 + \varepsilon) &= (I + \frac{\varepsilon}{2}A_2 + \frac{\varepsilon^2}{8}A_2^2 + \cdots)(I + \frac{\varepsilon}{2}A_1 + \frac{\varepsilon^2}{8}A_1^2 + \cdots)x_0 \\ &= (I + \varepsilon[\tfrac{1}{2}A_1 + \tfrac{1}{2}A_2] + \frac{\varepsilon^2}{8}[A_1^2 + A_2^2 + 2A_2A_1] + \cdots)x_0. \end{aligned}$$

If we compare the above expression to the power series development for $\exp[\varepsilon(\frac{1}{2}A_1 + \frac{1}{2}A_2)]$ which is given by

$$\exp[\varepsilon(\tfrac{1}{2}A_1 + \tfrac{1}{2}A_2)] \;=\; I + \varepsilon[\tfrac{1}{2}A_1 + \tfrac{1}{2}A_2] + \frac{\varepsilon^2}{8}[A_1^2 + A_2^2 + A_1A_2 + A_2A_1] + \cdots$$

we see that the constant and the linear term are the same, whereas the quadratic term is off by an amount of $A_1A_2 - A_2A_1$ (the "commutator" of A_1 and A_2). So the difference between the solution of the switched system and that of the smooth system $\dot{x} = (\frac{1}{2}A_1 + \frac{1}{2}A_2)x$ on a time interval of length ε is of the order ε^2. If we let ε tend to zero, then on any fixed time interval the solution of the switched hybrid system tends to the solution (with the same initial condition) of the "averaged" system

$$\dot{x} \;=\; (\tfrac{1}{2}A_1 + \tfrac{1}{2}A_2)x \;. \tag{5.25}$$

In particular, the stability properties of the switched system will for small ε be determined by the stability properties of the averaged system (5.25), that is, by the location of the eigenvalues of $\frac{1}{2}A_1 + \frac{1}{2}A_2$. Now it is well-known that the eigenvalues are nonlinear functions of the matrix entries and so it may well happen that the matrices A_1 and A_2 are both Hurwitz (all eigenvalues have negative real parts) whereas the matrix $\frac{1}{2}A_1 + \frac{1}{2}A_2$ is unstable, or vice versa. This is illustrated in the following example.

Example 5.4.1 Consider the switched hybrid system of Figure 5.3 with

$$A_1 \;=\; \begin{bmatrix} -0.5 & 1 \\ 100 & -1 \end{bmatrix}, \quad A_2 \;=\; \begin{bmatrix} -1 & -100 \\ -0.5 & -1 \end{bmatrix} \;. \tag{5.26}$$

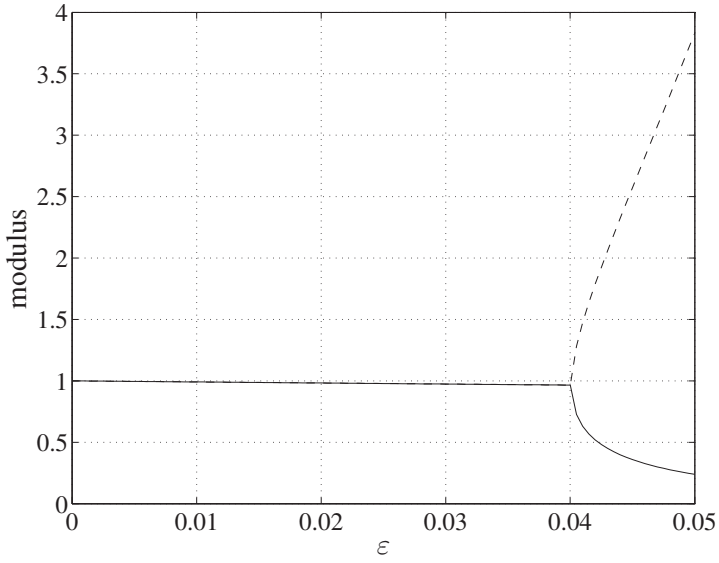

Figure 5.4: Modulus of the eigenvalues of $\exp(\frac{1}{2}\varepsilon A_2)\exp(\frac{1}{2}\varepsilon A_1)$ as a function of ε.

Although A_1 and A_2 are both unstable, the matrix

$$\tfrac{1}{2}(A_1 + A_2) \;=\; \begin{bmatrix} -0.75 & -49.5 \\ 49.75 & -1 \end{bmatrix}$$

is Hurwitz. Therefore, the switched system should be stable if the frequency of switching is sufficiently high. The switching frequency that is minimally needed can be found by computing the eigenvalues of the mapping $\exp(\frac{1}{2}\varepsilon A_2)\exp(\frac{1}{2}\varepsilon A_1)$ as a function of ε. Stability is achieved when both eigenvalues are inside the unit circle. For the present case, it turns out that a switching frequency of at least 50 Hz is needed (cf. Figure 5.4 — note that in the system every $\frac{1}{2}\varepsilon$ time units a switch occurs). The plot in Figure 5.5 shows a trajectory of the system when it is switched at 100 Hz (i.e., for $\varepsilon = 0.02$).

Instead of staying in mode 1 and in mode 2 for a time period of length $\frac{1}{2}\varepsilon$ each, we can also consider the situation in which mode 1 is followed during a time $h\varepsilon$ and mode 2 during a time interval $(1-h)\varepsilon$, where h is a number between 0 and 1. By the same reasoning as above, the behavior of such systems on fixed time intervals is well approximated by that of the system $\dot{x} = A_h x$, where $A_h = hA_1 + (1-h)A_2$. The choice of the parameter h influences the system dynamics and so h might be considered as a control input; the averaging analysis requires, however, that if h varies, it should be on a time scale that is much slower than the time scale at which the switching takes place. When mode 1 corresponds to "power on" and mode 2 is "power off", the parameter h is known as the *duty ratio*. In power electronics the use of switches is popular because theoretically it provides a possibility to regulate power without loss of energy. Fast switching is a particular form of this method of regulation.

Remark 5.4.2 It is interesting to observe that we obtained a differential equation of the form $\dot{x} = [A_2 + h(A_1 - A_2)]x$ with input $h \in [0,1]$ as an approximation of a switched system.

The above analysis can be applied broader. Indeed, in a more general form one encounters control systems of the following form:

$$\dot{x} = f(x,u), \quad x \in X, \quad u \in U, \tag{5.27}$$

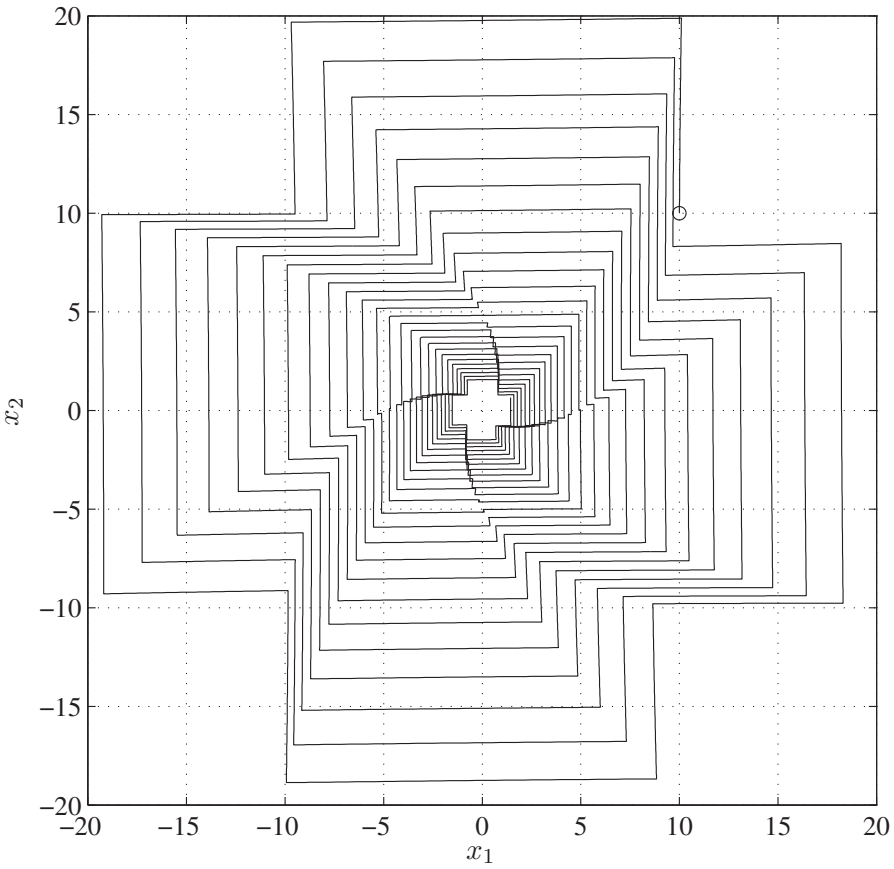

Figure 5.5: Trajectory of the switched system defined by (5.26) for a simulation period of 3 time units and with switching frequency of 100 Hz. The initial point $(10, 10)$ is marked by an "o".

where X is some continuous space, say $\mathbb{R}^n$, while the input space U is a *finite* space (or, more generally, the product of a continuous and a finite space). An appealing class of examples consists of *power converters*, see, e.g., the Boost converter example of Section 1.6.6, where the discrete inputs correspond to the switches being closed or open. Since the input space is finite, such systems can be considered as a special case of hybrid systems. In such cases, it makes sense to relate the behavior of these hybrid systems to the behavior of an associated control system with *continuous* inputs, using the notion of *Pulse Width Modulation* (PWM). Consider for concreteness a control system with $U = \{0, 1\}$, that is,

$$\dot{x} = f(x, u), \quad u \in \{0, 1\}. \tag{5.28}$$

The PWM control scheme for (5.28) can be described as follows. Consider a so-called *duty cycle* of fixed small duration Δ. On every duty cycle the input u is switched exactly one time from 1 to 0. The fraction of the duty cycle on which the input holds the fixed value 1 is known as the *duty ratio* and is denoted by α. The duty ratio may also depend on the state x (or, more precisely, on the value of the state sampled at the beginning of the duty cycle). On every duty cycle $[t, t + \Delta]$ the input is therefore defined by

$$\begin{array}{rcll} u(\tau) & = & 1, & \text{for } t \leqslant \tau < t + \alpha\Delta \\ u(\tau) & = & 0, & \text{for } t + \alpha\Delta \leqslant \tau < t + \Delta \end{array} \tag{5.29}$$

It follows that the state x at the end of this duty cycle is given by

$$x(t+\Delta) = x(t) + \int_t^{t+\alpha\Delta} f(x(\tau),1)d\tau + \int_{t+\alpha\Delta}^{t+\Delta} f(x(\tau),0)d\tau \tag{5.30}$$

The ideal averaged model of the PWM controlled system is obtained by letting the duty cycle duration Δ go to zero. In the limit, the above formula (5.30) then yields

$$\dot{x}(t) = \lim_{\Delta\to 0} \frac{x(t+\Delta)-x(t)}{\Delta} = \alpha f(x(t),1) + (1-\alpha) f(x(t),0) \tag{5.31}$$

The duty ratio α can be thought of as a *continuous-valued input*, taking its values in the *interval* $[0,1]$. Hence for sufficiently small Δ the trajectories of the continuous-time system (5.31) will be close to trajectories of the hybrid system (5.28). Note that in general the full behavior of the hybrid system (5.28) will be richer than that of (5.31); not every trajectory of (5.28) can be approximated by trajectories of (5.31).

The PWM control scheme originates from the control of switching power converters, where usually it is reasonable to assume that the switches can be opened ($u=0$) and closed ($u=1$) sufficiently fast at any ratio $\alpha \in [0,1]$.

A similar analysis can be performed for a control system (5.28) with an *arbitrary* finite input space $U \subset \mathbb{R}^m$. Then the PWM-associated continuous-time system has continuous inputs α taking values in the *convex hull* of U.

5.5 Sliding mode control

A classical example of switching control is *variable structure control*. Consider a control system described by equations of the form

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

where u is the (scalar) control input. Suppose that a switching control scheme is employed that uses a state feedback law $u(t) = g_+(x(t))$ when the scalar variable $y(t)$ defined by $y(t) = \xi(x(t))$ is positive, and a feedback $u(t) = g_-(x(t))$ whenever $y(t)$ is negative. Writing $f_+(x) = f(x, g_+(x))$ and $f_-(x) = f(x, g_-(x))$, we obtain a dynamical system that obeys the equation $\dot{x}(t) = f_+(x(t))$ on the subset where $\xi(x)$ is positive, and that follows $\dot{x}(t) = f_-(x(t))$ on the subset where $\xi(x)$ is negative. The surface $\{x|\,\xi(x)=0\}$ is called the *switching surface*. Since now typically the controller is switching, we use Utkin's equivalent control definition. See Section 3.2. This yields

$$\dot{x} \in F(x) \tag{5.32}$$

with

$$F(x) = \begin{cases} \{f(x,g_+(x))\}, & \text{for } \xi(x) > 0, \\ f(x,U(x)) = \{f(x,u) \mid u \in U(x)\}, & \text{for } \xi(x) = 0, \\ \{f(x,g_-(x))\}, & \text{for } \xi(x) < 0. \end{cases} \tag{5.33}$$

where $U(x)$ is the convex hull of $g_+(x)$ and $g_-(x)$, i.e., $U(x) := \{\lambda g_+(x) + (1-\lambda) g_-(x) \mid \lambda \in [0,1]\}$.

The extension to multi-variable inputs u and outputs y is straightforward.

In *sliding mode control*, or briefly, *sliding control*, the theory of variable structure systems is actually taken as the starting point for control design. Indeed, some scalar variable s depending on the state x and

possibly the time t is considered such that the system has a "desired behavior" whenever the constraint $s = 0$ is satisfied. The set $s = 0$ is called the *sliding surface*. Now a control law u is sought such that the following *sliding condition* is satisfied (see, e.g., [195])

$$\frac{1}{2}\frac{d}{dt}s^2 \leqslant -\alpha|s| \tag{5.34}$$

where α is a strictly positive constant. Essentially, (5.34) states that the squared "distance" to the sliding surface, as measured by s^2, decreases along all system trajectories. Thus it constrains trajectories to point towards the sliding surface. Note that the left-hand side can be also written as $s\dot{s}$; a more general sliding condition can therefore be formulated as designing u such that $s\dot{s} < 0$ for $s \neq 0$.

The methodology of sliding control can be best demonstrated by giving a typical example.

Example 5.5.1 Consider the second-order system

$$\ddot{q} = f(q, \dot{q}) + u$$

where u is the control input, q is the scalar variable of interest (e.g., the position of a mechanical system), and the dynamics described by f (possibly non-linear or time-varying) is not exactly known, but estimated by $\hat{f}$. The estimation error on f is assumed to be bounded by some known function $F(q, \dot{q})$:

$$|\hat{f} - f| \leqslant F \ .$$

In order to have the system track some desired trajectory $q(t) = q_{\mathrm{d}}(t)$, we define a sliding surface $s = 0$ with

$$s = \dot{e} + \lambda e \ ,$$

where $e = q - q_{\mathrm{d}}$ is the tracking error, and the constant $\lambda > 0$ is such that the error dynamics $s = 0$ has desirable properties (e.g., sufficiently fast convergence of the error to zero). We then have

$$\dot{s} = f + u - \ddot{q}_{\mathrm{d}} + \lambda\dot{e} \ .$$

The best approximation $\hat{u}$ of a continuous control law that would achieve $\dot{s} = 0$ is therefore

$$\hat{u} = -\hat{f} + \ddot{q}_{\mathrm{d}} - \lambda\dot{e} \ .$$

Note that $\hat{u}$ can be interpreted as our best estimate of the equivalent control. In order to satisfy the sliding condition (5.34) we add to $\hat{u}$ a term that is *discontinuous* across the sliding surface:

$$u = \hat{u} - k\,\mathrm{sgn}(s) \ . \tag{5.35}$$

where sgn is the sign function defined in (1.10). By choosing k in (5.35) large enough we can guarantee that the sliding condition (5.34) is satisfied. Indeed, we may take $k = F + \alpha$. Note that the control discontinuity k across the sliding surface $s = 0$ increases with the extent of parametric uncertainty F.

The occurrence of a sliding mode may not be desirable from an engineering point of view; depending on the actual implementation of the switching mechanism, a quick succession of switches may occur, which may lead to increased wear and to high-frequency vibrations in the system. Hence, for the actual implementation of sliding mode control — in order to avoid very fast switching — we usually have to embed the sliding surface in a thin *boundary layer*, such that switching will only occur outside this boundary layer (hysteresis switching logic). Furthermore, the discontinuity $\mathrm{sgn}(s)$ in the control law can

be further smoothened, say, by replacing sgn by a steep sigmoid function. Of course, such modifications may deteriorate the performance of the closed-loop system.

One of the main advantages of sliding mode control, in addition to its conceptual simplicity, is its robustness with respect to uncertainty in the system data. A possible disadvantage is the excitation of unmodeled high-frequency modes.

5.6 Stabilization of non-holonomic systems using hybrid feedback control

Let us now return to the non-holonomic integrator of Section 5.1:

$$\dot{x} = u \tag{5.36}$$
$$\dot{y} = v \tag{5.37}$$
$$\dot{z} = xv - yu \ . \tag{5.38}$$

For the non-holonomic integrator, we could consider the following sliding mode control law (taken from [32]):

$$u = -x + y\,\mathrm{sgn}(z) \tag{5.39}$$
$$v = -y - x\,\mathrm{sgn}(z) \ . \tag{5.40}$$

Now consider a Lyapunov function for the (x, y) subspace:

$$V(x, y) = \frac{1}{2}(x^2 + y^2) \ .$$

The time-derivative of V along the trajectories along the closed-loop system (5.36)–(5.40) is negative:

$$\dot{V} = -x^2 + xy\,\mathrm{sgn}(z) - y^2 - xy\,\mathrm{sgn}(z) = -(x^2 + y^2) = -2V \ . \tag{5.41}$$

Hence, we can already deduce that the x and y variables will converge to 0. Let us now consider the z variable. We have

$$\dot{z} = xv - yu = -(x^2 + y^2)\,\mathrm{sgn}(z) = -2V\,\mathrm{sgn}(z) \ . \tag{5.42}$$

Since V does not depend on z and since it is a positive function of the time t, the absolute value of z will thus decrease during the evolution of the system and reach 0 provided that the following inequality holds:

$$2\int_0^\infty V(\tau)d\tau > |z(0)| \ . \tag{5.43}$$

If (5.43) holds, then z will reach 0 in finite time; and afterwards it will remain 0, since by (5.42), all trajectories are directed towards the surface $z = 0$ (the sliding surface). If (5.43) is an equality instead of an inequality, then $z(t)$ converges to 0 only as $t \to \infty$. If

$$2\int_0^\infty V(\tau)d\tau < |z(0)| \ ,$$

then for $t \to \infty$ $z(t)$ will converge to some constant non-zero value that has the same sign as $z(0)$.

From (5.41) it follows that

$$V(t) = V(0)e^{-2t} = \frac{1}{2}(x^2(0) + y^2(0))e^{-2t} \ .$$

Substituting this expression in (5.43) and integrating, we find that the condition for the system to be asymptotically stable is that

$$\frac{1}{2}(x^2(0) + y^2(0)) \geqslant |z(0)| \ .$$

Hence, if the initial conditions of the system do *not* belong to the parabolic region defined by

$$\mathcal{P} = \left\{ (x, y, z) \,\middle|\, \frac{1}{2}(x^2 + y^2) < |z| \right\} \ ,$$

then the control (5.39)–(5.40) asymptotically stabilizes the system (5.36)–(5.38).

If the initial state is inside $\mathcal{P}$, we can use any control law that first steers it outside. In fact, any nonzero constant control can be applied. Indeed, if we take $u(t) = u_0$ and $v(t) = v_0$ for all t and for some non-zero constants u_0 and v_0, we have

$$\begin{aligned} x(t) &= u_0 t + x_0 \\ y(t) &= v_0 t + y_0 \\ z(t) &= x(t)v_0 - y(t)u_0 = t(x_0 v_0 - y_0 u_0) + z_0 \ , \end{aligned}$$

and then $\frac{1}{2}(x^2(t)+y^2(t))$ is a quadratic function of the time t, whereas $|z(t)|$ is linear. Hence, eventually the state will always leave the region $\mathcal{P}$ in finite time, after which we switch to the control given by (5.39)–(5.40). The resulting control strategy is inherently hybrid in nature: first, we apply the constant control $u(t) = u_0$, $v(t) = v_0$ as long the state belongs to the parabolic region $\mathcal{P}$, and if the system is outside this region, then we switch to the sliding mode control (5.39)–(5.40).

Note that a possible drawback of the above hybrid feedback scheme is caused by the application of sliding mode control: in principle, we will get chattering around the sliding surface $z = 0$. However, this can be remedied with the usual tools in sliding mode control (such as putting a boundary layer around the sliding surface, or approximating the sgn function by a steep sigmoid). Alternative hybrid feedback schemes are also given in [212].

