I. Troch, F. Breitenecker, eds. ISBN 978-3-901608-35-3

Modeling and Control of Hybrid Dynamical Systems: The Hybrid Toolbox for MATLAB

A. Bemporad
University of Siena, Italy

Corresponding author: A. Bemporad[1], Department of Information Engineering, University of Siena, Italy,
bemporad@dii.unisi.it

Abstract. Hybrid systems describe the dynamical interaction between continuous and discrete signals in one common framework. This paper focuses on discrete-time models of hybrid systems that are suitable for solving model predictive control problems and presents the Hybrid Toolbox, a MATLAB/Simulink tool for modeling, simulating, and verifying hybrid dynamical systems, for designing and simulating model predictive controllers for hybrid systems subject to constraints, and for generating linear and hybrid MPC control laws in piecewise affine form that can be directly embedded as C-code in real-time applications.

1 Hybrid models

The mathematical model of a dynamical system is traditionally associated with differential or difference equations, typically derived from physical laws governing the dynamics of the system. Consequently, systems science has mainly focused on models describing the evolution of continuous signals according to smooth linear or nonlinear state transition functions, typically differential or difference equations. In many applications, however, a dynamical system also involves discrete signals satisfying Boolean relations, if-then-else conditions, on/off conditions, etc., that interact with the continuous signals.

The lack of a general theory and of systematic control design tools for systems having such a heterogeneous dynamical discrete and continuous nature led to a considerable interest in the study of *hybrid systems*. After the seminal work [86] in the sixties, only in the mid-nineties there has been a renewed interest in the study of hybrid systems. The main reason for such an interest is probably the advent of technological innovations, in particular in the domain of embedded systems, where a logical/discrete decision device is "embedded" in a physical dynamical environment to change the behavior of the environment itself. Another reason is the availability of several software packages for simulation and numerical/symbolic computation that support the theoretical developments.

Several modelling frameworks for hybrid systems have appeared in the literature. We refer the interested reader to [5] and the references therein. Each class is usually tailored to solve a particular problem, and many of them look largely dissimilar, at least at first sight. Two main categories of hybrid systems were successfully adopted for analysis and synthesis [26]: *hybrid control systems* [62,63], where continuous dynamical systems and discrete/-logic automata interact (see Figure 1), and *switched systems* [27,53,77,78,87], where the state-space is partitioned into regions, each one being associated with a different continuous dynamics (see Figure 2). Today, there is a widespread agreement in defining hybrid systems as dynamical systems that switch among many operating modes, where each mode is governed by its own characteristic dynamical laws, and mode transitions are triggered by variables crossing specific thresholds (state events), by the elapse of certain time periods (time events), or by external inputs (input events) [5].

Hybrid systems arise in a large number of application areas and are attracting increasing attention in both academic theory-oriented circles as well as in industry, for instance in the automotive industry [6,25,29,41,54,71]. Moreover, many physical phenomena admit a natural hybrid description, like circuits involving relays or diodes [7], biomolecular networks [3], and TCP/IP networks in [47].

In this paper we focus on hybrid models formulated in discrete time, that we will call *discrete hybrid automata* (DHA), whose continuous dynamics is described by linear difference equations and whose discrete dynamics is described by finite state machines, both synchronized by the same clock. Despite the fact that the effects of sampling can be neglected in most applications, we note, however, that interesting mathematical phenomena occurring in hybrid systems, such as Zeno behaviors [55] do not exist in discrete time, as switches can only occur at sampling instants. On the other hand, most of these phenomena are usually a consequence of the continuous-time switching model, rather than the real behavior. Our main motivation for concentrating on discrete-time models stems from the need to solve optimal control problems, for which the continuous-time counterpart would not be easily computable.

[1]The Ford Motor Company is acknowledged for continuous support for the development of the Hybrid Toolbox, and in particular D.H. Hrovat, I.V. Kolmanovsky, S. Di Cairano, and H.E. Tseng. The European Commission is also acknowledged for supporting several theoretical developments through project CC (IST-2001-33520), the network of excellence HYCON (FP6-IST-511368), and project WIDE (FP7-IST-224168).

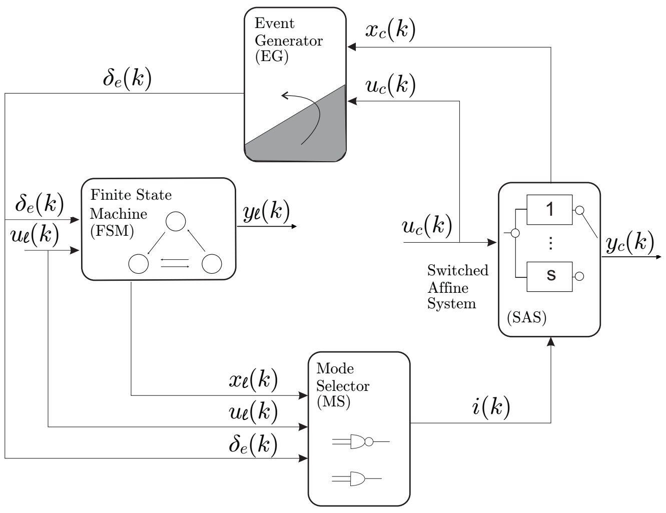

Figure 1: A discrete hybrid automaton (DHA) is the connection of a finite state machine (FSM) and a switched affine system (SAS), through a mode selector (MS) and an event generator (EG)

1.1 Discrete hybrid automata

Discrete hybrid automata (DHA) [83] are the interconnection of a finite state machine and a switched linear dynamic system through a mode selector and an event generator (see Figure 1).

1.1.1 Switched affine system (SAS)

A switched affine system is a collection of linear affine systems:

$$x_c(k+1) = A_{i(k)}x_c(k) + B_{i(k)}u_c(k) + f_{i(k)} \tag{1a}$$
$$y_c(k) = C_{i(k)}x_c(k) + D_{i(k)}u_c(k) + g_{i(k)} \tag{1b}$$

where $k \in \mathbb{Z}^+ \triangleq \{0,1,\ldots\}$ is the time indicator, $x_c \in \mathscr{X}_c \subseteq \mathbb{R}^{n_c}$ is the continuous state vector, $u_c \in \mathscr{U}_c \subseteq \mathbb{R}^{m_c}$ is the exogenous continuous input vector, $\mathscr{X}_c$ and $\mathscr{U}_c$ are convex and closed polyhedra, $y_c \in \mathbb{R}^{p_c}$ is the continuous output vector, $\{A_i, B_i, f_i, C_i, D_i, g_i\}_{i\in\mathscr{I}}$ is a collection of matrices of opportune dimensions, and the mode $i(k) \in \mathscr{I} \triangleq \{1,\ldots,s\}$ is an input signal that chooses the affine state update dynamics.

1.1.2 Boolean Algebra

Before dealing in detail with the other blocks constituting the DHA and introduce further notation, we recall here some basic definitions of Boolean algebra. A variable δ is a Boolean variable if $\delta \in \{0,1\}$, where "$\delta = 0$" means something is false, "$\delta = 1$" that is true. A Boolean expression is obtained by combining Boolean variables through the logic operators $\neg$ (not), $\vee$ (or), $\wedge$ (and), $\leftarrow$ (implied by), $\rightarrow$ (implies), and $\leftrightarrow$ (iff). A Boolean function $f : \{0,1\}^{n-1} \mapsto \{0,1\}$ is used to define a Boolean variable δ_n as a logic function of other variables $\delta_1,\ldots,\delta_{n-1}$, $\delta_n = f(\delta_1,\delta_2,\ldots,\delta_{n-1})$. Given n Boolean variables $\delta_1,\ldots,\delta_n$, a Boolean formula F defines a relation $F(\delta_1,\ldots,\delta_n)$ that must hold true. Every Boolean formula $F(\delta_1,\delta_2,\ldots,\delta_n)$ can be rewritten in the conjunctive normal form (CNF)

$$\text{(CNF)} \quad \bigwedge_{j=1}^{m}\left(\bigvee_{i\in P_j}\delta_i \bigvee_{i\in N_j} \sim \delta_i\right) \tag{2}$$
$$N_j, P_j \subseteq \{1,\ldots,n\},\ \forall j = 1,\ldots,m$$

1.1.3 Event generator (EG)

An event generator is a mathematical object that generates a logic signal according to the satisfaction of a linear affine constraint:

$$\delta_e(k) = f_{\mathrm{H}}(x_c(k), u_c(k), k) \tag{3}$$

where $f_{\mathrm{H}} : \mathscr{X}_c \times \mathscr{U}_c \times \mathbb{Z}^+ \rightarrow \{0,1\}^{r_\ell}$ is a vector of descriptive functions of a linear hyperplane, and $\mathbb{Z}^+ \triangleq \{0,1,\ldots\}$ is the set of nonnegative integers. In particular *threshold events* are modeled as $[\delta_e^i(k) = 1] \leftrightarrow [a^T x_c(k) + b^T u_c(k) \leq c]$, where ^{i} denotes the i-th component of a vector. *Time events* can be also modeled as: $[\delta_e^i(k) = 1] \leftrightarrow [t(k) \geq t_0]$, where $t(k+1) = t(k) + T_s$ denotes time, T_s is the sampling time, and t_0 is a given time.

