SIAM J. CONTROL OPTIM. © 2008 Society for Industrial and Applied Mathematics
Vol. 47, No. 5, pp. 2470–2489

EFFICIENT ON-LINE COMPUTATION OF CONSTRAINED OPTIMAL CONTROL*

MATO BAOTIĆ†, FRANCESCO BORRELLI‡, ALBERTO BEMPORAD§, AND MANFRED MORARI¶

Abstract. We consider constrained finite-time optimal control problems for discrete-time linear time-invariant systems with constraints on inputs and outputs based on linear and quadratic performance indices. The solution to such problems is a time-varying piecewise affine (PWA) state-feedback law and can be computed by means of multiparametric programming. By exploiting the properties of the value function and the piecewise affine optimal control law of the constrained finite-time optimal control (CFTOC), we propose two new algorithms that avoid storing the polyhedral regions. The new algorithms significantly reduce the on-line storage demands and computational complexity during evaluation of the PWA feedback control law resulting from the CFTOC.

Key words. constrained finite time optimal control, multiparametric programming, piecewise affine function evaluation

AMS subject classifications. 49N05, 93C05, 93C55, 90C05, 90C20, 90C31

DOI. 10.1137/060659314

1. Introduction.

Recently, in [4, 3], the authors have shown how to compute the solution to the constrained finite-time optimal control (CFTOC) problem as a piecewise affine (PWA) state-feedback law. Such a law is computed off-line by using a multiparametric programming solver [4, 7, 13], which divides the state space into polyhedral regions, and for each region determines the linear gain and offset which produces the optimal control action.

This method reveals its effectiveness when a receding horizon control (RHC) strategy is used [14, 15]. RHC requires solving at each sampling time an open-loop CFTOC problem. The optimal command signal is applied to the process only during the sampling interval that follows. At the next time step a new optimal control problem based on new measurements of the state is solved over a shifted horizon. Having a precomputed solution as an explicit PWA function of the state vector reduces the on-line computation of the RHC control law to a function evaluation, thus avoiding the on-line solution of a quadratic or linear program.

The only drawback of such a PWA feedback control law is that the number of polyhedral regions could grow dramatically with the number of constraints in the optimal control problem. In this paper we focus on efficient on-line methods for the evaluation of such a PWA control law. The simplest algorithm would require (i) the storage of the list of polyhedral regions and of the corresponding affine control laws

*Received by the editors May 8, 2006; accepted for publication (in revised form) May 1, 2008; published electronically September 8, 2008. All of the authors were attending the Automatic Control Laboratory, ETH Zurich, Switzerland, when initial research reported in this paper was carried out.
http://www.siam.org/journals/sicon/47-5/65931.html

†Corresponding author. Automatic Control Laboratory, ETH Zentrum - ETL, Physikstrasse 3, CH-8092 Zürich, Switzerland and Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, HR-10000 Zagreb, Croatia (mato.baotic@fer.hr).

‡Department of Mechanical Engineering, University of California, 5132 Etcheverry Hall, Berkeley, CA 94720-1740 (fborrelli@me.berkeley.edu).

§Dip. Ingegneria dell'Informazione, University of Siena, I-53100 Siena, Italy (bemporad@unisi.it).

¶Automatic Control Laboratory, ETH Zentrum - ETL, Physikstrasse 3, CH-8092 Zürich, Switzerland (morari@control.ee.ethz.ch).

and (ii) a sequential search through the list of polyhedra for the ith polyhedron that contains the current state in order to implement the ith control law. By exploiting the properties of the value function and the optimal control law, for CFTOC problems based on linear programming (LP) and quadratic programming (QP), we propose two new algorithms that avoid storing the polyhedral regions. The new algorithms significantly reduce the on-line storage demands and computational complexity during evaluation of the explicit solution of the CFTOC problem.

The same problem has been recently approached in a different manner in [23]. The algorithm proposed there—with the controller gains of the PWA control law organized on a balanced search tree—is less efficient in terms of memory requirements, but has a logarithmic average computation complexity. At the expense of the optimality of the solution a similar computational complexity can be achieved with an approximate point location algorithm described in [12].

Several papers also propose the use of fast solvers for the on-line solution of constrained predictive control problems (cf. [16, 10]); these algorithms pursue the same goal as this paper, namely, to reduce the on-line computational burden in RHC, but from a different perspective. They use fast LP or QP solvers (in place of a general-purpose solver) tailored to the special dynamic structure of the underlying optimal control problem [16]. Note that a proper comparison of the proposed algorithms with "fast" on-line LP and QP solvers requires the simultaneous analysis of several issues such as speed of computation, storage demand, and real time code verifiability. This is an involved study and as such is outside the scope of this paper.

The paper is organized as follows. For discrete-time linear time-invariant systems the basics of CFTOC problems and of RHC are summarized in section 2. In section 3, for LP-based and QP-based optimal control we present two new algorithms to evaluate on-line explicit optimal control laws and compare their complexity in terms of time and storage against the simplest algorithm mentioned above. Finally, in section 4 an example is given that confirms the efficiency of the new methods.

2. CFTOC, RHC, and their state-feedback PWA solution.

Throughout this paper (lower and upper case) italic letters denote scalars, vectors, and matrices (e.g., $A, a, \dots$), while upper case calligraphic letters denote sets (e.g., $\mathcal{A}, \mathcal{B}, \dots$). For a matrix (vector) A, A' denotes its transpose, while $A_{(i)}$ denotes the ith row (element), $\mathbb{R}$ is the set of real numbers, and $\mathbb{N}$ is the set of positive integer numbers.

2.1. CFTOC problem formulation.

Consider the discrete-time linear time-invariant system

$$x(t+1) = Ax(t) + Bu(t) \tag{1}$$

subject to the constraints

$$E^x x(t) + E^u u(t) \leq E \tag{2}$$

at all time instants $t \geq 0$.

In (1)–(2), $n_x \in \mathbb{N}$, $n_u \in \mathbb{N}$, and $n_E \in \mathbb{N}$ are the number of states, inputs, and constraints; respectively, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ is the input vector, $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $E^x \in \mathbb{R}^{n_E \times n_x}$, $E^u \in \mathbb{R}^{n_E \times n_u}$, $E \in \mathbb{R}^{n_E}$, the pair (A, B) is stabilizable, and the vector inequality (2) is considered elementwise.

Let $x_0 = x(0)$ be the initial state and consider the constrained finite-time optimal control problem

$$
\begin{aligned}
J^*(x_0) := \min_U \quad & J(x_0, U) \\
\text{s.t.} \quad & \begin{cases} x_{k+1} = Ax_k + Bu_k, \\ E^x x_k + E^u u_k \le E, \quad k = 0, \dots, N-1, \end{cases}
\end{aligned} \tag{3}
$$

where $N \in \mathbb{N}$ is the horizon length, $U := [u_0', \dots, u_{N-1}']' \in \mathbb{R}^{n_u N}$ is the optimization vector, x_i denotes the state at time i if the initial state is x_0 and the control sequence $\{u_0, \dots, u_{i-1}\}$ is applied to the system (1), $J^* : \mathbb{R}^{n_x} \to \mathbb{R}$ is the value function, and the cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u N} \to \mathbb{R}$ is given either as a piecewise linear function (i.e., sum of l_1 or l_∞ norms)

$$
J(x_0, U) = \|Q^{x_N} x_N\|_\ell + \sum_{k=0}^{N-1} \|Q^x x_k\|_\ell + \|Q^u u_k\|_\ell, \quad \ell \in \{1, \infty\}, \tag{4a}
$$

or as a quadratic function

$$
J(x_0, U) = x_N' Q^{x_N} x_N + \sum_{k=0}^{N-1} x_k' Q^x x_k + u_k' Q^u u_k. \tag{4b}
$$

In the following, we will assume that Q^x, Q^u, Q^{x_N} are full column rank matrices when the cost function (4a) is used, and that $Q^x = (Q^x)' \succeq 0$, $Q^u = (Q^u)' \succ 0$, $Q^{x_N} \succeq 0$, when the cost function (4b) is used, where $Q \succ 0$ denotes positive definiteness (resp., $Q \succeq 0$ positive semidefiniteness).

The optimization problem (3) can be translated into a linear program (LP) when the piecewise linear cost function (4a) is used [3] or into a quadratic program (QP) when the quadratic cost function (4b) is used [4]. We denote by $U^* = [(u_0^*)', \dots, (u_{N-1}^*)']'$ one of the possible optimizers of problem (3)–(4). An optimizer U^* can be computed by solving an LP or a QP once x_0 is fixed or can be computed explicitly for all x_0 within a given range of values as explained in subsections 2.3 and 2.4.

2.2. RHC strategy. Consider the problem of regulating the discrete-time linear time-invariant system (1) to the origin while fulfilling the constraints (2). The solution U^* to CFTOC problem (3)–(4) is an open-loop optimal control trajectory over a finite horizon. A receding horizon control strategy employs it to obtain a feedback control law in the following way: Assume that a full measurement of the state $x(t)$ is available at the current time $t \ge 0$. Then, the CFTOC problem (3)–(4) is solved at each time t for $x_0 = x(t)$, and

$$
u(t) = u_0^* \tag{5}
$$

is applied as an input to system (1). For a detailed discussion on RHC strategy, see, e.g., [21, 9, 18, 4, 3, 14].