5.7 Summary

To motivate the need for switched and hybrid control, we started by presenting a smooth system in the form of a non-holonomic integrator and showed that there does not exist a continuous feedback law that stabilizes this system. However, these type of systems can be stabilized by using switched controllers, as shown by the sliding mode controller at the end of the chapter. This is a strong motivation for the use of hybrid control as it can overcome problems that cannot be solved by the continuous or smooth controllers that we are used to. Also it is known that the performance of control loops can be improved by using switched controllers instead of smooth controllers [92].

We introduced the concept of switched control and described a variety of switched control strategies for several situations. In Sections 5.2.1 and 5.2.2 by designing the switching law (the "discrete control" so to speak) as a function of the state-variable that stabilizes switched linear systems. In Section 5.3.1 we computed a switched state feedback controller ("the continuous control" part of the system) such

that the resulting closed-loop systems is globally uniform asymptotically stable (GUAS) under arbitrary switching. Hence, by knowing the active mode this controller is "robustly stable" with respect to switching strategies. In Section 5.3.2 we both computed the switching strategy and the switched state feedback controller that stabilizes the switched linear system. Finally, in Section 5.3.3 we computed a stabilizing switched feedback law for a piecewise affine system, which means that now the switching law is given and only the continuous control part could be designed. A common denominator of these control problems is that the switching law is state-dependent. In Section 5.4 we presented approaches for stabilization of switched controllers using time-controlled switching and pulse width modulation control. Finally, we introduced a classical control scheme based on sliding mode control, which successfully stabilized the non-holonomic integrator.

In conclusion: although in some of the presented solutions there is quite some elegance and structure in the design, there currently seems to be no general design methodology for constructing hybrid feedback stabilization schemes. E.g., we have presented a sliding mode control scheme for the non-holonomic integrator, but it is not easy to extract a systematic design methodology from this example. Furthermore, the actual proofs that the proposed hybrid feedback schemes do work are typically complicated and rather ad hoc. This indicates that the field of hybrid controller design is still widely open and thus much work to do! In the next chapter we present optimization-based approaches to hybrid controller design, which is on one hand general in structure (so there are some systematic procedures involved), but on the other hand involves often optimization problems of which the solutions (either off-line or on-line in, e.g., model predictive approaches) are hard to obtain.

Chapter 6

Optimization-based control

In this chapter we consider some methods for control design classes of hybrid systems using numerical optimization techniques. We focus on optimal control and on model predictive control, and we also briefly discuss game-theoretic control approaches. Section 6.1 is based on [61] and Section 6.2 is based on [27].

6.1 Optimal control of a class of hybrid systems

In this section we consider an approach for optimal control of a class of hybrid systems as it is presented in [59–61]. Although the scope of this approach is general, it is largely motivated by the structure of many manufacturing systems. In these systems, discrete entities (referred to as jobs) move through a network of work centers, which process the jobs so as to change their physical characteristics according to certain specifications. Associated with each job is a *temporal state* and a *physical state*. The temporal state of a job evolves according to event-driven dynamics and includes information such as the waiting time or departure time of the job at the various work centers. The physical state evolves according to time-driven dynamics modeled through differential (or difference) equations, which, depending on the particular problem being studied, describe changes in such quantities as the temperature, size, weight, chemical composition, or some other measure of the quality of the job. The interaction of time-driven with event-driven dynamics leads to a natural trade-off between temporal requirements on job completion times and physical requirements on the quality of the completed jobs. For example, while the physical state of a job can be made arbitrarily close to a desired quality target, this usually comes at the expense of long processing times resulting in excessive inventory costs or violation of constraints on job completion deadlines. Our objective, therefore, is to formulate and solve optimal control problems associated with such trade-offs.

6.1.1 A hybrid model for a class of manufacturing systems

Consider a manufacturing system in which the switch from one mode to another corresponds to processing a new job i ($i = 1, \ldots, N$). We shall limit ourselves to a single-stage process modeled as a single-server queueing system (cf. Figure 6.1). The objective is to process N jobs. The server processes one job at a time on a first-come first-served non-preemptive basis (i.e., once a job begins service, the server cannot be interrupted, and will continue to work on it until the operation is completed). Jobs

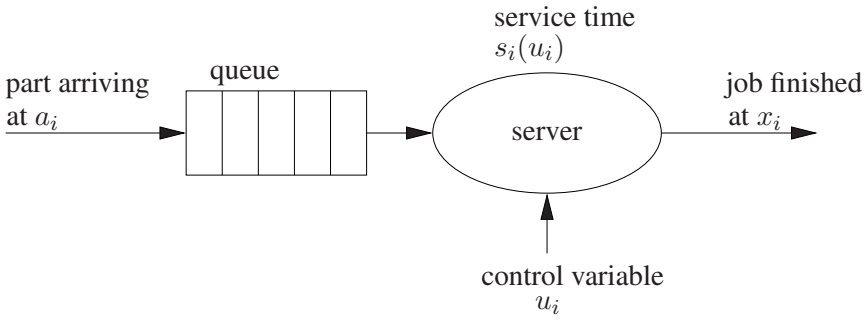

Figure 6.1: A single-stage single-server manufacturing process.

arriving when the server is busy wait in a queue whose capacity is larger than N (so no buffer overflow will take place). As job i is being processed, its physical state, denoted by $z_i \in \mathbb{R}^n$, evolves according to time-driven dynamics of the general form

$$\dot{z}_i = g_i(z_i, u_i, t) \qquad \text{with } z_i(\tau_i) = \zeta_i \tag{6.1}$$

where τ_i is the time instant at which the processing of job i begins, and ζ_i is the initial state at that time. The control variable u_i (assumed here to be scalar and not time-dependent for simplicity) is used to attain a final desired physical state corresponding to a target "quality level". Specifically, if the *service time* for the ith job is $s_i(u_i)$ and $\Gamma_i(u_i) \subseteq \mathbb{R}^n$ is a given set (e.g., a threshold above which z_i satisfies a desired "quality level"), then the control u_i is chosen to satisfy the stopping rule

$$s_i(u_i) = \min\{t \geqslant 0 \mid z_i(\tau_i + t) \in \Gamma_i(u_i)\} \tag{6.2}$$

with

$$z_i(\tau_i + t) = \int_{\tau_i}^{\tau_i + t} g_i(z_i, u_i, \nu) d\nu + \zeta_i,$$

and where u_i takes a fixed constant value during the interval $[\tau_i, \tau_i + t]$, and the minimum is assumed to exist.

The *temporal state* of the ith job is denoted by x_i, and represents the time when the job completes processing and departs from the system. Letting a_i be the arrival time of the ith job, the event-driven dynamics describing the evolution of the temporal state are given by the following "max-plus" recursive equation:

$$x_i = \max(x_{i-1}, a_i) + s_i(u_i) \tag{6.3}$$

where we set[1] $x_0 = -\infty$. It is assumed that the job arrival time sequence $a_1, a_2, \ldots, a_N$ is given (in [99] arrival times are considered to be controllable). The recursive relationship (6.3) is known in queueing theory as the Lindley equation [57].

Note that a situation where $a_{i+1} > x_i$ gives rise to an "idle period", in which case there is an interval $[x_i, a_{i+1}]$ on the temporal state axis during which the physical state is undefined.

Now we consider the following optimal control problem:

$$\min_{u_1, \ldots, u_N} \sum_{i=1}^{N} L_i(x_i, u_i) \tag{6.4}$$

[1] Hence, $x_1 = a_1 + s_1(u_1)$, i.e., the first job begins service as soon as it arrives.

subject to (6.1)–(6.3),

where $L(x_i, u_i)$ is a cost function associated with job i. Note that this formulation does not require an explicit cost on the physical state z_i, since (6.2) ensures that each job satisfies a given quality requirement, i.e., $z_i(\tau_i + s_i(u_i)) = z_i(x_i) \in \Gamma_i(u_i)$. Note that this stopping rule defines a separate optimization problem, which must be solved to obtain the service time and its derivative.

The problem defined above appears similar to classical discrete-time optimal control problems commonly found in the literature (e.g., [49]) except for two issues. First, the index i does not count time steps, but rather asynchronously departing jobs. Second, the presence of the max-function in the state equation (6.3) prevents us from using standard gradient-based techniques, since it introduces a non-differentiability at the point where $a_i = x_{i-1}$.

Regarding the first issue, note that the mathematical treatment of the recursive equation (6.3) is in fact no different than that of any other similar recursion where the index represents synchronized time steps as in classical discrete-time optimal control problems. Therefore, this issue is not really problematic.

Regarding the second issue, in [99, 179] it has been shown that the non-differentiability problem can be overcome in at least special cases of the problem formulated above, and that the max-function exhibits certain useful structural properties that can be exploited to simplify the analysis and lead to efficient numerical solutions. For the more general class of problems considered here, we will invoke ideas and results from non-differentiable calculus (e.g., [71]) to deal with the non-differentiability issue.

Example 6.1.1 To illustrate the use of the framework and problem formulation presented above, we outline below an optimal control problem for steel heating and annealing manufacturing processes involving a furnace integrated with plant-wide planning and scheduling operations; full details and solutions based on the methods presented in this section may be found in [67].

The set-up is such that individual steel "parts" (i.e., ingots or strips) of say a few 10s of meters are put into a furnace of 400-500 m length via a roller line. The strips undergo various operations to achieve certain metallurgical properties that define the quality of the finished products. In particular, the steel heating and annealing process is an important step that involves slowly heating and cooling strips to some desired temperatures. Before heating and cooling each strip, a higher level controller determines the furnace reference temperature (more generally, a "furnace heating profile") which the strip should follow, as well as the amount of time that this strip is held in a furnace. Raw material, (e.g., a cold-rolled strip) is put on a pay-off reel on the entry side of the line and runs through with a certain line speed.

The physical state of the ith strip in this process is denoted by z_i and represents the temperature at each point of the strip as it evolves through the heating furnace. The strip temperature is basically dependent on the *line speed* u_i, which usually remains constant during the process, and the *furnace reference temperature* F_i, which is pre-designed at a plant-wide planning level. The thermal process in the heating furnace can be represented by a nonlinear heat-transfer equation describing the dynamic response of each strip temperature so that the temporal change in heat energy at a particular location is equal to the transport heat energy plus the radiation heat energy as follows [135]:

$$\dot{z}_i(t) = -\frac{F_i - z_i(t_0)}{L} u_i + K_{\mathrm{s}}\big(F_i^4 - z_i^4(t)\big) \qquad \text{for } t \geqslant t_0 \ , \tag{6.5}$$

where the parameter K_{s} depends on the properties of the strip (such as specific heat and thickness) and where L is the furnace length, and t_0 the heating start time.

The temporal state of the ith strip consists of two variables, x_i and y_i, where x_i represents the time when the job starts processing at the furnace (i.e., when the strip is completely inside the furnace) and

y_i represents the time when the job completes processing and departs from the system. The need for two variables is due to the fact that we must distinguish between the starting time of the $(i+1)$th job and the completion time of the ith job (i.e., $x_{i+1} \neq y_i$), since each job is a continuous strip of a typical length, not a discrete entity. Letting a_i be the arrival time of the ith strip at the entrance of the furnace, the event-driven dynamics describing the evolution of these temporal states are given by

$$\begin{aligned} & x_i = \max(a_i, x_{i-1}) + s_1(u_i) \quad \text{and} \quad y_i = x_i + s_2(u_i) \\ & \text{subject to } u_{\min} \leqslant u_i \leqslant u_{\max} \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{6.6}$$

where $s_1(u_i)$ is the elapsed time for the whole body of the strip to enter the furnace, which is dependent on the length of the strip, and $s_2(u_i)$ is the processing time for each point of the strip to run through the furnace, which is dependent on the length of the furnace. In addition, $u_{\min}$ and $u_{\max}$ are the minimum and maximum allowable line speed respectively, and we assume that $x_0 = -\infty$. In this system, we consider two control objectives:

1. to reduce temperature errors with respect to the furnace reference temperature, and
2. to reduce the entire processing time for timely delivery using acceptable levels of line speed u_i.

Thus, the optimal control problem of interest is

$$\begin{aligned} & \min_{u_1,\ldots,u_N} \sum_{i=1}^{N} \big(\theta(u_i) + \phi(y_i)\big) \\ & \text{subject to (6.5)–(6.6)} \end{aligned}$$

where the function $\phi(y_i)$ is the cost related to jobs departing at time y_i (e.g., $\phi(y_i) = (y_i - d_i)^2$, such that a job departing after the due date d_i incurs a tardiness cost, and completing before its due date incurs an inventory (backlog) cost). The function $\theta(u_i)$ is selected so as to penalize the deviation of the ith strip temperature from the reference temperature, F_i:

$$\theta(u_i) = |F_i - z_i(L/u_i)|^2 + \beta \int_0^{L/u_i} (F_i - z_i(t))^2 dt$$

where L/u_i is the time each point of the strip stays in the furnace and β is a weighting factor.

6.1.2 Optimality conditions

We begin by invoking basic variational calculus techniques to study the minimization problem (6.4) subject to (6.3). As in standard discrete-time optimal control problems, we define the augmented cost

$$\bar{J}(x, \lambda, u) = \sum_{i=1}^{N} \big(L_i(x_i, u_i) + \lambda_i(\max(x_{i-1}, a_i) + s_i(u_i) - x_i)\big) \ , \tag{6.7}$$

where x and u are N-dimensional vectors for the temporal state and the control, and λ is an N-dimensional vector for the co-state sequence used to adjoin the temporal dynamics in (6.3) to the cost J in (6.4).

Throughout the rest of our analysis, we will make the following assumptions.

Assumption A1 : The one-step costs L_i and the service functions s_i are continuously differentiable for all i.

Assumption A2 : The service functions s_i are monotonically increasing for all i.

Ignoring for the moment the non-differentiabilities associated with the max-operation in the expression for $\bar{J}$, the standard first-order necessary conditions for optimality require that

$$\frac{\partial \bar{J}}{\partial u_i} = 0, \quad \frac{\partial \bar{J}}{\partial \lambda_i} = 0, \quad \frac{\partial \bar{J}}{\partial x_i} = 0 \quad \text{for } i = 1, \ldots, N \, .$$

The first equation above gives the stationarity condition

$$\frac{\partial L_i(x_i, u_i)}{\partial u_i} + \lambda_i \frac{ds_i(u_i)}{du_i} = 0 \ . \tag{6.8}$$

The second equation recovers the temporal state equation

$$x_i = \max(x_{i-1}, a_i) + s_i(u_i) \tag{6.9}$$

with initial condition $x_0 = -\infty$. Finally, the third equation gives the co-state equation

$$\lambda_i = \frac{\partial L_i(x_i, u_i)}{\partial x_i} + \lambda_{i+1} \frac{d \max(x_i, a_{i+1})}{dx_i} \tag{6.10}$$

with final boundary condition

$$\lambda_N = \frac{\partial L_N(x_N, u_N)}{\partial x_N} \ . \tag{6.11}$$

Equations (6.8)–(6.11) define a *two-point boundary-value problem* (TPBVP), the solution of which provides a control sequence satisfying the necessary conditions for optimality. TPBVPs are notoriously hard; in our case, matters are further complicated by the presence of the max-function in the co-state equation (6.10). This function is Lipschitz continuous, differentiable everywhere except at the single point where $x_i = a_{i+1}$ with

$$\frac{d \max(x_i, a_{i+1})}{dx_i} = \begin{cases} 0 & \text{if } x_i < a_{i+1} \\ 1 & \text{if } x_i > a_{i+1} \ . \end{cases} \tag{6.12}$$

As the system operates, the sequence of arrival and departure times defines a state trajectory (or sample path). On any sample path, the points where $x_i = a_{i+1}$ acquire special significance, since they are responsible for the non-differentiability of the max-function in the co-state equation. When such points are part of the optimal solution, the necessary conditions above cannot be used to establish optimality, and we must appeal to non-smooth optimization theory, and use the *generalized gradient*.

Definition 6.1.2 (Generalized gradient) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz continuous function of $u \in \mathbb{R}^n$, and let $S(u)$ denote the set of all sequences $\{u_m\}_{m=1}^{\infty}$ with $u_m \in \mathbb{R}^n$ that satisfy the following three conditions:

1. $u_m \to u$ as $m \to \infty$

2. the gradient $\nabla f(u_m)$ exists for all m
3. $\lim_{m\to\infty} \nabla f(u_m) = \phi$ exists.

Then the *generalized gradient* of f at u is denoted by $\partial f(u)$ and is defined as the convex hull of all limits ϕ corresponding to some sequence $\{u_m\}_{m=1}^{\infty}$ in $S(u)$.

The generalized gradient has the following three fundamental properties [71]:

1. $\partial f(u)$ is a nonempty, compact and convex set in $\mathbb{R}^n$,
2. $\partial f(u)$ is a singleton if and only if f is continuously differentiable in some open set containing u, in which case $\partial f(u) = \{\nabla f(u)\}$,
3. if u is a local minimum of f, then $0 \in \partial f(u)$.

The last property is an extension of the classical stationarity condition in (6.8), and becomes the first-order optimality condition in non-smooth optimization.

As described above, the necessary condition for the optimization of non-smooth Lipschitz functions is given in terms of the generalized gradient. Our task now, therefore, is to identify $\partial \bar{J}$. In order to do so, we introduce the following terminology that will be essential to all subsequent analysis:

Definition 6.1.3 An *idle period* is a time interval $(x_k, a_{k+1}]$ such that $x_k < a_{k+1}$ for any $k = 1, \ldots, N-1$.
A *busy period* is a time interval $(a_k, x_n]$ defined by a subsequence $\{k, k+1, \ldots, n\}$ such that

1. $x_{k-1} < a_k$
2. $x_i \geqslant a_{i+1}$ for all $i = k, \ldots, n-1$, and
3. $x_n < a_{n+1}$.

These terms are borrowed from classical queueing theory. An idle period is simply a time interval of strictly positive duration during which the server has no jobs to process, and a busy period is a time interval during which the server is processing jobs without any interruption caused by an empty input queue. A busy period, initiated at time a_k, must always follow an idle period, be followed by another idle period, and allow no other idle periods within it. We also set $a_{N+1} = \infty$ for consistency. The next term is introduced to capture an important special feature which we will show characterizes optimal sample paths for our problem.

Definition 6.1.4 A critical job with index i is one that satisfies $x_i = a_{i+1}$.

Note that a critical job corresponds precisely to the situation where the max-function is not differentiable in (6.10). Moreover, note that a critical job cannot end a busy period; however, a busy period may contain one or more critical jobs.

In order to identify the busy period structure and the locations of critical jobs within a busy period, we associate with every job i the following two indices

$$n(i) = \min\{n \geqslant i \mid x_n < a_{n+1}\}$$

$$m(i) = \min\{m \geqslant i \mid x_m \leqslant a_{m+1}\} \ .$$

In words, $n(i)$ is the index of the last job in the busy period containing job i. Regarding $m(i)$, if job i is critical or there are critical jobs between job i and the end of its busy period, then $m(i)$ is the index of the first such critical job; in this case, $m(i) < n(i)$ and we have $x_{m(i)} = a_{m(i)+1}$. If, on the other hand, job i is not critical and there are no critical jobs between job i and the end of its busy period, then $m(i)$ is the index of the job that ends the busy period, i.e., $m(i) = n(i)$.

The *m(i)=n(i)* case

This is the simpler of the two cases, where job i is not critical, there are no critical jobs between job i and the end of its busy period, and we have $\max(x_j, a_{j+1}) = x_j > a_{j+1}$ for all $j = i, \ldots, n(i) - 1$ and $\max(x_{n(i)}, a_{n(i)+1}) = a_{n(i)+1} > x_{n(i)}$. Therefore, all derivatives in the co-state equation (6.10) exist and we get

$$\lambda_i = \sum_{j=i}^{n(i)} \frac{\partial L_j}{\partial x_j} \ .$$

Then, $\bar{J}$ is locally continuously differentiable in u_i and the optimality condition (6.8) becomes

$$\bar{J}(u_i) \equiv \frac{\partial \bar{J}}{\partial u_i} = \frac{\partial L_i}{\partial u_i} + \frac{ds_i}{du_i} \sum_{j=i}^{n(i)} \frac{\partial L_j}{\partial x_j} = 0$$

where we have omitted the arguments of the functions. Clearly, the same result holds when there are critical jobs in the busy period containing job i, as long as these critical jobs precede job i in this busy period.

Thus, if critical jobs were to never occur on an optimal sample path (i.e., if $m(i) = n(i)$ for all $i = 1, \ldots, N$), then the function $\bar{J}$ would be differentiable at its minimum, the standard conditions for optimality would apply, and a numerical solution could be obtained by solving the TPBVP defined by (6.8)–(6.10).