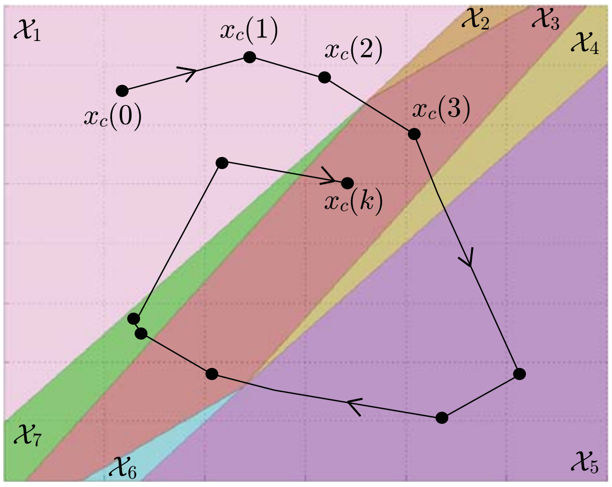

Figure 2: Piecewise affine systems. Mode switches are triggered by threshold events

1.1.4 Finite state machine (FSM)

A finite state machine[2] (or automaton) is a discrete dynamic process that evolves according to a logic state update function:

$$x_\ell(k+1) = f_{\rm B}(x_\ell(k), u_\ell(k), \delta_e(k)) \tag{4a}$$

where $x_\ell\{0,1\}^{n_\ell}$ is the Boolean state, $u_\ell \in \{0,1\}^{m_\ell}$ is the exogenous Boolean input, $\delta_e(k)$ is the endogenous input coming from the EG, and $f_{\rm B} : \{0,1\}^{n_\ell+m_\ell+r_\ell} \to \{0,1\}^{n_\ell}$ is a deterministic logic function. A FSM can be conveniently represented using an oriented graph. A FSM may also have an associated Boolean output

$$y_\ell(k) = g_{\rm B}(x_\ell(k), u_\ell(k), \delta_e(k)) \tag{4b}$$

where $y_\ell \in \{0,1\}^{p_\ell}$ and $g_\ell : \{0,1\}^{n_\ell+m_\ell+r_\ell} \to \{0,1\}^{p_\ell}$.

1.1.5 Mode selector (MS)

The logic state $x_\ell(k)$, the Boolean inputs $u_\ell(k)$, and the events $\delta_e(k)$ select the dynamic mode $i(k)$ of the SAS through a Boolean function $f_{\rm M} : \{0,1\}^{n_\ell+m_\ell+r_\ell} \to \mathscr{I}$, which is therefore called *mode selector*. The output of this function

$$i(k) = f_{\rm M}(x_\ell(k), u_\ell(k), \delta_e(k)) \tag{5}$$

is called *active mode*. We say that a *mode switch* occurs at step k if $i(k) \neq i(k-1)$. Note that, in contrast to continuous-time hybrid models, where switches can occur at any time, in our discrete-time setting a mode switch can only occur at sampling instants. In the following we will use the fact that any discrete variable $\alpha \in \{\alpha_1, \ldots, \alpha_j\}$, admits a Boolean encoding $a \in \{0,1\}^{d(j)}$, where $d(j)$ is the number of bits used to represent $\alpha_1, \ldots, \alpha_j$. From now on we will refer to either the variable or its encoding with the same name. In this way, $f_{\rm M}$ can be also represented as a Boolean function.

DHA are related to *Hybrid Automata* (HA) [4], the main difference is in the time model: DHA admit time in the natural numbers, while in HA the time is a real number. Moreover DHA models do not allow instantaneous transitions, and are deterministic, opposed to HA where any enabled transition may occur in zero time. This has two consequences: (*i*) DHA do not admit live-locks (infinite switches in zero time); (*ii*) DHA do not admit Zeno behaviors (infinite switches in finite time). Finally in DHA models, guards, reset maps and continuous dynamics are limited to linear affine functions. Moreover, contrarily to HA, in DHA the continuous dynamics is not a property of the state of the automaton but is selected by the mode selector (MS) according also to discrete inputs and events. For equivalence results between linear hybrid automata and continuous-time piecewise affine systems see [28]. Reset maps in DHA can be dealt with as described in [83].

1.2 Piecewise Affine Systems

A particular case of DHA is the popular class of *piecewise affine* (PWA) systems introduced by Sontag [78]. Essentially, PWA systems are switched affine systems whose mode only depends on the current location of the state vector, as depicted in Figure 2. PWA systems are defined by partitioning the set of states and inputs into

[2]Here we will only refer to synchronous finite state machines, where the transitions may happen only at sampling times. The adjective synchronous will be omitted for brevity.

polyhedral regions and associating with each region different linear state-update and output equations:

$$\begin{aligned} x_c(k+1) &= A_{i(k)}x(k)_c + B_{i(k)}u_c(k) + f_{i(k)} && \text{(6a)}\\ y_c(k) &= C_{i(k)}x_c(k) + D_{i(k)}u_c(k) + g_{i(k)} && \text{(6b)}\\ & \quad i(k) \text{ such that } H_{i(k)}x_c(k) + J_{i(k)}u_c(k) \le K_{i(k)} && \text{(6c)} \end{aligned}$$

where $x_c(k) \in \mathbb{R}^{n_c}$ is the state vector at time $t \in \mathbb{Z}^+$ is the set of nonnegative integers, $u_c(k) \in \mathbb{R}^{m_c}$ is the input vector, $y_c(k) \in \mathbb{R}^{p_c}$ is the output vector, $i(k) \in \mathscr{I} \triangleq \{1,\dots,s\}$ is the current *mode* of the system, the matrices $A_{i(k)}$, $B_{i(k)}$, $f_{i(k)}$, $C_{i(k)}$, $D_{i(k)}$, $g_{i(k)}$, $H_{i(k)}$, $J_{i(k)}$, $K_{i(k)}$ are constant and have suitable dimensions, and the inequalities in (6c) should be interpreted component-wise. Each linear inequality in (6c) defines a half-space in $\mathbb{R}^n$ and a corresponding hyperplane, that will be also referred to as *guardline*. Each vector inequality (6c) defines a polyhedron $\mathscr{X}_{i(k)}$ in state+input space $\mathbb{R}^{n+m}$ that will be referred to as *cell*, and the union of such polyhedral cells as *partition*.

If the mappings $(x(k),u(k)) \to x(k+1)$ and $(x(k),u(k)) \to y(k)$ are continuous across the guardlines that are facets of two or more cells (and, therefore, they are continuous on their domain of definition), such mappings are single valued and the PWA system is well posed. Discontinuous dynamical behaviors can only be approximated by disconnecting the domain. For instance, the discontinuous state-update equation

$$x_c(k+1) = \begin{cases} x_c(k) + u_c(k) & \text{if} \quad x_c(k) \le 1 \\ 0 & \text{if} \quad x_c(k) > 1 \end{cases} \tag{7a}$$

can be approximated by

$$x_c(k+1) = \begin{cases} x_c(k) + u_c(k) & \text{if} \quad x_c(k) \le 1 \\ 0 & \text{if} \quad x_c(k) \ge 1+\varepsilon \end{cases} \tag{7b}$$

where $\varepsilon > 0$ is an arbitrarily small number, for instance the machine precision. Clearly, system (7) is not defined for $1 < x_c(k) < 1+\varepsilon$, i.e., for the values of the state that cannot be represented in the machine. However, note that the trajectories produced by (7a) and (7b) are identical as long as $x_c(k) > 1+\varepsilon$ or $x_c(k) \le 1$.

In case the partition $\mathscr{X}$ does not cover the whole space $\mathbb{R}^{n+m}$, well-posedness does not imply that trajectories are persistent, i.e., that for all $t \in \mathbb{N}$ a successor state $x(k+1)$ and an output $y(k)$ are defined. A typical case of $\mathscr{X} \neq \mathbb{R}^{n+m}$ is when we are dealing with bounded inputs and bounded states $u_{\min} \le u(k) \le u_{\max}$, $x_{\min} \le x(k) \le x_{\max}$. By embedding such ranges in the inequalities (6c), the system becomes undefined outside the bounds, as no index i exists that satisfies any of the set of inequalities (6c).

PWA systems can model a large number of physical processes, as they can model static nonlinearities through a piecewise-affine approximation, or approximate nonlinear dynamics via multiple linearizations at different operating points. Moreover, tools exist nowadays for obtaining piecewise affine approximations automatically [17, 38, 76].

1.3 Mixed logical dynamical systems

Despite the fact that DHA expressive enoughand are therefore quite suitable for modeling and simulating a wide class of hybrid dynamical systems, they are not directly suitable for solving optimal control problems, because of their heterogeneous discrete and continuous nature. In this section we want to describe how DHA can be translated into a different hybrid model that is more suitable for optimization. In particular, we describe how to transform a DHA into an equivalent hybrid model described by linear mixed-integer equations and inequalities, by generalizing several results already appeared in the literature [20, 49, 70, 73, 85].

1.3.1 Logical functions

Boolean formulas can be equivalently represented as integer linear inequalities. For instance, $\delta_1 \vee \delta_2 = 1$ is equivalent to $\delta_1 + \delta_2 \ge 1$ [85]. Some recurrent equivalences are reported in Table 1.

In general, for every Boolean formula $F(\delta_1,\delta_2,\dots,\delta_n)$ there exists a polyhedral set P such that a set of binary values $\{\delta_1,\delta_2,\dots,\delta_n\}$ satisfies the Boolean formula F if and only if $\delta = [\delta_1\ \delta_2\ \dots\ \delta_n]' \in P$.

