Constrained Optimal Control of Discrete-Time Linear Hybrid Systems

Francesco Borrelli†, Mato Baotic†, Alberto Bemporad‡, Manfred Morari†

†Automatic Control Laboratory, ETH Zentrum, ETL K12, CH-8092 Zurich, Switzerland
borrelli,baotic,morari@aut.ee.ethz.ch
‡Dip. Ingegneria dell’Informazione, University of Siena, 53100 Siena, Italy,bemporad@dii.unisi.it

Abstract

In this paper we study the solution to optimal control problems for constrained discrete-time linear hybrid systems based on quadratic or linear performance criteria. The aim of the paper is twofold. First we give basic theoretical results on the structure of the optimal state feedback solution and of the value function. Second we describe how the state feedback optimal control law can be efficiently constructed by combining multiparametric programming and dynamic programming.

1 Introduction

Recent technological innovations have caused a considerable interest in the study of dynamical processes of a mixed continuous and discrete nature, denoted as hybrid systems. In their most general form hybrid systems are characterized by the interaction of continuous-time models (governed by differential or difference equations), and of logic rules and discrete event systems (described, for example, by temporal logic, finite state machines, if-then-else rules) and discrete components (on/off switches or valves, gears or speed selectors, etc.). Such systems can switch between many operating modes where each mode is governed by its own characteristic dynamical laws. Mode transitions are triggered by variables crossing specific thresholds (state events), by the lapse of certain time periods (time events), or by external inputs (input events) [2]. A detailed discussion of different modelling frameworks for hybrid systems that appeared in the literature goes beyond the scope of this paper, the main concepts can be found in [2, 16, 15, 9, 37, 38].

Different methods for the analysis and design of controllers for hybrid systems have emerged over the last few years [48, 50, 17, 38, 9]. Among them, the class of optimal controllers is one of the most studied. The publications differ greatly in the hybrid models adopted, in the formulation of the optimal control problem and in the approach used to solve it.
The first main difference concerns the use of discrete-time or continuous-time hybrid models. Most of the literature deals with the optimal control of continuous time hybrid systems and has focuses on two issues: the study of necessary conditions for a trajectory to be optimal [49, 44] and the computation of optimal or sub-optimal control laws by means of Hamilton-Jacobi-Bellmann (HJB) equations or the Maximum Principle [29, 30, 45, 18, 55]. In order to compute the optimal control law all approaches resort to a discretization of the state and input space either by gridding or by using a finite dimensional set of basis functions such as cubic Hermite polynomials [18]. In both cases two computational issues arise from the hybrid nature of the problem. Firstly, the HJB equations are discontinuous and their solution can be discontinuous even if the cost function and the boundary conditions of the optimal control problem are differentiable. This causes a variety of numerical problems that can be avoided by using special numerical techniques which employ an accurate choice of the grid, level set methods or differential inclusion [50]. The second issue concerns complexity. The well known “curse of dimensionality” arising from the discretization becomes a “double curse” in the hybrid framework: for each grid point a continuous and a discrete optimization is involved. The continuous optimization involves the continuous inputs of the hybrid system, the discrete one deals with discrete inputs and with all the possible internal and exogenous events that can occur, the number of which can be infinite even with finite horizon problems. Often, the explosion of the complexity is mitigated by imposing a bound on the number of possible events that can occur in a certain time interval and by using dynamic programming [30, 56] or branch and bound techniques which can avoid the enumeration of all possible events sequences of the hybrid system. Also, the complexity issues involved with the discretization has been studied by several researchers who have proposed techniques to reduce the computational burden for computing optimal control laws for special classes of hybrid systems. In [29, 20] the authors use a hierarchical decomposition approach to break down the overall problem into smaller ones. In doing so, discretization is not involved and the main computational complexity arises from a higher-level nonlinear programming problem. A similar approach has been used by Xu and Antsaklis in [57].

In this paper we focus on discrete-time linear hybrid models. In our hybrid modeling framework we allow (i) the system to be discontinuous, (ii) both states and inputs to assume continuous and logic values, (iii) events to be both internal, i.e., caused by the state reaching a particular boundary, and exogenous, i.e., one can decide when to switch to some other operating mode and (iv) states and inputs to fulfill piecewise-linear constraints. We will focus on discrete-time piecewise affine (PWA) models. Discrete-time PWA models can describe a large number of processes, such as discrete-time linear systems with static piecewise-linearities, discrete-time linear systems with logic states and inputs or switching systems where the dynamic behaviour is described by a finite number of discrete-time linear models, together with a set of logic rules for switching among these models. Moreover, PWA systems can approximate nonlinear discrete-time dynamics via multiple linearizations at different operating points.

In discrete-time hybrid systems an event can occur only at instants that are multiples of the sampling time and many interesting mathematical phenomena occurring in continuous-time hybrid systems such as Zeno behaviors [35] do not exist. However, the solution to optimal control problems is still complex: the solution to the HJB equation can be discontinuous [36] and the number of possible switches grows exponential with the length of the horizon of the optimal control problem. Nevertheless, we will show that for the class of linear discrete-time hybrid systems we can characterize and compute the optimal
control law without gridding the state space.

The solution to optimal control problems for discrete-time hybrid systems was first outlined by Sontag in [48]. In his plenary presentation [39] at the 2001 European Control Conference, Mayne presented an intuitively appealing characterization of the state feedback solution to optimal control problems for linear hybrid systems with performance criteria based on quadratic and linear norms. The detailed exposition presented in the initial part of this paper follows a similar line of argumentation and shows that the state feedback solution to the finite time optimal control problem is a time-varying piecewise affine feedback control law, possibly defined over non-convex regions. Moreover, we give insight into the structure of the optimal state feedback solution and the value function.

In the second part of the paper we describe how the optimal control law can be efficiently computed by means of multiparametric programming. In particular, we propose a novel algorithm that solves the Hamilton-Jacobi-Bellman equation by using a simple multiparametric solver.

In collaboration with different companies and institutes, the results described in this paper have been applied to a wide range of problems ([26, 41, 9, 52, 5, 8, 5, 13, 4, 42]). Such applications and their complexity are briefly discussed in Section 11. In this paper we will focus on simple examples that will highlight the main features of the hybrid system approach.

Before formulating optimal control problems for hybrid systems we will give a short overview on multiparametric programming and on discrete-time linear hybrid systems.

## 2 Definitions

We give the following definitions:

**Definition 1** A collection of sets $\mathcal{R}_1, \ldots, \mathcal{R}_N$ is a partition of a set $\Theta$ if (i) $\bigcup_{i=1}^N \mathcal{R}_i = \Theta$, (ii) $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, $\forall i \neq j$. Moreover $\mathcal{R}_1, \ldots, \mathcal{R}_N$ is a polyhedral partition of a polyhedral set $\Theta$ if $\mathcal{R}_1, \ldots, \mathcal{R}_N$ is a partition of $\Theta$ and the $\mathcal{R}_i$’s are polyhedral sets, where $\overline{\mathcal{R}_i}$ denotes the closure of the set $\mathcal{R}_i$.

**Definition 2** A function $h : \Theta \to \mathbb{R}^k$, where $\Theta \subseteq \mathbb{R}^s$, is piecewise affine (PWA) if there exists a partition $\mathcal{R}_1, \ldots, \mathcal{R}_N$ of $\Theta$ and $h(\theta) = H^i \theta + k^i$, $\forall \theta \in \mathcal{R}_i$, $i = 1, \ldots, N$.

**Definition 3** A function $h : \Theta \to \mathbb{R}^k$, where $\Theta \subseteq \mathbb{R}^s$, is PWA on polyhedra (PPWA) if there exists a polyhedral partition $\mathcal{R}_1, \ldots, \mathcal{R}_N$ of $\Theta$ and $h(\theta) = H^i \theta + k^i$, $\forall \theta \in \mathcal{R}_i$, $i = 1, \ldots, N$.

Piecewise quadratic functions (PWQ) and piecewise quadratic functions on polyhedra (PPWQ) are defined analogously.

**Definition 4** A function $q : \Theta \to \mathbb{R}$, where $\Theta \subseteq \mathbb{R}^s$, is a multiple quadratic function of multiplicity $d \in \mathbb{N}^+$ if $q(\theta) = \min \{ q^1(\theta) \triangleq \theta^T Q^1 \theta + l^1 \theta + c^1, \ldots, q^d(\theta) \triangleq \theta^T Q^d \theta + l^d \theta + c^d \}$ and $\Theta$ is a convex polyhedron.

**Definition 5** A function $q : \Theta \to \mathbb{R}$, where $\Theta \subseteq \mathbb{R}^s$, is a multiple PWQ on polyhedra (multiple PPWQ) if there exists a polyhedral partition $\mathcal{R}_1, \ldots, \mathcal{R}_N$ of $\Theta$ and $q(\theta) = \min \{ q^1_1(\theta) \triangleq \theta^T Q^1_1 \theta + l^1_1 \theta + c^1_1, \ldots, q^d_i(\theta) \triangleq \theta^T Q^d_i \theta + l^d_i \theta + c^d_i \}, \forall \theta \in \mathcal{R}_i$, $i = 1, \ldots, N$. We define $d_i$ to be the multiplicity of the function $q$ in the polyhedron $\mathcal{R}_i$ and $d = \sum_{i=1}^N d_i$ to be the multiplicity of the function $q$. (Note that $\Theta$ is not necessarily convex.)
3 Basics on Multiparametric programming

Consider the nonlinear mathematical program dependent on a parameter $x$ appearing in the cost function and the constraints

$$J^*(x) = \inf_z f(z, x)$$
subj. to $g(z, x) \leq 0$
$z \in M$

where $z \in \mathbb{R}^s$ is the optimization vector, $x \in \mathbb{R}^n$ is the parameter, $f : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}$ is the cost function, $g : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}^n$ are the constraints and $M \subseteq \mathbb{R}^s$.

A small perturbation of the parameter $x$ in the mathematical program (1) can cause a variety of results, i.e., depending on the properties of the functions $f$ and $g$ the solution $z^*(x)$ of the mathematical program may vary smoothly or change abruptly as a function of $x$. We denote by $K^*$ the set of feasible parameters, i.e.,

$$K^* = \{ x \in \mathbb{R}^n \mid \exists z \in M, g(z, x) \leq 0 \},$$

by $R : \mathbb{R}^n \to 2^{\mathbb{R}^s}$ the point-to-set map that assigns to a parameter $x$ the set of feasible $z$, i.e.

$$R(x) = \{ z \in M \mid g(z, x) \leq 0 \},$$

by $J^* : K^* \to \mathbb{R} \cup \{-\infty\}$ the real-valued function, which expresses the dependence on $x$ of the minimum value of the objective function over $K^*$, i.e.

$$J^*(x) = \inf z \{ f(z, x) \mid x \in K^*, z \in R(x) \},$$

and by $Z^* : K^* \to 2^{\mathbb{R}^s}$ the point-to-set map which expresses the dependence on $x$ of the set of optimizers, i.e., $Z^*(\bar{x})$ is the set of all infimizers of (1) for $x = \bar{x}, \bar{x} \in K^*$.

$J^*(x)$ will be referred to as optimal value function or simply value function, $Z^*(x)$ will be referred to as optimal set. We will denote by $z^* : \mathbb{R}^n \to \mathbb{R}$ one of the possible single valued functions that can be extracted from $Z^*$. $z^*(x)$ will be called optimizer function. If $Z^*(x)$ is a singleton for all $x$, then $z^*(x)$ is the only element of $Z^*(x)$. In general $z^*$ may have infinite components, however throughout the whole paper $z^*$ will be a real-valued function.