The *m(i)<n(i)* case

Since, in general, $\bar{J}$ will exhibit the non-differentiabilities associated with critical jobs, it is necessary to study next the case where $m(i) < n(i)$. For any such job i on an optimal sample path, we have $\max(x_{m(i)}, a_{m(i)+1}) = x_{m(i)} = a_{m(i)+1}$ and the corresponding derivative in the co-state equation (6.10) does not exist. Hence, the derivative $\frac{\partial \bar{J}}{\partial u_i}$ also fails to exist. To obtain the generalized gradient in this case we proceed as follows. First, since jobs i and $m(i)$ are in the same busy period and $m(i) \geqslant i$, we have

$$x_{m(i)} = \max(x_i, a_i) + s_i(u_i) + \sum_{j=i+1}^{m(i)} s_j(u_j) \tag{6.13}$$

where the max accounts for the fact that job i may be the first in the busy period. Hence, we see that the control for job i affects the departure time of job $m(i)$. Now suppose that we fix all controls at their optimal values and perturb u_i. Recalling (6.12), the following one-sided derivatives exist:

$$\lim_{x_{m(i)} \uparrow a_{m(i)+1}} \frac{d}{dx_{m(i)}} \max(x_{m(i)}, a_{m(i)+1}) = 0 \tag{6.14}$$

$$\lim_{x_{m(i)} \downarrow a_{m(i)+1}} \frac{d}{dx_{m(i)}} \max(x_{m(i)}, a_{m(i)+1}) = 1 \ . \tag{6.15}$$

Conceptually, the first limit corresponds to the process of changing u_i so that $x_{m(i)}$ increases toward a fixed $a_{m(i)+1}$. Similarly, the second limit corresponds to the process of changing u_i so that $x_{m(i)}$ decreases toward $a_{m(i)+1}$ and the same is true for all other critical jobs between $m(i)$ and $n(i)$.

Consider (6.10) and note that

$$\frac{d\max(x_j, a_{j+1})}{dx_j} = 1 \quad \text{for all } j = i, \ldots, m(i) - 1 \,.$$

Then, combining (6.7) and (6.10), we get

$$\frac{\partial \bar{J}}{\partial u_i} = \frac{\partial L_i}{\partial u_i} + \frac{ds_i}{du_i} \cdot \left[\sum_{j=i}^{m(i)} \frac{\partial L_j}{\partial x_j} + \lambda_{m(i)+1} \frac{d\max(x_j, a_{j+1})}{dx_j} \right] \,.$$

By Assumption A2 and (6.13), $x_{m(i)}$ is monotonically increasing in $u(i)$ and, using (6.14)–(6.15), the preceding equation leads to the one-sided left derivative

$$\xi_i^- \equiv \left(\frac{\partial \bar{J}}{\partial u_i} \right)^- = \frac{\partial L_i}{\partial u_i} + \frac{ds_i}{du_i} \sum_{j=i}^{m(i)} \frac{\partial L_j}{\partial x_j} \,. \tag{6.16}$$

Similarly, we obtain

$$\xi_i^+ \equiv \left(\frac{\partial \bar{J}}{\partial u_i} \right)^+ = \frac{\partial L_i}{\partial u_i} + \frac{ds_i}{du_i} \sum_{j=i}^{n(i)} \frac{\partial L_j}{\partial x_j} \tag{6.17}$$

for the one-sided right derivative regardless of whether one or more critical jobs are present between i and $m(i)$. It easy to verify that

Lemma 6.1.5 *Under Assumptions A1 and A2 we have*

$$\xi_i^+ = \xi_i^- + \frac{ds_i}{du_i} \sum_{j=m(i)+1}^{n(i)} \frac{\partial L_j}{\partial x_j}$$

for every $i = 1, \ldots, N$.

Recalling the definition of $\partial \bar{J}$, it is easy to see that when $m(i) < n(i)$, we have

$$\partial \bar{J}_i = [\min(\xi_i^-, \xi_i^+), \max(\xi_i^-, \xi_i^+)] \subset \mathbb{R} \,.$$

Note that when $m(i) = n(i)$, we get $\xi_i^- = \xi_i^+$ in which case the set $\partial \bar{J}_i$ defined by the closed interval above is a singleton equal to the gradient $\partial \bar{J} / \partial u_i$ as required. As the necessary condition of non-smooth optimization requires that $0 \in \partial \bar{J}_i$, we obtain the following result:

Theorem 6.1.6 *Under Assumptions A1 and A2, an optimal control* u_i, $i = 1, \ldots, N$, *satisfies the following conditions:*

1. $0 \in \partial \bar{J}_i = [\min(\xi_i^-, \xi_i^+), \max(\xi_i^-, \xi_i^+)]$
2. $x_i = \max(a_i, x_{i-1}) + s_i(u_i)$ *with* $x_0 = -\infty$

Remark 6.1.7 Recalling Lemma 6.1.5, we see that when $m(i) = n(i)$, i.e., when job i is not critical and there are no critical jobs between job i and the end of its busy period, then the first condition of the theorem simply requires that $\xi_i^- = \xi_i^+ = 0$.

6.1.3 Properties of optimal solutions

Based on the necessary conditions for optimality in Theorem 6.1.6, in this section we present a fundamental property of optimal sample paths, i.e., the decoupling properties. The presence of the max function appearing in the state and co-state equations leads to a decoupling property that decomposes sample paths into independent segments. This property is a consequence of the "regenerative" nature of the state trajectory. Because of the max function in the state equation, information is not propagated in the forward direction across idle periods. In addition, because of the max function in the co-state equation, information does not propagate in the backward direction across idle periods. As a result, we obtain what we call idle period decoupling.

Lemma 6.1.8 *Consider a busy period defined by* $\{k, \ldots, n(k)\}$ *and let* $i \in \{k, \ldots, n(k)\}$. *The optimal control* u_i^* *depends only on* $a_k, \ldots, a_{n(k)}$ *(i.e., it does not depend on the arrival times of jobs in any other busy period).*

Proof. In view of Theorem 6.1.6, observe that the state equation does not propagate information in the forward direction across the idle period that precedes the busy period containing job i, i.e.,

$$x_{k-1} < a_k \Rightarrow \max(x_{k-1}, a_k) = a_k \Rightarrow x_i = a_k + s_k(u_k) + \cdots + s_i(u_i) \ .$$

Hence, the control for job i does not depend on the arrival times of jobs in earlier busy periods. Moreover, the co-state equation does not propagate information in the backward direction across the idle period that follows the busy period containing job i, i.e.,

$$x_{n(k)} < a_{n(k)+1} \Rightarrow \frac{d \max(x_{n(k)}, a_{n(k)+1})}{dx_{n(k)}} = 0 \Rightarrow \xi_i^+ = \frac{\partial L_i}{\partial u_i} + \frac{ds_i}{du_i} \sum_{j=i}^{n(k)} \frac{\partial L_j}{\partial x_j}$$

and the same is true for ξ_i^- since $m(i) \leqslant n(k)$. Since, by Theorem 6.1.6, the optimal control u_i^* is determined by ξ_i^- and ξ_i^+, it follows that it does not depend on the arrival times of jobs in subsequent busy periods. □

Because of idle period decoupling, the controls for individual busy periods can be determined independently of each other. Therefore, idle period decoupling decomposes a large TPBVP consisting of N jobs into several smaller subproblems, one for each busy period. Of course, since the identification of busy periods themselves is not a simple matter, this only partially simplifies the solution approach. Nonetheless, this decomposition can be used to develop efficient numerical algorithms (see [68, 178, 180, 215]). Moreover, it is also useful in the theoretical analysis of the optimal sample path, since it allows us to study its properties by analyzing a single isolated busy period.

For more information and further properties we refer the interested reader to [61].

Other methods for optimal control of hybrid systems are presented by Branicky in [39, 41–43], and Hedlund and Rantzer in [107–109, 151, 189] (using convex dynamic programming).

6.2 Model predictive control for MLD systems

6.2.1 Model predictive control

Model predictive control (MPC) was pioneered simultaneously by Richalet *et al.* [191], and Cutler and Ramaker [74]. In the last decades MPC has shown to respond effectively to control demands imposed by tighter product quality specifications, increasing productivity demands, new environmental regulations, and fast changes in the market. As a result, MPC is now widely accepted in the process industry. There are several other reasons why MPC is probably the most applied advanced control technique in this industry:

- MPC is a model-based controller design procedure that can easily handle multi-input multi-output processes, processes with large time-delays, non-minimum phase processes, and unstable processes.
- It is an easy-to-tune method: in principle only three parameters have to be tuned.
- MPC can handle constraints on the inputs and the outputs of the process (due to, e.g., limited capacity of buffers, actuator saturation, output quality specifications, etc.) in a systematic way during the design and the implementation of the controller.
- MPC can handle structural changes, such as sensor or actuator failures, and changes in system parameters or system structure, by adapting the model and by using a receding horizon approach, in which the model and the control strategy are regularly updated.

Conventional MPC uses discrete-time models (i.e., models consisting of a system of difference equations). In this section and in the next one we discuss some extensions and adaptations of the MPC framework to classes of hybrid systems that ultimately result in "tractable" control approaches. For each of these cases the proposed MPC approach has the following ingredients (which are also present in conventional MPC): a prediction horizon, a receding horizon procedure, and a regular update of the model and re-computation of the optimal control input.

Readers not familiar with the basic concepts of MPC are referred to Appendix A in which a short and simplified introduction to conventional MPC for nonlinear discrete-time systems is presented.

6.2.2 The MLD-MPC problem

This subsection and the next one are based on the seminal paper [27].

Consider an MLD system:

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \tag{6.18}$$

$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \tag{6.19}$$

$$E_1x(k) + E_2u(k) + E_3\delta(k) + E_4z(k) \leqslant g_5, \tag{6.20}$$

where $x(k) = [\, x_{\mathrm{r}}^{\mathrm{T}}(k) \;\; x_{\mathrm{b}}^{\mathrm{T}}(k) \,]^{\mathrm{T}}$ with $x_{\mathrm{r}}(k) \in \mathbb{R}^{n_{\mathrm{r}}}$ and $x_{\mathrm{b}}(k) \in \{0,1\}^{n_{\mathrm{b}}}$ ($y(k)$ and $u(k)$ have a similar structure), and where $z(k) \in \mathbb{R}^{r_{\mathrm{r}}}$ and $\delta(k) \in \{0,1\}^{r_{\mathrm{b}}}$ are auxiliary variables.

An important control problem for MLD systems is to stabilize the system to an equilibrium state or to track a desired reference trajectory. In general, finding a control law that attains these objectives for

an MLD system is not an easy task, as in general MLD systems are neither linear[2] nor even smooth. MPC provides a successful tool to perform this task, as will be shown next. For the sake of brevity we will concentrate on the stabilization to an equilibrium state.

Consider the MLD system (6.18)–(6.20) and an equilibrium state/input/output triple $(x_{\rm eq}, u_{\rm eq}, y_{\rm eq})$, and let $(\delta_{\rm eq}, z_{\rm eq})$ be the corresponding pair of auxiliary variables. Let $\hat{x}(k+j|k)$ denote the estimate of the state x at sample step $k+j$ based on the information available at sample step k. In a similar way we also define $\hat{y}(k+j|k)$, $\hat{\delta}(k+j|k)$, and $\hat{z}(k+j|k)$. Now we consider the MLD-MPC problem at sample step k with objective function

$$\begin{aligned} J(k) = \sum_{j=1}^{N_{\rm p}} & \|\hat{x}(k+j|k) - x_{\rm eq}\|_{Q_x}^2 + \|u(k+j-1) - u_{\rm eq}\|_{Q_u}^2 + \\ & \|\hat{y}(k+j|k) - y_{\rm eq}\|_{Q_y}^2 + \|\hat{\delta}(k+j-1|k) - \delta_{\rm eq}\|_{Q_\delta}^2 + \\ & \|\hat{z}(k+j-1|k) - z_{\rm eq}\|_{Q_z}^2 \end{aligned}$$

where Q_u, Q_x are positive definite matrices, and Q_y, Q_δ, Q_z are nonnegative definite matrices. Furthermore, in addition to the MLD system equations we have the end-point condition

$$\hat{x}(k+N_{\rm p}|k) = x_{\rm eq} \ , \tag{6.21}$$

and possibly also a control horizon constraint[3] of the form

$$u(k+j) = u(k+N_{\rm c}-1) \ \text{ for } j = N_{\rm c}, \ldots, N_{\rm p}-1.$$

A discussion on the stability of this scheme will be postponed to Section 6.4. We will first focus on how to solve such optimization problems.

6.2.3 Algorithms for the MLD-MPC optimization problem

Let us now show that the MLD-MPC problem can be recast as a mixed integer quadratic programming (MIQP) problem. For the MLD case using successive substitution for (6.18) results in the following prediction equation for the state:

$$\begin{aligned} \hat{x}(k+j|k) = {} & A^j x(k) + \\ & \sum_{i=0}^{j-1} A^{j-i-1}\big(B_1 u(k+i) + B_2\hat{\delta}(k+i) + B_3\hat{z}(k+i)\big) \ . \end{aligned}$$

For $\hat{y}(k+j|k)$ we have a similar expression. Now we define

$$\tilde{u}(k) = \big[\, u^{\rm T}(k) \ \ \ldots \ \ u^{\rm T}(k+N_{\rm p}-1) \,\big]^{\rm T} \ ,$$

and in similar way also $\tilde{\delta}(k)$ and $\tilde{z}(k)$. If we define

$$\tilde{V}(k) = \big[\tilde{u}^{\rm T}(k) \ \ \tilde{\delta}^{\rm T}(k) \ \ \tilde{z}^{\rm T}(k)\big]^{\rm T} \ ,$$

[2]Due to the integer constraints $\delta_i \in \{0,1\}$, the linear inequality (6.20) results in a nonlinear relation between δ and x, u, and between z and x, u.

[3]While in other contexts introducing a control horizon constraint amounts to hugely down-sizing the optimization problem at the price of a reduced performance, for MLD systems the computational gain is only partial, since all the (auxiliary) variables $\hat{\delta}(k+\ell|k)$ and $\hat{z}(k+\ell|k)$ for $\ell = N_{\rm c}, \ldots, N_{\rm p}-1$ remain in the optimization.

we obtain the following equivalent formulation for the MLD-MPC problem:

$$\min_{\tilde{V}(k)} \ \tilde{V}^{\mathrm{T}}(k)S_1\tilde{V}(k) + 2(S_2 + x^{\mathrm{T}}(k)S_3)\tilde{V}(k) \tag{6.22}$$

$$\text{subject to } \ F_1\tilde{V}(k) \leqslant F_2 + F_3x(k) \ , \tag{6.23}$$

for appropriately defined matrices S_1, S_2, S_3, F_1, F_2, F_3. Note that $\tilde{V}(k)$ contains both real-valued and integer-valued components. As the objective function is quadratic, the problem (6.22)–(6.23) is an MIQP problem.

MIQP problems are classified as NP-hard [98, 194], which — loosely speaking — means that, in the worst case, the solution time grows exponentially with the problem size. Several algorithmic approaches have been applied successfully to medium and large-size application problems [96], the four major ones being cutting plane methods, decomposition methods, logic-based methods, and branch-and-bound methods. In [27] the authors use a branch-and-bound method as several authors seem to agree on the fact that branch-and-bound methods are the most successful for mixed integer programming problems [95].

As described by [27, 95], the branch-and-bound algorithm for MIQP consists of solving and generating new quadratic programming (QP) problems in accordance with a tree search, where the nodes of the tree correspond to QP subproblems. The QP subproblems involve real-valued variables only, and are thus efficiently solvable using a modified simplex method or an interior point method [172, 175, 222].

For a worked example of the MLD-MPC approach we refer the interested reader to [27].

If the objective function $J(k)$ contains 1-norms or ∞-norms, then the problem (6.22)–(6.23) becomes a mixed integer *linear* programming (MILP) problem. In that case there exists an approach based on multi-parametric linear programming in which the MPC input for a given current state $x(k)$ is computed explicitly. It can be shown that the optimal solution of the multi-parametric MILP problem is a PWA function of $x(k)$. This function can then be computed explicitly off-line and stored in a look-up table. For the on-line control, one then only has to retrieve the appropriate control input value in the look-up table, which is much faster than solving the MILP on-line at each MPC step. On the other hand, the number of regions of the PWA function also increases rapidly as the size of the MPC problem increases. For more information we refer to [23, 24, 28, 29, 36].

6.3 MPC for continuous PWA systems

In Section 6.2 we already discussed how the MPC problem for PWA (and MLD) systems can be recast as a mixed integer quadratic (or linear) programming problem. In this section we will briefly show how MPC for MMPS systems (cf. Section 2.5 for a definition) — which are equivalent to *continuous* PWA systems (cf. Section 2.1) — can be transformed into a set of real (so not integer!) linear programming problems.

First, we will present a direct connection between *continuous* PWA systems and MMPS systems (without the need to introduce additional auxiliary variables or extra constraints as was done in Section 2.6). Next, we use the link between PWA systems and MMPS systems to present a new approach to MPC for continuous PWA systems. In order to compute an MPC controller for a PWA system or for an MMPS system we have to solve a nonlinear non-convex optimization problem at each sample step. We propose an optimization algorithm that is based on canonical forms for MMPS functions and that is similar to the cutting-plane algorithm for convex optimization problems. The proposed algorithm consists in solving several linear programming problems and is more efficient than the algorithms used in [83], which are

based on multi-start nonlinear local optimization (sequential quadratic programming) or on the extended linear complementarity problem.

6.3.1 Continuous PWA systems and MMPS systems

Recall that an MMPS function f of the variables $x_1, \ldots, x_n$ is defined by the recursive grammar

$$f := x_i | \alpha | \max(f_k, f_l) | \min(f_k, f_l) | f_k + f_l | \beta f_k \ ,$$

with $i \in \{1, \ldots, n\}$, $\alpha, \beta \in \mathbb{R}$, and where f_k and f_l are again MMPS functions.

An MMPS system can be described by state space equations of the following form:

$$x(k) = \mathcal{M}_x(x(k-1), u(k)) \tag{6.24}$$
$$y(k) = \mathcal{M}_y(x(k), u(k)) \ , \tag{6.25}$$

where $\mathcal{M}_x$ and $\mathcal{M}_y$ are vector-valued MMPS functions.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be a *continuous* PWA function if and only if the following conditions hold [70]:

1. The domain space $\mathbb{R}^n$ is divided into a finite number of polyhedral regions $R_{(1)}, \ldots, R_{(N)}$.
2. For each $i \in \{1, \ldots, N\}$, f can be expressed as
$$f(x) = \alpha_{(i)}^{\mathrm{T}} x + \beta_{(i)} \tag{6.26}$$
for any $x \in R_{(i)}$ with $\alpha_{(i)} \in \mathbb{R}^n$ and $\beta_{(i)} \in \mathbb{R}$.
3. f is continuous on any boundary between two regions.

For more information on PWA functions we refer to [70, 145] and the references therein. Note that the main difference with the PWA functions and systems introduced in Section 2.1 is that now we require the functions to be continuous on the boundary between the regions that make up the partition of the domain.

A continuous PWA system is a system of the form

$$x(k) = \mathcal{P}_x(x(k-1), u(k)) \tag{6.27}$$
$$y(k) = \mathcal{P}_y(x(k), u(k)) \ , \tag{6.28}$$

where $\mathcal{P}_x$ and $\mathcal{P}_y$ are vector-valued continuous PWA functions.

6.3.2 Equivalence of continuous PWA and MMPS systems

Theorem 6.3.1 [103, 174] *If f is a continuous PWA function of the form (6.26), then there exist index sets $I_1, \ldots, I_\ell \subseteq \{1, \ldots, N\}$ such that*

$$f = \max_{j=1,\ldots,\ell} \min_{i \in I_j} \left(\alpha_{(i)}^{\mathrm{T}} x + \beta_{(i)}\right) \ .$$

From the definition of MMPS functions it follows that (see also [103, 174]):

Lemma 6.3.2 *Any MMPS function is also a continuous PWA function.*

From Theorem 6.3.1 and Lemma 6.3.2 it follows that continuous PWA systems and MMPS systems are equivalent, i.e., for a given continuous PWA model there exists an MMPS model (and vice versa) such that the input-output behavior of both models coincides.

Corollary 6.3.3 *Continuous PWA models and MMPS models are equivalent.*

Note that this is an extension of the results of Section 2.6 and of [113, 114], which already prove an equivalence between (not necessarily continuous) PWA models and MMPS models, but there some extra auxiliary variables and some additional algebraic MMPS constraints between the states, the inputs and the auxiliary variables were required to transform the PWA model into an MMPS model.

6.3.3 Canonical forms of MMPS functions

Let $\alpha, \beta, \gamma, \delta \in \mathbb{R}$. Now we consider some easily verifiable properties of the max and min operators that will be used in the proof of the main theorem of this section.