2.3. Solution of CFTOC, linear cost case. Consider the problem (3) with the piecewise linear cost function (4a) and $\ell = \infty$. Using a standard transformation [3], we introduce the vector $v := [u_0', \dots, u_{N-1}', \varepsilon_1^x, \dots, \varepsilon_N^x, \varepsilon_0^u, \dots, \varepsilon_{N-1}^u]' \in \mathbb{R}^{n_v}$, $n_v :=$

$(n_u+2)N$, $\varepsilon_k^x \geq \|Q^x x_k\|_\infty$, $\varepsilon_N^x \geq \|Q^{x_N} x_N\|_\infty$, $\varepsilon_k^u \geq \|Q^u u_k\|_\infty$, and substitute $x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$ in (3)–(4), which can be rewritten as the linear program[1]

$$J^*(x) := \min_v \quad c'v \qquad \text{s.t.} \quad L^v v \leq L + L^x x, \tag{6}$$

where $x = x_0$, and matrices $c \in \mathbb{R}^{n_v}$, $L^v \in \mathbb{R}^{n_L \times n_v}$, $L^x \in \mathbb{R}^{n_L \times n_x}$, $L \in \mathbb{R}^{n_L}$ are easily obtained from Q^x, Q^u, Q^{x_N} and (3)–(4), as explained in [3]. For a given x we denote with $\mathcal{V}^*(x)$ the set of optimizers for the problem (6). Note that, in general, $\mathcal{V}^*(x)$ is a set valued function, i.e., $\mathcal{V}^* : \mathbb{R}^{n_x} \to 2^{\mathbb{R}^{n_v}}$.

Because the problem depends on x the implementation of RHC can be performed in two different ways: solve the LP (6) on-line at each time step for a given x or solve (6) *off-line* for all x within a given range of values, i.e., by considering (6) as a *multiparametric linear program* (mp-LP) [11].

Solving an mp-LP means computing the value function $J^*(x) : \mathbb{R}^{n_x} \to \mathbb{R}$ and one (out of possibly many) optimizer function $v^*(x) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_v}$ *for all* possible vectors x in a given set $\mathcal{X}$. The solution to mp-LP problems can be approached simply by exploiting the properties of the primal and dual optimality conditions as proposed in [7, 11].

In [11] the following results about the properties of the solution are proved.

THEOREM 1. *Consider the mp-LP* (6)*. Then the set of feasible parameters $\mathcal{X}_f$ is convex. The value function $J^* : \mathcal{X}_f \to \mathbb{R}$ is convex and piecewise affine. There always exists a continuous and PWA selection of an optimizer function $v^* : \mathcal{X}_f \to \mathbb{R}^{n_v}$. In particular, if the optimizer $\mathcal{V}^*(x)$ is unique for all $x \in \mathcal{X}_f$, then $v^*(x) = \mathcal{V}^*(x)$.*

Once the multiparametric problem (6) has been solved off-line for a polyhedral set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ of states, the explicit solution $v^*(x)$ of CFTOC problem (6) is available as a PWA function of x, and the receding horizon controller (3)–(5) is also available explicitly, as the optimal input $u(t)$ consists simply of n_u components of $v^*(x(t))$,

$$u(t) = [I_{n_u}\ 0\ \cdots\ 0] v^*(x(t)). \tag{7}$$

COROLLARY 1. *The RHC* (7)*, defined by the optimization problem* (3), (4a)*, and* (5)*, is a continuous and piecewise affine function, $u : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$, and has the form*

$$u(x) = F_i x + G_i \quad \forall x \in \mathcal{P}_i, \quad i = 1, \ldots, N_{\mathcal{P}}, \tag{8}$$

where $F_i \in \mathbb{R}^{n_u \times n_x}$, $G_i \in \mathbb{R}^{n_u}$, $\{\mathcal{P}_i\}_{i=1}^{N_\mathcal{P}}$ is a polyhedral partition of $\mathcal{X}_f$ (i.e., $\cup_i \mathcal{P}_i = \mathcal{X}_f$, and $\mathcal{P}_i$ and $\mathcal{P}_j$ have disjoint interiors $\forall i \neq j$), with $\mathcal{P}_i = \{x \in \mathbb{R}^{n_x} \mid P_i^x x \leq P_i\}$, $P_i^x \in \mathbb{R}^{p_i \times n_x}$, $P_i \in \mathbb{R}^{p_i}$, and p_i is the number of halfspaces defining polyhedron $\mathcal{P}_i$, $i = 1, \ldots, N_\mathcal{P}$.

In the rest of this paper we will assume, without loss of generality, that $\mathcal{P}_i$ in (8) and $i = 1, \ldots, N_\mathcal{P}$, are full dimensional sets in $\mathbb{R}^{n_x}$ corresponding to the so-called critical regions of the optimization problem (6) (see [7] for more details). We will denote with $N_\mathcal{H}$ the total number of halfspaces defining the polyhedral partition of $\mathcal{X}_f$,

$$N_\mathcal{H} := \sum_{i=1}^{N_\mathcal{P}} p_i. \tag{9}$$

[1]The same holds for $\ell = 1$ with a different optimization vector [3].

Remark 2.1. Typically the total number of halfspaces defining polyhedral partition of feasible set $\mathcal{X}_f$ is much bigger than the number of polyhedral regions in it, i.e., $N_{\mathcal{H}} \gg N_{\mathcal{P}}$. The reasoning is the following. Assume, as is the case in practical applications, that all $\mathcal{P}_i$ are bounded. Since the smallest number of halfspaces defining a bounded polyhedron in $\mathbb{R}^{n_x}$ is $n_x + 1$ (achieved by a simplex), we have $N_{\mathcal{H}} \geq (n_x + 1)N_{\mathcal{P}}$.

2.4. Solution of CFTOC, quadratic cost case. Consider the problem (3) with the quadratic cost function (4b). By substituting $x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$ in (3)–(4), this can be rewritten as the quadratic program

$$\begin{aligned} J^*(x) = \tfrac{1}{2}x'Yx \ + \min_U \quad & \tfrac{1}{2}U'HU + x'FU \\ \text{s.t.} \quad & M^U U \leq M + M^x x, \end{aligned} \tag{10}$$

where $x = x_0$, the column vector $U := [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^{n_U}$, $n_U := n_u N$, is the optimization vector, $H = H' \succ 0$, and H, F, Y, M^U, M^x, M are easily obtained from Q^x, Q^u, Q^{x_N} and (3)–(4) (see [4] for details).

As in the linear cost case, because the problem depends on x, the implementation of RHC can be performed by either solving the QP (10) on-line or, as shown in [4, 22], by solving problem (10) *off-line* for all x within a given range of values, i.e., by considering (10) as a *multiparametric quadratic program* (mp-QP).

Once the multiparametric problem (10) is solved off-line, i.e., the solution $U^*(x)$ of the CFTOC problem (10) is found, the state-feedback PWA RHC law is simply

$$u(t) = [I_{n_u} \ 0 \ \cdots \ 0] U^*(x(t)). \tag{11}$$

In [4] the authors give a self-contained proof of the following properties of the solution.

THEOREM 2. *Consider the multiparametric quadratic program* (10)*, and let $H \succ 0$. Then the set of feasible parameters $\mathcal{X}_f$ is convex, the optimizer $U^* : \mathcal{X}_f \to \mathbb{R}^s$ is continuous and piecewise affine, and the value function $J^* : \mathcal{X}_f \to \mathbb{R}$ is continuous, convex, and piecewise quadratic.*

The proof of the properties listed in Theorem 2 can be found in [5, "Maximum Theorem" on page 116]. It also follows from [1, Theorem 3.2.1(I) and Theorem 3.3.3].

COROLLARY 2. *The RHC control law* (11)*, defined by the optimization problem* (3)*,* (4b)*, and* (5)*, is continuous and piecewise affine and has the form* (8).

The optimization problem (10), where $\mathcal{X}_f$ is a lower dimensional set, can be dealt with in the same way as in the linear cost case (see [7] for details). Hence, in the following we will assume, without loss of generality, that $\mathcal{P}_i$, $i = 1, \ldots, N_{\mathcal{P}}$, are full dimensional sets in $\mathbb{R}^{n_x}$ corresponding to the so-called critical regions of the optimization problem (10); cf. [4].

Corollaries 1 and 2 state that by using a multiparametric solver the computation of RHC action becomes a simple PWA function evaluation. In the next section we propose a method to efficiently evaluate such a PWA function without storing the polyhedral regions $\mathcal{P}_i$, $i = 1, \ldots, N_{\mathcal{P}}$.