Our interest in problem (1) will become clear in the next sections. We can anticipate here that optimal control problems for nonlinear systems can be reformulated as the mathematical program (1) where $z$ is the input sequence to be optimized and $x$ the initial state of the system. Therefore, the study of the properties of $J^*$ and $Z^*$ is fundamental for the study of properties of state feedback optimal controllers.

Fiacco ([27, Chapter 2]) provides conditions under which the solution of nonlinear multiparametric programs (1) is locally well behaved and establishes properties of the solution as a function of the parameters. In the following we summarize some of the results of [46, 22, 33] by focusing our attention on a restricted set of functions $f(z, x)$ and $g(z, x)$.

**Theorem 1 ([33])** Consider the multiparametric nonlinear program (1). Assume that $M$ is a convex and bounded set in $\mathbb{R}^s$, $f$ is continuous and the components of $g$ are convex on $M \times \mathbb{R}^n$. Then, $J^*(x)$ is continuous for each $x$ belonging to $K^*$. 

4
Theorem 2 ([27]) Consider the multiparametric nonlinear program (1). If in addition to the assumptions of Theorem 1 we assume that \( f \) is strictly quasi-convex in \( z \) for each fixed \( x \), then \( z^* \) is a continuous real-valued function.

Unfortunately very little can be said without convexity assumptions on \( f \) and \( g \). Below we restrict our attention to two special classes of multiparametric programming.

**Multiparametric Linear Program**

Consider the multi-parametric program

\[
J^*(x) = \min_z \ g'z \\
\text{s.t.} \quad Cz \leq c + Sx,
\]

where \( z \in \mathbb{R}^{n_z} \) is the optimization vector, \( x \in \mathbb{R}^n \) is the vector of parameters, and \( g \in \mathbb{R}^{n_z}, C \in \mathbb{R}^{q \times n_z}, c \in \mathbb{R}^q, S \in \mathbb{R}^{q \times n} \) are constant matrices. We refer to (5) as a (right-hand-side) multi-parametric linear program (mp-LP) [28, 14].

**Theorem 3 ([28])** Consider the mp-LP (5). The set \( K^* \) is a polyhedral set, the value function \( J^*: K^* \rightarrow \mathbb{R} \) is PPWA, convex and continuous and there exists a continuous and PPWA optimizer function \( z^*: K^* \rightarrow \mathbb{R}^{n_z} \).

**Proof:** See [28]. \( \square \)

When in (5) we add constraints that restrict some of the optimization variables to be 0 or 1, \( z \triangleq [z_c, z_\ell] \), where \( z_c \in \mathbb{R}^{n_c}, z_\ell \in \{0,1\}^{n_\ell} \), we refer to (5) as a (right-hand-side) multi-parametric mixed-integer linear program (mp-MILP) [25].

**Theorem 4** Consider the mp-MILP (5). The set \( K^* \) is a (possibly non-convex and disconnected) polyhedral set \(^1\), the value function \( J^*: K^* \rightarrow \mathbb{R} \) is PPWA and there exist PPWA optimizer functions \( z^*_c: K^* \rightarrow \mathbb{R}^{n_c}, z^*_\ell: K^* \rightarrow \{0,1\}^{n_\ell} \).

**Proof:** Easily follows from the Algorithm described in [25]. \( \square \)

**Multiparametric Quadratic Program**

Consider the multi-parametric program

\[
V(x) = \frac{1}{2}x'Yx + \min_z \frac{1}{2}z'Hz + z'Fx \\
\text{subj. to} \quad Cz \leq c + Sx
\]

where \( z \in \mathbb{R}^{n_z} \) is the optimization vector, \( x \in \mathbb{R}^n \) is the vector of parameters, and \( g \in \mathbb{R}^{n_z}, C \in \mathbb{R}^{q \times n_z}, c \in \mathbb{R}^q, S \in \mathbb{R}^{q \times n} \) are constant matrices. We refer to (6) as a (right-hand-side) multi-parametric quadratic program (mp-QP).

\(^1\)We define here a non-convex polyhedral set as a non-convex set given by the union of a finite number of convex polyhedra with mutually disjoint interiors.
Theorem 5 ([10]) Consider the mp-QP (6). Assume $H \succ 0$ and $[\bar{Y} \bar{F}'] \succeq 0$. The set $K^*$ is a polyhedral set, the value function $J^*: K^* \to \mathbb{R}$ is PPWQ, convex and continuous and the optimizer $z^*: K^* \to \mathbb{R}^{n_z}$ is PPWA and continuous.

When in (6) we add constraints that restrict some of the optimization variables to be 0 or 1, $z \triangleq [z_c, z_{\ell}]$, where $z_c \in \mathbb{R}^{n_c}$, $z_{\ell} \in \{0, 1\}^{n_{\ell}}$, we refer to a (right-hand-side) multi-parametric mixed-integer quadratic program (mp-MIQP)

Theorem 6 ([11]) Consider the mp-MIQP (6). The set $K^*$ is a (possibly non-convex and disconnected) polyhedral set, the value function $J^*: K^* \to \mathbb{R}$ is PWQ and there exist PWA optimizer functions $z^*_c: K^* \to \mathbb{R}^{n_c}$, $z^*_\ell: K^* \to \{0, 1\}^{n_{\ell}}$

4 Hybrid Systems

Several modeling frameworks have been introduced for discrete-time hybrid systems. Among them, piecewise affine (PWA) systems [48] are defined by partitioning the state space into polyhedral regions and associating with each region a different linear state-update equation

$$
\begin{align*}
x(t+1) &= A_i x(t) + B_i u(t) + f_i \\
& \text{if } [x(t) u(t)] \in \mathcal{P}_i, \ i = \{1, \ldots, s\},
\end{align*}
$$

(7)

where $x \in \mathbb{R}^{n_c} \times \{0, 1\}^{n_{\ell}}$, $u \in \mathbb{R}^{m_c} \times \{0, 1\}^{m_{\ell}}$, $\{\mathcal{P}_i\}_{i=1}^s$ is a polyhedral partition of the set of the state+input space $\mathcal{P} \subset \mathbb{R}^{n+m}$, $n \triangleq n_c + n_{\ell}$, $m \triangleq m_c + m_{\ell}$. We assume that $\mathcal{P}$ is closed and bounded and we denote by $x_c \in \mathbb{R}^{n_c}$ and $u_c \in \mathbb{R}^{m_c}$ the real components of the state and input vector, respectively. We will give the following definitions of continuous PWA system.

**Definition 6** We say that the PWA system (7) is continuous if the mapping $(x_c(t), u_c(t)) \mapsto x_c(t+1)$ is continuous and $n_{\ell} = m_{\ell} = 0$.

**Definition 7** We say that a PWA system (7) is continuous in the input space if the mapping $(x_c(t), u_c(t)) \mapsto x_c(t+1)$ is continuous in the $u_c$ space and $n_{\ell} = m_{\ell} = 0$.

**Definition 8** We say that a PWA system (7) is continuous in the real input space if the mapping $(x_c(t), u_c(t)) \mapsto x_c(t+1)$ is continuous in the $u_c$ space.

Our main motivation for focusing on discrete-time models stems from the need to analyze these systems and to solve optimization problems, such as optimal control or scheduling problems, for which the continuous time counterpart would not be easily computable. For this reason we need a way to describe discrete-time PWA systems in a form that is suitable for recasting analysis/synthesis problems into a more compact optimization problem. The MLD framework [9] described below has been developed for such a purpose. In particular, MLD models can be used to recast hybrid dynamical optimization problems into mixed-integer linear and quadratic programs, solvable via branch and bound techniques [43].

MLD systems [9] allow 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. 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 [21, 9, 53, 54, 43]. The MLD modeling framework relies on the idea of translating logic relations into mixed-integer linear inequalities [9, 54]. Alternative methods for translating any logical relation between Boolean variables into a set of linear integer inequalities can be found in [41, Chapter 2]. In [41] the reader can also find a more comprehensive and complete treatment of the topic.

The state update law of finite state machines can be described by logic propositions involving binary states, their time updates, and binary signals, under the assumptions that the transitions are clocked and synchronous with the sampling time of the continuous dynamical equations, and that the automaton is well-posed (i.e., at each time step a transition exists and is unique):

\[ x_\ell(t + 1) = F(x_\ell(t), u_\ell(t)), \]  

where \( u_\ell \) is the vector of Boolean signals triggering the transitions of the automaton. Therefore, the automaton is equivalent to a nonlinear discrete-time system where \( F \) is a purely Boolean function.

The translation technique mentioned above can be applied directly to translate the automaton (8) into a set of linear integer equalities and inequalities.

By collecting the equalities and inequalities derived from the representation of the hybrid system we obtain the Mixed Logical Dynamical (MLD) system [9]

\[ \begin{align*}
    x(t + 1) &= Ax(t) + B_1 u(t) + B_2 \delta(t) + B_3 z(t) \\
    y(t) &= Cx(t) + D_1 u(t) + D_2 \delta(t) + D_3 z(t) \\
    E_2 \delta(t) + E_3 z(t) &\leq E_1 u(t) + E_4 x(t) + E_5,
\end{align*} \]  

where \( x \in \mathbb{R}^{n_x} \times \{0,1\}^{n_\ell} \) is a vector of continuous and binary states, \( u \in \mathbb{R}^{m_u} \times \{0,1\}^{m_\ell} \) are the inputs, \( y \in \mathbb{R}^{p_y} \times \{0,1\}^{p_\ell} \) the outputs, \( \delta \in \{0,1\}^{r_\ell}, z \in \mathbb{R}^{r_c} \) represent auxiliary binary and continuous variables, respectively, which are introduced when transforming logic relations into mixed-integer linear inequalities, and \( A, B_1-3, C, D_1-3, E_1-5 \) are matrices of suitable dimensions.

**Remark 1** In (9c), we only allowed nonstrict inequalities, as we are interested in using numerical solvers with MLD models. Therefore, strict inequalities of the form \( a'x > b \) must be approximated by \( a'x \geq b + \epsilon \) for some \( \epsilon > 0 \) (e.g., the machine precision), with the assumption that \( 0 < a'x - b < \epsilon \) cannot occur due to the finite precision of the computer. After such approximation some regions of the state space will be considered infeasible for the MLD model (9) even if they were feasible for the original PWA model (7). However, the measure of this regions tends to zero as \( \epsilon \) goes to zero. Note that this issue arises only in the case of discontinuous PWA systems.

We assume that system (9) is completely well-posed [9], which means that for all \( x, u \) within a bounded set the variables \( \delta, z \) are uniquely determined by the inequalities (9c). Then it follows that \( x(t+1) \) and \( y(t) \) are uniquely defined once \( x(t), u(t) \) are given, and therefore that \( x \)- and \( y \)-trajectories exist and are uniquely determined by the initial state \( x(0) \) and input signal \( u(t) \).

Note that the constraints (9c) allow us to specify additional linear constraints on continuous variables (e.g., constraints on physical variables of the system), and logic constraints. The ability to include constraints, constraint prioritization, and heuristics adds to the expressiveness and generality of the MLD framework.
Remark 2 MLD and PWA system classes are equivalent only if the original PWA system is defined over a bounded set in the state-input space. In general an unconstrained PWA system cannot be translated into an equivalent MLD system. This issue has no practical importance.

The translation of a description of a hybrid dynamical system into mixed integer inequalities is the objective of the tool HYSDEL (HYbrid Systems DEscription Language), which automatically generates an MLD model from a high-level textual description of the system [52]. Given a textual description of the logic and dynamical parts of the hybrid system, HYSDEL returns the matrices $A, B_{1-3}, C, D_{1-3}, E_{1-5}$ of the corresponding MLD form (9). A full description of HYSDEL is given in [52]. The compiler is available at http://control.ethz.ch/~hybrid/hysdel.