Given a formula F, one way of constructing one of such polyhedra P is to rewrite F in an equivalent conjunctive normal form (2) and then simply define P as

$$P = \left\{ \delta \in \{0,1\}^n : \begin{array}{cc} 1 \le \sum_{i\in P_1} \delta_i & + \sum_{i \in N_1}(1-\delta_i) \\ \vdots & \\ 1 \le \sum_{i\in P_m} \delta_i & + \sum_{i \in N_m}(1-\delta_i) \end{array} \right\} \tag{8}$$

The smallest polyhedron P associated with formula F has the following geometric interpretation: Assume to list all the 0-1 combinations of δ_i's satisfying F (namely, to generate the truth table of F), and think each combination as an n-dimensional binary vector in $\mathbb{R}^n$, then P is the convex hull of such vectors [33, 48, 69]. For methods to compute convex hulls, we refer the reader to [39].

relation	Boolean	linear constraints
AND	$\delta_1 \wedge \delta_2$	$\delta_1 = 1$, $\delta_2 = 1$ **or** $\delta_1 + \delta_2 \geq 2$
OR	$\delta_1 \vee X_2$	$\delta_1 + \delta_2 \geq 1$
NOT	$\sim \delta_1$	$\delta_1 = 0$
XOR	$\delta_1 \oplus \delta_2$	$\delta_1 + \delta_2 = 1$
IMPLY	$\delta_1 \rightarrow \delta_2$	$\delta_1 - \delta_2 \leq 0$
IFF	$\delta_1 \leftrightarrow \delta_2$	$\delta_1 - \delta_2 = 0$
ASSIGNMENT $\delta_3 = \delta_1 \wedge \delta_2$	$\delta_3 \leftrightarrow \delta_1 \wedge \delta_2$	$\delta_1 + (1-\delta_3) \geq 1$ $\delta_2 + (1-\delta_3) \geq 1$ $(1-\delta_1)+(1-\delta_2)+\delta_3 \geq 1$

Table 1: Basic conversion of Boolean relations into mixed-integer inequalities. Relations involving the inverted literals $\sim \delta$ can be obtained by substituting $(1-\delta)$ for δ in the corresponding inequalities. More conversions are reported in [68], or can be derived by (2)–(8)

1.3.2 Continuous-logic interfaces

By using the so-called "big-M" technique, events of the form (3) can be equivalently expressed as

$$f^i_{\mathrm{H}}(x_c(k), u_c(k), k) \leq M^i(1-\delta^i_e) \tag{9a}$$

$$f^i_{\mathrm{H}}(x_c(k), u_c(k), k) > m^i \delta^i_e, \qquad i = 1, \ldots, n_e \tag{9b}$$

where M^i, m^i are upper and lower bounds, respectively, on $f^i_{\mathrm{H}}(x_c(k), u_c(k), k)$. Sometimes from a computational point of view it may be convenient to have a system of inequalities without strict inequalities. In this case we will follow the common practice [85] to replace the strict inequality (9b) as

$$f^i_{\mathrm{H}}(x_c(k), u_c(k), k) \geq \varepsilon + (m^i - \varepsilon)\delta^i_e \tag{9c}$$

where ε is a small positive scalar, e.g., the machine precision, although the equivalence does not hold for $0 < f^i_{\mathrm{H}}(x_c(k), u_c(k), k) < \varepsilon$, the numbers in the interval $(0, \varepsilon)$ cannot be represented in a computer.

The most common *logic to continuous* interface is the if-then-else construct

$$\text{IF } \delta \text{ THEN } z = a_1^T x + b_1^T u + f_1 \text{ ELSE } z = a_2^T x + b_2^T u + f_2 \tag{10}$$

which can be translated into [23]

$$\begin{aligned}
(m_2 - M_1)\delta + z &\leq a_2 x + b_2 u + f_2 && \text{(11a)}\\
(m_1 - M_2)\delta - z &\leq -a_2 x - b_2 u - f_2 && \text{(11b)}\\
(m_1 - M_2)(1-\delta) + z &\leq a_1 x + b_1 u + f_1 && \text{(11c)}\\
(m_2 - M_1)(1-\delta) - z &\leq -a_1 x - b_1 u - f_1 && \text{(11d)}
\end{aligned}$$

where M_i, m_i are upper and lower bounds on $a_i x + b_i u + f_i$, $i = 1, 2$, $\delta \in \{0, 1\}$, $z \in \mathbb{R}$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$. Note that when a_2, b_2, f_2 are zero, (10)–(11) coincide with the product $z = \delta \cdot (ax + bu + f)$ described in [85].

1.3.3 Continuous dynamics

A SAS can be rewritten as the combination of linear terms and *if-then-else* rules, as the state-update equation (1a) is equivalent to

$$z_1(k) = \begin{cases} A_1 x_c(k) + B_1 u_c(k) + f_1, & \text{if } (i(k) = 1) \\ 0, & \text{otherwise} \end{cases} \tag{12a}$$

$$\vdots$$

$$z_s(k) = \begin{cases} A_s x_c(k) + B_s u_c(k) + f_s, & \text{if } (i(k) = s) \\ 0, & \text{otherwise} \end{cases} \tag{12b}$$

$$x_c(k+1) = \sum_{i=1}^{s} z_i(k) \tag{12c}$$

where $z_i(k) \in \mathbb{R}^{n_c}, i = 1, \ldots, s$, and (1b) admits a similar transformation.

1.3.4 Mixed logical dynamical models

Given a DHA representation of a hybrid process, by following the techniques for converting logical relations into inequalities that we just described, we obtain an equivalent representation of the DHA as a *mixed logical dynamical*

(MLD) system [20] described by the following relations

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + B_5 \quad (13a)$$

$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_5 \quad (13b)$$

$$E_2\delta(k) + E_3z(k) \leq E_1u(k) + E_4x(k) + E_5 \quad (13c)$$

where $x \in \mathscr{X}_c \times \{0,1\}^{n_\ell}$ is a vector of continuous and binary states, $u \in \mathscr{U}_c \times \{0,1\}^{m_\ell}$ are the inputs, $y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$ the outputs, $\delta \in \{0,1\}^{r_\ell}$ are auxiliary binary variables, $z \in \mathbb{R}^{r_c}$ are continuous auxiliary variables, and A, B_1, B_2, B_3, C, D_1, D_2, D_3, $E_1,\ldots,E_5$ are matrices of suitable dimensions. Given the current state $x(k)$ and input $u(k)$, the time-evolution of (13) is determined by finding a feasible value for $\delta(k)$ and $z(k)$ satisfying (13c), and then by computing $x(k+1)$ and $y(k)$ from (13a)–(13b).

As MLD models consist of a collection of linear difference equations involving both real and binary variables and a set of linear inequality constraints, they are model representations of hybrid systems that can be easily used in optimization algorithms, as described in Section 3. The hybrid systems modeling language *HYSDEL* introduced in [83] and described in Section 2 was developed to describe DHA and to automatically operate the transformations.

A definition of well-posedness of the MLD system (13) can be given by requiring that for all $x(k)$ and $u(k)$ within a given bounded set the pair of variables $\delta(k)$, $z(k)$ satisfying (13c) is unique, so that the successor state $x(k+1)$ and output $y(k)$ are also uniquely defined functions of $x(k)$, $u(k)$ through (13a)–(13b)[3]. The well-posedness assumption is usually guaranteed by the procedure described in Section 1.1.2 used to generate the linear inequalities (13c). Nevertheless, a numerical test for well-posedness is reported in [20, Appendix 1].

Note that the constraints (13c) allow one to specify additional linear constraints on continuous variables (e.g., constraints over physical variables of the system), and logical constraints over Boolean variables. The ability to include constraints, constraint prioritization, and heuristics adds to the expressiveness and generality of the MLD framework. Note also that despite the fact that the description (13) seems to be linear, clearly the nonlinearity is concentrated in the integrality constraints over binary variables.

Finally we recall that the MLD model is similar to the model presented in [37] for verification of safety properties as they both aim at translating a hybrid system in a set of mixed integer linear equalities and inequalities using similar techniques.

1.4 Model Equivalences

In the previous chapters we have presented three different classes of discrete-time hybrid models: PWA systems, DHA, and MLD systems. For what we described in Section 1.1.2, under the assumption that the set of valid states and inputs is bounded, DHA systems can always be equivalently described as MLD systems. Also, a PWA system is a special case of a DHA whose threshold events and mode selector function are defined by the PWA partition. Therefore, a PWA system with bounded partition $\mathscr{X}$ can always be described as an MLD system (an efficient way of modeling PWA systems in MLD form is reported in [20]).

The converse result, namely that MLD systems (and therefore DHA) can be represented as PWA systems, is less obvious. Under the condition that the MLD system is well-posed, this result was proved in [16]. A slightly different and more general proof is reported in [8], where the author also provides efficient MLD to PWA translation algorithms. A different algorithm for obtaining a PWA representation of a DHA is reported in [40].

Such equivalence results are of interest because DHA are most suitable in the modeling phase, but MLD are most suitable for solving open-loop finite-time optimal control problems, and PWA are most suitable for solving finite-time optimal control problems in state-feedback form, as will be described in Section 3.

The fact that PWA systems are equivalent to interconnections of linear systems and finite automata was also pointed out by Sontag [79]. In [46] the authors proved the equivalence of discrete-time PWA/MLD systems with other classes of discrete-time hybrid systems (possibly under some hypothesis) such as linear complementarity (LC) systems [44, 45, 84], extended linear complementarity (ELC) systems [35], and max-min-plus-scaling (MMPS) systems [36].

2 The HYSDEL Modeling Language

A modeling language was proposed in [83] to describe DHA models, called HYbrid System DEscription Language (HYSDEL). The HYSDEL description of a DHA is an abstract modeling step. The associated HYSDEL compiler then translates the description into several computational models, in particular into a MLD using the technique presented in Section 1.3, and PWA form using either the approach of [40] or the approach of [8]. HYSDEL can generate also a simulator that runs as a function in MATLAB. The HYSDEL compiler is available at `http://control.ee.ethz.ch/~hybrid/hysdel` and is supported by the Hybrid Toolbox [9].

[3]For a more general definition of well-posedness of MLD systems see [20].

Any HYSDEL list is composed of two parts. The first one, called INTERFACE, contains the declaration of all variables and parameters, so that it is possible to make the proper type checks. The second part, IMPLEMENTATION, is composed of specialized sections where the relations among the variables are described. These sections are described next (see Table 2 for an example).

AUX Section The HYSDEL section AUX contains the declaration of the auxiliary variables used in the model. These variables will become the δ and z variables in the MLD model (13).

AD Section The HYSDEL section AD allows one to define Boolean variables from continuous ones, and is based exactly on the same semantics of the event generator (EG) described earlier. HYSDEL does not provide explicit access to the time instance, however this limitation can be easily overcome as described in Section 1.1.3.

LOGIC Section The section LOGIC allows one to specify arbitrary functions of Boolean variables; in particular the mode selector is a Boolean function that can be modeled in this section.