3. Efficient on-line algorithms. The on-line implementation of the control law (8) is executed simply according to the following algorithm.

```
ALGORITHM 1.
1. Measure the current state $x(t)$
2. Search for the $i$th polyhedron that contains $x(t)$, $(P_i^x x(t) \leq P_i)$
3. Implement the $i$th control law $(u(t) = F_i x(t) + G_i)$
```

In Algorithm 1, step 2 is critical and is the only step whose efficiency can be improved. A simple implementation of step 2 would consist of searching for the polyhedral region that contains the state $x(t)$ as in the following algorithm.

```
ALGORITHM 2.
1. $i = 0$, notfound=TRUE
2. WHILE $i \leq N_{\mathcal{P}}$ AND notfound
  2.1. $j = 0$, feasible=TRUE
  2.2. WHILE $j \leq p_i$ AND feasible
      2.2.1. IF $(P_i^x)_{(j)} x(t) > (P_i)_{(j)}$ THEN feasible=FALSE
  2.3. END
  2.4. IF feasible THEN notfound=FALSE
3. END
```

Recalling the expression (9) for $N_{\mathcal{H}}$ (the total number of halfspaces defining the polyhedral partition of the feasible set $\mathcal{X}_f$), it is easy to see that Algorithm 2 requires $(n_x + 1)N_{\mathcal{H}}$ real numbers to store all polyhedra $\mathcal{P}_i$, and in the worst case (when the state is contained in the last region of the list) Algorithm 2 will give a solution after $n_x N_{\mathcal{H}}$ multiplications, $(n_x - 1)N_{\mathcal{H}}$ sums, and $N_{\mathcal{H}}$ comparisons.

Remark 3.1. In the algorithms presented in the following sections we implicitly assume that $x(t)$ belongs to the feasible set $\mathcal{X}_f$. If this (reasonable) assumption does not hold, then we should include a set of *boundaries* of feasible parameter space $\mathcal{X}_f$, and we should (before using any of proposed algorithms) first check if the point $x(t)$ is inside the boundaries of $\mathcal{X}_f$. Note that such a step is not needed for Algorithm 2 since there we automatically detect if the point $x(t)$ is outside of the feasible set $\mathcal{X}_f$.

By using the properties of the value function, we will show how Algorithm 2 can be replaced by more efficient algorithms that have less computational complexity and *avoid storing the polyhedral regions* $\mathcal{P}_i$, $i = 1, \dots, N_{\mathcal{P}}$, therefore reducing the storage demand significantly.

In the following we will distinguish between optimal control based on LP and optimal control based on QP.

3.1. Efficient implementation, linear cost case.

From Theorem 1, the value function $J^*(x)$ corresponding to the solution of the CFTOC problem (3) with the piecewise linear cost (4a) is convex and PWA:

$$J^*(x) = T_i' x + V_i \quad \forall x \in \mathcal{P}_i, \quad i = 1, \dots, N_{\mathcal{P}}, \tag{12}$$

where $T_i \in \mathbb{R}^{n_x}$, $V_i \in \mathbb{R}$.

By exploiting the convexity of the value function the storage of the polyhedral regions $\mathcal{P}_i$ can be avoided. From the equivalence of the representations of PWA convex functions (cf. [17], [8, page 80]) the function $J^*(x)$ in (12) can be represented alternatively as

$$J^*(x) = \max \{T_i' x + V_i\}_{i=1}^{N_{\mathcal{P}}} \quad \text{for } x \in \mathcal{X}_f = \cup_{i=1}^{N_{\mathcal{P}}} \mathcal{P}_i. \tag{13}$$

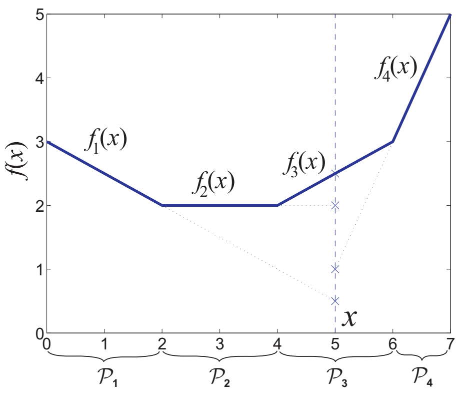


Fig. 1. *Example for Algorithm* 3 *in one dimension: For a given point* $x \in \mathcal{P}_3$ $(x = 5)$, *we have* $f_3(x) = \max(f_1(x), f_2(x), f_3(x), f_4(x))$.

Table 1
Complexity comparison of Algorithms 2 *and* 3.

	Algorithm 2	Algorithm 3
Storage demand (real numbers)	$(n_x + 1)N_{\mathcal{H}}$	$(n_x + 1)N_{\mathcal{P}}$
Number of flops (worst case)	$2n_x N_{\mathcal{H}}$	$2n_x N_{\mathcal{P}}$

From (12) and (13), the polyhedral region $\mathcal{P}_j$ containing x can be identified simply by searching for the maximum number in the list $\{T_i' x + V_i\}_{i=1}^{N_\mathcal{P}}$,

$$x \in \mathcal{P}_j \Leftrightarrow T_j' x + V_j = \max \{T_i' x + V_i\}_{i=1}^{N_\mathcal{P}}. \tag{14}$$

Therefore, instead of searching for the polyhedron j that contains the point x via Algorithm 2, we can just store the value function and identify region j by searching for the maximum in the list of numbers composed of the single affine function $T_i' x + V_i$ evaluated at x.

Algorithm 3.

```
1. Compute the list $\mathcal{T} = \{t_i := T_i' x + V_i\}_{i=1}^{N_\mathcal{P}}$
2. Find $i$ such that $t_i = \max_{t_j \in \mathcal{T}} t_j$
```

For illustration, see the example in Figure 1, where we have $N_\mathcal{P} = 4$, $f_1(x) = -0.5x+3$, $f_2(x) = 2$, $f_3(x) = 0.5x$, and $f_4(x) = 2x - 9$.

Algorithm 3 requires the storage of $(n_x + 1)N_\mathcal{P}$ real numbers and will give a solution after $n_x N_\mathcal{P}$ multiplications, $(n_x - 1)N_\mathcal{P}$ sums, and $N_\mathcal{P} - 1$ comparisons. In Table 1 we compare the complexity of Algorithm 3 against Algorithm 2 in terms of storage demand and number of flops.

Remark 3.2. Algorithm 3 will outperform Algorithm 2 since typically the total number of halfspaces defining polyhedral partition of feasible set $\mathcal{X}_f$ is much larger than the number of polyhedral regions, i.e., $N_\mathcal{H} \gg N_\mathcal{P}$ (see Remark 2.1).

Remark 3.3. Note that Algorithm 3 cannot be applied if the solution to (6) contains dual degenerate regions, i.e., two regions $\mathcal{P}_i$ and $\mathcal{P}_j$ that have the same cost expressions but different control expressions (i.e., $[T_i' \; V_i] = [T_j' \; V_j]$, $[F_i' \; G_i] \neq [F_j' \; G_j]$). If dual degeneracy occurs one can use Algorithm 4 combined with the procedure described in section 3.2.2. An alternative approach consists of modifying Algorithm 3 in order to be able to discern between dual degenerate regions (e.g., by means of Algorithm 2).

3.2. Efficient implementation, quadratic cost case. Consider the explicit solution of CFTOC problem (3) with the quadratic cost (4b). Theorem 2 states that the value function $J^*(x)$ is convex and piecewise quadratic and the simple Algorithm 3 described in the previous subsection cannot be used here. Instead, a modified approach is described below.

We will first establish the following general result: given a general polyhedral partition of the state space, we can locate where the state lies (i.e., in which polyhedron) by using a search procedure based on the information provided by an "appropriate" PWA continuous function defined over the same polyhedral partition. We will refer to such an "appropriate" PWA function as a *PWA descriptor function*. In the following, first we outline the properties of the PWA descriptor function and then we describe the search procedure itself. In later subsections we will finally show how the gradient of the value function (under certain regularity conditions) and the optimizer (always) can be used for the construction of PWA descriptor functions.

DEFINITION 1. *Two polyhedra $\mathcal{P}_i$, $\mathcal{P}_j$ of $\mathbb{R}^{n_x}$ are called neighboring polyhedra if their interiors are disjoint and $\mathcal{P}_i \cap \mathcal{P}_j$ is $(n_x - 1)$-dimensional (i.e., is a common facet).*

Let $\{\mathcal{P}_i\}_{i=1}^{N_\mathcal{P}}$ be the polyhedral partition obtained by solving the mp-QP (10). For each polyhedra $\mathcal{P}_i$ we denote with $\mathcal{C}_i$ the list of all its neighbors,

$$\mathcal{C}_i := \{j \mid \mathcal{P}_j \text{ is a neighbor of } \mathcal{P}_i,\ j = 1, \dots, N_\mathcal{P}, j \neq i\}, \quad i = 1, \dots, N_\mathcal{P}. \tag{15}$$

In the following, we will assume that every facet is shared by only two neighboring polyhedral regions. This so-called *facet-to-facet* property is almost always satisfied by the solution of the mp-QP (10); cf. [19]. In such a case the list $\mathcal{C}_i$ has at most p_i elements (p_i is the number of halfspaces defining polyhedron $\mathcal{P}_i$) since some of the boundaries of $\mathcal{P}_i$ may be outer boundaries of the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{N_\mathcal{P}}$ and they would not introduce element in the list $\mathcal{C}_i$. We give the following definition of a PWA descriptor function.