In [32] the authors prove the equivalence of MLD systems and PWA systems to other hybrid systems classes such as linear complementarity (LC) systems [31] and max-min-plus-scaling (MMPS) systems [23].

5 Examples

In this section we describe three examples of hybrid systems and show their PWA and MLD description. Optimal control problems for these systems will be solved later in the paper. The examples shown here are extremely simple and they are introduced to highlight the main features of the hybrid system approach. References to more complex hybrid systems are reported in Section 11. Example 5.1 is a PWA system where the dynamical matrix is a piecewise constant function of the state. Example 5.2 shows a mechanical system with a binary input and a piecewise-linear model of the friction coefficient. Example 5.3 is a one dimensional system with a logic state.

Example 5.1

Simple System

The following switching system

$$
\begin{align*}
x(t+1) &= 0.8 \begin{bmatrix} \cos \alpha(t) & -\sin \alpha(t) \\ \sin \alpha(t) & \cos \alpha(t) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\
\alpha(t) &= \begin{cases} \frac{\pi}{3} & \text{if } [1,0]^T x(t) \geq 0 \\ -\frac{\pi}{3} & \text{if } [1,0]^T x(t) < 0 \end{cases} \\
x(t) &\in [-10,10] \times [-10,10] \\
u(t) &\in [-1,1]
\end{align*}
$$

was first proposed in [9]. System (10) can be rewritten in form (9) as shown in [9].
where $B = [0 \ 1]'$, $A_1, A_2$ are obtained from (10) by setting respectively $\alpha = \frac{\pi}{3}, -\frac{\pi}{3}$, $M = 4(1 + \sqrt{3})[1 \ 1]' + B$, $N \triangleq 5[1 \ 1]'$, and $\epsilon$ is a properly chosen small positive scalar.

Example 5.2

Spring-Mass System

Consider the spring-mass system depicted in Figure 1. We assume here that the spring coefficient is a nonlinear function of the position as depicted in Figure 2. Moreover, we assume that the friction coefficient can be instantaneously switched from one value $b_1$ to another different value $b_2$ by using a logic input $u_2$.

The system dynamics can be described as:

$$M \ddot{x}_2 = u_1 - k(x_1)x_1 - b(u_2)x_2$$

where $x_1$ and $x_2$ denote the position and the speed of the mass, respectively, $u_1$ a continuous force input, and $u_2$ logic input switching the airflow. The spring coefficient is

$$k(x_1) = \begin{cases} 
  k_1x_1 + d_1 & \text{if } x_1 \leq x_m \\
  k_2x_1 + d_2 & \text{if } x_1 > x_m,
\end{cases}$$

and the friction coefficient is

$$b(u_2) = \begin{cases} 
  b_2 & \text{if } u_2 = 1 \\
  b_1 & \text{if } u_2 = 0.
\end{cases}$$

We also assume that the states are constrained to assume value between $-10$ and $+10$.\(^2\)

Two hybrid features of the example system can be also observed in real applications. For instance, the throttle valve that regulates air inflow to the engine of the car is equipped with two counteracting springs [4] and the resulting spring torque can be described with the nonlinearity depicted in Figure 2. The manipulated friction coefficient can be seen as the simplification of a bench whose kinematic friction coefficient is reduced by letting air flow from beneath (see Figure 1.b). If we neglect static friction and assume that the air flow is stopped when $u_2 = 1$, we obtain above the expression for the friction coefficient $b(u)$ above.

The system has four modes, depending on the binary input and the region of linearity. Assuming that normalized values of the system parameters are $M = 1$, $b_1 = 1$, $b_2 = 50$, $k_1 = 1$, $k_2 = 3$, $d_1 = 1$, $d_2 = 7.5$, $x_m = 1$, after sampling the dynamics in each mode with a sampling time of 0.1 seconds we obtain the following discrete-time PWA system

---

\(^2\)In the example we assume that all parameters and system variables are normalized to some nominal values. Hence, all values are expressed as dimensionless numbers.
A logic state \(x\) certain lower bound \(x\) holds for certain time intervals.

The MLD model obtained by compiling the HYSDEL list reported in the Appendix is:

\[
x(t + 1) = \begin{bmatrix}
0.9988 & 0.0987 \\
-0.0247 & 0.97410
\end{bmatrix} x(t) + \begin{bmatrix}
0.0012 \\
0.0247
\end{bmatrix} u_1(t) \quad \text{if}\ [1, 0] x(t) \leq 1 \text{ and } u_2(t) \leq 0.5
\]

\[
x(t + 1) = \begin{bmatrix}
0.9991 & 0.0571 \\
-0.0143 & 0.2859
\end{bmatrix} x(t) + \begin{bmatrix}
0.0009 \\
0.0143
\end{bmatrix} u_1(t) \quad \text{if}\ [1, 0] x(t) \leq 1 \text{ and } u_2(t) > 0.5
\]

\[
x(t + 1) = \begin{bmatrix}
0.9963 & 0.0986 \\
-0.0740 & 0.9716
\end{bmatrix} x(t) + \begin{bmatrix}
0.0012 \\
0.0247
\end{bmatrix} u_1(t) \quad \text{if}\ [1, 0] x(t) > 1 \text{ and } u_2(t) \leq 0.5
\]

\[
x(t + 1) = \begin{bmatrix}
0.9974 & 0.0570 \\
-0.0428 & 0.2848
\end{bmatrix} x(t) + \begin{bmatrix}
0.0009 \\
0.0143
\end{bmatrix} u_1(t) \quad \text{if}\ [1, 0] x(t) > 1 \text{ and } u_2(t) > 0.5
\]

\(x(t) \in [-10, 10] \times [-10, 10]\)

The MLD model obtained by compiling the HYSDEL list reported in the Appendix is:

\[
x(t + 1) = \begin{bmatrix}
0 & 1 \\
0 & 1
\end{bmatrix} x(t) + \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} u(t) + \begin{bmatrix}
-0.25 & 0.00 \\
0.25 & 0.00
\end{bmatrix} x(t) \quad \text{if}\ [1, 0] x(t) \leq 1 \text{ and } u_2(t) \leq 0.5
\]

\[
x(t + 1) = \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix} x(t) + \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix} u(t) + \begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix} x(t) \quad \text{if}\ [1, 0] x(t) > 1 \text{ and } u_2(t) > 0.5
\]

\(x(t) \in [-10, 10] \times [-10, 10]\)

Figure 3 shows the open-loop simulation of the system for a constant continuous input \(u_1 = 2\).

**Example 5.3**

**Simple System with a Logic State**

Consider the following SISO system:

\[
x_1(t + 1) = ax_1(t) + bu(t).
\]

A logic state \(x_2 \in [0, 1]\) stores the information whether the state of system (12) has ever gone below a certain lower bound \(x_{lb}\) or not:

\[
x_2(t + 1) = x_2(t) \bigvee [x_1(t) \leq x_{lb}],
\]

\(x_1(t + 1) = x_1(t) + u(t)\).
Figure 1: Example 5.2: Spring mass system

(a) System with high friction coefficient

(b) System with low friction coefficient

Figure 2: Nonlinear spring coefficient

Figure 3: Open-loop simulation of system (11)
Assume that the input coefficient is a function of the logic state:

\[ b = \begin{cases} 
  b_1 & \text{if } x_2 = 0 \\
  b_2 & \text{if } x_2 = 1.
\end{cases} \]  

The system can be described as the PWA system:

\[
x(t + 1) = \begin{cases} 
  \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} b_2 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \leq x_{lb} \\
  \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} b_1 \\ 0 \end{bmatrix} u(t) & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) > \begin{bmatrix} x_{lb} \\ -0.5 \end{bmatrix} \\
  \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} b_2 \\ 0 \end{bmatrix} u(t) & \text{if } x(t) > \begin{bmatrix} x_{lb} \\ 0.5 \end{bmatrix}
\end{cases}
\]

\[ x(t) \in [-10, 10] \times [-10, 10] \]

The MLD model obtained by compiling the HYSDEL list reported in the Appendix is:

\[
x(t + 1) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} (x(t) + \delta(t)) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} x(t) \leq \begin{bmatrix} 0.0 \\ 0.0 \\ -1.00 \\ 1.00 \\ 0.00 \\ -1.00 \\ 0.00 \\ 1.00 \\ 0.00 \\ 0.00 \\ 1.00 \end{bmatrix} x(t) + \begin{bmatrix} 1.00 \\ 0.00 \\ -1.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} u(t) + \begin{bmatrix} 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix} x(t) + \begin{bmatrix} 20.00 \\ 0.00 \\ 20.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \\ 0.00 \end{bmatrix}
\]

Figures 4 and 5 show two open-loop simulations of the system, one for each value of the logic state. If the continuous state \( x_1 \) goes below \( x_{lb} = -1 \) then the input has a stronger effect on the state (Figure 4). Note that the steady state of \( x_1 \) is a function of the logic state \( x_2 \).

### 6 Problem Formulation

Consider the PWA system (7) subject to hard input and state constraints

\[ Ex(t) + Lu(t) \leq M \]

for \( t \geq 0 \), and denote by constrained PWA system (CPWA) the restriction of the PWA system (7) over the set of states and inputs defined by (16),

\[
x(t + 1) = A^i x(t) + B^i u(t) + f^i \quad \text{if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \tilde{P}^i,
\]

where \( \{\tilde{P}^i\}_{i=0}^{s-1} \) is the new polyhedral partition of the sets of state+input space \( \mathbb{R}^{n+m} \) obtained by intersecting the sets \( P^i \) in (7) with the polyhedron described by (16). The union of the polyhedral partition \( \tilde{P} \triangleq \cup_{i=1}^{s-1} \tilde{P}^i \) will implicitly define the feasible region \( R_{feas} \) in the input space as a function of \( x \):

\[ R_{feas}(x) = \{ u \in \mathbb{R}^m \times \{0, 1\}^m \mid (x, u) \in \tilde{P} \}. \]
Figure 4: Open-loop simulation of system (15)

Figure 5: Open-loop simulation of system (15)
We define the following cost function

\[
J(U_N, x(0)) \triangleq \|Px_N\|_p + \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p,
\]

and consider the constrained finite-time optimal control (CFTOC) problem

\[
J^*(x(0)) \triangleq \inf_{U_N} J(U_N, x(0))
\]

subject to

\[
\begin{align*}
x_{k+1} &= A^i x_k + B^i u_k + f^i \text{ if } [\hat{z}_k] \in \hat{\mathcal{P}}^i, \ i = 1, \ldots, s \\
x_N &\in \mathcal{X}_f \\
x_0 &= x(0)
\end{align*}
\]

where the column vector \(U_N \triangleq [u_0', \ldots, u_{N-1}'] \in \mathbb{R}^{mcN \times \{0, 1\}^mN}\), is the optimization vector, \(N\) is the time horizon and \(\mathcal{X}_f\) is the terminal region. In (18), \(\|Qx\|_p\) denotes the \(p\)-norm of the vector \(x\) weighted with the matrix \(Q\), \(p = 1, 2, \infty\). In (20) we have omitted the constraints \(u_k \in R_{\text{feas}}(x_k), k = 1, \ldots, N\) assuming that they are implicit in the first constraints, i.e., if there exists no \(\mathcal{P}^i\) that contains \([\hat{z}_k]_{\hat{u}_k}\) then this is an infeasible point. We will use this implicit notation throughout the paper.

Remark 3 Reference tracking problem can be easily translated to the form (18) by extending state vector and making appropriate variable substitution.