DA Section The HYSDEL section DA defines continuous variables according to if-then-else conditions. This section models part of the switched affine system (SAS), namely the variables z_i defined in (12a)–(12b). Note that HYSDEL can handle compound logic formulas in the DA section, therefore there is no need to explicitly define a Boolean variable for each mode.

CONTINUOUS Section The CONTINUOUS section describes the linear dynamics, expressed as difference equations. This section models (12c).

A HYSDEL description may have additional sections that we describe below. For a set of examples and the detailed syntax we refer the interested reader to [82] and to the demos of the Hybrid Toolbox [9].

LINEAR Section HYSDEL allows also one to define a continuous variable as an affine function of continuous variables in the LINEAR section. This section, together with the CONTINUOUS and AD sections allows more flexibility when modeling the SAS. This extra flexibility allows algebraic loops that may render undefined the trajectories of the model. The HYSDEL compiler integrates a semantic checker that is able to detect and report such abnormal situations.

AUTOMATA Section The AUTOMATA section specifies the state transition equations of the finite state machine (FSM) as a collection of Boolean functions $x_{\ell i}(k+1) = f_{\mathrm{B}i}(x_\ell(k), u_\ell(k), \delta_e(k))$, $i = 1, \ldots, n_\ell$.

OUTPUT Section The OUTPUT section allows one to specify static linear and logic relations for the output vector $y = \left[\begin{smallmatrix} y_c \\ y_\ell \end{smallmatrix}\right]$.

Finally HYSDEL allows one more section:

MUST Section This section specifies arbitrary linear and logic constraints on continuous and Boolean variables, and therefore it allows for defining (mixed) linear/logical constraints on states, inputs, and outputs).

HYSDEL allows generating MLD models in MATLAB. Thanks to the equivalences mentioned in the previous section, such models can be immediately (and efficiently) translated into PWA systems [8, 40].

3 Model predictive control of hybrid systems

Model Predictive Control (MPC) is a widely spread technology in industry for control design of highly complex multivariable processes [10, 32, 64, 72, 74]. The idea behind MPC is to start with a model of the open-loop process that explains the dynamical relations among system's variables (command inputs, internal states, and measured outputs). Then, constraint specifications on system variables are added, such as input limitations (typically due to actuator saturation) and desired ranges where states and outputs should remain. Desired performance specifications complete the control problem setup and are expressed through different weights on tracking errors and actuator efforts (as in classical linear quadratic regulation). At each sampling time, an open-loop optimal control problem based on the given model, constraints, weights, and with initial condition set at the current (measured or estimated) state, is repeatedly solved through numerical optimization. The result of the optimization is an optimal sequence of future control moves. Only the first sample of such a sequence is actually applied to the process; the remaining moves are discarded. At the next time step, a new optimal control problem based on new measurements is solved over a shifted prediction horizon.

After reviewing the basics of MPC based on linear models, in this section we focus on MPC based on MLD models for control design of hybrid systems.

3.1 Linear model predictive control

The simplest MPC algorithm is based on the linear discrete-time prediction model

$$\begin{cases} x(k+1) & = & = Ax(k) + Bu(k) \\ y(k) & = & Cx(k) \end{cases} \tag{14}$$

of the open-loop process, where $x(k) \in \mathbb{R}^n$ is the state vector at time t, $u(k) \in \mathbb{R}^m$ is the vector of manipulated variables to be determined by the controller, $y(k) \in \mathbb{R}^m$ is the vector of controlled outputs, and on the solution of the finite-time optimal control problem

$$\min_U \quad x_N' P x_N + \sum_{j=0}^{N-1} x_j' Q x_j + u_j' R u_j \tag{15a}$$

$$\text{s.t.} \quad x_{j+1} = Ax_j + Bu_j,\ j = 0, \dots, N-1 \tag{15b}$$

$$y_j = Cx_j \tag{15c}$$

$$x_0 = x(k) \tag{15d}$$

$$u_{\min} \leq u_j \leq u_{\max},\ j = 0, \dots, N-1 \tag{15e}$$

$$y_{\min} \leq y_j \leq y_{\max},\ j = 1, \dots, N \tag{15f}$$

where N is the prediction horizon, $U \triangleq [u_0' \cdots u_{N-1}']' \in \mathbb{R}^{Nm}$ is the sequence of manipulated variables to be optimized, $Q = Q' \geq 0$, $R = R' > 0$, and $P = P' \geq 0$ are weight matrices of appropriate dimensions defining the performance index, $u_{\min}, u_{\max} \in \mathbb{R}^m$, $y_{\min}, y_{\max} \in \mathbb{R}^p, C \in \mathbb{R}^{p \times n}$ define constraints on input and state variables, respectively, and "$\leq$" denotes component-wise inequalities. By substituting $x_j = A^j x(k) + \sum_{i=0}^{j-1} A^i B u_{j-1-i}$, Eq. (15) can be recast as the Quadratic Programming (QP) problem

$$U^*(x(k)) \triangleq \arg\min_U \quad \frac{1}{2} U' H U + x'(k) C' U + \frac{1}{2} x'(k) Y x(k) \tag{16a}$$

$$\text{s.t.} \quad GU \leq W + Sx(k) \tag{16b}$$

where $U^*(x(k)) = [u_0'^*(x(k)) \cdots u_{N-1}'^*(x(k))]'$ is the optimal solution, $H = H' > 0$ and C, Y, G, W, S are matrices of appropriate dimensions [9,21,22]. Note that Y is not needed to compute $U^*(x(k))$, it only affects the optimal value of (16a).

The MPC control algorithm is based on the following iterations: at time t, measure or estimate the current state $x(k)$, solve the QP problem (16) to get the optimal sequence of future input moves $U^*(x(k))$, apply

$$u(k) = u_0^*(x(k)) \tag{17}$$

to the process, discard the remaining optimal moves, repeat the procedure again at time $k+1$.

In the absence of constraints (15e) – (15f), for $N \to \infty$ (or, equivalently, for $N < \infty$ and by choosing P as the solution of the algebraic Riccati equation associated with matrices (A, B) and weights (Q, R)), the MPC control law (16) – (17) coincides with the Linear Quadratic Regulator (LQR) [21]. From a design viewpoint, the MPC setup (15) can be therefore thought as a way of bringing the LQR methodology to systems with constraints.

The basic MPC setup (15) can be extended in many ways. In particular in tracking problems usually one has to make the output vector $y(k)$ track a reference signal $r(k) \in \mathbb{R}^p$ under constraints (15e)–(15f). In order to do so, the cost function (15a) is replaced by

$$\sum_{j=0}^{N-1} (y_j - r(k)) Q_y (y_j - r(k)) + \Delta u_j' R \Delta u_j \tag{18}$$

where $Q_y = Q_y' \geq 0 \in \mathbb{R}^{p \times p}$ is a matrix of output weights, and the increments of command variables $\Delta u(k) \triangleq u(k) - u(k-1)$ are the new optimization variables, possibly further constrained by $\Delta u_{\min} \leq \Delta u_j \leq \Delta u_{\min}$. In the above tracking setup vector $[x'(k)\ r'(k)\ u'(k-1)]'$ replaces $x(k)$ in (16b) and the control law (17) becomes $u(k) = u(k-1) + \Delta u_0^*(x(k), r(k), u(k-1))$.

The standard way of computing the linear MPC control action, which is implemented in most commercial MPC packages, is to solve the QP problem (16) on line at each time k (for example in the *MPC Toolbox for MATLAB*, The Mathworks, Inc., [22]).

Besides MPC schemes based on linear prediction models, several formulations of MPC based on general smooth nonlinear prediction models (as well as on uncertain linear models) exist. Most of them rely on nonlinear optimization methods for generic nonlinear functions/constraints to compute the control actions, and are therefore more rarely deployed in practical applications.

3.2 Hybrid model predictive control

MPC based on hybrid dynamical models has emerged in recent years as a very promising approach to operate switching linear dynamics, on/off inputs, logic states, subject to combinations of linear and logical constraints on input and state variables [20]. Hybrid dynamics are often so complex that a satisfactory feedback controller cannot be synthesized by using analytical tools, and heuristic design procedures usually require trial and error sessions, extensive testing, are time consuming, costly and often inadequate to deal with the complexity of the hybrid control problem properly.

As for the linear MPC case, hybrid MPC design is a systematic approach to meet performance and constraint specifications in spite of the aforementioned switching among different linear dynamics, logical state transitions, and more complex logical constraints on system's variables. The approach consists of modeling the switching open-loop process and constraints as a discrete hybrid automaton using the language *HYSDEL* [83], and then automatically transforming the model into the MLD form (13) as described in Section 1.3.

The associated finite-horizon optimal control problem based on quadratic costs takes the form (16) with $U = [u_0' \cdots u_{N-1}' \; \delta_0' \cdots \delta_{N-1}' \; z_0' \cdots z_{N-1}']'$ and subject to the further restriction that some of the components of U must be either 0 or 1. The problem is therefore a Mixed-Integer Quadratic Programming (MIQP) problem, for which both commercial [34,51] and public domain solvers (such as the one of [19]) are available. When infinity norms $\|Qx_k\|_\infty$, $\|Ru_k\|_\infty$, $\|Px_k\|_\infty$ are used in (15a) in place of quadratic costs, the optimization problem becomes a Mixed-Integer Linear Programming (MILP) problem [9, 12], which can be also handled by efficient public domain solvers such as [65], as well as by commercial solvers [34, 51].

Unfortunately MIP's are $\mathcal{NP}$-complete problems. However, the state of the art in solving MIP problems is growing constantly, and problems of relatively large size can be solved quite efficiently. While MIP problems can always be solved to the global optimum, closed-loop stability properties can be guaranteed as long as the optimum value in (16) decreases at each time step. Usually MIP solvers provide good feasible solutions within a relatively short time compared to the total time required to find and certify the global optimum. In the worst-case the complexity of optimally computing the control action $u(k)$ in (17) on line at each time t depends exponentially on the number of integer variables [73]. In principle, this limits the scope of application of the proposed method to relatively slow systems, since the sampling time should be large enough for real-time implementation to allow the worst-case computation.