DEFINITION 2 (PWA descriptor function). *A scalar* continuous *real-valued PWA function $f : \mathcal{X}_f \to \mathbb{R}$*

$$f(x) = f_i(x) := A_i'x + B_i \quad \text{if} \quad x \in \mathcal{P}_i, \tag{16}$$

with $A_i \in \mathbb{R}^{n_x}$, $B_i \in \mathbb{R}$, is called a descriptor function if

$$A_i \neq A_j \quad \forall j \in \mathcal{C}_i, \quad i = 1, \dots, N_\mathcal{P}, \tag{17}$$

where $\cup_i \mathcal{P}_i = \mathcal{X}_f \subset \mathbb{R}^{n_x}$, and $\mathcal{C}_i$ is the list of neighbors of $\mathcal{P}_i$.

In the following, we will show that the PWA descriptor function defined above has all of the properties we need to be able to locate in which polyhedron the state x lies, because the sign of $f_i(x) - f_j(x)$ changes only when the point x crosses the separating hyperplane between $\mathcal{P}_i$ and $\mathcal{P}_j$. Thus for all x in $\mathcal{P}_i$, the difference $f_i(x) - f_j(x)$ has the same sign.

DEFINITION 3 (ordering function). *Let $f(x)$ be a PWA descriptor function on the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{N_\mathcal{P}}$. We define ordering function $O_i(x)$ as*

$$O_i(x) := [O_{i,j}(x)]_{j \in \mathcal{C}_i}, \tag{18}$$

where

$$O_{i,j}(x) := \begin{cases} +1 & \text{if} \quad f_i(x) \geqslant f_j(x), \\ -1 & \text{if} \quad f_i(x) < f_j(x), \end{cases} \tag{19}$$

with $i \in \{1, \dots, N_{\mathcal{P}}\}$, $j \in \mathcal{C}_i$. Note that, for simplicity, we will assume that the order in which the elements of $\mathcal{C}_i$ are used when creating $O_i(x)$ in (18) is uniquely defined. Namely, we use sorted (e.g., in an increasing order) list $\mathcal{C}_i$.

THEOREM 3. *Let $f(x)$ be a PWA descriptor function on the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{N_{\mathcal{P}}}$. Let $\xi_i \in \mathbb{R}^{n_x}$ be any point in the interior of $\mathcal{P}_i$, and define*

$$S_{i,j} := O_{i,j}(\xi_i), \tag{20}$$

$$S_i := O_i(\xi_i), \tag{21}$$

with $i = 1, \dots, N_{\mathcal{P}}$, $j \in \mathcal{C}_i$. Then the following holds:

$$x \in \text{int}(\mathcal{P}_i) \quad \Leftrightarrow \quad O_{i,j}(x) = S_{i,j} \quad \forall j \in \mathcal{C}_i \quad \Leftrightarrow \quad O_i(x) = S_i. \tag{22}$$

Proof. Let $\mathcal{F} = \mathcal{P}_i \cap \mathcal{P}_j$ be the common facet of two neighboring polyhedra $\mathcal{P}_i$ and $\mathcal{P}_j$. Define the linear function

$$g_{i,j}(x) = f_i(x) - f_j(x). \tag{23}$$

From the continuity of descriptor function $f(x)$, it follows that $g_{i,j}(x) = 0\ \forall x \in \mathcal{F}$. As $\mathcal{P}_i$ and $\mathcal{P}_j$ are disjoint convex polyhedra and $A_i \neq A_j$, it follows that $g_{i,j}(x) = 0$ is a separating hyperplane between $\mathcal{P}_i$ and $\mathcal{P}_j$. We have the following.

(i) "$\Rightarrow$" part. Since $g_{i,j}(x) = 0$ is a separating hyperplane between $\mathcal{P}_i$ and $\mathcal{P}_j$ it follows that $g_{i,j}(x)$ does not change sign for all $x \in \text{int}(\mathcal{P}_i)$. Hence, we have $O_{i,j}(x) = S_{i,j}$.

(ii) "$\Leftarrow$" part by contradiction. Assume that $O_{i,j}(\bar{x}) = S_{i,j}$ while $\bar{x} \notin \mathcal{P}_i$. It is easy to see that $\forall \bar{x} \notin \mathcal{P}_i$, $\exists j \in \mathcal{C}_i$ such that $(P_i^x)_{(j)}\bar{x} > (P_i)_{(j)}$. This, however, implies that $g_{i,j}(\bar{x})$ has a different sign compared to $g_{i,j}(\xi)$, $\xi \in \text{int}(\mathcal{P}_i)$. Hence we have $O_{i,j}(\bar{x}) \neq S_{i,j}$, a contradiction. ☐

Theorem 3 states that the ordering function $O_i(x)$ and the vector S_i uniquely characterize $\mathcal{P}_i$. Therefore, to check on-line if the polyhedral region $\mathcal{P}_i$ contains the state x it is sufficient to compute the binary vector $O_i(x)$ and compare it with S_i. Vectors S_i are calculated off-line for $i = 1, \dots, N_{\mathcal{P}}$, by comparing the values of $f_i(x)$ and $f_j(x)\, \forall j \in \mathcal{C}_i$, in a point ξ belonging to $\text{int}(\mathcal{P}_i)$, for instance, the Chebyshev center of $\mathcal{P}_i$.

In Figure 2 a one dimensional example illustrates the procedure with $N_{\mathcal{P}} = 4$ regions and for $f_1(x) = x$, $f_2(x) = 2$, $f_3(x) = x - 3$, $f_4(x) = -\frac{1}{3}x + \frac{19}{3}$. The list of neighboring regions $\mathcal{C}_i$ and the vector S_i can be constructed by simply looking at the figure: $\mathcal{C}_1 = \{2\}$, $\mathcal{C}_2 = \{1, 3\}$, $\mathcal{C}_3 = \{2, 4\}$, $\mathcal{C}_4 = \{3\}$, $S_1 = -1$, $S_2 = [-1\ 1]$, $S_3 = [1\ -1]$, $S_4 = -1$. The point $x = 4$ is in region 2 and we have $O_2(x) = [-1\ 1] = S_2$, while $O_3(x) = [-1\ -1] \neq S_3$, $O_1(x) = 1 \neq S_1$, $O_4(x) = 1 \neq S_4$. The failure of a match $O_i(x) = S_i$ provides information on good search direction(s). The solution can be found by searching in the direction where a constraint is violated, i.e., we should check the neighboring region $\mathcal{P}_j$ for which $O_{i,j}(x) \neq S_{i,j}$.

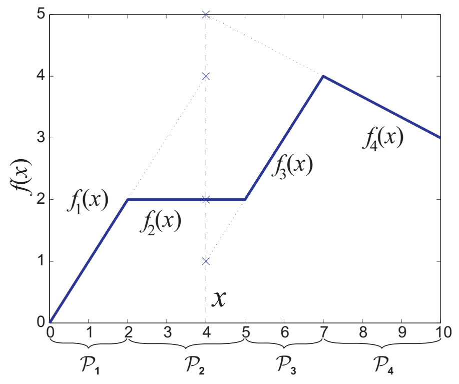


FIG. 2. *Example for Algorithm* 4 *in one dimension: For a given point* $x \in \mathcal{P}_2$ $(x = 4)$ *we have* $O_2(x) = [-1\ 1] = S_2$*, while* $O_1(x) = 1 \neq S_1 = -1$, $O_3(x) = [-1\ \ -1] \neq S_3 = [1\ \ -1]$, $O_4(x) = 1 \neq S_4 = -1$.

The overall procedure is composed of two parts:

1. *(off-line).* Construction of the scalar continuous real-valued PWA function $f(\cdot)$ in (16) satisfying (17), and computation of the list of neighbors $\mathcal{C}_i$ and the vector S_i,
2. *(on-line).* Search for the ith polyhedron that contains $x(t)$ by the execution of the following algorithm.

ALGORITHM 4.

```
1. $\mathcal{I} = \{1, \dots, N_{\mathcal{P}}\}$
2. $i \leftarrow \mathcal{I}$
3. $\mathcal{I} = \mathcal{I} \setminus \{i\}$, $\mathcal{C} = \mathcal{C}_i$
4. WHILE $\mathcal{C} \neq \emptyset$
  4.1. $j \leftarrow \mathcal{C}$, $\mathcal{C} = \mathcal{C} \setminus \{j\}$
  4.2. Compute $O_{i,j}(x)$
  4.3. IF $O_{i,j}(x) \neq S_{i,j}$
      4.3.1. IF $j \notin \mathcal{I}$ THEN GOTO step 2. ELSE $i = j$ and GOTO step 3.
  4.4. END
5. END
```

Algorithm 4 does not require the storage of the polyhedra $\mathcal{P}_i$, but only the storage of one affine function $f_i(x)$ per polyhedron, i.e., $N_{\mathcal{P}}(n_x + 1)$ real numbers and the list of neighbors $\mathcal{C}_i$ which demands (at most) $N_{\mathcal{H}}$ integers. Algorithm 4 in the worst case terminates after $N_{\mathcal{P}} n_x$ multiplications, $N_{\mathcal{P}}(n_x - 1)$ sums, and $N_{\mathcal{H}}$ comparisons.