In the following, we will assume that \(Q = Q' \succeq 0, R = R' \succ 0, P \succeq 0\), for \(p = 2\), and that \(Q, R, P\) are full column rank matrices for \(p = 1, \infty\). We will also denote by \(\mathcal{X}_k \subseteq \mathbb{R}^{mc} \times \{0, 1\}^{mc}\) the set of states \(x(k)\) at time \(k\) for which the optimal control problem (18)-(20) is feasible:

\[
\mathcal{X}_k = \left\{ x \in \mathbb{R}^{mc} \times \{0, 1\}^{mc} \mid \exists u \in \mathbb{R}^{mc} \times \{0, 1\}^{mc} \text{ and } i \in \{1, \ldots, s\} \text{ such that } \left[ \begin{array}{c} x_k \\ u \end{array} \right] \in \hat{\mathcal{P}}^i \text{ and } A^i x + B^i u + f^i \in \mathcal{X}_{k+1} \right\},
\]

\(k = 0, \ldots, N-1, \mathcal{X}_N = \mathcal{X}_f\).

Let \(U_N^*(x(0)) = [(u_0^*(x(0)))', \ldots, (u_{N-1}^*(x(0)))']'\) be a minimizer of problem (18)-(20). In general, the optimal control problem (18)-(20) may not have a minimizer for some feasible \(x(0)\) but only an infimizer. We will assume that problem (18)-(20) always admits a minimizer \(U_N^*(x(0))\) for all feasible \(x(0)\). In this case, the optimal control problem (18)-(20) becomes

\[
J^*(x(0)) \triangleq \min_{U_N} J(U_N, x(0))
\]

subject to

\[
\begin{align*}
x_{k+1} &= A^i x_k + B^i u_k + f^i \text{ if } [\hat{z}_k] \in \hat{\mathcal{P}}^i, \ i = 1, \ldots, s \\
x_N &\in \mathcal{X}_f \\
x_0 &= x(0)
\end{align*}
\]

The optimizer \(U_N^*(x(0))\) may not exist for some \(x(0)\) if the PWA system (17) is discontinuous in the space of continuous inputs and any component of the infimizer of problem (18)-(20) lies on a boundary of a polyhedron belonging to the polyhedral partition of the PWA system (17) for some \(x(0)\). Such a situation is of doubtful practical interest, in the sense that a small discrepancy between the “real”
system and its model can lead to an arbitrarily large deviation between the performance of the closed loop system and its infimum.

The minimizer $U_N^*(x(0))$ may not be unique for a given $x(0)$. In particular, for a fixed $x$, $U_N^*(x)$ is a subset of the space $\mathbb{R}^{Nm_e} \times \{0, 1\}^{Nm_e}$. If $p = 2$ the cardinality of the set $U_N^*(x)$ is finite for all $x$ while, for $p = 1$ or $p = \infty$, $U_N^*(x)$ can have an infinite dimension for some $x$.  

In the following we need to distinguish between optimal control based on the 2-norm and optimal control based on the 1-norm or $\infty$-norm. As last remark, we want to point out that it is almost immediate to extend the results of the following sections to hybrid optimal control problems where discrete quantities are weighted in the cost function, such as the switching between two different regions of operation.

7 Solution Properties

**Theorem 7** The solution to the optimal control problem (18)-(23) with $p = 2$ is a PWA state feedback control law of the form

$$ u_k^*(x(k)) = F_k^i x(k) + G_k^i \text{ if } x(k) \in R_k^i, $$

where $R_k^i$, $i = 1, \ldots, N_k$ is a partition of the set $X_k$ of feasible states $x(k)$, and the closure $\overline{R}_k^i$ of the sets $R_k^i$ has the following form:

$$ \overline{R}_k^i \triangleq \{ x : x(k)'L_k^i(j)x(k) + M_k^i(j)x(k) \leq N_k^i(j), \ j = 1, \ldots, n_k^i \}, \ k = 0, \ldots, N - 1. $$

**Proof:** The piecewise linearity of the solution was first mentioned by Sontag in [48]. In [39] Mayne sketched a proof. In the following we will give the proof for $u_0^*(x(0))$, the same arguments can be repeated for $u_1^*(x(1)), \ldots, u_{N-1}^*(x(N - 1))$.

Depending on the initial state $x(0)$ and on the input sequence $U = [u'_0, \ldots, u'_{N-1}]$, the state $x(k)$ is either infeasible or it belongs to a certain polyhedron $\overline{P}_k$. Suppose for the moment that there are no binary inputs, $m_e = 0$. The number of all possible locations of the state sequence $x_0, \ldots, x_{N-1}$ is equal to $s^N$. Denote by $\{v_i\}_{i=1}^s$ the set of all possible switching sequences over the horizon $N$, and by $v_k^i$ the $k$-th element of the sequence $v_i$, i.e., $v_k^i = j$ if $x_k \in \overline{P}_j$.

Fix a certain $v_i$ and constraine the state to switch according to the sequence $v_i$. Problem (18)-(23) becomes

$$ J_{v_i}^*(x(0)) \triangleq \min_{\{U_N\}} J(U_N, x(0)) $$

$$ \begin{align*}
    & x_{k+1} = A^j x_k + B^j u_k + f^j \\
    & \text{subj. to } \begin{cases} 
        \left[ \begin{array}{c} x_k \\ u_k \end{array} \right] \in \overline{P}_k^v \\
        x_N \in X_f \\
        x_0 = x(0) \\
    \end{cases} \\
    & k = 0, \ldots, N - 1
\end{align*} $$

Assume for the moment that the sets $\overline{P}_k^v$, $k = 0, \ldots, N - 1$ are closed. Problem (25)-(26) is equivalent to a finite-time optimal control problem for a linear time-varying system with time-varying constraints.

---

3For the sake of simplicity we will not introduce the concept of point-to-set map. From an engineering point of view it make sense to refer to the function $U_N^*(x(0))$ as one of the possible single-valued functions that can be extracted from the original point-to-set map.
and can be solved by using the approach outlined in [10]. Its solution is the PPWA feedback control law

\[ u_0^i(x(0)) = \tilde{F}^{i,j}x(0) + \tilde{G}^{i,j}, \quad \forall x(0) \in T^{i,j}, \quad j = 1, \ldots, N^{ri} \]  

(27)

where \( D^i = \bigcup_{j=1}^{N^{ri}} T^{i,j} \) is a polyhedral partition of the convex set \( D^i \) of feasible states \( x(0) \) for problem (25)-(26). \( N^{ri} \) is the number of regions of the polyhedral partition of the solution and it is a function of the number of constraints in problem (25)-(26). The upper index \( i \) in (27) denotes that the input \( u_0^i(x(0)) \) is optimal when the switching sequence \( v_i \) is fixed.

The optimal solution \( u_0^i(x(0)) \) to the original problem (18)-(23) can be found by solving problem (25)-(26) for all \( v_i \). The set \( X_0 \) of all feasible states at time 0 is \( X_0 = \bigcup_{i=1}^{s} D^i \) and in general it is not convex.

Indeed, as some initial states can be feasible for different switching sequences, the sets \( D^i, i = 1, \ldots, s \), in general, can overlap. The solution \( u_0^i(x(0)) \) can be computed in the following way. For every polyhedron \( T^{i,j} \) in (27),

1. If \( T^{i,j} \cap T^{l,m} = \emptyset \) for all \( l \neq i, l = 1, \ldots, s, m = 1, \ldots, N^{ri} \), then the switching sequence \( v_i \) is the only feasible one for all the states belonging to \( T^{i,j} \) and therefore the optimal solution is given by (27), i.e.

\[ u_0^i(x(0)) = \tilde{F}^{i,j}x(0) + \tilde{G}^{i,j}, \quad \forall x \in T^{i,j}. \]  

(28)

2. If \( T^{i,j} \) intersects one or more polyhedra \( T^{l_1,m_1}, T^{l_2,m_2}, \ldots \), the states belonging to the intersection are feasible for more than one switching sequence \( v_i, v_{l_1}, v_{l_2}, \ldots \) and therefore the corresponding value functions \( J^*_v(x(0)), J^*_{v_{l_1}}(x(0)), J^*_{v_{l_2}}(x(0)), \ldots \) in (25) have to be compared in order to compute the optimal control law.

Consider the simple case when only two polyhedra overlap, i.e., \( T^{i,j} \cap T^{l,m} \neq \emptyset \). We will refer to \( T^{(i,j),(l,m)} \) as a polyhedron of multiple feasibility. For all states belonging to \( T^{(i,j),(l,m)} \) the optimal solution is:

\[
\begin{align*}
  u_0^i(x(0)) = & \begin{cases} 
    \tilde{F}^{i,j}x(0) + \tilde{G}^{i,j}, & \forall x(0) \in T^{(i,j),(l,m)} : J^*_v(x(0)) < J^*_{v_{l_1}}(x(0)) \\
    \tilde{F}^{l,m}x(0) + \tilde{G}^{l,m}, & \forall x(0) \in T^{(i,j),(l,m)} : J^*_{v_{l_1}}(x(0)) > J^*_{v_{l_2}}(x(0)) \\
    \tilde{F}^{i,j}x(0) + \tilde{G}^{i,j} & \text{or} \tilde{F}^{l,m}x(0) + \tilde{G}^{l,m}, & \forall x(0) \in T^{(i,j),(l,m)} : J^*_v(x(0)) = J^*_{v_{l_1}}(x(0))
  \end{cases}
\end{align*}
\]

(29)

Because \( J^*_v(x(0)) \) and \( J^*_{v_{l_1}}(x(0)) \) are quadratic functions on \( T^{i,j} \) and \( T^{l,m} \) respectively, the theorem is proved. In general, a polyhedron of multiple feasibility where \( n \) value functions intersect is partitioned into \( n \) non convex subsets where in each one of them a certain value function is greater than all the others.

Assume now that there exists \( k \) such that \( \tilde{P}^{v_k} \) is not closed. Then we will proceed in the following way. In problem (25)-(26) we substitute \( \tilde{P}^{v_k} \) with its closure \( \overline{\tilde{P}^{v_k}} \). After having computed the PPWA solution (27), we exclude from the sets \( D^i \) all the initial states \( x(0) \) belonging to \( \overline{\tilde{P}^{v_k}} \) but not to the set \( \tilde{P}^{v_k} \). The proof follows as outlined above, except that the sets \( T^{i,j} \) may be neither open nor closed polyhedra.

The proof can be repeated in the presence of binary inputs, \( m_\ell \neq 0 \). In this case the switching sequences \( v_i \) are given by all combinations of region indices and binary inputs, i.e., \( i = 1, \ldots, (s \cdot m_\ell)^N \).
The continuous component of the optimal input is given by (28) or (29). Such an optimal continuous component of the input has an associated optimal sequence $v_i$, whose component provide the remaining binary components of the optimal input.

**Remark 4** Let $T^{(i,j),(l,m)}$ be a polyhedron of multiple feasibility and let $F = \{ x \in T^{(i,j),(l,m)} : J^*_v(x) = J^*_v(x) \}$ be the set where the quadratic functions $J^*_v(x)$ and $J^*_v(x)$ intersect (for the sake of simplicity we consider the case where only two polyhedra intersect). We distinguish four cases (sub-cases of case 2 in Theorem 7):

2.a $F = \emptyset$, i.e., $J^*_v(x)$ and $J^*_v(x)$ do not intersect over $\tilde{T}^{(i,j),(l,m)}$.

2.b $F = \{ x : Ux = P \}$ and $J^*_v(x)$ and $J^*_v(x)$ are tangent on $F$.

2.c $F = \{ x : Ux = P \}$ and $J^*_v(x)$ and $J^*_v(x)$ are not tangent on $F$.

2.d $F = \{ x : x'Yx + Ux = P \}$ with $Y \neq 0$.