- Minimization is distributive[4] w.r.t. maximization, i.e., $\min\big(\alpha, \max(\beta, \gamma)\big) = \max\big(\min(\alpha, \beta), \min(\alpha, \gamma)\big)$, which results in:

$$\min\big(\max(\alpha, \beta), \max(\gamma, \delta)\big) = \max\big(\min(\alpha, \gamma), \min(\alpha, \delta), \min(\beta, \gamma), \min(\beta, \delta)\big) \ . \tag{6.29}$$

- The $\max$ operation is distributive w.r.t. $\min$. Hence,

$$\max\big(\min(\alpha, \beta), \min(\gamma, \delta)\big) = \min\big(\max(\alpha, \gamma), \max(\alpha, \delta), \max(\beta, \gamma), \max(\beta, \delta)\big) \ . \tag{6.30}$$

- We have

$$\min(\alpha, \beta) + \min(\gamma, \delta) = \min(\alpha + \gamma, \alpha + \delta, \beta + \gamma, \beta + \delta) \tag{6.31}$$
$$\max(\alpha, \beta) + \max(\gamma, \delta) = \max(\alpha + \gamma, \alpha + \delta, \beta + \gamma, \beta + \delta) \ . \tag{6.32}$$

- The $\min$ and $\max$ operators are related as follows:

$$\max(\alpha, \beta) = -\min(-\alpha, -\beta) \ . \tag{6.33}$$

- If $\rho \in \mathbb{R}$ is positive, then

$$\rho \max(\alpha, \beta) = \max(\rho\alpha, \rho\beta), \qquad \rho \min(\alpha, \beta) = \min(\rho\alpha, \rho\beta) \ . \tag{6.34}$$

[4] If we use the operator symbols $\vee$ and $\wedge$ to denote $\max$ and $\min$ respectively, this distributivity property can be written as $\alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$.

Theorem 6.3.4 *Any MMPS function $f : \mathbb{R}^n \to \mathbb{R}$ can be rewritten in the min-max canonical form*

$$f = \min_{i=1,\dots,K} \max_{j=1,\dots,n_i} (\alpha_{(i,j)}^{\mathrm{T}} x + \beta_{(i,j)}) \tag{6.35}$$

or in the max-min canonical form

$$f = \max_{i=1,\dots,L} \min_{j=1,\dots,m_i} (\gamma_{(i,j)}^{\mathrm{T}} x + \delta_{(i,j)}) \tag{6.36}$$

for some integers K, L, $n_1, \dots, n_K$, $m_1, \dots, m_L$, vectors $\alpha_{(i,j)}, \gamma_{(i,j)}$, and real numbers $\beta_{(i,j)}, \delta_{(i,j)}$.

Proof. We will only prove the theorem for the min-max canonical form since the proof for the max-min canonical form is similar.
It is easy to verify that if f_k and f_l are affine functions, then the functions that result from applying the basic constructors of an MMPS function ($\max$, $\min$, $+$, and scaling — cf. (2.40)) are in min-max canonical form[5].
Now we use a recursive argument that consists in showing that if we apply the basic constructors of an MMPS function to two (or more) MMPS functions in min-max canonical form, then the result can again be transformed into min-max canonical form. Consider two MMPS functions f and g in min-max canonical form[6]: $f = \min(\max(f_1, f_2), \max(f_3, f_4))$ and $g = \min(\max(g_1, g_2), \max(g_3, g_4))$. The following equivalences show that $\max(f, g)$, $\min(f, g)$, $f + g$ and βf can again be written in min-max canonical form:

- $$\begin{aligned}\max(f,g) &= \max\big[\ \min\big(\max(f_1,f_2),\max(f_3,f_4)\big),\ \min\big(\max(g_1,g_2),\max(g_3,g_4)\big)\ \big]\\ &= \max\big[\ \max\big(\min(f_1,f_3),\min(f_1,f_4),\min(f_2,f_3),\min(f_2,f_4)\big),\\ &\qquad \max\big(\min(g_1,g_3),\min(g_1,g_4),\min(g_2,g_3),\min(g_2,g_4)\big)\ \big] \qquad \text{(by (6.29))}\\ &= \max\big(\min(f_1,f_3),\min(f_1,f_4),\min(f_2,f_3),\min(f_2,f_4),\\ &\qquad \min(g_1,g_3),\min(g_1,g_4),\min(g_2,g_3),\min(g_2,g_4)\big)\\ &= \min\big(\max(f_1,f_1,f_2,f_2,g_1,g_1,g_2,g_2),\max(f_1,f_1,f_2,f_2,g_1,g_1,g_2,g_4),\dots\\ &\qquad \max(f_3,f_4,f_3,f_4,g_3,g_4,g_3,g_4)\big) \qquad \text{(since max is distributive w.r.t. min)}\end{aligned}$$

- $$\begin{aligned}\min(f,g) &= \min\big[\ \min\big(\max(f_1,f_2),\max(f_3,f_4)\big),\ \min\big(\max(g_1,g_2),\max(g_3,g_4)\big)\ \big]\\ &= \min\big(\max(f_1,f_2),\max(f_3,f_4),\max(g_1,g_2),\max(g_3,g_4)\big)\end{aligned}$$

- $$\begin{aligned}f+g &= \min\big(\max(f_1,f_2),\max(f_3,f_4)\big) + \min\big(\max(g_1,g_2),\max(g_3,g_4)\big)\\ &= \min\big(\max(f_1,f_2)+\max(g_1,g_2),\max(f_1,f_2)+\max(g_3,g_4),\\ &\qquad \max(f_3,f_4)+\max(g_1,g_2),\max(f_3,f_4)+\max(g_3,g_4)\big) \qquad \text{(by (6.31))}\\ &= \min\big(\max(f_1+g_1,f_1+g_2,f_2+g_1,f_2+g_2),\\ &\qquad \max(f_1+g_3,f_1+g_4,f_2+g_3,f_2+g_4),\\ &\qquad \max(f_3+g_1,f_3+g_2,f_4+g_1,f_4+g_2),\\ &\qquad \max(f_3+g_3,f_3+g_4,f_4+g_3,f_4+g_4)\big) \qquad \text{(by (6.32))}\end{aligned}$$

[5]We allow "void" min or max statements of the form $\min(s)$ or $\max(s)$, which by definition are equal to s for any expression s. Alternatively, we can write $\min(s, s)$ or $\max(s, s)$.

[6]For the sake of simplicity we only consider two min-terms in f and g, each of which consists of the maximum of two affine functions. However, the proof also holds if more terms are considered.

- $\beta f = \beta \min\big(\max(f_1, f_2), \max(f_3, f_4)\big)$

$$= \begin{cases} \min\big(\max(\beta f_1, \beta f_2), \max(\beta f_3, \beta f_4)\big) & \text{(by (6.34))} \quad \text{if } \beta \geqslant 0 \\ -|\beta| \min\big(\max(f_1, f_2), \max(f_3, f_4)\big) & \text{if } \beta < 0 \\ \quad = -\min\big(\max(|\beta| f_1, |\beta| f_2), \max(|\beta| f_3, |\beta| f_4)\big) & \text{(by (6.34)} \\ \quad = \max\big(-\max(|\beta| f_1, |\beta| f_2), -\max(|\beta| f_3, |\beta| f_4)\big) & \text{(by (6.33))} \\ \quad = \max\big(\min(-|\beta| f_1, -|\beta| f_2), \min(-|\beta| f_3, -|\beta| f_4)\big) & \text{(by (6.33))} \\ \quad = \max\big(\min(\beta f_1, \beta f_2), \min(\beta f_3, \beta f_4)\big) & \\ \quad = \min\big(\max(\beta f_1, \beta f_3), \max(\beta f_1, \beta f_4), \max(\beta f_2, \beta f_3), \max(\beta f_2, \beta f_4)\big) & \text{(by (6.30))} \ \square \end{cases}$$

6.3.4 MPC for MMPS systems

In this section we give a short overview of the main results of [83] in which we have extended the MPC framework to MMPS systems. Related results can be found in [27]. More extensive information on conventional MPC for (linear and nonlinear) discrete-time systems can be found in [51, 97, 158] and the references therein. We can use the deterministic model (6.24)–(6.25) either as a model of an MMPS system, as the equivalent model of a continuous PWA system, or as an approximation of a general smooth nonlinear system. Note that we do not include modeling errors or uncertainty in the model. However, since MPC uses a receding finite horizon approach, we can regularly update the model and the state estimate as new information and measurements become available.

In MPC we compute at each sample step k an optimal control input that minimizes a cost criterion over the period $[k, k + N_\mathrm{p} - 1]$ where N_p is the prediction horizon. Assume that at sample step k the current state can be measured, estimated or predicted using previous measurements. Then we can make an estimate $\hat{y}(k + j|k)$ of the output of the system (6.24)–(6.25) at sample step $k + j$ based on the state $x(k - 1)$ and the future inputs $u(k + i)$, $i = 0, \ldots, j$. Using successive substitution, we obtain an expression of the following form:

$$\hat{y}(k + j|k) = F_j(x(k - 1), u(k), \ldots, u(k + j))$$

for $j = 0, \ldots, N_\mathrm{p} - 1$. Clearly, $\hat{y}(k + j|k)$ is an MMPS function of $x(k - 1), u(k), \ldots, u(k + j)$.

The cost criterion J used in MPC reflects the reference tracking error (J_out) and the control effort (J_in):

$$J(k) = J_\mathrm{out}(k) + \lambda J_\mathrm{in}(k)$$

where λ is a nonnegative real number. Let r contain the reference signal and define the vectors

$$\begin{aligned} \tilde{u}(k) &= \big[\, u^\mathrm{T}(k) \ \ \ldots \ \ u^\mathrm{T}(k + N_\mathrm{p} - 1) \,\big]^\mathrm{T} \\ \tilde{y}(k) &= \big[\, \hat{y}^\mathrm{T}(k|k) \ \ \ldots \ \ \hat{y}^\mathrm{T}(k + N_\mathrm{p} - 1|k) \,\big]^\mathrm{T} \\ \tilde{r}(k) &= \big[\, r^\mathrm{T}(k) \ \ \ldots \ \ r^\mathrm{T}(k + N_\mathrm{p} - 1) \,\big]^\mathrm{T} \ . \end{aligned}$$

In this section we consider the following output and input cost functions[7]:

$$J_{\mathrm{out},1}(k) = \|\tilde{y}(k) - \tilde{r}(k)\|_1 \tag{6.37}$$

[7] In conventional MPC usually quadratic cost functions of the form $J_\mathrm{out}(k) = \|\tilde{y}(k) - \tilde{r}(k)\|_2^2$ and $J_\mathrm{in}(k) = \|\tilde{u}(k)\|_2^2$ are used. In a discrete-event context, however, other choices are more appropriate (see [82, 83]).

$$J_{\text{out},\infty}(k) = \|\tilde{y}(k) - \tilde{r}(k)\|_\infty \tag{6.38}$$
$$J_{\text{in},1}(k) = \|\tilde{u}(k)\|_1 \tag{6.39}$$
$$J_{\text{in},\infty}(k) = \|\tilde{u}(k)\|_\infty \ . \tag{6.40}$$

Since we have $|x| = \max(x, -x)$ for all $x \in \mathbb{R}$, it is easy to verify that these cost functions are also MMPS functions.

In practical situations, there will be constraints on the input and output signals (caused by limited capacity of buffers, limited transportation rates, saturation, etc.) In general, this is reflected in a nonlinear constraint of the form

$$C_{\text{c}}(k, x(k-1), \tilde{u}(k), \tilde{y}(k)) \geqslant 0 \ . \tag{6.41}$$

The MPC problem at sample step k consists in minimizing $J(k)$ over all possible future input sequences subject to the constraints. To reduce the complexity of the optimization problem a control horizon N_{c} is introduced in MPC, which means that the input is taken to be constant beyond sample step $k + N_{\text{c}}$:

$$u(k+j) = u(k+N_{\text{c}}-1) \ \text{ for } j = N_{\text{c}}, \ldots, N_{\text{p}} - 1. \tag{6.42}$$

Alternatively, we can set the input rate constant as was done in [83]:

$$\Delta u(k+j) = \Delta u(k+N_{\text{c}}-1) \ \text{ for } j = N_{\text{c}}, \ldots, N_{\text{p}} - 1\,, \tag{6.43}$$

where $\Delta u(k) = u(k) - u(k-1)$. In addition to a decrease in the number of optimization parameters and thus also the computational burden, a smaller control horizon N_{c} also gives a smoother control signal, which is often desired in practical situations.

MPC uses a receding horizon principle. This means that after computation of the optimal control sequence $u(k), u(k+1), \ldots, u(k+N_{\text{c}}-1)$, only the first control sample $u(k)$ will be implemented, subsequently the horizon is shifted one sample, next the model and the state are updated using new information from the measurements, and a new MPC optimization is performed for sample step $k+1$.

6.3.5 Algorithms for the MMPS-MPC optimization problem

Nonlinear optimization

In general, the MMPS-MPC optimization problem is a nonlinear, non-convex optimization problem. In [83] we have discussed some algorithms to solve the MMPS-MPC optimization problem: we can use multi-start nonlinear optimization based on sequential quadratic programming (SQP), or we can use a method based on the extended linear complementarity problem (ELCP). However, both methods have their disadvantages. If we use the SQP approach, then we usually have to consider a large number of initial starting points and perform several optimization runs to obtain (a good approximation of) the global minimum. In addition, the objective functions that appear in the MMPS-MPC optimization problem are non-differentiable and PWA (if we use the cost criteria given in (6.37)–(6.40) or in [82]), which makes the SQP algorithm less suitable for them. The main disadvantage of the ELCP approach is that the execution time of this algorithm increases exponentially as the size of the problem increases. This implies that this approach is not feasible if N_{c} or the number of inputs and outputs of the system are large.

An alternative option consists in transforming the MMPS system into an MLD system since MMPS systems are equivalent to MLD systems (cf. Section 2.6). The main difference between MLD-MPC and MMPS-MPC is that MLD-MPC requires the solution of *mixed integer-real* optimization problems. In general, these are also computationally hard optimization problems.

In the next section we will present another method to solve the MMPS-MPC optimization problem that is similar to the cutting-plane method used in convex optimization.

An algorithm based on linear programming

We assume that the cost criteria given in (6.37)–(6.40) are used[8]. Recall that these objective functions (and any linear combination of them) are MMPS functions. The same holds for the estimate of future output $\tilde{y}(k)$. So if we substitute $\tilde{y}(k)$ in the expression for $J(k)$, we finally obtain an MMPS function of $\tilde{u}(k)$ as objective function. From Theorem 6.3.4 it follows that this objective function can be written in min-max canonical form as follows (where — for the sake of simplicity of notation — we drop the index k):

$$J = \min_{i=1,\ldots,\ell} \max_{j=1,\ldots,n_i} (\alpha_{(i,j)}^{\mathrm{T}} \tilde{u} + \beta_{(i,j)})$$

for appropriately defined integers $\ell, n_1, \ldots, n_\ell$, vectors $\alpha_{(i,j)}$ and integers $\beta_{(i,j)}$. Note that in general the expression obtained by straightforwardly applying the manipulations of the proof of Theorem 6.3.4 will contain a large number of affine arguments $\alpha_{(i,j)}^{\mathrm{T}} \tilde{u} + \beta_{(i,j)}$. However, many of these terms are redundant[9] and can thus be removed. This reduces the number of affine arguments. Also note that the transformation into canonical form only has to be performed once — provided that we explicitly consider all arguments that depend on k as additional variables when performing the transformation, — and that it can be done off-line.

The derivation below is similar to the cutting-plane algorithm for convex optimization (see, e.g., [38]). Hence, it requires constraints that are linear (or convex) in $\tilde{u}$. Note that the control horizon constraints (6.42) and (6.43) satisfy this condition. However, even if the original MPC constraint (6.41) is linear in $\tilde{u}(k)$ and $\tilde{y}(k)$, then in general after substitution of $\tilde{y}(k)$ this constraint is not linear anymore. Therefore, from now on we assume that (after substitution of $\tilde{y}(k)$) there are only linear[10] constraints on the input $\tilde{u}(k)$:

$$P\tilde{u} + q \geqslant 0 \ . \tag{6.44}$$

Note that in general P and q may depend on $x(k-1)$ and k, but for the sake of simplicity of notation we do not explicitly indicate this dependence. In practice constraints of the form (6.44) occur if we have to guarantee that the control signal $\tilde{u}(k)$ or the control signal rate $\Delta\tilde{u}(k)$ stay within certain bounds.

To obtain the optimal MPC input signal at sample step k, we have to solve an optimization problem of the following form:

$$\begin{aligned} &\min_{\tilde{u}} \min_{i=1,\ldots,\ell} \max_{j=1,\ldots,n_i} (\alpha_{(i,j)}^{\mathrm{T}} \tilde{u} + \beta_{(i,j)}) \\ &\text{subject to } P\tilde{u} + q \geqslant 0 \, . \end{aligned}$$

[8] The result below also holds for any other cost criterion that is an MMPS function of $\tilde{y}(k)$ and $\tilde{u}(k)$. So it follows from Theorem 6.3.1 that any continuous PWA norm function can also be used.

[9] E.g., since they appear twice, or since there are other arguments in the max (min) expression that are always larger (smaller) than the given argument.

[10] The optimization algorithm used below, which is based on the cutting plane algorithm for convex optimization, can also deal with convex constraints. So we can also allow convex constraints instead of (6.44).

or equivalently

$$\begin{aligned} &\min_{i=1,\ldots,\ell} \min_{\tilde{u}} \max_{j=1,\ldots,n_i} (\alpha_{(i,j)}^{\mathrm{T}} \tilde{u} + \beta_{(i,j)}) \\ &\text{subject to } P\tilde{u} + q \geqslant 0 \ . \end{aligned} \tag{6.45}$$

Now let $i \in \{1, \ldots, \ell\}$ and consider

$$\begin{aligned} &\min_{\tilde{u}} \max_{j=1,\ldots,n_i} (\alpha_{(i,j)}^{\mathrm{T}} \tilde{u} + \beta_{(i,j)}) \\ &\text{subject to } P\tilde{u} + q \geqslant 0 \ . \end{aligned}$$

It is easy to verify that this problem is equivalent to the following linear programming (LP) problem:

$$\begin{aligned} &\min_{\tilde{u},t} \ t \\ &\text{subject to} \qquad t \geqslant \alpha_{(i,j)}^{\mathrm{T}} \tilde{u} + \beta_{(i,j)} \quad \text{for } j = 1, \ldots, n_i \\ &\qquad\qquad\ \ P\tilde{u} + q \geqslant 0 \ . \end{aligned} \tag{6.46}$$

This LP problem can be solved efficiently using (variants of) the simplex method or an interior-point algorithm (see, e.g., [172, 222]).
To obtain the solution of (6.45), we solve (6.46) for $i = 1, \ldots, \ell$ and afterward we select the solution $\tilde{u}_{(i)}^{\mathrm{opt}}$ for which $\max_{j=1,\ldots,n_i} (\alpha_{(i,j)}^{\mathrm{T}} \tilde{u}_{(i)}^{\mathrm{opt}} + \beta_{(i,j)})$ is the smallest[11]. This results in an algorithm to solve the MMPS-MPC problem that is more efficient than the SQP or the ELCP approach.

6.4 Stability and robustness of MPC laws

Next to the computational issues as discussed above, there are also various system theoretic questions that are important to be answered. Questions related to Lyapunov stability and robustness with respect to disturbances on the system, do not only have theoretical importance, but are also important for any implementation of the controller in practice as any plant is subject to disturbances and the model used in the MPC set-up is never exactly the same as the physical plant that is controlled. It is not always guaranteed that an MPC controller, although optimizing a performance criterion involving the state variable, indeed stabilizes the plant. For linear systems and smooth nonlinear systems several approaches are known that assure that particular MPC set-ups are asymptotically stable, see e.g., [162] for a survey. Typically, these are based on putting an explicit condition and weight in the cost criterion on the terminal state at the end of the prediction horizon. One such example is the use of an end-point equality constraint as in (6.21). Using such a condition in the MPC setup, a result on attractivity (all trajectories converging to the equilibrium state) can be proven for the MLD-MPC setup as done in [27]:

Theorem 6.4.1 *Consider an MLD system (6.18)–(6.20) and an equilibrium state/input/output triple* $(x_{\mathrm{eq}}, u_{\mathrm{eq}}, y_{\mathrm{eq}})$*, and let* $(\delta_{\mathrm{eq}}, z_{\mathrm{eq}})$ *be the corresponding pair of auxiliary variables. Let the end-point condition* (6.21) *be included in the MPC setup (next to possible other time-independent constraints on inputs and states). Assume that the initial state* $x(0)$ *is such that a feasible solution of the MLD-MPC*

[11] If we use a primal-dual simplex method or an interior-point method to solve the LP problems, we can improve the efficiency of the approach even further by stopping the optimization if we obtain a lower bound for the objective function of the current LP problem that is larger than the smallest final objective function of the LP problems that have already been solved.

problem exists for sample step 0*. The input signal resulting from applying the optimal MLD-MPC input signal in a receding horizon approach stabilizes the MLD system in the sense that*

$$\begin{aligned} \lim_{k\to\infty} x(k) &= x_{\rm eq}\,, & \lim_{k\to\infty} \|y(k)-y_{\rm eq}\|_{Q_y} &= 0\,,\\ \lim_{k\to\infty} u(k) &= u_{\rm eq}\,, & \lim_{k\to\infty} \|\delta(k)-\delta_{\rm eq}\|_{Q_\delta} &= 0\,,\\ & & \lim_{k\to\infty} \|z(k)-z_{\rm eq}\|_{Q_z} &= 0\ . \end{aligned}$$

In the proof of the above result it is also shown that all the MLD-MPC problems are guaranteed to be feasible for all $k \in \mathbb{N}$. Although the above result is of interest and one of the first within the context of hybrid MPC, it is rather conservative in two ways. First of all, it relies on a terminal equality constraint (6.21), which is a severe condition as it requires controllability properties of the underlying plant, while stabilizability should be sufficient. Also for the numerical implementation the equality constraint at the end of the prediction horizon complicates matters considerably. Secondly, the theorem only provides attractivity and not Lyapunov stability. Hence, no matter how close the MLD system is to its equilibrium, it might first move arbitrarily far away from the origin before it starts converging to the equilibrium again, which is an undesirable property. As such, results that relax the terminal equality constraint to milder conditions such as, for instance, inclusions of the form

$$\hat{x}(k+N_{\rm p}|k) \in \mathcal{X}_{\rm T}, \tag{6.47}$$

where $\mathcal{X}_{\rm T}$ is a suitably constructed target set, are of interest. Also the guarantee of Lyapunov stability is of practical importance. For hybrid systems, these issues are very subtle and one has to be careful in drawing conclusions as both the plant and the controller might be discontinuous. In [143] conditions for asymptotically stability (including Lyapunov stability) for PWA systems are provided. The issue of robust stability (in terms of the so-called input-to-state stability property) is studied for hybrid systems in, e.g., [140–142] (see also the references therein.)