In Table 2 we compare the complexity of Algorithm 4 against the standard Algorithm 2 in terms of storage demand and the number of flops.

Remark 3.4. Note that the computation of $O_i(x)$ in Algorithm 4 requires the evaluation of p_i linear functions, but the overall computation never exceeds $N_{\mathcal{P}}$ linear function evaluations. Consequently, Algorithm 4 will outperform Algorithm 2, since typically $N_{\mathcal{H}} \gg N_{\mathcal{P}}$.

Now that we have shown how to locate the polyhedron in which the state lies by using a PWA descriptor function, we need a procedure for the construction of such a function.

Table 2
Complexity comparison of Algorithms 2 *and* 4.

	Algorithm 2	Algorithm 4
Storage demand (real numbers)	$(n_x+1)N_{\mathcal{H}}$	$(n_x+1)N_{\mathcal{P}}$
Number of flops (worst case)	$2n_xN_{\mathcal{H}}$	$(2n_x-1)N_{\mathcal{P}}+N_{\mathcal{H}}$

The image of the descriptor function is the set of real numbers $\mathbb{R}$. In the following, we will show how a descriptor function can be generated from a vector valued function $m:\mathbb{R}^{n_x}\to\mathbb{R}^s$. This general result will be used in the next subsections.

Definition 4 (vector valued PWA descriptor function). *A continuous vector valued PWA function $m:\mathcal{X}_f\to\mathbb{R}^s$,*

$$m(x)=\bar{A}_ix+\bar{B}_i \quad \text{if} \quad x\in\mathcal{P}_i, \tag{24}$$

is called a vector valued PWA descriptor function if

$$\bar{A}_i\neq\bar{A}_j \quad \forall j\in\mathcal{C}_i, \quad i=1,\dots,N_{\mathcal{P}}, \tag{25}$$

where $\cup_i\mathcal{P}_i=\mathcal{X}_f\subset\mathbb{R}^{n_x}$, $\bar{A}_i\in\mathbb{R}^{s\times n_x}$, $\bar{B}_i\in\mathbb{R}^s$, $s\in\mathbb{N}$, $s\geq 2$, and $\mathcal{C}_i$ is the list of neighbors of $\mathcal{P}_i$.

Theorem 4. *Let $m:\mathbb{R}^{n_x}\to\mathbb{R}^s$ be a vector valued PWA descriptor function defined over a polyhedral partition $\{\mathcal{P}_i\}_{i=1}^{N_{\mathcal{P}}}$. Then $\exists w\in\mathbb{R}^s$ such that $f(x):=w'm(x)$ is a PWA descriptor function defined over the same polyhedral partition.*

Proof. Let $\mathcal{N}_{i,j}$ be the null-space of $(\bar{A}_i-\bar{A}_j)'$. Since, by definition, $\bar{A}_i-\bar{A}_j\neq\mathbf{0}$, it follows that $\mathcal{N}_{i,j}$ is not full dimensional, i.e., $\mathcal{N}_{i,j}\subseteq\mathbb{R}^{s-1}$. Consequently, it is always possible to find a vector $w\in\mathbb{R}^s$ such that $w'(\bar{A}_i-\bar{A}_j)\neq\mathbf{0}$ holds for all $i=1,\dots,N_{\mathcal{P}}$ and $\forall j\in\mathcal{C}_i$. Clearly, $f(x)=w'm(x)$ is then a valid PWA descriptor function. ☐

As shown in the proof of Theorem 4, once we have vector valued PWA descriptor function, practically any randomly chosen vector $w\in\mathbb{R}^s$ is likely to be satisfactory for the construction of a PWA descriptor function. From a numerical point of view, however, we would like to obtain w, which is as far away as possible from the null-spaces $\mathcal{N}_{i,j}$. We show one algorithm for finding such a vector w.

For a given vector valued PWA descriptor function we form a set of vectors $a_k\in\mathbb{R}^s$, $\|a_k\|=1$, $k=1,\dots,N_a$, by taking and normalizing one (and only one) nonzero column from each matrix $(\bar{A}_i-\bar{A}_j)\,\forall j\in\mathcal{C}_i$, $i=1,\dots,N_{\mathcal{P}}$. Here $N_a:=\sum_i|\mathcal{C}_i|/2\leq N_{\mathcal{H}}$ and $|\mathcal{C}_i|$ denotes cardinality of set $\mathcal{C}_i$. The vector $w\in\mathbb{R}^s$ satisfying the set of equations $w'a_k\neq 0$, $k=1,\dots,N_a$, can then be constructed by using the following algorithm.[2]

Algorithm 5.
```
1. $w \leftarrow [1,\dots,1]'$, $R \leftarrow 1$
2. WHILE $k \leq N_a$
  2.1. $d \leftarrow w'a_k$
  2.2. IF $0 \leq d \leq R$ THEN $w \leftarrow w+\frac{1}{2}(R-d)a_k$, $R \leftarrow \frac{1}{2}(R+d)$
  2.3. IF $-R \leq d < 0$ THEN $w \leftarrow w-\frac{1}{2}(R+d)a_k$, $R \leftarrow \frac{1}{2}(R-d)$
3. END
```

Essentially, Algorithm 5 constructs a sequence of balls $\mathcal{B}=\{x\mid x=w+r,\ \|r\|_2\leq R\}$. As depicted in Figure 3, we start with the initial ball of radius $R=1$,

[2]Index k goes to $N_a:=\sum_i|\mathcal{C}_i|/2$ since the term $(\bar{A}_j-\bar{A}_i)$ is the same as $(\bar{A}_i-\bar{A}_j)$, and thus there is no need to consider it twice.

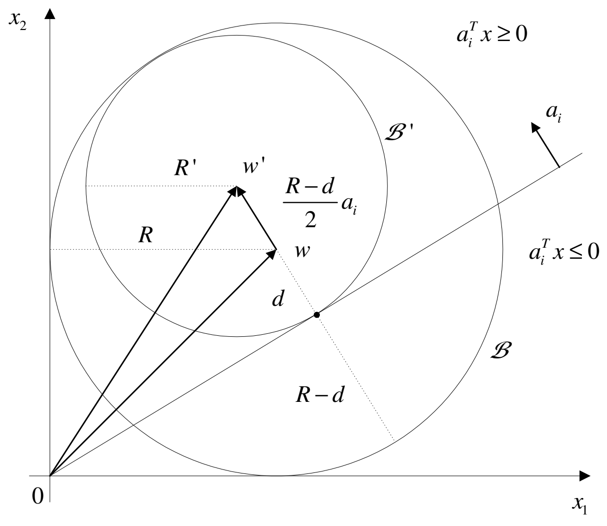


FIG. 3. *Illustration for Algorithm* 5 *in two dimensions.*

centered at $w = [1, \ldots, 1]'$. Iteratively one hyperplane $a_k' x = 0$ at a time is introduced and the largest ball $\mathcal{B}' \subseteq \mathcal{B}$ that does not intersect this hyperplane is constructed. The center w of the final ball is the vector w we want to obtain, while R provides information about the degree of nonorthogonality: $|w' a_k| \geq R \, \forall k$.

In the following subsections we will show that both the gradient of the value function (under certain regularity conditions) and the optimizer (always) are vector valued PWA descriptor functions, and thus we can use Algorithm 5 for the construction of the PWA descriptor function.

3.2.1. Generating a PWA descriptor function from the value function. We will first prove that $J^*(x)$ is a continuously differentiable function whenever the QP problem (10) is not degenarate, and then we will show that the gradient of $J^*(x)$ is a vector valued PWA descriptor function.

Let $J^*(x)$ be the convex and piecewise quadratic (CPWQ) value function corresponding to the explicit solution of CFTOC (3) for the quadratic cost function (4b):

$$J^*(x) = q_i(x) := x' Q_i x + T_i' x + V_i \quad \text{if} \quad x \in \mathcal{P}_i, \; i = 1, \ldots, N_{\mathcal{P}}. \tag{26}$$

Before going further we recall the following result [22].