In the first case $T^{(i,j),(l,m)}$ is not further partitioned, the optimal solution in $T^{(i,j),(l,m)}$ is either $\tilde{F}^{i,j}x(0) + \tilde{G}^{i,j}$ or $\tilde{F}^{l,m}x(0) + \tilde{G}^{l,m}$. In case 2.b, $T^{(i,j),(l,m)}$ is not further partitioned but there are multiple optimizers on the set $Ux = P$. In case 2.c, $T^{(i,j),(l,m)}$ is partitioned into two polyhedra. In case 2.d $T^{(i,j),(l,m)}$ is partitioned into two sets (not necessarily connected) as shown in Figure 6.

In the special situation where case 2.c or 2.d occur but the control laws are identical, i.e., $F^{i,j} = F^{l,m}$ and $\tilde{G}^{i,j} = \tilde{G}^{l,m}$, we will assume that the set $T^{(i,j),(l,m)}$ is not further partitioned.

**Example 7.1**
Consider the following simple system

\[
\begin{aligned}
x(t+1) &= \begin{cases} 
-1 & 2 \\
2 & 3 \\
1 & 2 \\
0 & 1 
\end{cases} x(t) + \begin{cases} 
0 \\
1 \\
1 \\
0 
\end{cases} u(t) & \text{if } x(t) \in \mathcal{P}^1 = \{x : [0,1]x \geq 0\} \\
\end{aligned}
\]

and the optimal control problem (18)-(23), with \( p = 2, N = 2, Q = I, R = 10, P = Q, \mathcal{X}_f = \{x \in \mathbb{R}^2 | \begin{bmatrix} -1 \\
-0.5 \end{bmatrix} \leq x \leq \begin{bmatrix} 1 \\
1 \end{bmatrix} \} \).

The possible switching sequences are \( v_1 = \{1,1\}, v_2 = \{1,2\}, v_3 = \{2,1\}, v_4 = \{2,2\} \). The solution to problem (25)-(26) is depicted in Figure (7) (note that the switch \( v_3 \) is infeasible for all \( x(0) \)). In Figure 8(a) the four solutions are intersected, the white region corresponds to polyhedra of multiple feasibility. The state-space partition of the optimal control law is depicted in Figure 8(b) (for lack of space, we do not report here the analytic expressions of the regions and the corresponding affine gains).

**Theorem 8** Assume that the PWA system (17) is continuous and has no state constraints. Then, the value function \( J^*(x(0)) \) in (23) is continuous.

**Proof:** The proof follows from the continuity of \( J(U_N, x(0)) \) and Theorem 1.

\[ \square \]

In general, even for continuous PWA systems with state constraints it is difficult to find weak assumptions ensuring the continuity of the value function \( J^*(x(0)) \). Ensuring the continuity of the optimal control law \( u_k^*(x(k)) \) is even more difficult. Theorem 2 provides a sufficient condition for \( U_N^*(x(0)) \) to be continuous. It requires the cost \( J(U_N, x(0)) \) to be continuous and strictly quasi-convex in \( U_N \) for each \( x(0) \) and the constraints in (23) to be convex in \( U_N \) for each \( x(0) \). Such conditions are clearly too restrictive since the cost and the constraints in problem (23) are a composition of quadratic and linear functions, respectively, with the piecewise affine dynamics of the system.

The next theorem provides a condition under which the solution \( u_k^*(x(k)) \) of the optimal control problem (18)-(23) is a PPWA state feedback control law.

**Theorem 9** Assume that the optimizer \( U_N^*(x(0)) \) of (18)-(23) is unique for all \( x(0) \). Then, the solution to the optimal control problem (18)-(23) is a PPWA state feedback control law of the form

\[
u_k^*(x(k)) = F_k^i x(k) + G_k^i \text{ if } x(k) \in \mathcal{R}_k^i \quad k = 0, \ldots, N - 1,
\]

where \( \mathcal{R}_k^i, i = 1, \ldots, N_k^i \), is a polyhedral partition of the set \( \mathcal{X}_k \) of feasible states \( x(k) \).

**Proof:** We will show that case 2.d in Remark 4 cannot occur by contradiction. Suppose case 2.d occurs. Since the optimizer \( u_0^*(x(0)) \) is unique and the value function \( J^*(x(0)) \) is continuous on \( \mathcal{F} \), by Theorem 7 this implies that \( \bar{F}^i x(0) + \bar{G}^i x(0) = \bar{F}^l x(0) + \bar{G}^l x(0), \forall x(0) \in \mathcal{F} \). This contradicts the hypothesis since the set \( \mathcal{F} \) is not a hyperplane. The same arguments can be repeated for \( u_k^*(x(k)), k = 1, \ldots, N - 1 \). \[ \square \]
Remark 5 Theorem 9 relies on a rather weak uniqueness assumption. As the proof indicates, the key point is to exclude case 2d in Remark 4. Therefore, it is reasonable to believe that there are other conditions or problem classes which satisfy this structural property without claiming uniqueness. We are also currently trying to identify and classify situations where the state transition structure guarantees the absence of disconnected sets as shown in Figure 6(b).

The following Corollary summarizes the properties enjoyed by the solution to problem (18)-(23) as a direct consequence of Theorems 7-9.

Corollary 1

1. $u^*_k(x(k))$ and $J^*(x(0))$ are, in general, discontinuous and $X_k$ may be non-convex.

2. $J^*(x(0))$ can be discontinuous only on a facet of a polyhedron of multiple feasibility.

3. If there exists a polyhedron of multiple feasibility with $\mathcal{F} = \{x : x'Yx + Ux = P\}$, $Y \neq 0$, then $u^*_k(x(k))$ is not unique on $\mathcal{F}$, except possibly at isolated points.
The previous results can be extended to piecewise linear cost functions, i.e., cost functions based on the $1$-norm or the $\infty$-norm.

**Theorem 10** The solution to the optimal control problem (18)-(23) with $p = 1$, $\infty$ is a PPWA state feedback control law of the form

$$u_k^*(x(k)) = F_k^i x(k) + G_k^i \text{ if } x(k) \in R_k^i,$$

where $R_k^i$, $i = 1, \ldots, N_k^r$ is a polyhedral partition of the set $X_k$ of feasible states $x(k)$.

**Proof:** The proof is similar to the proof of Theorem 7. Fix a certain switching sequence $v_i$, consider the problem (18)-(23) and constrain the state to switch according to the sequence $v_i$ to obtain problem (25)-(26). Problem (25)-(26) can be viewed as a finite time optimal control problem with performance index based on $1$-norm or $\infty$-norm for a linear time varying system with time varying constraints and can be solved by using the multiparametric linear program as described in [7]. Its solution is a PPWA feedback control law

$$u_0^i(x(0)) = \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, \quad \forall x \in T^{i,j}, \quad j = 1, \ldots, N^{r,i},$$

and the value function $J_v^*$ is piecewise affine on polyhedra and convex. The rest of the proof follows the proof of Theorem 7. Note that in this case the value functions to be compared are piecewise affine and not piecewise quadratic.  

By comparing Theorem 7 and Theorem 10 it is clear that it is simpler to solve problems with performance indices based on $1$ or $\infty$ norms: in this case the solution is piecewise affine on polyhedra and one does not need to deal with nonconvex ellipsoidal regions as in the $2$ norm case.
8 Computation of the Optimal Control Law

In the previous section the properties enjoyed by the solution to hybrid optimal control problems were investigated. The proof is constructive (as shown in the figures), but it is based on the enumeration of all the possible switching sequences of the hybrid system, the number of which grows exponentially with the time horizon. Although the computation is performed off-line (the on-line complexity is the one associated with the evaluation of the PWA control law (31)), more efficient methods than enumeration are desirable.

In the next sections we will describe how to compute in an efficient way the solution to finite time optimal control problems. In section 8.1 we review the method that has been proposed in [9]. The main idea is to translate problem (18)-(23) into a linear or quadratic mixed integer program that can be solved by using standard commercial software [34]. The translation of problem (18)-(20) into a mixed integer program is immediate if the equivalent MLD representation of the PWA system is available, e.g., through HYSDEL. This approach does not provide the state feedback law (24) or (32) but only the optimal control sequence \( U^*_N(x(0)) \) for a given initial state \( x(0) \).

In section 8.2 we describe how the state feedback law (24) or (32) can be computed by means of multiparametric mixed integer programming as proposed in [12]. We briefly describe the main idea of multi-parametric mixed integer programming and how multi-parametric solvers can be used to solve problem (18)-(23).

However, the use of multiparametric mixed integer programming has a major drawback: the solver does not exploit the structure of the optimal control problem. In fact, a large part of the information associated with problem (18)-(23) becomes hidden when it is reformulated as a mixed integer program. In Section 9 we propose a novel algorithm based on dynamic programming that allows one to compute the state feedback optimal control laws (24) or (32) in an efficient way without using a mixed-integer program solver.

8.1 Pointwise Computation via Mixed Integer Programming

The MLD framework can be used to avoid enumeration when solving optimal control problems. In fact, when the model of the system is an MLD model and the performance index is quadratic, the optimization problem can be cast as a Mixed-Integer Quadratic Program (MIQP) [9]. Similarly, 1-norm and \( \infty \)-norm performance indices lead to Mixed-Integer Linear Programming (MILP) problems. In the following we detail the translation of problem (18)-(23) into a mixed integer linear or quadratic program for which efficient solution algorithms exist [34].

Consider the equivalent MLD representation (9) of the PWA system (17). Problem (18)-(23) is
rewritten as:

\[ J^*(x(0)) = \min_{U_N} \|Px_N\|_p + \sum_{k=0}^{N-1} \|Q_1 u_k\|_p + \|Q_2 \delta_k\|_p + \|Q_3 z_k\|_p + \|Q_4 x_k\|_p \]

subject to \[
\begin{align*}
x_{k+1} &= A x_k + B_1 u_k + B_2 \delta_k + B_3 z_k \\
E_2 \delta_k + E_3 z_k &\leq E_1 u_k + E_4 x_k + E_5 \\
x_N &\in \mathcal{X}_f \\
x_0 & = x(0)
\end{align*}
\]

Note that the cost function (34) is more general than (18), and includes also weights on auxiliary variables \( z \) and \( (0,1) \)-variables \( \delta \).

The optimal control problem in (34)-(35) can be formulated as a Mixed Integer Quadratic Program (MIQP) when the squared Euclidean norm is used \( (p=2) \) [9], or as a Mixed Integer Linear Program (MILP), when \( p=\infty \) or \( p=1 \) [6],

\[ \min_{\varepsilon} \quad \varepsilon' H_1 \varepsilon + \varepsilon' H_2 x(0) + x(0)' H_3 x(0) + c_1' \varepsilon + c_2' x(0) + c_3 \]

subject to \[ G \varepsilon \leq W + S x(0) \]

where \( H_1, H_2, H_3, c_1, c_2, c_3, G, W, S \) are matrices of suitable dimensions, \( \varepsilon = [\varepsilon_c', \varepsilon_\ell'] \) where \( \varepsilon_c, \varepsilon_\ell \) represent continuous and discrete variables, respectively, and \( H_1, H_2, H_3, \) are null matrices if problem (36) is an MILP.

The translation of (34)-(35) into (36) for \( p=2 \) is simply obtained by substituting the state update equation

\[ x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j (B_1 u_{k-1-j} + B_2 \delta_{k-1-j} + B_3 z_{k-1-j}) \]

and the optimization vector \( \varepsilon = [u'_0, \ldots, u'_{N-1}, \delta'_0, \ldots, \delta'_{N-1}, z'_0, \ldots, z'_{N-1}]' \) in (36).