6.5 Discussion on MPC

In MPC for hybrid systems several issues play an important role, as we already indicated in the previous sections. One of the issues is complexity of the optimization problems to be solved. Especially, the general MLD set-up might become easily computationally prohibitive when the number of decision variables (especially the Boolean ones) and the number of constraints become relatively large. A remedy is the computation of the MPC off-line as a function of the state $x(k)$ of the systems (see end of Subsection 6.2.3) and using this as a kind of "look-up table" to find the control value that has to be implemented. This is in contrast with solving the optimization problems on-line. The storage of the explicit MPC approach in a computer or micro controller might require quite some memory. In the end, it has to be evaluated if the search algorithm with which one can find the correct value in stored memory will determine the feasibility of the explicit approach.

Another approach to solve the high computational burden that hybrid MPC might bring across, is to look at smaller subclasses of hybrid systems and exploit the special structure of these classes. As an example, above we saw several algorithms that might be used to improve the numerical feasibility of the optimization problems for continuous PWA systems.

Besides the computational issues, another topic of interest is the stability and robustness of the resulting MPC closed-loop systems. Not all choices of constraints and cost criteria lead to a stable

closed-loop. We presented a few results on (robust) stability of MPC for hybrid systems, but it is clear that these results can and should still be further improved.

6.6 Game-theoretic approaches

If we are considering safety-critical applications such as collision avoidance in (free-flight[12]) air space, or merging in automatic platooning in intelligent highway systems, then the primary goal of the controller is to guarantee safety. One of the approaches to design safe controllers is based on dynamic game theory.

Consider a continuous-time system

$$\dot{x} = f(x, u, d) \tag{6.48}$$

with $u \in U$ the control inputs (corresponding to the first player), and $d \in D$ the disturbance inputs (corresponding to the second player, who is known as the adversary). Note that we allow U and D to be arbitrary, possibly even (partially) discrete, sets.

Assume that the safety constraints can be represented by a set F of safe states given by inequality constraints:

$$F = \{x \in X \mid S(x) \geqslant 0\}$$

where $S : X \to \mathbb{R}$ is a differentiable function with $\frac{\partial S}{\partial x}(x) \neq 0$ on the boundary $\{x \mid S(x) = 0\}$. Let $t \in [t_0, t_{\text{end}}]$ and consider a cost function

$$J : X \times \mathcal{U} \times \mathcal{D} \times [t_0, t_{\text{end}}] \to \mathbb{R}$$

where $\mathcal{U}$ and $\mathcal{D}$ denote the admissible control and disturbance functions respectively. Assume that we have the end condition

$$J(x, u(\cdot), d(\cdot), t_{\text{end}}) = S(x(t_{\text{end}})) \ .$$

The function J may be interpreted as the cost associated with a trajectory of the system (6.48) starting at x at time $t = t_0$ according to the input functions $u(\cdot)$ and $d(\cdot)$, and ending at time $t = t_{\text{end}}$ at the final state $x(t_{\text{end}})$. Furthermore, for $t \in [t_0, t_{\text{end}}]$ define the value function

$$J^\star(x, t) = \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} J(x, u, d, t) \ .$$

Then the set

$$\{x \in X \mid \min_{\tau \in [t, t_{\text{end}}]} J^\star(x, \tau) \geqslant 0\}$$

contains all the states for which the system can be forced by the control u to remain in the safe set F for at least $|t_{\text{end}} - t|$ time units, irrespective of the disturbance function d.

In principle, the value function $J^\star$ can be computed by standard techniques from game theory using Hamilton-Jacobi equations [21]. Although in many cases the (numerical) solution of the Hamilton-Jacobi equations is a formidable task, this approach provides a systematic way to check safety properties

[12] Nowadays commercial aircraft usually have to follow certain predefined corridors in the airspace. As a consequence, a large part of the airspace is not utilized very well (especially when the planes are far away from an airport). Furthermore, the workload of the air traffic controllers increases very rapidly due to the growing number of planes. Therefore, the aviation community is advocating the concept of *free flight*. In the free-flight approach pilots can determine their own routes, altitudes and speeds. However, this can only be done in non-congested areas far away from any airport. If the pilot is in a congested area, say close to the airport, the pilot's preferences will be restricted.

for continuous-time systems, and is a starting point for checking safety properties for certain classes of hybrid systems (see [156]).

Note that in the free-flight framework we could either revert to non-cooperative[13] or cooperative[14] game theory.

6.7 Summary

In this chapter we have presented a selection of optimization-based control methods for hybrid systems, with an emphasis on model predictive control for MLD systems and for continuous PWA systems.

[13] E.g., if we assume that no (in)direct communication is possible with the neighboring airplane then — in order to be completely safe — the course of the plane has to be determined under the assumption that the other plane is actively working towards a collision

[14] If there is direct communication possible among all planes in a given region, then they can all cooperate and coordinate their actions in order to avoid any collisions.

Chapter 7

Model checking and timed automata

This chapter has been written by John Lygeros as part of his lecture notes used in Cambridge. We acknowledge his contribution and thank him for making it available to us.

Finite state systems are relatively easy to work with because one can investigate their properties by systematically exploring their states. For example, one can decide if a finite state system will eventually visit a particular set of states by following the system trajectories one by one. This is tedious to do by hand, but is easy to implement on a computer. Moreover, the process is guaranteed to terminate: since the number of states is finite, sooner or later we will find ourselves visiting the same states over and over again. At this point either the desired set has already been reached, or, if not, it will never be reached.

With hybrid systems it is in general impossible to do this. Because the number of states is infinite, it is impossible to enumerate them and try to explore them one by one. However, there are hybrid systems for which one can find a finite state system which is, in some sense, equivalent to the hybrid system. This is usually done by partitioning the state space into a finite number of sets with the property that any two states in a given set exhibit similar behavior. Then, to decide whether the hybrid system has certain properties, one has to work with the finite sets of the partition, as opposed to the infinite states of the original hybrid system. Moreover, the generation and analysis of the finite partition can be carried out automatically by a computer.

The process of automatically analyzing the properties of systems by exploring their state space is known as *model checking*. In this handout we discuss some fundamental ideas behind model checking, and introduce a class of hybrid systems, known as *timed automata*, that are amenable to this approach. As with deductive methods the discussion will be motivated by safety (reachability) analysis. Because of the complexity of the material we will not develop the results in their full beauty and generality. A good starting point for a deeper study is [7].

7.1 Transition systems

We first introduce a very general class of dynamical systems, known as transition systems, on which the above questions can be formulated.

Definition 7.1.1 (Transition System) A transition system, $T = (S, \delta, S_0, S_{\mathrm{f}})$ consists of

- A set of states S (finite or infinite);

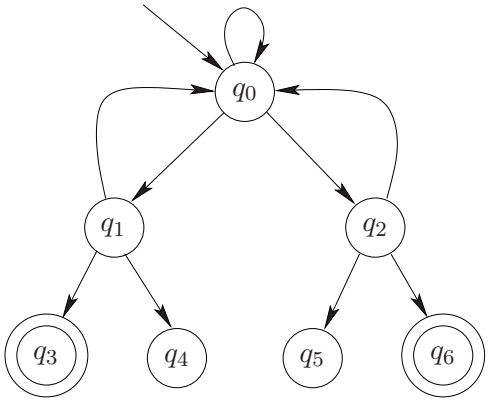

Figure 7.1: Finite state transition system.

- A transition relation $\delta : S \to P(S)$;
- A set of initial states $S_0 \subseteq S$;
- A set of final states $S_{\mathrm{f}} \subseteq S$.

The set of final states is included because we are interested in reachability type specifications. We will use it to encode desired final states, sets of states in which we want to stay, etc.

A *trajectory* of a transition system is a finite or infinite sequence of states $\{s_i\}_{i=0}^{N}$ such that

1. $s_0 \in S_0$; and,
2. $s_{i+1} \in \delta(s_i)$ for all $i = 0, 1, \ldots, N-1$.

Example 7.1.2 (Finite State Transition System) A transition system with finite states is shown in Figure 7.1. The formal definition of the system is

1. $S = \{q_0, \ldots, q_6\}$;
2. $\delta(q_0) = \{q_0, q_1, q_2\}$, $\delta(q_1) = \{q_0, q_3, q_4\}$, $\delta(q_2) = \{q_0, q_5, q_6\}$, $\delta(q_3) = \delta(q_4) = \delta(q_5) = \delta(q_6) = \varnothing$;
3. $S_0 = \{q_0\}$ (denoted by an arrow pointing to q_0);
4. $S_{\mathrm{f}} = \{q_3, q_6\}$ (denoted by a double circle).

Example 7.1.3 (Transition System of a Hybrid Automaton) Hybrid automata can be transformed into transition systems by abstracting away time. Consider a hybrid automaton $H = (Q, X, \mathrm{Init}, f, \mathrm{Inv}, E, G, R)$ together with a distinguished "final" set of states $F \subseteq Q \times X$. We will define a transition system for the hybrid automaton. Start with

- $S = Q \times X$, i.e., $s = (q, x)$
- $S_0 = \mathrm{Init}$

- $S_{\mathrm{f}} = F$.

The transition relation δ can be defined in many parts: a discrete transition relation $\delta_e : S \to P(S)$ for each edge $e \in E$ of the graph and a continuous transition relation $\delta_C : S \to P(S)$ to model the passage of time. For each $e = (q, q') \in E$ define

$$\delta_e(\hat{q}, \hat{x}) = \begin{cases} \{q'\} \times R(e, \hat{x}) & \text{if } \hat{q} = q \text{ and } \hat{x} \in G(e), \\ \varnothing & \text{if } \hat{q} \neq q \text{ or } \hat{x} \notin G(e). \end{cases}$$

For the continuous transition relation let

$$\delta_C(\hat{q}, \hat{x}) = \{(\hat{q}', \hat{x}') \in Q \times X \mid \left[\hat{q}' = \hat{q}\right] \wedge \left[\exists t_{\mathrm{f}} \geqslant 0,\ (x(t_{\mathrm{f}}) = \hat{x}') \wedge (\forall t \in [0, t_{\mathrm{f}}], x(t) \in \mathrm{Inv}(\hat{q}))\right]\}$$

where $x(\cdot)$ denotes the solution of the differential equation

$$\dot{x} = f(\hat{q}, x) \text{ starting at } x(0) = \hat{x} \ .$$

The overall transition relation is then

$$\delta(s) = \delta_C(s) \cup \bigcup_{e \in E} \delta_e(s) \ .$$

In words, a transition from $s \in S$ to $s' \in S$ is possible in the transition system if either a discrete transition $e \in E$ of the hybrid system will bring s to s', or s can flow continuously to s' after some time. Note that in the last statement time has been abstracted away. We do not care how long it takes to get from s to s', we only care whether it is possible to get there eventually. The transition system captures the sequence of "events" that the hybrid system may experience, but not the timing of these events.

Transition systems are designed to allow one to answer reachability (and other) questions algorithmically. For example, say we would like to answer the question

"Does there exist a trajectory of T such that $s_i \in S_{\mathrm{f}}$ for some i?"

If this is the case we say that S_{f} *is reachable*. We can approach this question using the predecessor operator

$$\mathrm{Pre} : P(S) \to P(S)$$

which for each set of states $\hat{S} \subseteq S$ is defined as

$$\mathrm{Pre}(\hat{S}) = \{s \in S \mid \exists \hat{s} \in \hat{S} \text{ with } \hat{s} \in \delta(s)\} \ .$$

The operator Pre takes a set of states, $\hat{S}$, and returns the states that can reach $\hat{S}$ in one transition. The following algorithm can then be used to determine if S_{f} is reachable.

Algorithm 1 (Backwards Reachability)

initialization: $W_0 = S_{\mathrm{f}}$, $i = 0$
repeat
  if $W_i \cap S_0 \neq \varnothing$
    return "S_{f} reachable"
  end if
  $W_{i+1} = \mathrm{Pre}(W_i) \cup W_i$
  $i = i + 1$
until $W_i = W_{i-1}$
return "S_{f} not reachable"

Using an induction argument it is easy to show that if the algorithm *terminates* (i.e., at some point it returns "S_{f} reachable" or "S_{f} not reachable") then it produces the right answer.

Exercise 7.1.4 *Show this.*

Exercise 7.1.5 *Define an appropriate operator* $\mathrm{Post} : P(S) \to P(S)$ *that for each set of states* $\hat{S} \subseteq S$ *returns the set of states that can be reached from* $\hat{S}$ *in one transition. Hence, develop a forward reachability algorithm.*

This algorithm is written in what is known as *pseudo-code*. It is conceptually useful, but is still a long way from being implementable on a computer. To effectively implement the algorithm one needs to figure out a way to explain to the computer how to

1. store sets of states,
2. compute the Pre of a set of states,
3. take the union and intersection of sets of states,
4. check whether a set of states is empty, and
5. check whether two sets of states are equal.

If the number of states S is finite all of these are relatively easy to do by enumerating the states. None of these steps are completely straightforward, however, if the state has real valued components (as is the case with hybrid systems).

Even if one was able to perform all of these operations using a computer program, it is still unclear whether the program would always produce an answer to the reachability question. The above algorithm may come up with the answer "S_{f} reachable", the answer "S_{f} not reachable", but it may also come up with no answer at all. This will be the case if new states get added to W_i each time we go around the repeat-until loop (hence, $W_{i+1} \neq W_i$) but none of these states belongs to S_0 (hence, $W_{i+1} \cap S_0 = \varnothing$).

Example 7.1.6 (Non-Terminating System) Consider the transition system $T = (S, \delta, S_0, S_{\mathrm{f}})$ with $S = \mathbb{R}$,

$$\delta(x) = 2x$$

$S_0 = \{-1\}$, $S_{\mathrm{f}} = \{1\}$. The Backwards Reachability algorithm produces the following sequence of sets:

$$W_0 = \{1\}, W_1 = \{1, \frac{1}{2}\}, \ldots, W_i = \{1, \frac{1}{2}, \ldots, \left(\frac{1}{2}\right)^i\}, \ldots$$

Note that W_{i+1} contains one more element than W_i, therefore we will never have $W_i = W_{i+1}$. Moreover, -1 will never be in W_i, therefore $W_i \cap S_0 = \varnothing$. Hence, the algorithm will not terminate.

With finite state systems termination is not a problem: the set W_i will sooner or later stop growing.

Example 7.1.7 (Finite State Transition System (cont.)) When applied to the finite state system of Figure 7.1 the Backwards Reachability Algorithm produces the following sequence of sets:

$$W_0 = \{q_3, q_6\}, W_1 = \{q_1, q_2, q_3, q_6\}, W_2 = \{q_0, q_1, q_2, q_3, q_6\} \ .$$

Note that $W_2 \cap S_0 = \{q_0\} \neq \varnothing$. Therefore, after two steps the algorithm terminates with the answer "S_{f} reachable".

Exercise 7.1.8 *Assume the set S is finite and contains M states. Give an upper bound on the number of times the algorithm will have to perform the "repeat-until" loop.*

7.2 Bisimulation

Since finite state systems are so much easier to work with, we are going to try to turn our infinite state systems into finite state ones, by grouping together states that have "similar" behavior. Such a grouping of states is called a *partition* of the state space. A partition is a collection of non-empty sets of states, $\{S_i\}_{i\in I}$, with $S_i \subseteq S$ and $S_i \neq \varnothing$, such that

1. Any two sets, S_i and S_j, in the partition are disjoint, i.e., $S_i \cap S_j = \varnothing$ for all $i, j \in I$ with $i \neq j$. (A family of sets with this property is called *mutually disjoint*).

2. The union of all sets in the partition is the entire state space, i.e.,

$$\bigcup_{i\in I} S_i = S$$

   (A family of sets with this property is said to *cover* the state space).

The index set, I, of the partition may be either finite or infinite. If I is a finite set (e.g., $I = \{1, 2, \ldots, M\}$ for $M < \infty$) then we say that the partition $\{S_i\}_{i\in I}$ is a *finite partition*.

Example 7.2.1 (Finite State Transition System (cont.)) The collection of sets

$$\{q_0\}, \{q_1, q_2\}, \{q_3, q_6\}, \{q_4, q_5\}$$

is a partition of the state space S of the finite system of Figure 7.1. The collection

$$\{q_0\}, \{q_1, q_3, q_4\}, \{q_2, q_5, q_6\}$$

is also a partition. However, the collection

$$\{q_1, q_3, q_4\}, \{q_2, q_5, q_6\}$$

is not a partition, and neither is

$$\{q_0, q_1, q_3, q_4\}, \{q_0, q_2, q_5, q_6\}$$

Given a transition system, $T = (S, \delta, S_0, S_\mathrm{f})$ and a partition of the state space $\{S_i\}_{i\in I}$ we can define a transition system whose states are the elements of the partition $S_i \subseteq S$, rather than individual states $s \in S$. This transition system $\hat{T} = (\hat{S}, \hat{\delta}, \hat{S}_0, \hat{S}_F)$ is defined as

- $\hat{S} = \{S_i\}_{i\in I}$, i.e., the states are the sets of the partition;
- $\hat{\delta}$ allows a transition from one set in the partition (say S_i) to another (say S_j) if and only if δ allows a transition from some state in S_i (say $s \in S_i$) to some state in S_j (say $s' \in S_j$). In mathematical notation,

$$\hat{\delta}(S_i) = \{S_j \mid \exists s \in S_i, \exists s' \in S_j \text{ such that } s' \in \delta(s)\}$$

- A set in the partition (say S_i) is in the initial set of $\hat{T}$ (i.e., $S_i \in \hat{S}_0$) if and only if some element of S_i (say $s \in S_i$) is an initial state of the original transition system (i.e., $s \in S_0$).
- A set in the partition (say S_i) is a final set of $\hat{T}$ (i.e., $S_i \in \hat{S}_F$) if and only if some element of S_i (say $s \in S_i$) is a final state of the original transition system (i.e., $s \in S_\mathrm{f}$).

Exercise 7.2.2 *Show that the above definitions are equivalent to* $\hat{\delta}(S_i) = \{S_j \mid \delta(S_i) \cap S_j \neq \varnothing\}$, $S_i \in \hat{S}_0 \Leftrightarrow S_i \cap S_0 \neq \varnothing$, $S_i \in \hat{S}_F \Leftrightarrow S_i \cap S_\mathrm{f} \neq \varnothing$. *You need to define* $\delta(S_i)$ *appropriately, but otherwise this is a tautology.*

The transition system generated by the partition is known as the *quotient transition system*. Note that if the partition is finite, then the quotient transition system $\hat{T}$ is a finite state system and therefore can be easily analyzed.

Using this method we can in principle construct finite state systems out of any transition system. The problem is that for most partitions the properties of the quotient transition system do not allow us to draw any useful conclusions about the properties of the original system. However, there is a special type of partition for which the quotient system $\hat{T}$ is in a sense equivalent to the original transition system, T. This type of partition is known as a *bisimulation*.

Definition 7.2.3 (Bisimulation) A bisimulation of a transition system $T = (S, \delta, S_0, S_\mathrm{f})$ is a partition $\{S_i\}_{i \in I}$ of the state space S of T such that

1. S_0 is a union of elements of the partition,
2. S_f is a union of elements of the partition,
3. if one state (say s) in some set of the partition (say S_i) can transition to another set in the partition (say S_j), then all other states, $\hat{s}$ in S_i must be able to transition to some state in S_j. More formally, for all $i, j \in I$ and for all states $s, \hat{s} \in S_i$, if $\delta(s) \cap S_j \neq \varnothing$, then $\delta(\hat{s}) \cap S_j \neq \varnothing$.