THEOREM 5. *Consider the set $\mathcal{A}^*(x)$ of active constraints at the optimum of QP* (10),

$$\mathcal{A}^*(x) = \left\{ i \mid M^U_{(i)} U^*(x) = M_{(i)} + M^x_{(i)} x \right\}, \tag{27}$$

and assume there is no degeneracy, i.e., that the rows of $M^U_{(\mathcal{A}^*(x))}$ are linearly independent. Therefore we have the following:

1. *$\mathcal{A}^*$ is constant on the interior of $\mathcal{P}_i$, i.e., $\mathcal{A}^*(x) := \mathcal{A}_i \; \forall x \in \operatorname{int}(\mathcal{P}_i)$, $i = 1, \ldots, N_{\mathcal{P}}$.*
2. *If $\mathcal{P}_i$ and $\mathcal{P}_j$ are neighboring polyhedra, then $\mathcal{A}_i \subset \mathcal{A}_j$ or $\mathcal{A}_j \subset \mathcal{A}_i$.*

DEFINITION 5 (nondegenerate QP). *We say that the QP* (10) *is nondegenerate if, for each $x \in \mathcal{X}_f$, the rows of $M^U_{(\mathcal{A}^*(x))}$ are linearly independent.*

LEMMA 1. *Suppose that the QP problem* (10) *is nondegenerate. Consider the value function $J^*(x)$ in* (26)*, and let $\mathcal{P}_i$, $\mathcal{P}_j$ be two neighboring polyhedra corresponding to the set of active constraints $\mathcal{A}_i$ and $\mathcal{A}_j$, respectively. Then*

$$Q_i - Q_j \preceq 0 \quad \text{or} \quad Q_i - Q_j \succeq 0 \quad \text{and} \quad Q_i \neq Q_j \tag{28}$$

and

$$Q_i - Q_j \preceq 0 \quad \text{iff} \quad \mathcal{A}_i \subset \mathcal{A}_j. \tag{29}$$

Proof. Let $\mathcal{P}_i$ and $\mathcal{P}_j$ be two neighboring polyhedra, and let $\mathcal{A}_i$ and $\mathcal{A}_j$ be the corresponding sets of active constraints at the optimum of QP (10). Let $\mathcal{A}_i \subset \mathcal{A}_j$. We want to prove that the difference between the quadratic terms of $q_i(x)$ and $q_j(x)$ is negative semidefinite, i.e., $Q_i - Q_j \preceq 0$ and $Q_i \neq Q_j$.

Without loss of generality, we can assume that $\mathcal{A}_i = \varnothing$. If this is not the case, then a simple substitution of variables based on the set of active constraints $M^U_{(\mathcal{A}_i)} U = M_{(\mathcal{A}_i)} + M^x_{(\mathcal{A}_i)} x$ transforms problem (10) into a QP in a lower dimensional space.

With the substitution $z = U + H^{-1}F'x$, problem (10) can be translated into the following:

$$\begin{aligned} J_z^*(x) = \min_z \ & \tfrac{1}{2} z' H z \\ \text{s.t. } & Gz \leq W + Sx, \end{aligned} \tag{30}$$

where $G := M^U$, $W := M$, $S := M^x + M^U H^{-1} F'$, and $J_z^*(x) = J^*(x) - \frac{1}{2}x'(Y - FH^{-1}F')x$. For the unconstrained case we have $z^* = 0$ and $J_z^*(x) = 0$. Consequently,

$$q_i(x) = \frac{1}{2} x'(Y - FH^{-1}F')x. \tag{31}$$

For the constrained case, as shown in [4], from the set of active constraints $G_{(\mathcal{A}_j)} z = W_{(\mathcal{A}_j)} + S_{(\mathcal{A}_j)} x$ and the Karush–Kuhn–Tucker (KKT) conditions we obtain

$$z = H^{-1} G'_{(\mathcal{A}_j)} \Gamma^{-1} (W_{(\mathcal{A}_j)} + S_{(\mathcal{A}_j)} x), \tag{32}$$

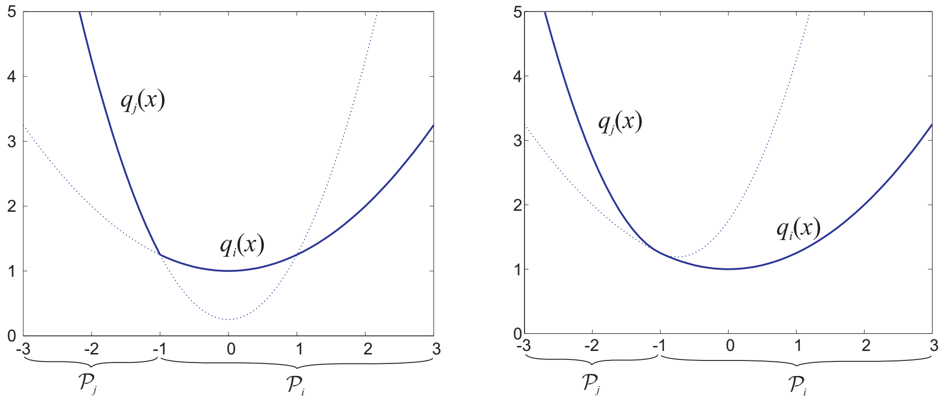
$$\lambda_{(\mathcal{A}_j)} = -\Gamma^{-1} (W_{(\mathcal{A}_j)} + S_{(\mathcal{A}_j)} x), \tag{33}$$

where $\Gamma = G_{(\mathcal{A}_j)} H^{-1} G'_{(\mathcal{A}_j)}$, $\Gamma = \Gamma' \succ 0$ as the rows of $G_{(\mathcal{A}_j)}$ are linearly independent, and $\lambda_{(\mathcal{A}_j)}$ are the Lagrange multipliers of the active constraints $\lambda_{(\mathcal{A}_j)} \geq 0$. The corresponding value function is

$$\begin{aligned} q_j(x) = \ & \frac{1}{2} x'(Y - FH^{-1}F' + S'_{(\mathcal{A}_j)} \Gamma^{-1} S_{(\mathcal{A}_j)}) x \\ & + W'_{(\mathcal{A}_j)} \Gamma^{-1} S_{(\mathcal{A}_j)} x + \frac{1}{2} W'_{(\mathcal{A}_j)} \Gamma^{-1} W_{(\mathcal{A}_j)}. \end{aligned} \tag{34}$$

The difference of the quadratic terms of $q_i(x)$ and $q_j(x)$ gives

$$Q_i - Q_j = -\frac{1}{2} S'_{(\mathcal{A}_j)} \Gamma^{-1} S_{(\mathcal{A}_j)} \preceq 0. \tag{35}$$



(a) not differentiable (b) differentiable

FIG. 4. *Two piecewise quadratic convex functions.*

What is left to prove is that $Q_i \neq Q_j$. We will prove it by showing that $Q_i = Q_j$ if and only if $\mathcal{P}_i = \mathcal{P}_j$. For this purpose we recall [4] that the polyhedron $\mathcal{P}_j$, where the set of active constraints $\mathcal{A}_j$ is constant, is defined as

$$\mathcal{P}_j = \Big\{x \mid GH^{-1}G'_{(\mathcal{A}_j)}\Gamma^{-1}(W_{(\mathcal{A}_j)} + S_{(\mathcal{A}_j)}x) \leq W + Sx,\ -\Gamma^{-1}(W_{(\mathcal{A}_j)} + S_{(\mathcal{A}_j)}x) \geq 0\Big\}. \tag{36}$$

From (35) we conclude that $Q_i = Q_j$ if and only if $S_{(\mathcal{A}_j)} = 0$. The continuity of $J_z^*(x)$ implies that $q_i(x) - q_j(x) = 0$ on the common facet of $\mathcal{P}_i$ and $\mathcal{P}_j$. Therefore, by comparing (31) and (34) we see that $S_{(\mathcal{A}_j)} = 0$ implies $W_{(\mathcal{A}_j)} = 0$. Finally, for $S_{(\mathcal{A}_j)} = 0$ and $W_{(\mathcal{A}_j)} = 0$, from (36) it follows that $\mathcal{P}_j = \mathcal{P}_i := \{x \mid 0 \leq W + Sx\}$. $\square$

The following property of CPWQ functions was proved in [20].

THEOREM 6. *Consider the value function $J^*(x)$ in* (26) *satisfying* (28) *and its quadratic expression $q_i(x)$ and $q_j(x)$ on two neighboring polyhedra $\mathcal{P}_i$, $\mathcal{P}_j$. Then*

$$q_i(x) = q_j(x) + (a'x - b)(\gamma a'x - \bar{b}), \tag{37}$$

where $\gamma \in \mathbb{R}/\{0\}$, $a \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$, $\bar{b} \in \mathbb{R}$.

Equation (37) states that the functions $q_i(x)$ and $q_j(x)$ in two neighboring regions $\mathcal{P}_i, \mathcal{P}_j$ of a CPWQ function satisfying (28) either intersect on two parallel hyperplanes, $a'x - b$ and $\gamma a'x - \bar{b}$ if $\bar{b} \neq \gamma b$ (see Figure 4(a)), or are tangent in one hyperplane, $a'x - b$ if $\bar{b} = \gamma b$ (see Figure 4(b)). We will prove next that if the QP problem (10) is nondegenerate, then $J^*(x)$ is a $C^{(1)}$ function by showing that the case depicted in Figure 4(a) is not consistent with Lemma 1. In fact, Figure 4(a) depicts case $Q^i - Q^j \preceq 0$, that implies $\mathcal{A}_i \subset \mathcal{A}_j$ by Lemma 1. However, $q_j(0) < q_i(0)$ and from the definition of q_i and q_j this contradicts the fact that $\mathcal{A}_i \subset \mathcal{A}_j$.