For \( p=1, \infty \), the translation requires the introduction of slack variables. In particular, for \( p=\infty \) the sum of the components of any vector

\[ [\varepsilon^u_0, \ldots, \varepsilon^u_{N-1}, \varepsilon^\delta_0, \ldots, \varepsilon^\delta_{N-1}, \varepsilon^\ell_0, \ldots, \varepsilon^\ell_{N-1}, \varepsilon^x_0, \ldots, \varepsilon^x_N]' \]

that satisfies

\[
\begin{align*}
-1_m \varepsilon^u_k &\leq Q_1 u_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_m \varepsilon^u_k &\leq -Q_1 u_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_r \varepsilon^\delta_k &\leq Q_2 \delta_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_r \varepsilon^\delta_k &\leq -Q_2 \delta_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_r \varepsilon^\ell_k &\leq Q_3 z_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_r \varepsilon^\ell_k &\leq -Q_3 z_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_n \varepsilon^x_k &\leq Q_4 x_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_n \varepsilon^x_k &\leq -Q_4 x_k, \quad k = 0, 1, \ldots, N - 1 \\
-1_n \varepsilon^N &\leq P x_N, \\
-1_n \varepsilon^N &\leq P x_N,
\end{align*}
\]
represents an upper bound on $J^*(x(0))$, where $1_h$ is a column vector of ones of length $h$, and where $x(k)$ is expressed as in (37). Similarly to what was shown in [19], it is easy to prove that vector
\[ \varepsilon \triangleq [\varepsilon^u_0, \ldots, \varepsilon^u_{N-1}, \varepsilon^\delta_0, \ldots, \varepsilon^\delta_{N-1}, \varepsilon^r_0, \ldots, \varepsilon^r_{N-1}, u'_0, \ldots, u'_{N-1}, \delta'_0, \ldots, \delta'_{N-1}, z'_0, \ldots, z'_{N-1}]^T \]
that satisfies equations (35) and (38), and simultaneously minimizes
\[ J(\varepsilon) = \varepsilon^u_0 + \ldots + \varepsilon^u_{N-1} + \varepsilon^\delta_0 + \ldots + \varepsilon^\delta_{N-1} + \varepsilon^r_0 + \ldots + \varepsilon^r_{N-1} \] (39)
also solves the original problem, i.e., the same optimum $J^*(U_N, x(0))$ is achieved. Therefore, problem (34)-(35) can be reformulated as the following MILP problem
\[
\begin{align*}
\min_{\varepsilon} & \quad J(\varepsilon) \\
\text{subj. to} & \quad -1_m^{\varepsilon^u_k} \leq \pm Q_1 u_k, \ k = 0, 1, \ldots, N - 1 \\
& \quad -1_r \varepsilon^\delta_k \leq \pm Q_2 \delta_k, \ k = 0, 1, \ldots, N - 1 \\
& \quad -1_r \varepsilon^r_k \leq \pm Q_3 z_k, \ k = 0, 1, \ldots, N - 1 \\
& \quad -1_n \varepsilon^\delta_k \leq \pm Q_4 (A^k x_0 + \sum_{j=0}^{k-1} A^j (B_1 u_{k-1-j} + B_2 \delta_{k-1-j} + B_3 z_{k-1-j}) ) \ k = 0, \ldots, N - 1 \\
& \quad -1_n \varepsilon^r_k \leq \pm P(A^k x_0 + \sum_{j=0}^{k-1} A^j (B_1 u_{k-1-j} + B_2 \delta_{k-1-j} + B_3 z_{k-1-j}) ) \\
& \quad x_{k+1} = A x_k + B_1 u_k + B_2 \delta_k + B_3 z_k, \ k \geq 0 \\
& \quad E_2 \delta_k + E_3 z_k \leq E_1 u_k + E_4 x_k + E_5, \ k \geq 0 \\
& \quad x_N \in X_f \\
& \quad x_0 = x(0)
\end{align*}
\] (40)
where the variable $x(0)$ appears only in the constraints in (40) as a parameter vector.

Given a value of the initial state $x(0)$, the MIQP (36) (or MILP (40)) can be solved to get the optimizer $\varepsilon^*$ and therefore the optimal input sequence $u'_0, \ldots, u'_{N-1}$.

### 8.2 State-Feedback Computation via multi-parametric Mixed Integer Programming

Multiparametric programming [28, 25, 10, 14] can be used to efficiently compute the PWA form (24) or (32) of the optimal state feedback control law $u^*(x(k))$. By generalizing the results for linear systems to hybrid systems, the state vector $x(0)$, which appears in the objective function and in the linear part of the right-hand-side of the constraints (36), can be handled as a vector of parameters that perturbs the solution of the mathematical program (36). Then, for performance indices based on the $\infty$-norm or 1-norm, the optimization problem can be treated as a multiparametric MILP (mp-MILP), while for performance indices based on the 2-norm, the optimization problem can be treated as a multiparametric MIQP (mp-MIQP). Solving an mp-MILP (mp-MIQP) amounts to expressing the solution of the MILP (MIQP) (36) as a function of the parameters $x(0)$.

In [1, 25] two approaches were proposed for solving mp-MILP problems. In both methods the authors use an mp-LP algorithm and a branch and bound strategy that avoids the enumeration of all combinations of 0-1 integer variables. A method for solving mp-MIQPs has appeared only recently in [24].
The solution of the mp-MILP (mp-MIQP) (36) provides the state feedback solution \( u^*(k) = f_k(x(k)) \) (32) or (24) of CFTOC (18)-(23) for \( k = 0 \) and it also provides the open loop optimal control laws \( u^*(k) = u^*(k)(x(0)) \). The state feedback piecewise-affine optimal control law \( f_k : x(k) \mapsto u^*(k) \) for \( k = 1, \ldots, N \) is obtained by solving \( N \) mp-MILP (mp-MIQP) over a shorter horizon [11].

More details on multiparametric programming can be found in [28, 14] for linear programs, in [10, 51, 47] for quadratic programs in [1, 25] for mixed integer linear programs and in [24, 11] for mixed integer quadratic programs.

9 Computation of the State Feedback Solution via Dynamic Programming

In this section we will show how linear and quadratic parametric programming can be used to solve the Hamilton-Jacobi-Bellman equations associated with CFTOC problem (18)-(23). The PWA solution (24) will be computed proceeding backwards in time using two tools: simple linear or quadratic multi-parametric programming solver (depending on the cost function used) and a special technique to store the solution which will be illustrated in the next sections. The algorithm will be presented for optimal control based on a quadratic performance criterion. Its extension to optimal control based on a linear performance criterion is straightforward.

9.1 Preliminaries and Basic Steps

Consider the PWA map \( \zeta(x) \)

\[
\zeta : x \mapsto F_i x + G_i \quad \text{if} \quad x \in R_i, \quad i = 1, \ldots, N_R,
\] (41)

where \( R_i, \ i = 1, \ldots, N_R \) are subsets of the \( x \)-space. If there exist \( l, m \in \{1, \ldots, N_R\} \) such that for \( x \in R_l \cap R_m, F_l x + G_l \neq F_m x + G_m \) the map \( \zeta(x) \) (41) is not single valued.

**Definition 9** Given a PWA map (41) we define the function \( f_{PWA}(x) = \zeta_0(x) \) as the ordered region single-valued function associated with (41) when

\[
\zeta_0(x) = \begin{cases} 
F_j x + G_j & \text{if } x \in R_j \\
\end{cases}
\]

\[
\forall i < j : x \notin R_i, j \in \{1, \ldots, N_R\},
\]

and we write it in the following form

\[
\zeta_0(x) = \begin{cases} 
F_1 x + G_1 & \text{if } x \in R_1 \\
\vdots & \\
F_{N_R} x + G_{N_R} & \text{if } x \in R_{N_R}.
\end{cases}
\]

Note that given a PWA map (41) the corresponding ordered region single-valued function \( \zeta_0(x) \) changes if the order used to store the regions \( R_i \) and the corresponding affine gains changes. For illustration purpose consider an example depicted in Figure 9, where \( x \in \mathbb{R}, N_R = 2, F_1 = 0, G_1 = 0, \quad R_1 = [-2, 1], F_2 = 1, G_2 = 0, \quad R_2 = [0, 2] \).

In the following we assume that the sets \( R_i^k \) in the optimal solution (24) can overlap. When we refer to the PWA function \( u_k^*(x(k)) \) in (24) we will implicitly mean the ordered region single-valued function associated with the mapping (24).
(a) Multiple valued PWA map \( \zeta(x) \)

(b) Ordered region single valued function \( \zeta_{12} \)

(c) Ordered region single valued function \( \zeta_{21} \)

Figure 9: Illustration of the ordered region single valued function.

**Proposition 1** Let \( J_1^* : \mathcal{R}_1 \to \mathbb{R} \) and \( J_2^* : \mathcal{R}_2 \to \mathbb{R} \) be two quadratic functions, \( J_i^*(x) \triangleq x' L_i x + M_i x + N_i \) and \( J_2^*(x) \triangleq x' L_2 x + M_2 x + N_2 \), where \( \mathcal{R}_1 \) and \( \mathcal{R}_2 \) are convex polyhedra and \( J_i^*(x) = +\infty \) if \( x \notin \mathcal{R}_i \), \( i \in \{1, 2\} \). Let \( u_1^* : \mathcal{R}_1 \to \mathbb{R}^m \), \( u_2^* : \mathcal{R}_2 \to \mathbb{R}^m \) be vector functions. Let \( \mathcal{R}_1 \cap \mathcal{R}_2 = \mathcal{R}_3 \neq \emptyset \) and define

\[
J^*(x) \triangleq \min\{J_1^*(x), J_2^*(x)\} \quad (42)
\]

\[
u^*(x) \triangleq \begin{cases} u_1^*(x) & \text{if } J_1^*(x) \leq J_2^*(x) \\ u_2^*(x) & \text{if } J_1^*(x) > J_2^*(x). \end{cases} \quad (43)
\]

Let \( L_3 = L_2 - L_1 \), \( M_3 = M_2 - M_1 \), \( N_3 = N_2 - N_1 \). Then, corresponding to the three following cases

1. \( J_1^*(x) \leq J_2^*(x) \) \( \forall x \in \mathcal{R}_3 \)
2. \( J_1^*(x) \geq J_2^*(x) \) \( \forall x \in \mathcal{R}_3 \)
3. \( \exists x_1, x_2 \in \mathcal{R}_3|J_1^*(x_1) < J_2^*(x_1) \) and \( J_1^*(x_2) > J_2^*(x_2) \)

the expressions (42) and (43) can be written equivalently as:
1. 

\[
J^*(x) = \begin{cases} 
J_1^*(x) & \text{if } x \in \mathcal{R}_1 \\
J_2^*(x) & \text{if } x \in \mathcal{R}_2
\end{cases}
\] (44)

\[
u^*(x) = \begin{cases} 
u_1^*(x) & \text{if } x \in \mathcal{R}_1 \\
u_2^*(x) & \text{if } x \in \mathcal{R}_2
\end{cases}
\] (45)

2. as in (44) and (45) by switching the indices 1 and 2

3. 

\[
J^*(x) = \begin{cases} 
\min\{J_1^*(x), J_2^*(x)\} & \text{if } x \in \mathcal{R}_3 \\
J_1^*(x) & \text{if } x \in \mathcal{R}_1 \\
J_2^*(x) & \text{if } x \in \mathcal{R}_2
\end{cases}
\] (46)

\[
u^*(x) = \begin{cases} 
u_1^*(x) & \text{if } x \in \mathcal{R}_3 \text{ and } x'L_3x + M_3x + N_3 \geq 0 \\
u_2^*(x) & \text{if } x \in \mathcal{R}_3 \text{ and } x'L_3x + M_3x + N_3 \leq 0 \\
u_1^*(x) & \text{if } x \in \mathcal{R}_1 \\
u_2^*(x) & \text{if } x \in \mathcal{R}_2
\end{cases}
\] (47)

where (44), (45), (46), and (47) have to be considered as PWA and PPWQ functions in the ordered region sense.