In general an MIP solver provides the solution after solving a sequence of relaxed standard linear (or quadratic) problems (LP,QP). A potential drawback of MIP is (1) the need for converting the discrete/logic part of the hybrid problem into mixed-integer inequalities, therefore losing most of the original discrete structure, and (2) the fact that its efficiency mainly relies upon the tightness of the continuous LP/QP relaxations. Such drawbacks are not suffered by techniques for solving constraint satisfaction problems (CSP), i. e., the problem of determining whether a set of constraints over discrete variables can be satisfied. Under the class of CSP solvers we mention constraint logic programming (CLP) [66] and satisfiability (SAT) solvers [43], the latter specialized for the satisfiability of Boolean formulas. The approach of [18] combines MIP and CSP techniques in a cooperative way. In particular, convex programming for optimization over real variables, and SAT solvers for determining the satisfiability of Boolean formulas (or logic constraints) are combined in a single branch and bound solver.

Another approach for reducing the complexity of on-line computations is to look for suboptimal solutions. For instance in [52] the authors propose to suitably constrain the mode sequence over the prediction horizon, so that on-line optimization is solved more quickly. Although closed-loop stability is still guaranteed by this approach, clearly in general the overall tracking performance of the feedback loop gets deteriorated.

In the last decade, *explicit model predictive control* has been proposed as a way to completely get rid of the need of on-line solvers (see [2] for a survey on explicit MPC).

For linear MPC, to get rid of on-line QP an approach to evaluate the MPC law (17) was proposed in [21]. Rather then solving the QP problem (16) on line for the current vector $x(k)$, the idea is to solve (16) *off line* for all vectors x within a given range and make the dependence of u on x *explicit* (rather than implicitly defined by the optimization procedure (16)). The key idea is to treat (16) as a *multiparametric* quadratic programming problem, where $x(k)$ is the vector of parameters. It turns out that the optimizer $U^* : \mathbb{R}^n \to \mathbb{R}^{mN_u}$ is a piecewise affine and continuous function, and consequently the MPC controller defined by (17) can be represented explicitly as

$$u(x) = \begin{cases} F_1 x + g_1 & \text{if} \quad H_1 x \le k_1 \\ \quad\vdots & \quad\vdots \\ F_M x + g_M & \text{if} \quad H_M x \le k_M. \end{cases} \tag{19}$$

It turns also out that the set of states X^* for which problem (16) admits a solution is a polyhedron, and that the optimum value in (16) is a piecewise quadratic, convex, and continuous function of $x(k)$. The controller structure (19) is simply a look-up table of linear gains (F_i, g_i), where the ith gain is selected according to the set of linear inequalities $H_i x \le K_i$ that the state vector satisfies. Hence, the evaluation of the MPC controller (17), once put in the form (19), can be carried out by a very simple piece of control code. In the most naive implementation, the number of operations depends linearly in the worst case on the number M of partitions, or even logarithmically if the partitions are properly stored [81].

An alternative way to solving MIP problems on line is to extend explicit MPC ideas to the hybrid case. For hybrid MPC problems based on infinity norms, [12] showed that an equivalent piecewise affine explicit reformulation – possibly discontinuous, due to binary variables – can be obtained through off-line multiparametric mixed-integer linear programming techniques.

Thanks to the possibility of converting hybrid models (such as those designed through *HYSDEL*) to an equivalent piecewise affine (PWA) form [8], an explicit hybrid MPC approach dealing with quadratic costs was proposed in [24], based on dynamic programming (DP) iterations. Multiparametric quadratic programs (mpQP) are solved at each iteration, and quadratic value functions are compared to possibly eliminate regions that are proved to never be optimal. A different approach still exploiting the PWA structure of the hybrid model was proposed in [67], where all possible switching sequences are enumerated, an mpQP is solved for each sequence, and quadratic costs are compared on-line to determine the optimal input (in this respect, one could define the approach semi-explicit). To overcome the problem of enumerating all switching sequences and storing all the corresponding mpQP solutions, backwards reachability analysis is exploited in [1] (and implemented in the *Hybrid Toolbox*). A procedure to post-process the mpQP solutions and eliminate all polyhedra (and their associated control gains) that never provide the lowest cost was suggested in [1]. Typically the DP approach provides simpler explicit solutions when long horizons N are chosen, but on the contrary tends to subdivide the state space in a larger number of polyhedra than the enumeration approach for short horizons.

For closed-loop convergence results of hybrid MPC the reader is referred to [20, 30, 59–61] and to the PhD thesis [58]. Extensions of hybrid MPC to stochastic hybrid systems was proposed in [13], and to event-based continuous-time hybrid systems in [14].

4 The Hybrid Toolbox for MATLAB

The Hybrid Toolbox for MATLAB [9] provides a nice development environment for hybrid and explicit MPC design. Hybrid dynamical systems described in HYSDEL are automatically converted to MATLAB MLD and PWA objects. MLD and PWA objects can be validated in open-loop simulation, either from the command line or through their corresponding Simulink blocks. Hybrid MPC controllers based on MILP/MIQP optimization can be designed and simulated, either from the command line or in Simulink, and can be converted to their explicit form for deployment. Several demos are available in the Hybrid Toolbox distribution. The toolbox can be freely downloaded from `http://www.dii.unisi.it/hybrid/toolbox`. Similar functionalities are also included in the Multi Parametric Toolbox [57].

4.1 Toolbox features

The toolbox offers the following features:

4.1.1 Hybrid model design, simulation, and reachability analysis

Hybrid models can be conveniently described in HYSDEL. HYSDEL models are converted to MLD models that are handled as MATLAB objects. MLD objects can be automatically converted into PWA objects, using an implementation of the conversion algorithm described in [8]. MLD and PWA objects can be simulated in open-loop for validation purposes through command-line functions or MATLAB scripts, or in Simulink. Safety properties can be verified through reachability analysis based on mixed-integer programming.

4.1.2 Control design

Model predictive controllers (MPC) based on on-line mixed-integer linear or quadratic programming (MILP/MIQP) can be designed for hybrid systems converted to MLD form. MPC performance functions based on both quadratic and infinity norms are supported.

4.1.3 Explicit control design

The toolbox provides a multi-parametric quadratic programming (mpQP) solver based on the algorithm described in [80], a multi-parametric linear programming (mpLP) solver based on a similar implementation, and a multi-parametric hybrid optimal control solver based on the method described in [1]. Several functions for manipulation and visualization of polyhedral objects and polyhedral partitions are provided by the toolbox.

MPC controllers designed for hybrid systems and for constrained linear systems can be converted to their equivalent explicit piecewise affine form via offline optimization (multiparametric programming). Linear MPC controllers designed with the Model Predictive Control Toolbox for MATLAB [22] can be also converted into piecewise affine form via multiparametric quadratic programming.

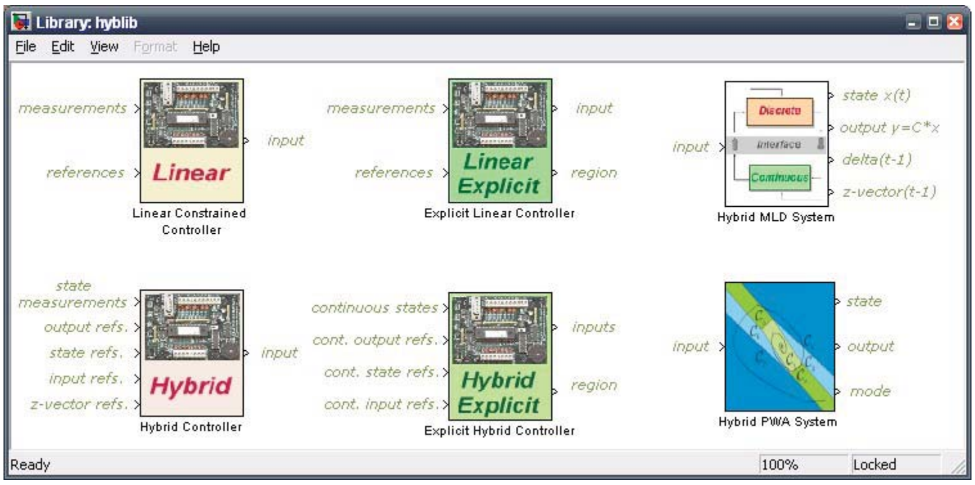

Figure 3: Simulink library of the Hybrid Toolbox

4.1.4 C-code generation

Explicit controllers can be easily exported as C-code for direct implementation and rapid prototyping, by either simply running Real Time Workshop or by embedding the generated C code in the application.

4.1.5 Simulink library

As depicted in Fig. 3, Simulink blocks are provided for controllers based on on-line optimization (MILP/MIQP or QP), explicit piecewise linear controllers (for hybrid or linear systems), and for simulation and validation of MLD and PWA models.

4.1.6 Solver support

The Hybrid Toolbox supports several solvers for linear, quadratic, and mixed-integer optimization. The list of supported solvers includes the GNU Linear Programming Kit for LP/MILP [65] through the GLPKMEX MATLAB interface, the QP solver of the MPC Toolbox, the LP/QP/MILP/MIQP solvers of Cplex [50] through the CPLEXMEX interface, the LP/QP/MILP/MIQP solvers of Xpress-MP [34] through the MEXPRESS interface, the LP/QP solvers of the NAG Foundation Toolbox, the LP/QP solvers of the Optimization Toolbox, and the free MIQP solver `MIQP.M` [19]. The source code of the MEX interfaces can be downloaded from `http://www.dii.unisi.it/hybrid/tools`.

4.1.7 Demos

Several demos are provided to highlight different functionalities of the toolbox. In the following we briefly describe the features of the toolbox through simple illustrating examples.