Exercise 7.2.4 *Show that a partition* $\{S_i\}_{i \in I}$ *is a bisimulation if and only if conditions 1 and 2 above hold and 3 is replaced by "for all* i, $\mathrm{Pre}(S_i)$ *is either empty or a union of elements of* $\{S_i\}_{i \in I}$*".*

Example 7.2.5 (Finite Transition System (cont.)) The partition

$$\{q_0\}, \{q_1, q_2\}, \{q_3, q_6\}, \{q_4, q_5\}$$

is a bisimulation of the finite system of Figure 7.1. Let us test this:

1. $S_0 = \{q_0\}$ which is an element of the partition;
2. $S_\mathrm{f} = \{q_3, q_6\}$ which is also an element of the partition;
3. Let us study the third condition for the set $\{q_1, q_2\}$. From q_1 one can jump to the following sets in the partition
$$\{q_0\}, \{q_3, q_6\}, \{q_4, q_5\}$$
From q_2 one can jump to exactly the same sets in the partition. Therefore the third condition is satisfied for set $\{q_1, q_2\}$. It is also easy to check this condition for the remaining sets (for the set $\{q_0\}$ the third condition is trivially satisfied).

The partition

$$\{q_0\}, \{q_1, q_3, q_4\}, \{q_2, q_5, q_6\}$$

on the other hand, is not a bisimulation. For example, S_f is not a union of elements of the partition. Also, from q_1 one can transition to q_0, whereas from q_3 and q_4 (the other elements of the set $\{q_1, q_3, q_4\}$) one cannot transition to q_0.

Bisimulations are important because of the following property.

Theorem 7.2.6 *Let $\{S_i\}_{i\in I}$ be a bisimulation of a transition system, T, and let $\hat{T}$ be the quotient transition system. S_f is reachable by T if and only if $\hat{S}_F$ is reachable by $\hat{T}$.*

Remark 7.2.7 This is a simplified version of a much deeper theorem. In fact bisimulations preserve not only reachability properties, but any kind of property that can be written as a formula in a temporal logic known as the Computation Tree Logic (CTL).

This is a very important and useful result. For finite state systems its implications are mostly in terms of computational efficiency. If we can find a bisimulation of the finite state system (like we did in the finite state example discussed above) we can study reachability in the quotient system instead of the original system. The advantage is that the quotient system will in general be much smaller than the original system. In the above example, the quotient system had 4 states whereas the original system had 7.

The implications are much more profound for infinite state systems. Even when the original transition system has an infinite number of states, sometimes we may be able to find a bisimulation that consists of a finite number of sets. Then we will be able to answer reachability questions for the infinite state system by studying an equivalent finite state system. Since finite state systems are so much easier to work with this could be a very big advantage. A class of hybrid systems for which we can always find finite bisimulations will be introduced in the next section.

The following algorithm can be used to find a bisimulation of a transition system $T = (S, \delta, S_0, S_\mathrm{f})$.

Algorithm 2 (Bisimulation)

initialization: $\mathcal{P} = \{S_0, S_\mathrm{f}, S \setminus (S_0 \cup S_\mathrm{f})\}$
while there exists $S_i, S_j \in \mathcal{P}$ such that $S_i \cap \mathrm{Pre}(S_j) \neq \varnothing$ and $S_i \cap \mathrm{Pre}(S_j) \neq S_i$ **do**
    $S_i' = S_i \cap \mathrm{Pre}(S_j)$
    $S_i'' = S_i \setminus \mathrm{Pre}(S_j)$
    $\mathcal{P} = (\mathcal{P} \setminus S_i) \cup \{S_i', S_i''\}$
end while
return $\mathcal{P}$

The symbol $\setminus$ in the algorithm stands for set difference: $S_i \setminus \mathrm{Pre}(S_j)$ is the set that contains all elements of S_i that are not elements of $\mathrm{Pre}(S_j)$, in other words

$$S_i \setminus \mathrm{Pre}(S_j) = \{s \in S \mid (s \in S_i) \wedge (s \notin \mathrm{Pre}(S_j))\}$$

The algorithm maintains a partition of the state space, denoted by $\mathcal{P}$, which gets refined progressively so that it looks more and more like a bisimulation. The definition of the bisimulation suggests that if $\mathcal{P}$ is to be a bisimulation then it must at least allow us to "distinguish" the initial and final states. We therefore start with a partition that contains three sets: S_0, S_f and everything else $S \setminus (S_0 \cup S_\mathrm{f})$, i.e.,

$$\mathcal{P} = \{S_0, S_\mathrm{f}, S \setminus (S_0 \cup S_\mathrm{f})\}$$

At each step of the algorithm, we examine the sets of the candidate partition. Assume we can find two sets $S_i, S_j \in \mathcal{P}$ such that $\mathrm{Pre}(S_j)$ contains some elements of S_i (i.e., $S_i \cap \mathrm{Pre}(S_j) \neq \varnothing$) but not all of them (i.e., $S_i \not\subseteq \mathrm{Pre}(S_j)$, or, equivalently, $S_i \cap \mathrm{Pre}(S_j) \neq S_i$). Then some states $s \in S_i$ may find themselves in S_j after one transition (namely those with $s \in S_i \cap \mathrm{Pre}(S_j)$), while others cannot do the same (namely those with $s \in S_i \setminus \mathrm{Pre}(S_j)$). This is not allowed if $\mathcal{P}$ is to be a bisimulation. We therefore replace S_i by two sets: the states in S_i that can transition to S_j ($S_i' = S_i \cap \mathrm{Pre}(S_j)$) and the states in S_i that cannot transition to S_j ($S_i'' = S_i \setminus \mathrm{Pre}(S_j)$). Note that after the replacement $\mathcal{P}$ has one more set than it did before. The process is repeated until for all sets $S_i, S_j \in \mathcal{P}$ either $S_i \cap \mathrm{Pre}(S_j) \neq \varnothing$ or $S_i \cap \mathrm{Pre}(S_j) \neq S_i$. The resulting collection of sets, $\mathcal{P}$ is a bisimulation. In fact:

Theorem 7.2.8 *If the Bisimulation Algorithm terminates it will produce the coarsest bisimulation of the transition system (i.e., the bisimulation containing the smallest number of sets).*

For finite state systems this algorithm is easy to implement (by enumerating the states) and will always terminate.

Example 7.2.9 (Finite State Transition System (cont.)) Let us apply the bisimulation algorithm to the finite system of Figure 7.1. Initially

$$\mathcal{P} = \{S_0, S_{\mathrm{f}}, S \setminus (S_0 \cup S_{\mathrm{f}})\} = \{\{q_0\}, \{q_3, q_6\}, \{q_1, q_2, q_4, q_5\}\}$$

Note that $\mathrm{Pre}(\{q_3, q_6\}) = \{q_1, q_2\}$. This is not an element of the partition $\mathcal{P}$. It intersects $\{q_1, q_2, q_4, q_5\}$ but is not equal to it. We therefore split $\{q_1, q_2, q_4, q_5\}$ into two sets

$$\{q_1, q_2, q_4, q_5\} \cap \mathrm{Pre}(\{q_3, q_6\}) = \{q_1, q_2\}$$

and

$$\{q_1, q_2, q_4, q_5\} \setminus \mathrm{Pre}(\{q_3, q_6\}) = \{q_4, q_5\}$$

After one iteration of the algorithm the partition becomes

$$\mathcal{P} = \{\{q_0\}, \{q_3, q_6\}, \{q_1, q_2\}, \{q_4, q_5\}\}$$

It is easy to check that this is a bisimulation. Clearly S_0 and S_{f} are elements of the partition. Moreover,

$$\mathrm{Pre}(\{q_0\}) = \{q_0, q_1, q_2\}$$

which is a union of elements of the partition, and so on.

The problem with using the bisimulation algorithm on finite state systems is that it may be more work to find a bisimulation than to investigate the reachability of the original system. Sometimes bisimulations can be computed by "inspection", by taking advantage of symmetries of the transition structure. In the above example, we can immediately see that the left sub-tree is a mirror image of the right sub-tree. This should make us suspect that there is a bisimulation lurking somewhere in the system. There is an entire community in computer science that develops methods for detecting and exploiting such symmetries.

When we try to apply the bisimulation algorithm to infinite state systems we face the same problems we did with the Backward Reachability algorithm: how to store sets of states in the computer, how to compute Pre, etc. Moreover, even in cases where we can do all these, the algorithm may never terminate. The reason is that not all infinite state transition systems have finite bisimulations. In the next section we will introduce a class of (infinite state) hybrid systems for which we can not only implement the above algorithm in a computer, but also ensure that it will terminate in a finite number of steps.

7.3 Timed automata

Timed automata are a class of hybrid systems that involve particularly simple continuous dynamics: all differential equations are of the form $\dot{x} = 1$ and all the domains, guards, etc. involve comparison of the real valued states with constants ($x = 1$, $x < 2$, $x \geqslant 0$, etc.). Clearly timed automata are somewhat limited when it comes to modeling physical systems. They are very good however for encoding timing constraints ("event A must take place at least 2 seconds after event B and not more than 5 seconds before event C", etc.). For some applications, such as multimedia, Internet and audio protocol verification, etc. this type of description is sufficient for both the dynamics of the system and the properties that we want the system to satisfy. We conclude this chapter with a brief discussion of the properties of timed automata. Because complicated mathematical notation is necessary to formally introduce the topic we will give a rather informal exposition. Students interested in the details are referred to the (rather technical but classic) paper [4].

A subset of $\mathbb{R}^n$ is called *rectangular* if it can be written as a finite boolean combination of constraints of the form $x_i \leqslant a$, $x_i < b$, $x_i = c$, $x_i \geqslant d$, and $x_i > e$, for $x \in \mathbb{R}^n$ where a, b, c, d, e are rational numbers. Roughly speaking, rectangular sets are "rectangles" in $\mathbb{R}^n$ whose sides are aligned with the axes, or unions of such rectangles. For example, in $\mathbb{R}^2$ the set defined by

$$(x_1 \geqslant 0) \wedge (x_1 \leqslant 2) \wedge (x_2 \geqslant 1) \wedge (x_2 \leqslant 2)$$

is rectangular, and so is the set defined by

$$((x_1 \geqslant 0) \wedge (x_2 = 0)) \vee ((x_1 = 0) \wedge (x_2 \geqslant 0))$$

The empty set is also a rectangular set (e.g., $\varnothing = (x_1 \geqslant 1) \wedge (x_1 \leqslant 0)$). However the set

$$\{x \in \mathbb{R}^2 \mid x_1 = 2x_2\}$$

is not rectangular.

Exercise 7.3.1 *Draw these sets in $\mathbb{R}^2$. You should immediately be able to see why rectangular sets are called rectangular.*

Note that rectangular sets are easy to encode in a computer. Instead of storing the set itself (which is impossible since the set is infinite) we can store and manipulate the list of constraints used to generate the set (which is finite).

Roughly speaking, a timed automaton is a hybrid automaton for which

- all differential equations are of the form $\dot{x}_i = 1$; continuous variables governed by this differential equation are known as *clocks*,
- the sets involved in the definition of the initial states, guards and domain are rectangular sets, and
- the reset is either a rectangular set, or may leave certain states unchanged.

Example 7.3.2 (Timed Automaton) An example of a timed automaton is given in Figure 7.2. We have

- $Q = \{q_1, q_2\}$;

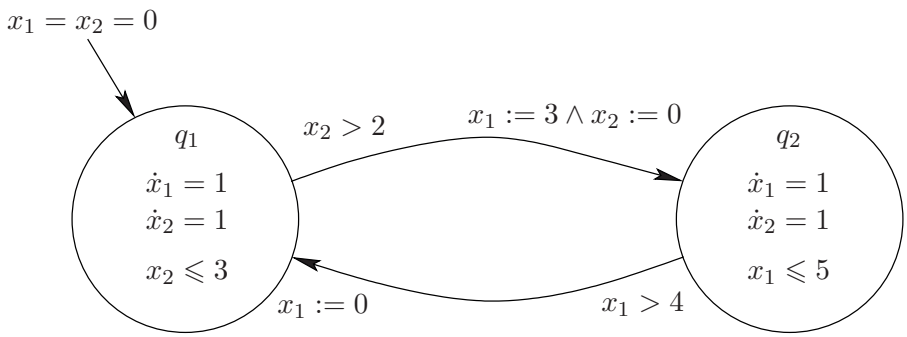

Figure 7.2: Example of a timed automaton.

- $X = \mathbb{R}^2$;
- $f(q_1, x) = f(q_2, x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$;
- $\text{Init} = \{(q_1, (0, 0))\}$;
- $\text{Inv}(q_1) = \{x \in \mathbb{R}^2 \mid x_2 \leqslant 3\}$, $\text{Inv}(q_2) = \{x \in \mathbb{R}^2 \mid x_1 \leqslant 5\}$;
- $E = \{(q_1, q_2), (q_2, q_1)\}$;
- $G(q_1, q_2) = \{x \in \mathbb{R}^2 \mid x_2 > 2\}$, $G(q_2, q_1) = \{x \in \mathbb{R}^2 \mid x_1 > 4\}$;
- $R(q_1, q_2) = \{((x_1, x_2),\ (3, 0)) \mid x_1, x_2 \in \mathbb{R}\}$, $R(q_2, q_1) = \{((x_1, x_2),\ (0, x_2)) \mid x_1, x_2 \in \mathbb{R}\}$.

Exercise 7.3.3 *Is this timed automaton non-blocking? Is it deterministic?*

Note that in the timed automaton of the example all the constants that appear in the definition are nonnegative integers. It turns out that we can in general assume that this is the case: given any timed automaton whose definition involves rational and/or negative constants we can define an equivalent timed automaton whose definition involves only nonnegative integers. This is done by "scaling" (multiplying by an appropriate integer) and "shifting" (adding an appropriate integer) some of the states.

We can turn timed automata into transition systems by "abstracting away" time, just like we did for general hybrid systems above. It turns out that the transition system corresponding to a timed automaton always has a finite bisimulation. One standard bisimulation that works for all timed automata is the *region graph*. The region graph for the timed automaton of Figure 7.2 is shown in Figure 7.3.

We will briefly describe the way the region graph is constructed. Assume that all the constants that appear in the definition of the timed automaton are nonnegative integers (this can be done without loss of generality as noted above). As usual, let us label the continuous variables (clocks) as $x_1, \ldots, x_n$. Let C_i be the largest constant with which x_i is compared in any of the sets used in the definition of the timed automaton (initial sets, guards, etc.). In the example $C_1 = 5$ and $C_2 = 3$. If all we know about the timed automaton is these bounds C_i, x_i could be compared with any integer M with $0 \leqslant M \leqslant C_i$ in some guard, reset or initial condition set. Therefore, the discrete transitions of the timed automaton may be able to "distinguish" states with $x_i < M$ from states with $x_i = M$ and from states with $x_i > M$, for all $0 \leqslant M \leqslant C_i$. Distinguish means, for example, that a discrete transition may be possible from a state with $x_i < M$ but not from a state with $x_i > M$ (because the guard contains the condition

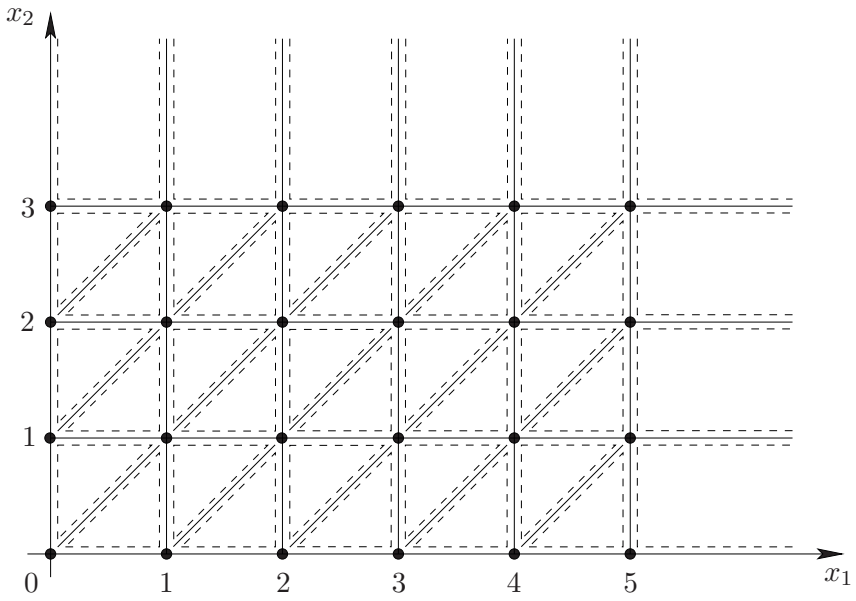

Figure 7.3: Region graph for the timed automaton of Figure 7.2.

$x_i < M$). Because these sets may be distinguished by the discrete transitions we add them to our candidate bisimulation. In the example this gives rise to the sets

$$\begin{aligned}
&\text{for } x_1 : x_1 \in (0,1), x_1 \in (1,2), x_1 \in (2,3), x_1 \in (3,4), x_1 \in (4,5), x_1 \in (5,\infty)\\
&\qquad x_1 = 0, x_1 = 1, x_1 = 2, x_1 = 3, x_1 = 4, x_1 = 5,\\
&\text{for } x_2 : x_2 \in (0,1), x_2 \in (1,2), x_2 \in (2,3), x_2 \in (3,\infty)\\
&\qquad x_2 = 0, x_2 = 1, x_2 = 2, x_2 = 3.
\end{aligned}$$

The product of all these sets, i.e., the sets

$$\begin{aligned}
&\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 \in (0,1)\}\\
&\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 = 1\}\\
&\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 \in (0,1)\}\\
&\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 = 1\}\\
&\{x \in \mathbb{R}^2 \mid x_1 \in (1,2) \wedge x_2 \in (3,\infty)\},\\
&\text{etc.}
\end{aligned}$$

define all the sets in $\mathbb{R}^2$ that the discrete dynamics (initial states, guards, domain and reset relations) can distinguish. Note that in the continuous state space $\mathbb{R}^2$ these product sets are open squares (squares without their boundary), open horizontal and vertical line segments (line segments without their end points), points on the integer grid and open, unbounded rectangles (when some $x_i > C_i$).

Since $\dot{x}_1 = \dot{x}_2 = 1$, time flow makes the continuous state move diagonally up along $45°$ lines. By allowing time to flow the timed automaton may therefore distinguish points below the diagonal of each square, points above the diagonal and points on the diagonal. For example, points above the diagonal of the square

$$\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 \in (0,1)\}$$

will leave the square through the line

$$\{x \in \mathbb{R}^2 \mid x_1 \in (0,1) \wedge x_2 = 1\}$$

points below the diagonal will leave the square through the line

$$\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 \in (0,1)\}$$

while points on the diagonal will leave the square through the point

$$\{x \in \mathbb{R}^2 \mid x_1 = 1 \wedge x_2 = 1\}$$

This leads us to split each open square in three: two open triangles and an open diagonal line segment.

It can in fact be shown that this is enough to generate a bisimulation.

Theorem 7.3.4 *The region graph is a finite bisimulation of the timed automaton.*

It should be stressed that the region graph is not necessarily the coarsest bisimulation. One may be able to find bisimulations with fewer sets; the elements of these bisimulation will be unions of sets of the region graph. It is generally considered bad form to try to construct the entire region graph to investigate properties of a timed automaton. Usually, one either uses the bisimulation algorithm to construct a coarse bisimulation, or uses the reachability algorithm directly to investigate reachability. The point of Theorem 7.3.4 is that it guarantees that there is at least one finite bisimulation for each timed automaton, which in turn guarantees that the Bisimulation and Reachability algorithms can be implemented and will terminate.

A counting exercise reveals that the total number of regions in the region graph is of the order of

$$n!\, 2^n \prod_{i=1}^{n} (2C_i + 2)$$

(! denotes factorial.) Even for relatively small n this is a very large number! (! denotes exclamation point) What is worse, the number grows very quickly (exponentially) as n increases. (In addition to $n!$ and 2^n, there is another hidden exponential in this formula. Because on a computer numbers are stored in binary, C_i is exponential in the number of bits used to store it).

This is bad news. It implies that a relatively simple timed automaton can give rise to a region graph with a very large number of sets, which will be a nightmare to analyze. It turns out that this is a general property of timed automata and has nothing to do with the way the region graph was constructed. Because timed automata have finite bisimulations, we know that they can be automatically analyzed by a computer. However, in the worst case, the number of operations that the computer will have to perform to analyze the system will be exponential in the size of the problem (e.g., the length of the input file we need to define our timed automaton for the computer). This is irrespective of how well we write the program. In computational complexity terminology, model checking for timed automata turns out to be *PSPACE Complete*.

7.4 Bibliography and further reading

Timed automata were the first class of hybrid systems that were shown to be amenable to model checking methods [4]. Since then a number of other classes of hybrid systems with this property have been

established: classes of multi-rate automata [3], classes of systems with continuous dynamics governed by constant differential inclusions [120] and classes of systems with continuous dynamics governed by linear differential equations [138]. It has also been shown that a very wide class of hybrid systems can be approximated arbitrarily closely by such "decidable" hybrid systems [188] (albeit at the cost of exponential computational complexity). For an excellent overview of the developments in this area see [7].