Proposition 1 allows us to

• avoid the storage of the intersections of two polyhedra in case 1 and 2

• avoid the storage of possibly non convex regions \(\mathcal{R}_1 \setminus \mathcal{R}_3\) and \(\mathcal{R}_2 \setminus \mathcal{R}_3\)

• work with multiple quadratic functions instead of quadratic functions defined over non-convex and non-polyhedral regions.

The three point listed above will be the three basic ingredients for storing and simplifying the optimal control law (24). Next we will show how to compute it.

Remark 6 To distinguish between cases 1, 2 and 3 of Proposition 1, in general, one needs to solve an indefinite quadratic program, namely

\[
\min_x \ x'L_3x + M_3x + N_3 \\
\text{s.t.} \quad x \in \mathcal{R}_3.
\]

In our approach, if such a test is avoided the form (47) corresponding to the third case, is used. The only drawback is that the form (47) is, in general, a non-minimal representation of the value function and therefore it increases the complexity of evaluating the optimal control action (24).
9.2 Multiparametric Programming with Multiple Quadratic Functions

Consider the multiparametric program

\[ J^\star(x) \triangleq \min_u \ l(x,u) + q(f(x,u)) \]

\[ \text{s.t.} \quad f(x,u) \in \mathcal{R}, \quad \] (48)

where \( \mathcal{R} \subseteq \mathbb{R}^n \), \( f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \), \( q : \mathcal{R} \to \mathbb{R} \), and \( l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) is a quadratic function of \( x \) and \( u \). We aim at determining the region \( \mathcal{X} \) of variables \( x \) such that the program (48) is feasible and the optimum \( J^\star(x) \) is finite, and at finding the expression \( u^\star(x) \) of (one of) the optimizer(s). We point out that the constraint \( f(x,u) \in \mathcal{R} \) implies a constraint on \( u \) as a function of \( x \) since \( u \) can assume only values where \( f(x,u) \) is defined.

Next we show how to solve several forms of the problem (48).

**Lemma 1 (one to one problem)** Problem (48) where \( f \) is linear, \( q \) is quadratic, and \( \mathcal{R} \) is a convex polyhedron can be solved by one mp-QP.

**Proof:** See [10]

**Lemma 2 (one to one problem of multiplicity \( d \))** Problem (48) where \( f \) is linear, \( q \) is a multiple quadratic function of multiplicity \( d \) and \( \mathcal{R} \) is a convex polyhedron can be solved by \( d \) mp-QP's.

**Proof:** The multiparametric program to be solved is

\[ J^\star(x) = \min \{ \min_u l(x,u) + q_1(f(x,u)), \ldots, \min_u l(x,u) + q_d(f(x,u)) \} \]

\[ \text{s.t.} \quad f(x,u) \in \mathcal{R} \] (50)

and it is equivalent to

\[ J^\star(x) = \min \left\{ \begin{array}{c} \min_u l(x,u) + q_1(f(x,u)), \ldots, \min_u l(x,u) + q_d(f(x,u)) \\ \text{s.t.} \quad f(x,u) \in \mathcal{R} \end{array} \right\} . \] (51)

The \( i \)-th sub-problems in (51)

\[ J^\star_i(x) \triangleq \min_u l(x,u) + q_i(f(x,u)) \]

\[ \text{s.t.} \quad f(x,u) \in \mathcal{R} \] (53)

is a one to one problem and therefore it is solvable by an mp-QP. Let the solution of the \( i \)-th mp-QPs be

\[ u^i(x) = \tilde{F}^{i,j} x + \tilde{G}^{n,j}, \quad \forall x \in T^{i,j}, \quad j = 1, \ldots, N^{r_i}, \] (54)

where \( T^i = \bigcup_{j=1}^{N^{r_i}} T^{i,j} \) is a polyhedral partition of the convex set \( T^i \) of feasible \( x \) for the \( i \)-th sub-problem and \( N^{r_i} \) is the number of polyhedral regions. The feasible set \( \mathcal{X} \) satisfies \( \mathcal{X} = T^1 = \ldots = T^d \) since the constraints of the \( d \) sub-problems are identical.

The solution \( u^\star(x) \) to the original problem (50) is obtained by comparing and storing the solution of \( d \) mp-QPs subproblems (53) as explained in Proposition 1. Consider the case \( d = 2 \), and consider the intersection of the polyhedra \( T^{1,i} \) and \( T^{2,l} \) for \( i = 1, \ldots, N^{r_1} \), \( l = 1, \ldots, N^{r_2} \). For all \( T^{1,i} \cap T^{2,l} \triangleq T^{(1,i),(2,l)} \neq \emptyset \) the optimal solution is stored in an ordered way as described in Proposition 1, while
paying attention to the fact that a region could be already stored. Moreover, when storing a new polyhedron with the corresponding value function and optimizer the relative order of the regions already stored must not be changed. The result of this Intersect and Compare procedure is

\[ u^*(x) = F^i x + G^i \quad \text{if} \quad x \in \mathcal{R}^i \triangleq \{ x : x' L^i(j)x + M^i(j)x \leq N^i(j), \ j = 1, \ldots, n^i \} \quad (55) \]

where \( \mathcal{R} = \bigcup_{j=1}^{N_R} \mathcal{R}^j \) is a convex polyhedron and the value function

\[ J^*(x) = \tilde{J}^*_j(x) \quad \text{if} \quad x \in \mathcal{D}^j, \ j = 1, \ldots, N^D, \quad (56) \]

where \( \tilde{J}^*_j(x) \) are multiple quadratic functions defined over the convex polyhedra \( \mathcal{D}^j \). The polyhedra \( \mathcal{D}^j \) can contain more regions \( \mathcal{R}^i \)'s or can coincide with one of them. Note that (55) and (56) have to be considered as PWA and PPWQ functions in the ordered region sense.

If \( d > 2 \) then the value function in (56) is intersected with the solution of the third mp-QP subproblem and the procedure is iterated making sure not to change the relative order of the polyhedra and corresponding gain of the solution constructed in the previous steps. The solution will still have the same form (55)--(56).

**Lemma 3 (one to r problem)** Problem (48) where \( f \) is linear, \( q \) is a PPWQ function defined over \( r \) polyhedral regions can be solved by \( r \) mp-QP's

**Proof:** Let \( q(x) \triangleq q_i \quad \text{if} \quad x \in \mathcal{R}_i \) the PWQ function where \( \mathcal{R}_i \) are convex polyhedra and \( q_i \) quadratic functions. The multiparametric program to solve is

\[ J^*(x) = \min \{ \min_{u^i} l(x,u) + q_1(f(x,u)) + \ldots + \min_{u^i} l(x,u) + q_r(f(x,u)) \} \quad (57) \]

The proof is similar to the proof of the previous theorem with the only difference that the constraints of the \( i \)-th mp-QP subproblem differ from the one of the \( j \)-th mp-QP subproblem, \( i \neq j \). Therefore the procedure based on solving mp-QPs and storing the results as in Proposition 1 will be the same but the domain \( \mathcal{R} = \bigcup_{j=1}^{N_R} \mathcal{R}^j \) of the solution

\[ u^*(x) = F^i x + G^i \quad \text{if} \quad x \in \mathcal{R}^i \triangleq \{ x : x' L^i x + M^i x \leq N^i \} \quad (58) \]

\[ J^*(x) = \tilde{J}^*_j(x) \quad \text{if} \quad x \in \mathcal{R}^j \quad (59) \]

can be a non-convex polyhedron and even disconnected if the domain of the PWA function \( f(x,u) \) is not connected. \hfill \Box

If \( f \) is PPWA defined over \( s \) regions then we have a \( s \) to \( X \) problem where \( X \) can belong to any of the ones listed above. In particular, we have a \( s \) to \( r \) problem of multiplicity \( d \) if \( f \) is PPWA and defined over \( s \) regions and \( q \) is a multiple PPWQ function of multiplicity \( d \), defined over \( r \) polyhedral regions. The following Lemmas can be proven along the same lines of the proofs given before.

**Lemma 4 (one to r problem of multiplicity d)** Problem (48) where \( f \) is linear and \( q \) is a PPWQ function of multiplicity \( d \), defined over \( r \) polyhedral regions can be solved by \( rd \) mp-QP’s

**Lemma 5 (s to one problem)** An \( s \) to one problem can be decomposed into \( s \) one to one problem.

**Lemma 6 (s to r problem)** An \( s \) to \( r \) problem can be decomposed into \( s \) one to \( r \) problems.

**Lemma 7 (s to r problem of multiplicity d)** An \( s \) to \( r \) problems of multiplicity \( d \) can be decomposed into \( s \) one to \( r \) problem of multiplicity \( d \).
9.3 Algorithmic Solution of the HJB Equations

In the following we will substitute the CPWA system equations (17) with the shorter form

$$x(k+1) = \tilde{f}_{PWA}(x(k), u(k))$$  \hspace{1cm} (60)

where $\tilde{f}_{PWA} : \tilde{P} \rightarrow \mathbb{R}^n$ and $\tilde{f}_{PWA}(x, u) = A_i x + B_i u + f_i$ if $\left[ \begin{array}{c} x \\ u \end{array} \right] \in \tilde{P}_i$, $i = 1, \ldots, s$, and $\{\tilde{P}_i\}$ is polyhedral partition of $\tilde{P}$.

Consider the dynamic programming formulation of the CFTOC problem (18)-(23),

$$J^*_j(x(j)) \triangleq \min_{u_j} \|Qx_j\|_2 + \|Ru_j\|_2 + J^*_{j+1}(\tilde{f}_{PWA}(x(j), u_j))$$  \hspace{1cm} (61)

subj. to $\tilde{f}_{PWA}(x(j), u_j) \in X_{j+1}$  \hspace{1cm} (62)

for $j = N-1, \ldots, 0$, with terminal conditions

$$X_N = X_f$$

$$J^*_N(x) = \|Px\|_2,$$  \hspace{1cm} (63) \hspace{1cm} (64)

where $X_j$ is the set of all states $x(j)$ for which problem (61)–(62) is feasible:

$$X_j = \{x \in \mathbb{R}^n \mid \exists u, \tilde{f}_{PWA}(x, u) \in X_{j+1}\}.$$  \hspace{1cm} (65)

Equations (61)-(65) are the discrete-time version of the well known Hamilton-Jacobi-Bellman equations for continuous-time optimal control problems.

Assume for the moment that there are no binary inputs and logic states, $m_{\ell} = n_{\ell} = 0$. The HJB equations (61)–(64) can be solved backwards in time by using a multiparametric quadratic programming solver and the results of the previous section.

Consider the first step of the dynamic program (61)–(64)

$$J^*_{N-1}(x(N-1)) \triangleq \min_{u_{N-1}} \|Qx(N-1)\|_2 + \|Ru(N-1)\|_2 +$$

$$+ J^*_N(\tilde{f}_{PWA}(x(N-1), u_{N-1}))$$  \hspace{1cm} (66)

subj. to $\tilde{f}_{PWA}(x(N-1), u_{N-1}) \in X_f$.  \hspace{1cm} (67)

The cost to go function $J^*_N(x)$ in (66) is quadratic, the terminal region $X_f$ is a polyhedron and the constraints are piecewise affine. Problem (66)–(67) is an $s$ to one problem that can be solved by solving $s$ mp-QPs (Lemma 5).