Consider the problem of regulating the temperatures T_1 and T_2 of two bodies in a room (see demo `heatcool.m` in the `demos/hybrid/` directory of the Hybrid Toolbox). The room is equipped with a heater and an air conditioning system, which are automatically activated according to the following rules:

- body #1 turns the heater on whenever its temperature T_1 is below a certain threshold $T_{\text{cold},1}$, and similarly turns the air conditioning system on when T_1 is above a certain threshold $T_{\text{hot},1}$;
- body #2 turns the heater on whenever its temperature T_2 is below a certain threshold $T_{\text{cold},2}$, unless body #1 is hot ($T_1 \geq T_{\text{hot},1}$); similarly, body #2 turns the air conditioning system on whenever T_2 is above a certain threshold $T_{\text{hot},2}$, unless body #1 is cold ($T_1 \leq T_{\text{cold},1}$);
- otherwise, both the heater and the air conditioning system are off.

The dynamics can be interpreted as a double thermostat, with thermostat #2 having lower priority than thermostat #1.

By letting u_{hot} be the amount of heat produced by the heater, and u_{cold} the amount of heat subtracted by the air conditioner, the dynamics of T_1, T_2 is

$$\dot{T}_1 = -\alpha_1(T_1 - T_{\text{amb}}) + k_1(u_{\text{hot}} - u_{\text{cold}}) \tag{20}$$
$$\dot{T}_2 = -\alpha_2(T_2 - T_{\text{amb}}) + k_2(u_{\text{hot}} - u_{\text{cold}}) \tag{21}$$

where the ambient temperature T_{amb} is treated as a continuous input to the system (assume for instance that the ambient temperature of the room can be changed by opening a window), and $k_1 = 0.8$, $k_2 = 0.4$, $\alpha_1 = 1$, $\alpha_2 = 0.5$,

```
/* Heat and cool example - (C) 2003 by A. Bemporad */
SYSTEM heatcool {
INTERFACE {
    STATE { REAL T1 [-10,50];
            REAL T2 [-10,50];
        }
    INPUT { REAL Tamb [-50,50];
        }
    OUTPUT {REAL y1;
            REAL y2;}
    PARAMETER {
        REAL Ts, alpha1, alpha2, k1, k2;
        REAL Thot1, Tcold1, Thot2, Tcold2, Uc, Uh;
    }
}
IMPLEMENTATION {
        AUX { REAL uhot, ucold;
              BOOL hot1, hot2, cold1, cold2;
        }
        AD  { hot1 = T1>=Thot1;
              hot2 = T2>=Thot2;
              cold1 = T1<=Tcold1;
              cold2 = T2<=Tcold2;
        }
        DA  { uhot = {IF cold1 | (cold2 & ~hot1) THEN Uh ELSE 0};
              ucold = {IF hot1 | (hot2 & ~cold1) THEN Uc ELSE 0};
        }
        CONTINUOUS { T1 = T1+Ts*(-alpha1*(T1-Tamb)+k1*(uhot-ucold));
                     T2 = T2+Ts*(-alpha2*(T2-Tamb)+k2*(uhot-ucold));
        }
        OUTPUT {y1=T1;
                y2=T2;
        }
    }
}
```

Table 2: HYSDEL model for the hybrid thermal system (`heatcoolmodel.hys`)

$U_c = U_h = 2$, $T_{\text{cold},1} = 15$, $T_{\text{hot},1} = 30$, $T_{\text{cold},2} = 10$, $T_{\text{hot},2} = 35$ (no physical units are provided here as the model is a toy example with little physical significance).

To obtain a HYSDEL model of the system, we sample the continuous dynamics with sampling time $T_s = 0.5$, by replacing $\dot{T}$ with $\frac{T(k+1)-T(k)}{Ts}$ in (21) (Euler approximation). The overall hybrid dynamics is described by the HYSDEL model `heatcoolmodel.hys` reported in Table 2.

The corresponding MLD model is simply obtained through the MATLAB command

```
>> S=mld('heatcoolmodel',Ts);
```

which returns the MLD object S containing all MLD matrices, dimensions of system variables, etc. In this case the system has 2 continuous states, 1 continuous input, 2 continuous auxiliary variables, 6 binary auxiliary variables, and consists of 20 mixed-integer linear inequalities.

The PWA equivalent P of S is obtained as follows:

```
>> P=pwa(S);
```

and consists of 5 regions in the three-dimensional space $(T_1, T_2, T_{\text{amb}})$, as depicted in Fig. 4.

The MLD system S or its equivalent PWA system P can be now simulated in open loop for inspecting the dynamical behavior of the hybrid system and for validation purposes. For example, given the initial condition $T_1(0) = 30$, $T_2(0) = 20$ and input signal $T_{\text{amb}}(k) = 20 + 10\cos(k/10)$, $k = 0, 1, \ldots$, the command

```
>> [X,T,D,Z,Y]=sim(S,[30;20],Tamb);
```

generates the trajectories of state, auxiliary, and output vectors of the MLD system, and similarly

```
>> [X,T,I,Y]=sim(P,[30;20],Tamb);
```

generates state, mode, and output trajectories of the PWA system. Clearly, the state and output trajectories generated by the MLD and PWA system coincide.

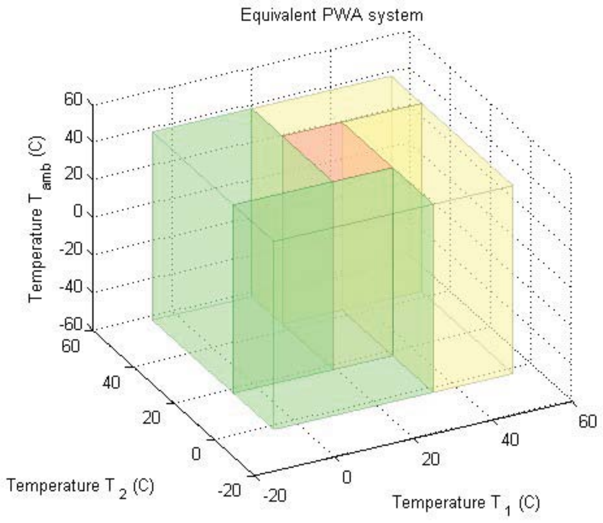

Figure 4: PWA equivalent of thermal system. Note that the partition does not depend on the input variable T_{amb}

4.2 Verification of safety properties

Simulation tools allow probing a model for a certain initial condition and input excitation. Reachability analysis aims at detecting whether a hybrid model will eventually reach unsafe state configurations for *all* possible initial conditions and input excitations within a prescribed set.

For MLD systems the problem is to test whether a certain terminal unsafe set $X_f = \{x : S_x x \le T_x\}$ can be reached after exactly N steps, starting from a set of initial states $X_0 = \{x : S_0 x \le T_0\}$ for some input sequence $u(0), \dots, u(N-1)$ with $u_{\min} \le u(k) \le u_{\max}$, $\forall k = 0,\dots,N-1$. Such a sequence exists if the following set of mixed-integer linear inequalities is feasible:

$$\begin{cases} S_0 x(0) & \le T_0 \\ x(k+1) & = Ax(k) + B_1 u(k) + B_2 \delta(k) + B_3 z(k),\ k = 0,1,\dots,N-1 \\ E_2 \delta(k) & + E_3 z(k) \le E_1 u(k) + E_4 x(k) + E_5,\ k = 0,1,\dots,N-1 \\ u_{\min} & \le u(k) \le u_{\max},\ k = 0,1,\dots,N-1 \\ S_x x(N) & \le T_x \end{cases} \tag{22}$$

with respect to the unknowns $u(0)$, $\delta(0)$, $z(0)$, $\dots$, $u(N-1)$, $\delta(N-1)$, $z(N-1)$, $x(0)$. Complex conditions can be posed in the reachability analysis by adding constraints (such as linear constraints, logical constraints, and combinations of both) in the MUST section of the HYSDEL file that defines the MLD model.

Consider again the thermal example. We want to test if the set of states $X_f = \{(T_1,T_2) : 10 \le T_1, T_2 \le 15\}$ can be reached after $N = 10$ steps from an initial state $x(0)$ in the set $X_0 = \{(T_1,T_2) : 35 \le T_1, T_2 \le 40\}$ with a bounded input $10 \le T_{\mathrm{amb}} \le 30\}$ The reachability analysis question is answered through the following code:

```
>> Xf.A=[eye(2);-eye(2)];
>> Xf.b=[15;15;-10;-10];
>> X0.A=[eye(S.nx);-eye(S.nx)];
>> X0.b=[40;40;-35;-35];
>> umin=10;
>> umax=30;
>> [flag,x0,U,xf,X,T]=reach(S,N,Xf,X0,umin,umax);
```

The answer is `flag=1`, which means that the set of states X_f is reachable from X_0 under the above dynamics and input bounds. The code also returns an example of initial state $x(0) = x_0$, sequence of inputs $[u(0)\ u(1)\ \dots\ u(N-1)] = U$, and the corresponding final state $x(N) = x_f$ and state sequence $[x(0)\ x(1)\ \dots\ x(N-1)] = X$, satisfying the reachability analysis problem. With the lower bound $T_{\mathrm{amb}} \ge 20$, the MILP problem (22) is infeasible, meaning that no input exists driving x_0 to X_f.

A more extended “safety” question is whether the state can never reach X_f at any time k, $1 \le k \le N$, can be also tested. Instead of solving this extended problem by solving the MILP (22) for all horizons between 1 and N, the Hybrid Toolbox provides the answer with one MILP, by introducing an auxiliary binary variable as described in [9]. The corresponding command is

```
>> [flag,x0,U]=reach(S,[1 N],Xf,X0,umin,umax);
```

For a constructive way to embed Linear Temporal Logic (LTL) specifications within the MLD framework the reader is referred to [56].