THEOREM 7. *If the QP problem* (10) *is nondegenerate, then the value function $J^*(x)$ in* (26) *is $C^{(1)}$.*

Proof. To show that J^* is $C^{(1)}$ we need to prove that the cost functions $q_i(x)$ and $q_j(x)$ of any two neighboring regions $\mathcal{P}_i$ and $\mathcal{P}_j$ have the same gradient on the common boundary. It is straightforward to see from (37) that this is indeed the case for $\bar{b} = \gamma b$. We will prove by contradiction that $\bar{b} = \gamma b$. Suppose there exist two neighboring polyhedra $\mathcal{P}_i$ and $\mathcal{P}_j$ such that $\bar{b} \neq \gamma b$. Without loss of generality, assume that (i) $Q_i - Q_j \preceq 0$ and (ii) $\mathcal{P}_i$ is in the halfspace $a'x \leq b$ defined by the common boundary. Let $\mathcal{F}_{ij}$ be the common facet between $\mathcal{P}_i$ and $\mathcal{P}_j$ and relint$(\mathcal{F}_{ij})$ its relative interior.

From both (i) and (37), either $\gamma < 0$ or $\gamma = 0$ if $Q_i - Q_j = 0$. Take $x_0 \in \text{relint}(\mathcal{F}_{ij})$. For sufficiently small $\varepsilon \geq 0$, the point $x := x_0 - a\varepsilon$ belongs to $\mathcal{P}_i$.

Let $J^*(\varepsilon) := J^*(x_0 - a\varepsilon)$, $q_i(\varepsilon) := q_i(x_0 - a\varepsilon)$, and consider that

$$q_i(\varepsilon) = q_j(\varepsilon) + (a'a\varepsilon)(\gamma a'a\varepsilon + (\bar{b} - \gamma b)). \tag{38}$$

From convexity of $J^*(\varepsilon)$, $J^{*-}(\varepsilon) \leq J^{*+}(\varepsilon)$, where $J^{*-}(\varepsilon)$ and $J^{*+}(\varepsilon)$ are the left and right derivatives of $J^*(\varepsilon)$ with respect to ε. This implies $q_j'(\varepsilon) \leq q_i'(\varepsilon)$, where $q_j'(\varepsilon)$ and $q_i'(\varepsilon)$ are the derivatives of $q_j(\varepsilon)$ and $q_i(\varepsilon)$, respectively. Condition $q_j'(\varepsilon) \leq q_i'(\varepsilon)$ is true if and only if $-(\bar{b} - \gamma b) \leq 2\gamma(a'a)\varepsilon$, which implies $-(\bar{b} - \gamma b) < 0$ since $\gamma < 0$ and $\varepsilon > 0$.

From (38) $q_j(\varepsilon) < q_i(\varepsilon) \, \forall \varepsilon \in (0, \frac{-(\bar{b}-\gamma b)}{\gamma a'a})$.

Thus there exists $x \in \mathcal{P}_i$ with $q_j(x) < q_i(x)$. This is a contradiction since from Theorem 5, $\mathcal{A}_i \subset \mathcal{A}_j$. ◻

Note that, in case of degeneracy, the value function $J^*(x)$ in (26) may not be $C^{(1)}$; counterexamples are given in [6].

In Theorem 7 we have proven that the value function is $C^{(1)}$ for the nondegenerate QP (10). Now we want to show that the gradient of $J^*(x)$ is a vector valued PWA descriptor function.

THEOREM 8. *Consider the value function $J^*(x)$ in* (26)*, and assume that the CFTOC problem* (3) *leads to a nondegenerate QP* (10)*. Then the gradient $m(x) := \nabla J^*(x)$ is a vector valued PWA descriptor function.*

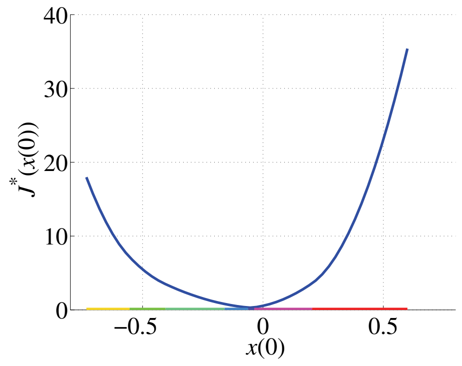
Proof. From Theorem 7 we see that $m(x)$ is a continuous vector valued PWA function, while from (26) we get

$$m(x) := \nabla J^*(x) = 2Q_i x + T_i \quad \text{if} \quad x \in \mathcal{P}_i, \; i = 1, \ldots, N_{\mathcal{P}}. \tag{39}$$

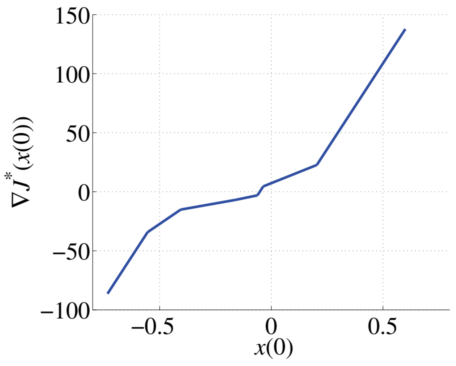
Since from Lemma 1 we know that $Q_i \neq Q_j$ for all neighboring polyhedra, it follows that $m(x)$ satisfies all conditions for a vector valued PWA descriptor function. ◻

Combining results of Theorems 8 and 4, it follows that by using Algorithm 5 we can construct a PWA descriptor function from the gradient of the value function $J^*(x)$.

Remark 3.5. Note that some CFTOC problems may lead to a degenerate QP (10). Identifying the classes of control problems which are guaranteed to be nondegenerate is currently an unsolved problem and the topic of ongoing research. An example of a control problem where the nondegeneracy fails is when, for some initial point, the optimal control is saturated on the whole time horizon (this gives as many active constraints as the dimension of U^*) and the state hits the state constraint at least once (so there are more active constraints than the dimension of U^*). However, degeneracy does not necessarily imply the loss of continuous differentiability of the value function, which thus might have no effect on our procedure. Next, we provide a simple example of a nondegenerate CFTOC which, by adding a terminal constraint, becomes degenerate and whose cost is not continuously differentiable.



(a) value function $J^*(x(0))$



(b) gradient of value function $\nabla J^*(x(0))$

FIG. 5. *Solution to the CFTOC problem* (42).

Example. Consider the system

$$x_{k+1} = -1.5x_k + u_k, \tag{40}$$

where $x_k \in \mathbb{R}$ and $u_k \in \mathbb{R}$ are state and input at time k, respectively, subject to the following constraints $\forall k \geq 0$:

$$-1 \leq x_k \leq 1, \ -0.5 \leq u_k \leq -0.1. \tag{41}$$

Consider the CFTOC problem (3) for system (40)–(41) with quadratic cost (4a), weighting matrices $Q^x = 0.1, Q^u = 10$, $Q^{x_N} = 0$, and horizon $N = 3$, i.e.,

$$\begin{aligned} J^*(x(0)) = \min_{u_0,u_1,u_2} \quad & \textstyle\sum_{k=0}^{2} 0.1x_k^2 + 10u_k^2 \\ \text{s.t.} \quad & x_{k+1} = -1.5x_k + u_k, \quad k = 0,\dots,2, \\ & -1 \leq x_k \leq 1, \quad k = 0,\dots,2, \\ & -0.5 \leq u_k \leq -0.1, \quad k = 0,\dots,2, \\ & x_0 = x(0). \end{aligned} \tag{42}$$

Then the corresponding QP (10) is nondegenerate and $J^*(x(0))$ is continuously differentiable everywhere in its domain (see Figure 5). However, if the additional constraint $x_3 = 0$ is added to the CFTOC problem (42), then the corresponding QP (10) becomes degenerate and $J^*(x(0))$ is not continuously differentiable at $x(0) = -0.0518$ (see Figure 6).