In the case of hybrid control systems, related methods have been developed for automatically synthesizing controllers to satisfy specifications (e.g., given in temporal logic) whenever possible [15,160,221].

Based on the theoretical results, computational tools have been developed to automatically perform verification or synthesis tasks [9,31,77,121].

Chapter 8

Suggestions for further reading

These lecture notes cover, of course, not all results that are available on hybrid systems. Basic system theoretic results on controllability and observability are, for instance, missing. Also topics related to observer design, identification and simulation of hybrid systems have not been touched upon. Readers might consider one of the books [127, 139] for general overviews on hybrid systems. Below we provide references for more dedicated topics within hybrid systems theory. The lists of references are not complete, but can be used as a first introduction to these challenging topics in the appealing field of hybrid systems.

8.1 Controllability, observability and related topics

Notions such as controllability, stabilizability, observability and detectability have played a central role throughout the history of modern control theory. Conceived by Kalman, the controllability concept has been studied extensively in the context of finite-dimensional linear systems, nonlinear systems, infinite-dimensional systems, hybrid systems, and behavioral systems. One may refer for instance to Sontag's book [200] for historical comments and references.

Outside the linear context, characterizations of global controllability have been hard to obtain. In the setting of smooth nonlinear systems, results have been obtained for local controllability but there is no hope to obtain general algebraic characterizations of controllability in the large. The complexity of characterizing controllability and stabilizability has been studied by [33] for some classes of hybrid systems, and these authors show that even within quite limited classes there is no algorithm to decide the controllability status of a given system. Hence, this indicates that there is no hope to find complete conditions for general hybrid systems. The best to obtain seems to be characterizations for some specific classes of hybrid systems some results are available.

Controllability problems for piecewise linear systems and various related model classes have been studied in [25, 33, 54, 105, 144]. A similar story holds for observability and detectability [17, 25, 56, 72].

8.2 Observer design

Observer design for hybrid systems is also a topic of interest. In many practical situations the full state is not available for feedback, while most control design methods as mentioned in these lecture notes are

based on state feedback. As such, it is of interest to use output feedback controllers using for instance a "certainty equivalence principle" in which the state feedback will be based on an estimated state coming from an observer or another estimation scheme.

Several interesting papers are available on observer design for hybrid systems, especially in the context of switched and piecewise linear systems. Still the problem is still of interest as it turns out to be a complicated problem, especially when the mode of the system is not known or cannot be directly reconstructed on the basis of the measurements. The situation becomes even more complicated when resets of the continuous state variable occur. Good starting points for investigating hybrid observer design can be found in [2, 19, 20, 90, 133, 177, 183, 208]. For application of hybrid observer structures on experimental setups we refer the reader to [86].

8.3 Identification

Models of hybrid systems typically contain parameters that can not always be determined using a first principles modeling approach. In such cases the parameters have to be determined based on input-output data. This process is called identification. Some results on identification of hybrid systems are described in [26, 91, 134, 192, 214]. A comparison between various identification procedures is given in [132]. This comparison paper also presents interesting problems that hybrid identification methods have to face.

8.4 Simulation

Finally, note that in practice the most widely used technique for hybrid systems is computer simulation, via a combination of discrete-event simulation and differential algebraic equation (DAE) solvers. Some computer simulation and verification tools that are (also) used for hybrid systems are BaSiP, Modelica, HyTech, KRONOS, Hybrid Chi, 20-sim, and UPPAAL. Simulation models can represent the plant with a high degree of detail, providing a close correspondence between simulated behavior and real plant behavior. This approach is, for any large system, computationally very demanding, and moreover it is difficult to understand from a simulation how the behavior depends on model parameters. This difficulty is even more pronounced in the case of large hybrid systems which consist of many interacting modules. Fast simulation techniques based on variance reduction, and perturbation analysis techniques [57] have been developed in order to partially overcome these limitations.

Appendix A

Model predictive control

In this appendix we give a short and simplified introduction to conventional model predictive control (MPC) for nonlinear discrete-time systems.

For the sake of simplicity we will only consider the deterministic, i.e., noiseless, case in this brief introduction. More extensive information on MPC can be found in [51, 158, 190] and the references therein.

A.1 Prediction model

Consider a plant with m inputs and l outputs that can be modeled by a nonlinear discrete-time state space description of the following form:

$$x(k+1) = f(x(k), u(k)) \tag{A.1}$$

$$y(k) = h(x(k), u(k)) \tag{A.2}$$

where f and h are smooth functions of x and u.

In MPC we consider the future evolution of the system over a given prediction period $[k+1, k+N_\mathrm{p}]$, which is characterized by the prediction horizon N_p, and where k is the current sample step (see Figure A.1). For the system (A.1)–(A.2) we can make an estimate $\hat{y}(k+j|k)$ of the output at sample step $k+j$ based on the state[1] $x(k)$ at step k and the future input sequence $u(k), u(k+1), \ldots, u(k+j-1)$. Using successive substitution, we obtain an expression of the form

$$\hat{y}(k+j|k) = F_j(x(k), u(k), u(k+1), \ldots, u(k+j-1)) \tag{A.3}$$

for $j = 1, \ldots, N_\mathrm{p}$. If we define the vectors

$$\tilde{u}(k) = \left[\, u^\mathrm{T}(k) \;\; \ldots \;\; u^\mathrm{T}(k+N_\mathrm{p}-1) \,\right]^\mathrm{T} \tag{A.4}$$

$$\tilde{y}(k) = \left[\, \hat{y}^\mathrm{T}(k+1|k) \;\; \ldots \;\; \hat{y}^\mathrm{T}(k+N_\mathrm{p}|k) \,\right]^\mathrm{T} \;, \tag{A.5}$$

we obtain the following prediction equation:

$$\tilde{y}(k) = F(x(k), \tilde{u}(k)) \;. \tag{A.6}$$

[1] For the sake of simplicity, we assume that all the components of the state can be measured, or that the system is observable such that the current state can be reconstructed from the past output sequence (using, e.g., an (extended) Kalman filter).

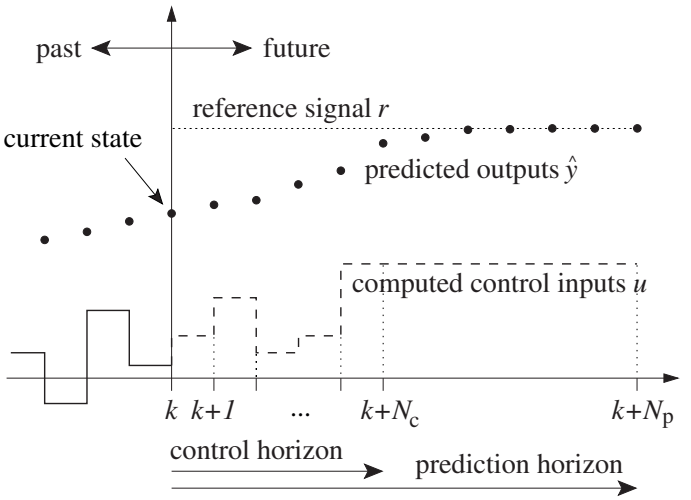

Figure A.1: Representation of the MPC control scheme.

A.2 Cost criterion and constraints

The cost criterion J used in conventional MPC reflects the reference tracking error (J_{out}) and the control effort (J_{in}):

$$\begin{aligned} J(k) &= J_{\mathrm{out}}(k) + \lambda J_{\mathrm{in}}(k) \\ &= \|\tilde{y}(k) - \tilde{r}(k)\|_Q^2 + \lambda \|\tilde{u}(k)\|_R^2 \\ &= \big(\tilde{y}(k) - \tilde{r}(k)\big)^{\mathrm{T}} Q \big(\tilde{y}(k) - \tilde{r}(k)\big) + \lambda \tilde{u}^{\mathrm{T}}(k) R \tilde{u}(k) \end{aligned}$$

where λ is a nonnegative integer, and $\tilde{r}(k)$ contains the reference signal (defined similarly to $\tilde{y}(k)$ (cf. (A.5))), and Q, R are positive definite matrices.

In practical situations, there will be constraints on the input and output signals of the plant (caused by limited capacity of buffers, limited transportation rates, saturation, etc.). This is reflected in the nonlinear constraint function

$$C_{\mathrm{c}}(k, \tilde{u}(k), \tilde{y}(k)) \leqslant 0 \ , \tag{A.7}$$

which is often taken to be linear:

$$A_{\mathrm{c}}(k)\tilde{u}(k) + B_{\mathrm{c}}(k)\tilde{y}(k) \leqslant c_{\mathrm{c}}(k) \ . \tag{A.8}$$

The MPC problem at sample step k consists in minimizing $J(k)$ over all possible future input sequences subject to the constraints. This is usually a nonconvex optimization problem. To reduce the complexity of the optimization problem a control horizon N_{c} is introduced in MPC, which means that the input is taken to be constant beyond sample step $k + N_{\mathrm{c}} - 1$:

$$u(k+j) = u(k+N_{\mathrm{c}}-1) \ \text{ for } j = N_{\mathrm{c}}, \ldots, N_{\mathrm{p}} - 1. \tag{A.9}$$

In addition to a decrease in the number of optimization parameters and thus also the computational burden, a smaller control horizon N_{c} also gives a smoother control signal, which is often desired in practical situations.

A.3 Receding horizon approach

MPC uses a receding horizon principle. At time step k the future control sequence $u(k), \ldots, u(k+N_\mathrm{p}-1)$ is determined such that the cost criterion is minimized subject to the constraints. At time step k the first element of the optimal sequence ($u(k)$) is applied to the process. At the next time instant the horizon is shifted, the model is updated with new information from the measurements, and a new optimization at time step $k+1$ is performed.

A.4 The standard MPC problem

The MPC problem at sample step k for the nonlinear discrete-time system described by (A.1)–(A.2) is defined as follows:

> Find the input sequence $\{u(k), \ldots, u(k+N_\mathrm{p}-1)\}$ that minimizes the cost criterion $J(k)$ subject to the evolution equations (A.1)–(A.2) of the system, the nonlinear constraint (A.7), and the control horizon constraint (A.9).

Recall that due to the receding horizon approach this problem has to be solved at each sample step k. Note however that the use of a control horizon leads to a reduction of the number of optimization variables. This results in a decrease of the computational burden, a smoother controller signal (because of the emphasis on the average behavior rather than on aggressive noise reduction), and a stabilizing effect.

For *linear* discrete-time systems and with linear constraints (A.8) only, the MPC problem boils down to a convex quadratic programming problem, which can be solved very efficiently. Furthermore, in this case the solution can be even computed off-line and reduces to the simple evaluation of an explicitly defined piecewise affine function [28, 29].

A.5 Tuning

The parameters N_p, N_c and λ are the three basic MPC tuning parameters: The prediction horizon N_p is related to the length of the step response of the process, and the time interval $(1, N_\mathrm{p})$ should contain the crucial dynamics of the process. The control horizon $N_\mathrm{c} \leqslant N_\mathrm{p}$ is usually taken equal to the system order. An important effect of a small control horizon is the smoothing of the control signal. The control signal is then rapidly forced towards its steady-state value, which is important for stability properties. Another important consequence of decreasing N_c is the reduction in computational effort, because the number of optimization parameters is reduced. The parameter $\lambda \geqslant 0$ makes a trade-off between the tracking error and the control effort, and is usually chosen as small as possible (while still getting a stabilizing controller).

Bibliography

[1] M. Aganagic. *Contributions to Complementarity Theory*. PhD thesis, Dept. of Operations Research, Stanford University, California, USA, 1978.

[2] A. Alessandri and P. Coletta. Design of Luenberger observers for a class of hybrid linear systems. In *Hybrid Systems: Computation and Control*, pages 7–18, 2001.

[3] R. Alur, C. Courcoubetis, N. Halbawachs, T.A. Henzinger, P.H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[4] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.

[6] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR 1997)*, volume 1243 of *Lecture Notes in Computer Science*, pages 74–88. Springer Verlag, 1997.

[7] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[8] R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III.* (Proc. of the Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995.), volume 1066 of *Lecture Notes in Computer Science*. Springer, 1996.

[9] R. Alur and R.P. Kurshan. Timing analysis in COSPAN. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 220–231. Springer-Verlag, 1996.

[10] P. Antsaklis, W. Kohn, M.D. Lemmon, A. Nerode, and S. Sastry, editors. *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1999. (Proceedings of the 5th International Hybrid Systems Workshop, Notre Dame, Indiana, September 1997).

[11] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems II.* (Proc. Workshop on Hybrid Systems, Cornell University, USA, October 1994.), volume 999 of *Lecture Notes in Computer Science*. Springer, 1995.

[12] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems IV.* (Proc. of the Fourth Intern. Workshop on Hybrid Systems, Ithaca, New York, October 1996.), volume 1273 of *Lecture Notes in Computer Science*. Springer, 1997.

[13] P.J. Antsaklis and A. Nerode, eds. Special issue on hybrid systems. *IEEE Transactions on Automatic Control*, 43(4), April 1998.

[14] P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon. Hybrid system modeling and autonomous control systems. In R.L. Grossmand, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 366–392, New York, 1993. Springer.

[15] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7):1011–1025, July 2000.

[16] J.P. Aubin and A. Cellina. *Differential Inclusions: Set-Valued Maps and Viability Theory*. Springer, Berlin, 1984.

[17] M. Babaali and M. Egerstedt. Observability for switched linear systems. In *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*. Springer, 2004.

[18] F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat. *Synchronization and Linearity*. John Wiley & Sons, New York, 1992.

[19] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, and A. L. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In C.J. Tomlin and J.R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 76–89. Springer-Verlag, Stanford, CA, 2002.

[20] G.I. Bara, J. Daafouz, F. Kratz, and J. Ragot. Parameter dependent state observer design for affine LPV systems. *International Journal of Control*, 74(16):1601–1611, 2001.

[21] T. Basar and G.J. Olsder. *Dynamic Non-Cooperative Game Theory*. Academic Press, Boston, 2nd edition, 1995.

[22] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Transactions on Automatic Control*, 49(5):832–838, May 2004.

[23] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proceedings of the 2000 American Control Conference*, pages 1190–1194, Chicago, Illinois, June 2000.

[24] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming — The explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, December 2002.

[25] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, 2000.

[26] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Transactions on Automatic Control*, 50(10):1567–1580, 2005.

[27] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[28] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.

[29] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. Corrigendum to: "the explicit linear quadratic regulator for constrained systems" [Automatica 38(1) (2002) 3–20]. *Automatica*, 39(10):1845–1846, October 2003.

[30] 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.

[31] J. Bengtsson, K.G. Larsen, F. Larsson, P. Petterson, and W. Yi. UPPAAL: A tool suite for automatic verification of real-time systems. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer Verlag, 1996.

[32] A. Bloch and S. Drakunov. Stabilization of a nonholonomic system via sliding modes. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 2961–2963, Lake Buena Vista, December 1994.

[33] V.D. Blondel and J.N. Tsitsiklis. Complexity of stability and controllability of elementary hybrid systems. *Automatica*, 35(3):479–490, 1999.

[34] R.K. Boel, L. Ben-Naoum, and V. Van Breusegem. On forbidden state problems for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, 40(10):1717–1731, October 1995.

[35] R.K. Boel, B. De Schutter, G. Nijsse, J.M. Schumacher, and J.H. van Schuppen. Approaches to modelling, analysis, and control of hybrid systems. *Journal A*, 40(4):16–27, December 1999.

[36] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, 2003.

[37] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, 1994.

[38] S.P. Boyd and C.H. Barratt. *Linear Controller Design: Limits of Performance*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[39] M.S. Branicky. Analyzing and synthesizing hybrid control systems. In G. Rozenberg and F. Vaandrager, editors, *Lectures on Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, pages 74–113. Springer, Berlin, 1998.

[40] M.S. Branicky. Multiple Lyapunov theory and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, 1998.

[41] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, January 1998.

[42] M.S. Branicky and S.K. Mitter. Algorithms for optimal hybrid control. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2661–2666, New Orleans, Louisiana, December 1995.

[43] M.S. Branicky and G. Zhang. Solving hybrid control problems: Level sets and behavioral programming. In *Proceedings of the 2000 American Control Conference*, pages 1175–1180, Chicago, IL, June 2000.

[44] K.E. Brenan, S.L. Campbell, and L.R. Petzhold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14 of *Classics in Applied Mathematics*. SIAM, 1995.

[45] R.W. Brockett. Hybrid models for motion control systems. In H.L. Trentelman and J.C. Willems, editors, *Essays on Control: Perspectives in the Theory and its Applications*, volume 14 of *Progress in Systems and Control Theory*, pages 29–53, Boston, 1993. Birkhäuser.

[46] B. Brogliato. *Nonsmooth Impact Mechanics. Models, Dynamics and Control*, volume 220 of *Lecture Notes in Control and Information Sciences*. Springer, London, 1996.

[47] B. Brogliato, S.-I. Niculescu, and P. Orthant. On the control of finite-dimensional mechanical systems with unilateral constraints. *IEEE Transactions on Automatic Control*, 42(2):200–215, 1997.

[48] B. Brogliato and A. Zavala-Rio. On the control of complementary-slackness mechanical juggling systems. *IEEE Transactions on Automatic Control*, 45(3), 2000.

[49] A.E. Bryson and Y.C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere, Bristol, PA, 1975.

[50] C. Cai, A. Teel, and R. Goebel. Smooth Lyapunov functions for hybrid systems. Part I: Existence is equivalent to robustness. *IEEE Transactions on Automatic Control*, 52(7):1264–1277, 2007.

[51] E.F. Camacho and C. Bordons. *Model Predictive Control in the Process Industry*. Springer-Verlag, Berlin, Germany, 1995.

[52] M.K. Camlibel, W.P.M.H. Heemels, and J.M. Schumacher. Consistency of a time-stepping method for a class of piecewise-linear networks. *IEEE Transactions on Circuits and Systems-I*, 49(3):349–357, 2002.

[53] M.K. Camlibel, W.P.M.H. Heemels, and J.M. Schumacher. On linear passive complementarity systems. *European Journal of Control*, 8(3):220–237, 2002.

[54] M.K. Camlibel, W.P.M.H. Heemels, and J.M. Schumacher. Algebraic necessary and sufficient conditions for the controllability of conewise linear systems. *IEEE Transactions on Automatic Control*, 53(3):762–774, 2008.

[55] M.K. Camlibel, W.P.M.H. Heemels, A.J. van der Schaft, and J.M. Schumacher. Switched networks and complementarity. *IEEE Transactions on Circuits and Systems-I*, 50:1036–1046, 2003.

[56] M.K. Camlibel, J.S. Pang, and J. Shen. Conewise linear systems: non-zenoness and observability. *SIAM Journal on Control and Optimizaton*, 45:1769–1800, 2006.

[57] C.G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis*. The Aksen Associates Series in Electrical and Computer Engineering. Richard D. Irwin, Inc., Burr Ridge, Illinois, 1993.

[58] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, Boston, 1999.

[59] C.G. Cassandras and D.L. Pepyne. Optimal control of a class of hybrid systems. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 133–138, San Diego, California, December 1997.

[60] C.G. Cassandras, D.L. Pepyne, and Y. Wardi. Optimal control of systems with time-driven and event-driven dynamics. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 7–12, Tampa, Florida, December 1998.

[61] C.G. Cassandras, D.L. Pepyne, and Y. Wardi. Optimal control of a class of hybrid systems. *IEEE Transactions on Automatic Control*, 46(3):398–415, March 2001.

[62] M. K. Çamlıbel, W. P. M. H. Heemels, and J.M. Schumacher. On linear passive complementarity systems. *European Journal of Control*, 8:220–237, 2002.

[63] M.K. Çamlıbel. *Complementarity Methods in the Analysis of Piecewise Linear Dynamical Systems*. PhD thesis, Tilburg University, 2001.

[64] M.K. Çamlıbel, W.P.M.H. Heemels, and J.M. Schumacher. Well-posedness of a class of linear network with ideal diodes. In *Proc. of the 14th International Symposium of Mathematical Theory of Networks and Systems*, Perpignan (France), 2000.

[65] Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

[66] H. Chen. Synthesis of feedback logic for controlled Petri nets with forward and backward conflict-free uncontrolled subnets. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 3098–3103, Lake Buena Vista, Florida, December 1994.

[67] Y. Cho and C.G. Cassandras. Optimal control for steel annealing processes as hybrid systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, 2000.

[68] Y. Cho, C.G. Cassandras, and D.L. Pepyne. Forward decomposition algorithms for optimal control of a class of hybrid systems. *Int. J. Robust Nonlin. Control*, 2:369–394, 2001.

[69] D. Christiansen. *Electronics Engineers' Handbook*. IEEE Press/McGraw Hill, New York, 4th edition, 1997.

[70] L.O. Chua and A.C. Deng. Canonical piecewise-linear representation. *IEEE Transactions on Circuits and Systems*, 35(1):101–111, January 1988.

[71] F.H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley-Interscience, New York, 1983.

[72] P. Collins and J.H. van Schuppen. Observability of piecewise–affine hybrid systems. In R. Alur and G.J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 265–279. Springer Verlag, Berlin, 2004.