From the second step $j = N - 2$ to the last one $j = 0$ the cost to go function $J^*_{j+1}(x)$ is PPWQ with a certain multiplicity $d_{j+1}$, the terminal region $X_{j+1}$ is a polyhedron (not necessary convex) and the constraints are piecewise affine. Therefore, problem (61)–(64) is an $s$ to $N_{j+1}$ problem with multiplicity $d_{j+1}$ (where $N_{j+1}$ is the number of polyhedra of the cost to go function $J^*_{j+1}$), that can be solved by solving $s N_{j+1} d_{j+1}$ mp-QPs (Lemma 7). The resulting optimal solution will have the form (24) considered in the ordered region sense.

In the presence of binary inputs the procedure can be repeated, with the difference that all the possible combinations of binary inputs have to be enumerated. Therefore a one to one problem becomes a $2^{m_{\ell}}$ to one problem and so on. Clearly the procedure becomes prohibitive for a system with a high number

\[29\]
of binary inputs. In this case multiparametric mixed integer programming (or a combination of the two techniques) may be preferable. The multiparametric programming approach described in Section 8.2 and the dynamic programming approach have been compared in [3].

In the presence of logic states the procedure can be repeated either by relaxing the logic states to assume continuous values between 0 and 1 or by enumerating them all. For a large number of binary states and large horizons the mixed integer programming approach described in Section 8.1 could be the only one viable to compute optimal control actions.

9.4 Examples

Example 5.1 - Finite Time Optimal Control ($\infty$-norm)

Consider the problem of steering the piecewise affine system (10) to a small region around the origin in three steps while minimizing the cost function (18). The finite-time constrained optimal control problem (18)-(23) is solved with $p = \infty$, $N = 3$, $P = 0$, $Q = 700I$, $R = 1$, and $X_f = [-0.01 0.01] \times [-0.01 0.01]$. The solution was computed with the algorithm presented in Section 9 in 71 s with Matlab 6.1 on a Pentium IV 2.2 GHz machine. The polyhedral regions corresponding to the state feedback solution $u^*_k(x(k))$, $k = 0, 2, 2$ in (32) are depicted in Fig. 10. The state feedback control law at time 0 comprises 106 polyhedral regions.

![Figure 10: State space partition corresponding to the state feedback finite time optimal control law $u^*(x(k))$ of system (10).](image)

Example 5.1 - Finite Time Optimal Control (2-norm)

Consider the problem of steering the piecewise affine system (10) to a small region around the origin in three steps while minimizing the cost function (18). The finite-time constrained optimal control
problem (18)-(23) is solved with $p = 2$, $N = 3$, $Q = I$, $R = 1$, $P = I$, and $X_f = [-0.01 \ 0.01] \times [-0.01 \ 0.01]$. The solution is computed with the algorithm presented in Section 9 in 29 s with Matlab 6.1 on a Pentium IV 2.2 GHz machine. The polyhedral regions corresponding to the state feedback solution $u^*_k(x(k))$, $k = 0, 1, 2$ in (24) are depicted in Fig. 11. The state feedback control law at time 0 comprises of 77 polyhedral regions, and some of them have multiplicity 2 (note that the enumeration of all possible switching sequences could lead to a multiplicity of $2^3$ in all regions).

Figure 11: State space partition corresponding to the state feedback finite time optimal control law $u^*(x(k))$ of system (10).

As explained in Section 9.2 the optimal control law is stored in a special data structure where:

1. The ordering of the regions is important.

2. The polyhedra can overlap.

3. The polyhedra can have an associated value function of multiplicity $s > 1$. Thus, in general $s$ quadratic functions have to be compared on-line in order to evaluate the optimal control action.

**Example 5.2 - Finite Time Optimal Control**

We solved the finite-time constrained optimal control problem (18)-(23) for system (11) with $p = 2$, $N = 3$, $Q = I$, $R = 0.01$, $P = Q$, and $X_f = [-10, -10] \times [10, 10]$. The equilibrium point was chosen as $x_e = [1, 0]$ and $u_e = 2$. The solution was computed with the algorithm presented in Section 9 in less than 5 minutes with Matlab 5.3 on a Pentium II-500 MHz. The state feedback control law at time 0 comprises of 78 polyhedral regions, the maximum multiplicity is 2.
Example 5.3 - Finite Time Optimal Control

The steady-state value of the continuous state $x_1$ of system (15) is a function of the logic state. For such a case a formulation of the optimal control problem in terms of the variation of the input is more appropriate:

$$ J^*(x(0)) \triangleq \min_{\Delta U_N} \| P x_N \|_p + \sum_{k=0}^{N-1} \| Q (x_k - x_e) \|_p + \| R \delta u_k \|_p $$  \hspace{1cm} (68)

subject to

$$ \begin{cases} 
  x_{k+1} = A^i x_k + B^i u_k + f^i \text{ if } [u_k] \in \tilde{P}^i \\
  x_N \in X_f \\
  x_0 = x(0) \\
  u_k = u_{k-1} + \delta u_k \\
  u_{-1} = 0.
\end{cases} $$  \hspace{1cm} (69)

In the problem above no reference for the input is needed and the controller will automatically decide the “cheapest” logic state and corresponding steady-state input. The reference $x_e$ can be considered as an additional constant state over the horizon $N$. The piecewise optimal controller (24) will be defined in a space of dimension four, i.e. the two original states, the reference on the continuous state $x_3(t) = x_e$ and the previous input $x_4(t) = u(t-1)$.

We solved the finite-time constrained optimal control problem (18)-(23) for system (15) with $p = 2$, $N = 3$, $Q = I$, $R = 0.1$, $P = I$, $X_f = [-10, -10] \times [10, 10]$ with the additional constraints $-10 \leq \delta u(t) \leq 10$ and $-10 \leq x_e \leq 10$. The state feedback control law at time 0 comprises of 78 polyhedral regions, the maximum multiplicity is 2.

10 Receding Horizon Control

An infinite horizon controller can be obtained by implementing in a receding horizon fashion a finite-time optimal control law. In this case the control law is simply obtained by repeatedly evaluating at each time $t$ the PWA controller (24) for $k = 0$:

$$ u(t) = u^*_0(x(t)) \text{ for } x(t) \in X_0. $$  \hspace{1cm} (70)

In general the control law (70) is not stabilizing. For stability and feasibility of the closed-loop system (7), (70) we refer the reader to previously published results on RHC, see e.g. [40, 9], although general stability criteria for the case of logic states and logic inputs, to the best of our knowledge, have appeared only in [9].

Example 5.2 - Receding Horizon Control

Consider the problem of regulating the piecewise affine system described in Example (5.2) to the equilibrium point $x_e = [1, 0]'$, $u_e = 2$. We implemented the state feedback solution computed above in a receding horizon fashion, i.e. $u(x(t)) = u^*_0(x(t))$. In Figure 12 we show the corresponding closed-loop trajectories starting from the initial state $x(0) = [0, 0]'$. The controller behaves as expected: First, the fast dynamics is chosen and then, as soon as the state approaches the reference, the system is switched to the slow dynamics, to avoid large overshoots.
Example 5.3 - Receding Horizon Control

We implemented the state feedback solution (24) computed for system (15) in a receding horizon fashion. The corresponding closed-loop trajectories starting from the initial state $x(0) = [0 0]'$ are reported in Figure 13 ($x_e = 5$) and Figure 14 ($x_e = 9$). Note that if the reference is high enough the controller first goes down to the value $x_{lb}$. This cause the change of the logic state that implies a higher input coefficient and therefore faster dynamics. If the reference is not high enough no switch in the logic state will occur.

Figure 13: Receding horizon control of system (15)
11 Application Examples

In collaboration with different companies and institutes, the results described in the previous chapters have been applied to a wide range of problems.

Constrained optimal control for hybrid systems was used for the control of co-generation power plants [26], of a benchmark multi-tank system [41], of a gas supply system [9], of a gear shift operation on automotive vehicles [52, 5], of the direct injection stratified charge engine [8], of the integrated management of the power-train [5], of a traction control problem [13], of the electronic throttle [4] and in the multi-objective adaptive cruise control [42].

We point out that the associated feedback solution was not computed in all the above mentioned problems. In some cases it was deemed to complex, and in others it was not needed. In general for problems of high dimension with a high sampling time, on line optimization is preferable. Nevertheless, in those cases when the explicit solution was computed the size of the systems varied on a wide range. Sometimes the number of switched dynamics was only 2, while in other cases more than 10 dynamics were used. Similar variation of values was present in the dimension of the state vector (2 – 10) and length of the time horizon (2 – 10), while the number of logic inputs was between 0 and 3.

Although there is no general rule for the complexity of the optimal PWA controller (a huge problem could have a very simple PWA solution), in most applications we observe that larger number of binary inputs and/or large number of PWA dynamics may lead to the significant increase of the complexity of the PWA solution. In our examples whenever one of the critical parameters - dimension of the problem, time horizon, number of dynamics or number of binary inputs - was large (at its upper limit), other parameters were chosen small. Otherwise problem was too complex to solve.

12 Appendix

Here we report HYSDEL list for Spring-Mass System in Example 5.2:
/* Sprin-Mass System
* HYSDEL v2 description
*/

SYSTEM springmass {

INTERFACE { /* Description of variables and constants */

STATE {
    REAL x1;
    REAL x2;
}

INPUT { REAL u1;
        BOOL u2;
}

PARAMETER {

    /* Break Point */
    REAL bpm = 1;

    /* Other parameters */
    REAL zxmin = -100;
    REAL zxmax = 100;
    REAL x1min = -100;
    REAL x1max = 100;
    REAL e = 1e-6;

    /* Dynamic behaviour of the model (Matlab Generated) */
    REAL A11=0;
    REAL A12=1;
    REAL A1=0;
    REAL A21a=-0.25;
    REAL A22a=-0.25;
    REAL A21b=-0.25;
    REAL A22b=-12.5;
    REAL A21c=-0.75;
    REAL A22c=-0.25;
    REAL A21d=-0.75;
    REAL A22d=-12.5;
    REAL B2a=0.25;
    REAL B2b=0.25;
    REAL B2c=0.25;
    REAL B2d=0.25;

    }
}

IMPLEMENTATION {

}
In the following we report HYSDEL list for Simple System with a Logic State in Example 5.3:

/* System with logic state
* HYSDEL v2 description
*/

SYSTEM SLS {

INTERFACE { /* Description of variables and constants */

    STATE {
        REAL x1;
        BOOL x2;
    }

    INPUT { REAL u;
    }

    PARAMETER {

/* Lower Bound Point */
REAL xlb = -1;

/* Other parameters */
REAL x1min = -10;
REAL x1max = 10;
REAL e = 1e-6;

/* Dynamic behaviour of the model */
REAL A = 1.00000;
REAL B1 = 1.00000;
REAL B2 = 1.00000;

IMPLEMENTATION {
  AUX {
    BOOL region;
    REAL zx1;
  }
  AD {
    /* PWA Region */
    region = x1-xlb <= 0 [x1min, x1max, e];
  }
  DA { zx1={IF x2 THEN A*x1+B2*u [x1min, x1max, e] ELSE A*x1+B1*u [x1min, x1max, e]};
  }
  CONTINUOUS { x1=zx1;
  }
  AUTOMATA { x2=(x2) | region;
  }
  MUST { x1<=10;
    x1>=-10;
  }
}

Acknowledgements

This work was partially supported by the Swiss National Science Foundation and the European Project “Computation and Control”.

We thank David Q. Mayne, Christos G. Cassandras and Nicolo’ Giorgetti for constructive and helpful comments on the original manuscript.
References