4.3 Hybrid model predictive control design

Consider the problem to command T_{amb} so that T_2 tracks a given reference trajectory $r(k) = 15\sin(\frac{2}{5}t)$ under the constraint $T_1(k) \geq 25$, $\forall t \geq 0$. To this end we design the hybrid MPC controller based on the optimization problem

$$\min_{\{u,\delta,z\}_0^{N-1}} \sum_{k=1}^{N-1} |Q_x(T_2(t+k|t) - r(k))| \tag{23}$$

$$\text{s. t. } \; x(t+k|t) \geq 25, \; k = 1,\dots,N \tag{24}$$

Using the Hybrid Toolbox the problem is formulated as

```
>> refs.x=2;   % only state x(2) is weighted
>> Q.x=1;      % weight on state x(2)
>> Q.rho=Inf;  % hard constraints
>> Q.norm=Inf; % infinity norm
>> N=2;        % prediction horizon
>> limits.xmin=[25;-Inf];
>> C=hybcon(S,Q,N,limits,refs);
```

where C is now an MPC controller object of type `@hybcon`. A closed-loop simulation over a time interval of $T_{\text{stop}} = 100$ time units from the initial condition $T_1(0) = 30$, $T_2(0) = 30$ is obtained through the commands

```
>> r.x=30+15*sin(2/5*(0:Ts:Tstop-Ts)'/5);
>> [X,U,D,Z,T,Y]=sim(C,S,r,[30;20],Tstop);
```

producing the closed-loop trajectories reported in Fig. 5. The closed-loop control system can also be simulated in Simulink using the "Hybrid Controller" block reported in Fig. 3.

In case we want to use the quadratic penalty $(T_2(t+k|t) - r(k))^2$ and the MIQP solver of CPLEX, it is enough to set

```
>> Q.norm=2; % use quadratic costs
>> C=hybcon(S,Q,N,limits,refs);
>> C.mipsolver='cplex';
```

4.4 Explicit hybrid model predictive control

The hybrid MPC controller C can be converted to its equivalent explicit PWA form through the following command

```
>> E=expcon(C,range,options);
```

where `range` is a structure defining the bounds of interest for measured states and references, and `options` is a structure defining parameters for the multiparametric solvers. The output is an explicit `@excpon` controller object consisting of 12 affine gains, defined over the polyhedral partition depicted in Fig. 6. The closed-loop system containing the explicit controller can be either simulated in MATLAB through command `sim` or in Simulink using the "Explicit Hybrid Controller" block reported in Fig. 3. In the latter case Real-Time Workshop can be employed to rapidly prototype the controller.

The Hybrid Toolbox handles constrained *linear* MPC designs `@lincon` and their explicit counterparts `@expcon` in a similar fashion.

5 Conclusions

Since its first release at the end of 2004 the toolbox has been requested to the author by approximately 2200 users to date, and employed in several industrial applications in numerous different domains. The modeling and control tools provided by the Hybrid Toolbox are mainly useful for the following areas [10]:

- *MPC of complex dynamical plants that cannot be handled by linear models due to logical states, switching dynamics, nonlinearities, and to the enforcement of logical rules and other constraints.* A hybrid MPC controller can be designed through the toolbox and embedded in existing Simulink diagrams for realistic simulation based on mixed-integer programming solutions. Real-time implementation was successfully carried out through both the xPC-Target and OPC Toolbox for MATLAB (sampling time in the second to hour range). Typical applications include on-line rescheduling of production plants and supervisory control of industrial processes.

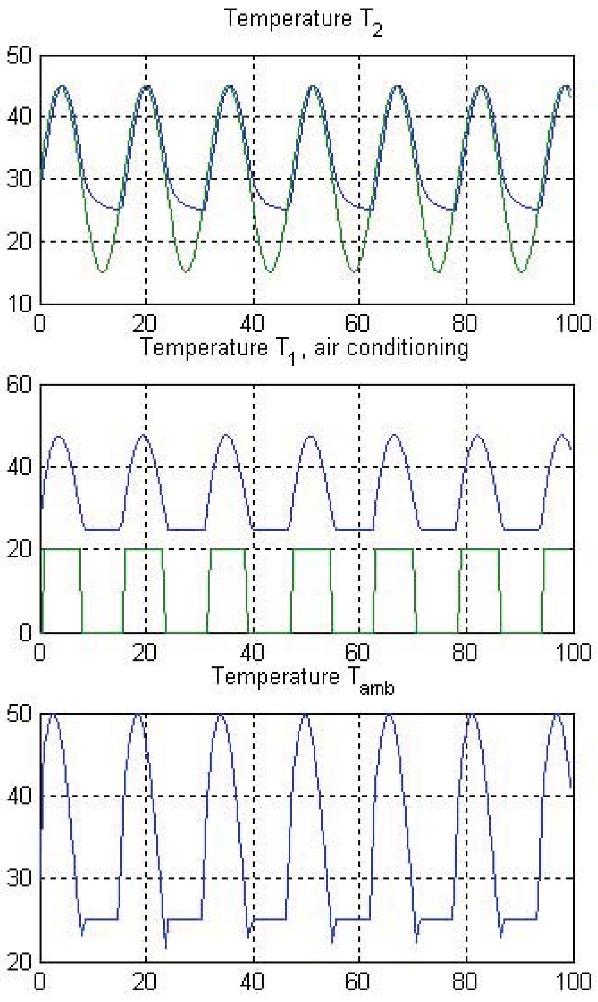

Figure 5: Closed-loop hybrid MPC simulation

- *MPC of small-size fast dynamical processes without on-line optimization.* Thanks to the capabilities of the toolbox to convert linear or hybrid MPC laws into a lookup-table of linear gains and automatically generate the corresponding C code, MPC can be smoothly embedded in any real-time application (sampling frequency up to 100kHz, depending on the control platform).

Typical applications of explicit and hybrid MPC of fast dynamical processes are in the automotive industry, as testified by numerous successful test cases: control of semi-active suspensions [41], control of magnetic actuators subject to position and velocity contraints [15, 29], optimal management strategies for the simultaneous control of engine, electrical mootors, and battery in a hybrid vehicle [75], idle speed control of combustion engines [31], traction control [25], air-to-fuel ratio and torque control in advanced technology gasoline direct injection stratified charge (DISC) engines [42], control of robotized gearbox for reduction of consumptions and emissions [11], adaptive cruise control [71].

6 References

[1] A. Alessio and A. Bemporad. Feasible mode enumeration and cost comparison for explicit quadratic model predictive control of hybrid systems. In *2nd IFAC Conference on Analysis and Design of Hybrid Systems*, pages 302–308, Alghero, Italy, 2006.

[2] A. Alessio and A. Bemporad. A survey on explicit model predictive control. In D.M. Raimondo L. Magni, F. Allgower, editor, *Proc. Int. Workshop on Assessment and Future Directions of Nonlinear Model Predictive Control (Pavia, Italy)*, Lecture Notes in Control and Information Sciences, 2009.

[3] R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In M.D. Di Benedetto and A. Sangiovanni Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, 2001.

[4] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In A.P. Ravn R.L. Grossman, A. Nerode and H. Rischel,

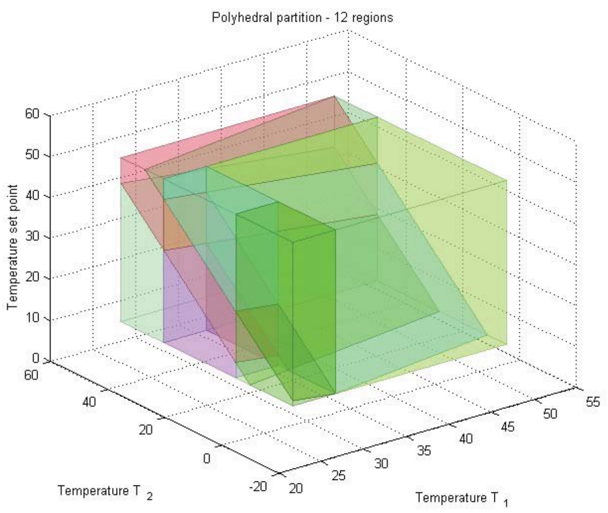

Figure 6: Equivalent explicit MPC controller

editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[5] P.J. Antsaklis. A brief introduction to the theory and applications of hybrid systems. *Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 88(7):879–886, July 2000.

[6] A. Balluchi, L. Benvenuti, M. Di Benedetto, C. Pinello, and A. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proc. IEEE*, 88(7):888–912, 2000.

[7] A.G. Beccuti, G. Papafotiou, R. Frasca, and M. Morari. Explicit hybrid model predictive control of the DC-DC boost converter. In *IEEE PESC*, Orlando, Florida, USA, June 2007.

[8] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Trans. Automatic Control*, 49(5):832–838, 2004.

[9] A. Bemporad. *Hybrid Toolbox – User's Guide*. January 2004. `http://www.dii.unisi.it/hybrid/toolbox`.

[10] A. Bemporad. Model-based predictive control design: New trends and tools. In *Proc. 45th IEEE Conf. on Decision and Control*, pages 6678–6683, San Diego, CA, 2006.

[11] A. Bemporad, P. Borodani, and M. Mannelli. Hybrid control of an automotive robotized gearbox for reduction of consumptions and emissions. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 81–96. Springer-Verlag, 2003.

[12] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proc. American Contr. Conf.*, pages 1190–1194, Chicago, IL, June 2000.

[13] A. Bemporad and S. Di Cairano. Optimal control of discrete hybrid stochastic automata. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, number 3414 in Lecture Notes in Computer Science, pages 151–167. Springer-Verlag, 2005.

[14] A. Bemporad, S. Di Cairano, and J. Júlvez. Event-based model predictive control and verification of integral continuous-time hybrid automata. In J.P. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, number 3927 in Lecture Notes in Computer Science, pages 93–107. Springer-Verlag, 2006.

[15] A. Bemporad, S. Di Cairano, I. V. Kolmanovsky, and D. Hrovat. Hybrid modeling and control of a multibody magnetic actuator for automotive applications. In *Proc. 46th IEEE Conf. on Decision and Control*, pages 5270–5275, New Orleans, LA, 2007.

[16] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Automatic Control*, 45(10):1864–1876, 2000.

[17] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A bounded-error approach to piecewise affine system identification. *IEEE Trans. Automatic Control*, 50(10):1567–1580, October 2005.