[73] R.W. Cottle, J.-S. Pang, and R.E. Stone. *The Linear Complementarity Problem*. Academic Press, Boston, 1992.

[74] C.R. Cutler and B.L. Ramaker. Dynamic matrix control – a computer control algorithm. In *Proceedings of the 86th AIChE National Meeting*, Houston, Texas, April 1979.

[75] R. David and H. Alla. Petri nets for modelling of dynamic systems - a survey. *Automatica*, 30(2):175–202, 1994.

[76] J. Davies. *Specification and proof in real-time CSP*. Cambridge University Press, 1993.

[77] C. Daws, A. Olivero, S. Trypakis, and S. Yovine. The tool KRONOS. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer Verlag, 1996.

[78] B. De Schutter. Optimal control of a class of linear hybrid systems with saturation. *SIAM Journal on Control and Optimization*, 39(3):835–851, 2000.

[79] B. De Schutter and B. De Moor. The extended linear complementarity problem and the modeling and analysis of hybrid systems. 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 70–85. Springer, 1999.

[80] B. De Schutter, W.P.M.H. Heemels, J. Lunze, and C. Prieur. Survey of modeling, analysis, and control of hybrid systems. In J. Lunze and F. Lamnabhi-Lagarrigue, editors, *Handbook of Hybrid Systems Control – Theory, Tools, Applications*, chapter 2, pages 31–55. Cambridge University Press, Cambridge, UK, 2009.

[81] B. De Schutter and T. van den Boom. Model predictive control for max-min-plus systems. In R. Boel and G. Stremersch, editors, *Discrete Event Systems: Analysis and Control*, volume 569 of *The Kluwer International Series in Engineering and Computer Science*, pages 201–208. Kluwer Academic Publishers, Boston, 2000.

[82] B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, July 2001.

[83] B. De Schutter and T.J.J. van den Boom. Model predictive control for max-min-plus-scaling systems. In *Proceedings of the 2001 American Control Conference*, pages 319–324, Arlington, Virginia, June 2001.

[84] B. De Schutter and T.J.J. van den Boom. Model predictive control for max-min-plus-scaling systems — Efficient implementation. In M. Silva, A. Giua, and J.M. Colom, editors, *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02)*, pages 343–348, Zaragoza, Spain, October 2002.

[85] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1995.

[86] A. Doris, A.Lj. Juloski, N. Mihajlovic, W.P.M.H. Heemels, N. van de Wouw, and H. Nijmeijer. Observer designs for experimental non-smooth and discontinuous systems. *IEEE Transactions on Control Systems Technology*, 16(6):1323–1332, 2008.

[87] B.C. Eaves and C.E. Lemke. Equivalence of LCP and PLS. *Mathematics of Operations Research*, 6(4):475–484, November 1981.

[88] G. Escobar, A.J. van der Schaft, and R. Ortega. A Hamiltonian viewpoint in the modeling of switching power converters. *Automatica*, 35:445–452, 1999.

[89] E. Feron. Quadratic stabilizability of switched systems via state and output feedback. Technical Report CICS-P-468, Center for intelligent control systems, Massachusetts Institute of Technology, Cambridge, 1996.

[90] G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Transactions on Automatic Control*, 47:1663–1676, 2002.

[91] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[92] A. Feuer, G.C. Goodwin, and M. Salgado. Potential benefits of hybrid control for linear time-invariant systems. In *Proceedings of the 1997 American Control Conference*, pages 2790–2794, Albuquerque, New Mexico, 1997.

[93] A.F. Filippov. Differential equations with discontinuous right-hand side. *Matemat. Sbornik.*, 51:99–128, 1960. In Russian. English translation: *Am. Math. Soc. Transl.* 62 (1964).

[94] A.F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Mathematics and Its Applications. Kluwer, Dordrecht, The Netherlands, 1988.

[95] R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM Journal on Optimization*, 8(2):604–616, May 1998.

[96] C.A. Floudas. *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, Oxford, UK, 1995.

[97] C.E. García, D.M. Prett, and M. Morari. Model predictive control: Theory and practice — A survey. *Automatica*, 25(3):335–348, May 1989.

[98] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[99] M. Gazarik and Y. Wardi. Optimal release times in a single server: An optimal control perspective. *IEEE Transactions on Automatic Control*, 43(7):998–1002, July 1998.

[100] D.N. Godbole, J. Lygeros, and S. Sastry. Hierarchical hybrid control: a case study. In *[11]*, pages 166–190, 1995.

[101] R. Goebel, R. Sanfelice, and A.R. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, 2009.

[102] R. Goebel and A.R. Teel. Solution to hybrid inclusions via set and graphical convergence with stability theory applications. *Automatica*, 42:573–587, 2006.

[103] V.V. Gorokhovik and O.I. Zorko. Piecewise affine functions and polyhedral sets. *Optimization*, 31:209–221, 1994.

[104] F.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors. *Hybrid Systems.* (Proc. Workshop on the Theory of Hybrid Systems, Lyngby, Denmark, October 1992.), volume 736 of *Lecture Notes in Computer Science*. Springer, 1993.

[105] L.C.G.J.M. Habets and J.H. van Schuppen. A controllability result for piecewise-linear hybrid systems. In *Proceedings of the 2001 European Control Conference*, Porto, Portugal, 2001.

[106] M.L.J. Hautus and L.M. Silverman. System structure and singular control. *Linear Algebra and its Applications*, 50:369–402, 1983.

[107] S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *Proceedings 38th IEEE Conference on Decision and Control(CDC'99)*, Phoenix, Arizona, December 1999.

[108] S. Hedlund and A. Rantzer. Hybrid control laws from convex dynamic programming. In *Proceedings of IEEE Conference of Decision and Control*, December 2000.

[109] S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Transactions on Automatic Control*, 47(9):1536–1540, September 2002.

[110] W.P.M.H. Heemels. *Linear Complementarity Systems: A Study in Hybrid Dynamics*. PhD thesis, Dept. of Electrical Engineering, Eindhoven University of Technology, The Netherlands, 1999.

[111] W.P.M.H. Heemels, M.K. Camlibel, and J.M. Schumacher. On the dynamic analysis of piecewise linear networks. *IEEE Transactions on Circuits and Systems I*, 49:315–327, 2002.

[112] W.P.M.H. Heemels, M.K. Camlibel, A.J. van der Schaft, and J.M. Schumacher. On the existence and uniqueness of solution trajectories to hybrid dynamical systems. In *Chapter 18 in "Nonlinear and Hybrid Control in Automotive Applications," Springer London (Editor: R. Johannson and A. Rantzer)*, pages 391–421, 2002.

[113] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[114] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. On the equivalence of classes of hybrid dynamical models. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 364–369, Orlando, Florida, December 2001.

[115] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. The rational complementarity problem. *Linear Algebra and its Applications*, 294(1-3):93–135, 1999.

[116] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Well-posedness of linear complementarity systems. In *38-th IEEE Conference on Decision and Control,* Phoenix (USA), 1999.

[117] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000.

[118] W.P.M.H. Heemels, A.R. Teel, N. van de Wouw, and D. Nešić. Networked control systems with communication constraints: tradeoffs between sampling intervals and delays. In *Proc. European Control Conference in Budapest, Hungary*, 2009.

[119] M.F. Heertjes, M.J.G. van den Molengraft, J.J. Kok, and D.H. van Campen. Vibration reduction of a harmonically excited beam with one-sided springs using sliding computed torque control. *Dynamics and control*, 7:361–375, 1997.

[120] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata. In *Proceedings of the 27th Annual Symposium on the Theory of Computing, STOC'95*, pages 373–382. ACM Press, 1995.

[121] T.A. Henzinger, P.H. Ho, and H. Wong Toi. A user guide to HYTECH. In E. Brinksma, W. Cleaveland, K. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019, pages 41–71. Springer Verlag, 1995.

[122] T.A. Henzinger, P.W. Kopke, A. Puria, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, August 1998.

[123] T.A. Henzinger and S. Sastry, editors. *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1998. (Proceedings of the First International Workshop on Hybrid Systems: Computation and Control (HSCC'98), Berkeley, California, April 1998).

[124] J.P. Hespanha and A.S. Morse. Stability of switched systems with average dwell-time. In *Proc.38th IEEE Conference on Decision and Control*, pages 2655–2660, 1999.

[125] C.A.R. Hoare. *Communicating sequential processes*. Prentice-Hall, 1985.

[126] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 7(2):151–190, April 1997.

[127] D. Hristu-Varsakelis and W.S. Levine, editors. *Handbook of Networked and Embedded Control Systems*. Birkhäuser, 2005.

[128] N. Inaba and S. Mori. Chaos via torus breakdown in a piecewise-linear forced van der Pol oscillator with a diode. *IEEE Transactions on Circuits and Systems*, 38(4):398–409, 1991.

[129] K.J. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of zeno hybrid automata. *Systems & Control Letters*, 38:141–150, 1999.

[130] M. Johansson. *Piecewise Linear Control Systems*, volume 284 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, Germany, 2003.

[131] M. Johansson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, April 1998.

[132] A.Lj. Juloski, W.P.M.H. Heemels, G. Ferrari-Trecate, R. Vidal, S. Paoletti, and J.H.G. Niessen. Comparison of four procedures for the identification of hybrid systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 354–369. Springer Verlag, 2005.

[133] A.Lj. Juloski, W.P.M.H. Heemels, and S. Weiland. Observer design for a class of piecewise linear systems. *International Journal of Robust and Nonlinear Control*, 17(15):1387–1404, 2007.

[134] A.Lj. Juloski, S. Wieland, and W.P.M.H. Heemels. A Bayesian approach to identification of hybrid systems. *IEEE Transactions on Automatic Control*, 50(10):1520–1533, 2005.

[135] C.D. Kelly, D. Watanapongse, and K.M. Gaskey. Application of modern control to a continuous anneal line. *IEEE Control Systems Magazine*, 8:32–37, 1988.

[136] T.A.M. Kevenaar and D.M.W. Leenaerts. Comparison of piecewise-linear model descriptions. *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, 39(12):996–1004, 1992.

[137] H.K. Khalil. *Nonlinear Systems*. MacMillan, New York, 1992.

[138] G. Lafferriere, G.J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, March 2000.

[139] F. Lamnabhi-Lagarrigue and J. Lunze, editors. *HYCON Handbook of Hybrid Systems: Control Theory, Tools and Applications*. Cambridge University Press, Cambridge, UK, 2009.

[140] M. Lazar, D. Munoz de la Pena, W.P.M.H. Heemels, and T. Alamo. On the stability of min-max nonlinear model predictive control. *Systems & Control Letters*, 57(1):39–48, 2008.

[141] M. Lazar and W.P.M.H. Heemels. Predictive control of hybrid systems: Input-to-state stability results for sub-optimal solutions. *Automatica*, 45(1):180–185, 2009.

[142] M. Lazar, W.P.M.H. Heemels, B.J.P. Roset, H. Nijmeijer, and P.P.J. v.d. Bosch. Input-to-state stabilizing sub-optimal nmpc with an application to dc-dc converters. *International Journal of Robust and Nonlinear Control*, 18(8):890–904, 2008.

[143] M. Lazar, W.P.M.H. Heemels, S. Weiland, and A. Bemporad. Stabilizing model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 51(11):1813–1818, 2006.

[144] K.K. Lee and A. Arapostathis. On the controllability of piecewise-linear hypersurface systems. *Systems & Control Letters*, 9:89–96, 1987.

[145] D.M.W. Leenaerts and W.M.G. van Bokhoven. *Piecewise Linear Modeling and Analysis*. Kluwer Academic Publishers, Boston, 1998.

[146] D. Liberzon. *Lecture Notes of the Workshop on the IEEE Conference on Decision and Control*. Las Vegas, USA, 2001.

[147] D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, MA, 2003.

[148] D. Liberzon, J.P. Hespanha, and A.S. Morse. Stability of switched systems: a Lie-algebraic condition. *Systems & Control Letters*, 37:117–122, 1999.

[149] D. Liberzon and A.S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazin*, pages 59–70, 1999.

[150] H. Lin and P.J. Antsaklis. Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Transactions on Automatic Control*, 54(2):308–322, 2009.

[151] B. Lincoln and A. Rantzer. Optimizing linear system switching. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 2063–2068, Orlando, Florida, December 2001.

[152] Y.J. Lootsma, A.J. van der Schaft, and M.K. Çamlıbel. Uniqueness of solutions of relay systems. *Automatica*, 35(3):467–478, 1999.

[153] J. Lygeros. *Lecture Notes on Hybrid Systems*. Department of Engineering, University of Cambridge, Cambridge, UK, 2003.

[154] J. Lygeros, D.N. Godbole, and S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43:522–539, 1998.

[155] J. Lygeros, K.H. Johansson, S.N. Simic, J. Zhang, and S.S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, 2003.

[156] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35:349–370, 1999.

[157] N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer Verlag, 1996.

[158] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, England, 2002.

[159] O. Maler, editor. *Hybrid and Real-Time Systems.* (Proc. Intern. Workshop HART'97, Grenoble, France, March 1997.), volume 1201 of *Lecture Notes in Computer Science*, Berlin, 1997. Springer.

[160] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 229–242. Springer Verlag, 1995.

[161] J.L. Mancilla-Aguilar and R.A. Garcia. A converse Lyapunov theorem for nonlinear switched systems. *System and Control Letters*, 41:67–71, 2000.

[162] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[163] K. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous cicuits. In G. von Bochmann and D.K. Probst, editors, *Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1993.

[164] A.M. Meilakhs. Design of stable control systems subject to parametric perturbation. *Automat. Remote Control*, 39:1409–1418, 1978.

[165] D. Mignone, G. Ferrari-Trecate, and M. Morari. Stability and stabilization of piecewise affine and hybrid systems: an lmi approach. In *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, 2000.

[166] A.P. Molchanov and Y.S. Pyatnitskiy. Criteria of absolute stability of differential and difference inclusions encountered in control theory. *Systems & Control Letters*, 13:59–64, 1989.

[167] J. Moody and P. Antslakis. *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Press, 1998.

[168] A.S. Morse. Control using logic-based switching. In A. Isidori, editor, *Trends in Control: A European Perspective*, pages 69–114. Springer-Verlag, Berlin, Germany, 1995.

[169] A.S. Morse, C.C. Pantelides, S. Sastry, and J.M. Schumacher, eds. Special issue on hybrid systems. *Automatica*, 35(3), March 1999.

[170] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[171] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In R.L. Grossmand, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 317–356, New York, 1993. Springer.

[172] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia, Pennsylvania, 1994.

[173] D. Nešić and A.R. Teel. Input-output stability properties of networked control systems. *IEEE Transactions on Automatic Control*, 49(10):1650–1667, 2004.

[174] S. Ovchinnikov. Max-min representation of piecewise linear functions. *Beiträge zur Algebra und Geometrie/Contributions to Algebra and Geometry*, 43(1):297–302, 2002.

[175] P.M. Pardalos and M.G.C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, Oxford, UK, 2002.

[176] K.M. Passino and K.L. Burgess. *Stability Analysis of Discrete Event Systems*. John Wiley & Sons, New York, 1998.

[177] A. Pavlov, N. van de Wouw, and H. Nijmeijer. Convergent piecewise affine systems: Analysis and design – part i: continuous case. In *Proc. CDC/ECC*, Sevilla, Spain, 2005.

[178] D.L. Pepyne. *Performance Optimization Strategies for Discrete Event and Hybrid Systems*. PhD thesis, Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, February 1999.

[179] D.L. Pepyne and C.G. Cassandras. Modeling, analysis, and optimal control of a class of hybrid systems. *Discrete Event Dynamic Systems: Theory and Applications*, pages 175–201, 1998.

[180] D.L. Pepyne and C.G. Cassandras. Hybrid systems in manufacturing. *Proceedings of the IEEE*, 88:1108–1123, July 2000.

[181] J.L. Peterson. Petri nets. *Computing Surveys*, 9(3):223–252, September 1977.

[182] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[183] S. Petterson. Switched state jump observers for switched systems. In *Proceedings of the IFAC World Congress*, Prague, Czech Republic, 2005.

[184] F. Pfeiffer and C. Glocker. *Multibody Dynamics with Unilateral Contacts*. Wiley, Chichester, 1996.

[185] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on the Foundations of Computer Science*, pages 46–57. IEEE Computer Science Press, New York, 1977.

[186] A. Pnueli and J. Sifakis (guest eds.). Special issue on hybrid systems. *Theoretical Computer Science* 138, 1995.

[187] A. Yu. Pogromsky, M. Jirstrand, and P. Spångéus. On stability and passivity of a class of hybrid systems. In *Proceedings of the 37-th IEEE Conference on Decision and Control'98,* Tampa (USA), pages 3705–3710, 1998.

[188] A. Puri, P. Varaiya, and V. Borkar. ϵ-approximation of differential inclusions. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2892–2897, New Orleans, Louisiana, December 1995.

[189] A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. *IEEE Trans. on Automatic Control*, 2000.

[190] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, Wisconsin, 2009.

[191] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, September 1978.

[192] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

[193] C. Scherer and S. Weiland. DISC Lecture Notes on Linear Matrix Inequalities in Control, 2002.

[194] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, UK, 1986.

[195] J.J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[196] U. Söderman and J.-E. Strömberg. Switched bond graphs: Multiport switches, mathematical characterization and systematic composition of computational models. Technical report, Linköping University, Dept. of Computer and Information Science, Sweden, available at `http://www.ida.liu.se/ext/pur/enter/`, 1995.

[197] V. Solo. On the stability of slowly time-varying linear systems. *Mathematics of Control, Signals and Systems*, 7:331–350, 1994.

[198] E.D. Sontag. Nonlinear regulation: the piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–357, 1981.

[199] 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*, volume 1066 of *Lecture Notes in Computer Science*, pages 436–448. Springer, 1996.

[200] E.D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer, New York, 1998. Texts in applied Mathematics, vol. 6.

[201] G. Stremersch and R.K. Boel. On the influencing net and forbidden state control of timed Petri nets with forced transitions. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 3287–3292, Tampa, Florida, December 1998.

[202] L. Tavernini. Differential automata and their discrete simulators. *Nonlinear Analysis, Theory, Methods & Applications*, 11(6):665–683, 1987.

[203] C. Tomlin, G. Pappas, J. Lygeros, D. Godbole, and S. Sastry. Hybrid models of next generation of air traffic management. In *[12]*, pages 378–404, 1997.

[204] F.D. Torrisi, A. Bemporad, and D. Mignone. HYSDEL — A language for describing hybrid systems. Technical Report AUT00-03, ETH Zurich, 2000. `http://control.ethz.ch/~hybrid/hysdel`.

[205] V.I. Utkin. Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control*, 22(2):212–222, 1977.

[206] F.W. Vaandrager and J.H. van Schuppen, editors. *Hybrid systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*. Berlin, Germany: Springer, 1999. (Proceedings of the Second International Workshop on Hybrid Systems: Computation and Control (HSCC'99), Berg en Dal, The Netherlands, March 1999).

[207] W.M.G. van Bokhoven. *Piecewise Linear Modeling and Analysis*. Kluwer, Deventer, The Netherlands, 1981.

[208] N. van de Wouw and A. Pavlov. Tracking and synchronisation for a class of PWA systems. *Automatica*, 44(11):2909–2915, 2008.

[209] T.J.J. van den Boom and B. De Schutter. Model predictive control for perturbed max-plus-linear systems. *Systems & Control Letters*, 45(1):21–33, January 2002.

[210] A.J. van der Schaft and J.M. Schumacher. The complementary-slackness class of hybrid systems. *Mathematics of Control, Signals and Systems*, 9:266–301, 1996.

[211] A.J. van der Schaft and J.M. Schumacher. Complementarity modelling of hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):483–490, 1998.

[212] A.J. van der Schaft and J.M. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, London, 2000.

[213] L. Vandenberghe, B. De Moor, and J. Vandewalle. The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits. *IEEE Transactions on Circuits and Systems*, 36(11):1382–1391, November 1989.

[214] R. Vidal, S. Soatto, and S. Sastry. An algebraic geometric approach for identification of linear hybrid systems. In *Proceedings of 42nd IEEE Conference on Decision and Control*, 2003.

[215] Y. Wardi, D.L. Pepyne, and C. G. Cassandras. A backward algorithm for computing optimal controls for single-stage manufacturing systems. *Int. J. Prod. Res.*, 2001.

[216] M.A. Wicks and R.A. DeCarlo. Solution of coupled Lyapunov equations for the stabilization of multimodal linear systems. In *Proceedings of the American Control Conference*, pages 1709–1713, 1997.

[217] J.C. Willems. Dissipative dynamical systems. *Archive for Rational Mechanics and Analysis*, 45:321–393, 1972.

[218] J.C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36:259–294, 1991.

[219] H.P. Williams. *Model Building in Mathematical Programming*. Wiley, New York, 3rd edition, 1993.

[220] H.S. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167, 1966.

[221] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 4607–4613, San Diego, California, December 1997.

[222] S.J. Wright. *Primal-Dual Interior Point Methods*. SIAM, Philadelphia, Pennsylvania, 1997.