[18] A. Bemporad and N. Giorgetti. Logic-based methods for optimal control of hybrid systems. *IEEE Trans. Automatic Control*, 51(6):963–976, 2006.

[19] A. Bemporad and D. Mignone. *MIQP.M: A Matlab function for solving mixed integer quadratic programs*, 2000. `http://www.dii.unisi.it/~hybrid/tools/miqp`.

[20] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[21] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[22] A. Bemporad, M. Morari, and N.L. Ricker. *Model Predictive Control Toolbox for Matlab – User's Guide*. The Mathworks, Inc., 2004. `http://www.mathworks.com/access/helpdesk/help/toolbox/mpc/`.

[23] A. Bemporad, F.D. Torrisi, and M. Morari. Discrete-time hybrid modeling and verification of the batch

evaporator process benchmark. *European Journal of Control*, 7(4):382–399, July 2001.

[24] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, October 2005.

[25] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An MPC/hybrid system approach to traction control. *IEEE Trans. Contr. Systems Technology*, 14(3):541–552, May 2006.

[26] M.S. Branicky. *Studies in hybrid systems: modeling, analysis, and control*. PhD thesis, LIDS-TH 2304, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[27] M.S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Automatic Control*, 43(4):475–482, April 1998.

[28] S. Di Cairano and A. Bemporad. An equivalence result between linear hybrid automata and piecewise affine systems. In *Proc. 45th IEEE Conf. on Decision and Control*, pages 2631–2636, San Diego, CA, 2006.

[29] S. Di Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat. Model predictive control of magnetically actuated mass spring dampers for automotive applications. *Int. J. Control*, 80(11):1701–1716, 2007.

[30] S. Di Cairano, M. Lazar, A. Bemporad, and M. Heemels. A control Lyapunov approach to predictive control of hybrid systems. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computation and Control*, number 4981 in Lecture Notes in Computer Science, pages 130–143. Springer-Verlag, Berlin Heidelberg, 2008.

[31] S. Di Cairano, D. Yanakiev, A. Bemporad, I.V. Kolmanovsky, and D. Hrovat. An MPC design flow for automotive control and applications to idle speed regulation. In *Proc. 47th IEEE Conf. on Decision and Control*, pages 5686–5691, Cancun, Mexico, 2008.

[32] E.F. Camacho and C. Bordons. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer-Verlag, London, 2$^{\text{nd}}$ edition, 2004.

[33] V. Chandru and J.N. Hooker. *Optimization methods for logical inference*. Wiley-Interscience, 1999.

[34] Dash Associates. *XPRESS-MP User Guide*, 2004. `http://www.dashoptimization.com`.

[35] 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.

[36] 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.

[37] V.D. Dimitriadis, N. Shah, and C.C. Pantelides. Modeling and safety verification of discrete/continuous processing systems. *AIChE Journal*, 43:1041–1059, 1997.

[38] 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, February 2003.

[39] K. Fukuda. *cdd/cdd+ Reference Manual*. Institute for Operations Research, ETH-Zentrum, CH-8092 Zurich, Switzerland, 0.61 (cdd) 0.75 (cdd+) edition, December 1997. `http://www.ifor.math.ethz.ch/˜fukuda/cdd_home`.

[40] T. Geyer, F.D. Torrisi, and M. Morari. Efficient Mode Enumeration of Compositional Hybrid Models. In A. Pnueli and O. Maler, editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, 2003.

[41] N. Giorgetti, A. Bemporad, H. E. Tseng, and D. Hrovat. Hybrid model predictive control application towards optimal semi-active suspension. *Int. J. Control*, 79(5):521–533, 2006.

[42] N. Giorgetti, G. Ripaccioli, A. Bemporad, I.V. Kolmanovsky, and D. Hrovat. Hybrid model predictive control of direct injection stratified charge engines. *IEEE/ASME Transactions on Mechatronics*, 11(5):499–506, August 2006.

[43] J. Gu, P.W. Purdom, J. Franco, and B. Wah. Algorithms for the satisfiability (SAT) problem: A survey. In *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, number 35, pages 19–151. American Mathematical Society, 1997.

[44] 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.

[45] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000.

[46] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[47] J. P. Hespanha, S. Bohacek, K. Obraczka, and J. Lee. Hybrid modeling of TCP congestion control. In M.D. Di Benedetto and A. Sangiovanni Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 291–304. Springer-Verlag, 2001.

[48] J.N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.

[49] T. Huerlimann. *Reference Manual for the LPL Modeling Language, Version 4.42*. Departement for Informatics, Université de Fribourg, Switzerland, `http://www2-iiuf.unifr.ch/tcs/lpl/TonyHome.htm`, 2001.

[50] ILOG, Inc. *CPLEX 9.0 User Manual*. Gentilly Cedex, France, 2004.

[51] ILOG, Inc. *CPLEX 11.0 User Manual*. Gentilly Cedex, France, 2008.

[52] A. Ingimundarson, C. Ocampo-Martinez, and A. Bemporad. Model predictive control of hybrid systems based on mode-switching constraints. In *Proc. 46th IEEE Conf. on Decision and Control*, pages 5265–5269, New Orleans, LA, 2007.

[53] M. Johannson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.

[54] T. A. Johansen, J. Kalkkuhl, J. Lüdemann, and I. Petersen. Hybrid control strategies in ABS. In *Proc. American Contr. Conf.*, Arlington, Virginia, June 2001.

[55] K H Johansson, M Egerstedt, J Lygeros, and S Sastry. On the regularization of Zeno hybrid automata. *System & Control Letters*, 38:141–150, 1999.

[56] S. Karaman, R.G. Sanfelice, and E. Frazzoli. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *Proc. 47th IEEE Conf. on Decision and Control*, pages 2117–2122, Cancun, Mexico, 2008.

[57] M. Kvasnica, P. Grieder, and M. Baotić. *Multi Parametric Toolbox (MPT)*, 2006. `http://control.ee.ethz.ch/~mpt/`.

[58] M. Lazar. *Model predictive control of hybrid systems: stability and robustness*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.

[59] M. Lazar, M. Heemels, S. Weiland, and A. Bemporad. On the stability of 2-norm based model predictive control of constrained PWA systems. In *Proc. American Contr. Conf.*, pages 575–580, Portland, OR, 2005.

[60] M. Lazar, M. Heemels, S. Weiland, and A. Bemporad. Stabilizing model predictive control of hybrid systems. *IEEE Trans. Automatic Control*, 51(11):1813–1818, 2006.

[61] M. Lazar, M. Heemels, S. Weiland, A. Bemporad, and O. Pastravanu. Infinity norms as Lyapunov functions for model predictive control of constrained PWA systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, number 3414 in Lecture Notes in Computer Science, pages 417–432. Springer-Verlag, 2005.

[62] J. Lygeros, D.N. Godbole, and S. Sastry. A game theoretic approach to hybrid system design. In R. Alur and T. Henzinger, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1996.

[63] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

[64] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, UK, 2002.

[65] A. Makhorin. *GLPK (GNU Linear Programming Kit) User's Guide*, 2004.

[66] K. Marriot and P.J. Stuckey. *Programming with constraints: an introduction*. MIT Press, 1998.

[67] D.Q. Mayne and S. Rakovic. Optimal control of constrained piecewise affine discrete time systems using reverse transformation. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 1546–1551, Las Vegas, Nevada, USA, December 2002.

[68] D. Mignone. *Control and Estimation of Hybrid Systems with Mathematical Optimization*. PhD thesis, Automatic Control Labotatory - ETH, Zurich, 2002.

[69] D. Mignone, A. Bemporad, and M. Morari. A framework for control, fault detection, state estimation and verification of hybrid systems. *Proc. American Contr. Conf.*, pages 134–138, June 1999.

[70] G. Mitra, C. Lucas, and Moody S. Tool for reformulating logical forms into zero-one mixed integer programs. *European Journal of Operational Research*, 71:262–276, 1994.

[71] R. Möbus, M. Baotić, and M. Morari. Multi-objective adaptive cruise control. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 359–374. Springer-Verlag, 2003.

[72] M. Morari and J.H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4–5):667–682, 1999.

[73] R. Raman and I.E. Grossmann. Relation between MILP modeling and logical inference for chemical process synthesis. *Computers & Chemical Engineering*, 15(2):73–84, 1991.

[74] J.B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, pages 38–52, June 2000.

[75] G. Ripaccioli, A. Bemporad, F. Assadian, C. Dextreit, S. Di Cairano, and I.V. Kolmanovsky. Hybrid modeling, identification, and predictive control: an application to hybrid electric vehicle energy management. In R. Majumdar and P. Tabuada, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg, 2009. To appear.

[76] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

[77] C. Seatzu, D. Corona, A. Giua, and A. Bemporad. Optimal control of continuous-time switched affine systems. *IEEE Trans. Automatic Control*, 51(5):726–741, 2006.

[78] E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automatic Control*, 26(2):346–358, April 1981.

[79] E.D. Sontag. Interconnected automata and linear systems: A theoretical framework in discrete-time. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III - Verification and Control*, number 1066 in Lecture Notes in Computer Science, pages 436–448. Springer-Verlag, 1996.

[80] P. Tøndel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489–497, March 2003.

[81] P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39(5):945–950, May 2003.

[82] F. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and Mignone D. Hysdel 2.0.5 - user manual. Technical Report AUT02-28, Automatic Control Laboratory, ETH Zurich, 2002.

[83] F.D. Torrisi and A. Bemporad. HYSDEL — A tool for generating computational hybrid models. *IEEE Trans. Contr. Systems Technology*, 12(2):235–249, March 2004.

[84] A.J. van der Schaft and J.M. Schumacher. Complementarity modelling of hybrid systems. *IEEE Trans. Automatic Control*, 43:483–490, 1998.

[85] H.P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Third Edition, 1993.

[86] H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Trans. Automatic Control*, 11(2):161–167, 1966.

[87] X. Xu and P.J. Antsaklis. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Trans. Automatic Control*, 49(1):2–16, 2004.