3.2.2. Generating a PWA descriptor function from the optimizer. A clear drawback of the procedure described in the previous subsection is the requirement that the QP (10) be nondegenerate. Luckily, there is another way to construct a vector valued PWA descriptor function $m(x)$ that always works and does not require any additional checking. This second procedure emerges naturally if we look at the properties of the optimizer $U^*(x)$ corresponding to the state feedback solution of the CFTOC problem (3). From Theorem 2, the optimizer $U^*(x)$ is continuous in x and

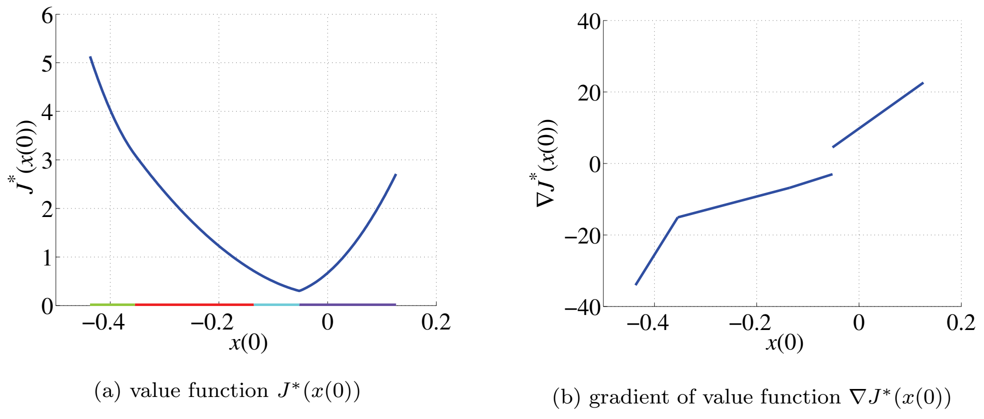


FIG. 6. *Solution to the CFTOC problem* (42) *with additional constraint* $x_3 = 0$.

PWA:

$$U^*(x) = l_i(x) := \bar{F}_i x + \bar{G}_i \quad \text{if} \quad x \in \mathcal{P}_i,\ i = 1, \dots, N_{\mathcal{P}}, \tag{43}$$

where $\bar{F}_i \in \mathbb{R}^{s \times n_x}$, $\bar{G}_i \in \mathbb{R}^s$, and $\mathcal{P}_i$, $i = 1, \dots, N_{\mathcal{P}}$, are the full dimensional critical regions of the optimization problem (10); cf. [4].

All we need to show now is the following lemma.

LEMMA 2. *Consider the CFTOC problem* (3) *and corresponding QP* (10). *Let* $\mathcal{P}_i$, $\mathcal{P}_j$ *be two (full dimensional) neighboring polyhedra of the state feedback solution* (43). *Then* $\bar{F}_i \neq \bar{F}_j$.

Proof. We prove Lemma 2 by contradiction. Suppose that the optimizer is the same for both polyhedra, i.e., $[\bar{F}_i\ \bar{G}_i] = [\bar{F}_j\ \bar{G}_j]$. Let $\mathcal{A}_i$ and $\mathcal{A}_j$ be the sets of active constraints for $\mathcal{P}_i$ and $\mathcal{P}_j$. From (27) we see that for an active constraint in $\mathcal{P}_i$, $v \in \mathcal{A}_i$, the following holds:

$$M^U_{(v)}(\bar{F}_i x + \bar{G}_i) = M_{(v)} + M^x_{(v)} x \quad \forall x \in \mathcal{P}_i, \forall v \in \mathcal{A}_i, \tag{44}$$

because (44) must hold $\forall x \in \mathcal{P}_i$, where $\mathcal{P}_i$ is a full dimensional polyhedron in $\mathbb{R}^{n_x}$. Since $M^U_{(v)}\bar{F}_i - M^x_{(v)} \in \mathbb{R}^{n_x}$ the above statement is satisfied if and only if

$$M^U_{(v)}\bar{F}_i - M^x_{(v)} = \mathbf{0}', \quad M^U_{(v)}\bar{G}_i - M_{(v)} = 0 \quad \forall v \in \mathcal{A}_i. \tag{45}$$

By assumption $[\bar{F}_i\ \bar{G}_i] = [\bar{F}_j\ \bar{G}_j]$, hence the expression (45) implies that

$$M^U_{(v)}(\bar{F}_j x + \bar{G}_j) = M_{(v)} + M^x_{(v)} x \quad \forall x, \forall v \in \mathcal{A}_i. \tag{46}$$

Clearly, (46) holds $\forall x \in \mathcal{P}_j$, and we conclude that all active constraints of $\mathcal{P}_i$ are also active in $\mathcal{P}_j$. The same reasoning holds for $\mathcal{P}_j$ and $\forall v \in \mathcal{A}_j$, thus implying that $\mathcal{A}_i = \mathcal{A}_j$. However, the same sets of active constraints and the same optimal control law in QP (10) can generate only one region, thus implying that $\mathcal{P}_i = \mathcal{P}_j$, which contradicts the assumption of Lemma 2. Therefore we have $[\bar{F}_i\ \bar{G}_i] \neq [\bar{F}_j\ \bar{G}_j]$. Finally, observing that for the two neighboring polyhedra, due to the continuity of $U^*(x)$, $\bar{F}_i = \bar{F}_j$ implies $\bar{G}_i = \bar{G}_j$, for which we prove that $\bar{F}_i \neq \bar{F}_j$. ☐

From Lemma 2 and Theorem 4 it follows that an appropriate PWA descriptor function $f(x)$ can be calculated from the optimizer $U^*(x)$ by using Algorithm 5.

Remark 3.6. Note that even if we are implementing RHC strategy, the construction of the PWA descriptor function is based on the full optimization vector $U^*(x)$ and the corresponding matrices $\bar{F}_i$ and $\bar{G}_i$.

Remark 3.7. In some cases the use of the optimal control profile $U^*(x)$ for the construction of a descriptor function $f(x)$ can be extremely simple. If there is a row $r,\ r \leq n_u$ (n_u is the dimension of u), for which $(\bar{F}_i)_{(r)} \neq (\bar{F}_j)_{(r)}$ $\forall i = 1, \ldots, N_{\mathcal{P}}$, $\forall j \in \mathcal{C}_i$, it is enough to set $A_i{}' = (\bar{F}_i)_{(r)}$ and $B^i = (\bar{G}_i)_{(r)}$, where $(\bar{F}_i)_{(r)}$ and $(\bar{G}_i)_{(r)}$ denote the rth row of the matrices $\bar{F}_i$ and $\bar{G}_i$, respectively. In this way we *avoid* the storage of the descriptor function, since it is equal to one component of the control law, which is stored anyway.

Remark 3.8. A natural question is whether this approach can be readily extended to the control of PWA systems. In general, the answer is no. However, we point out that whenever a PWA descriptor function can be found one can use Algorithm 4. For instance, the optimal control of a PWA system with piecewise linear cost results in a PWA control over polyhedral partition of the feasible state space (cf. [2]). If this control law satisfies conditions of the vector valued PWA descriptor function (see Definition 4), then we can use Algorithm 4. Note that the value function (12) in the linear cost case is also a descriptor function.

All described algorithms can be easily implemented and combined with available tools for deriving optimal controllers for linear systems (e.g., multiparametric toolbox [13] under MATLAB).

4. Example.

As an example, we compare the performance of Algorithms 2, 3, and 4 on a CFTOC problem for the discrete-time system

$$\left\{\begin{aligned} x(t+1) &= \begin{bmatrix} 4 & -1.5 & 0.5 & -0.25 \\ 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t), \\ y(t) &= [0.08333 \quad 0.2292 \quad 0.1146 \quad 0.02083]\, x(t) \end{aligned}\right. \tag{47}$$

resulting from the linear system $y = \frac{1}{s^4} u$, sampled at $T_s = 1$, and subject to the input and output constraints

$$-1 \leq u(t) \leq 1, \quad -10 \leq y(t) \leq 10. \tag{48}$$

4.1. CFTOC, linear cost case.

To regulate (47)–(48), we design a receding horizon controller based on the optimization problem (3) with the piecewise linear cost function (4a), $\ell = \infty$, $N = 2$, $Q = diag\{5, 10, 10, 10\}$, $R = 0.8$, $P = 0$. The PWA solution of the mp-LP problem was computed in 240 s on a Pentium III 900 MHz machine running MATLAB 6.0. The corresponding polyhedral partition of the state-space consists of $N_{\mathcal{P}} = 136$ regions with $N_{\mathcal{H}} = 1138$ halfspaces. In Table 3 we report the comparison between the complexity of Algorithms 2 and 3 for this example.

The average on-line RHC computation for a set of 1000 random points in the state space is 2259 flops (Algorithm 2) and 1088 flops (Algorithm 3).

4.2. CFTOC, quadratic cost case.

To regulate (47)–(48), we design a receding horizon controller based on the optimization problem (3) with the quadratic cost

Table 3
Complexity comparison of Algorithms 2 *and* 3 *for the example in section* 4.1.

	Algorithm 2	Algorithm 3
Storage demand (real numbers)	5690	680
Number of flops (worst case)	9104	1088

Table 4
Complexity comparison of Algorithms 2 *and* 4 *for the example in section* 4.2.

	Algorithm 2	Algorithm 4
Storage demand (real numbers)	9740	1065
Number of flops (worst case)	15584	3439

function (4b), $N = 7$, $Q = I$, $R = 0.01$, $P = 0$. The PWA solution of the mp-QP problem was computed in 560 s on a Pentium III 900 MHz machine running MATLAB 6.0. The corresponding polyhedral partition of the state space consists of $N_{\mathcal{P}} = 213$ regions with $N_{\mathcal{H}} = 1948$ halfspaces. For this example the choice of $w = [1\ 1\ 1\ 1\ 1\ 1\ 1]'$ is satisfactory to obtain a descriptor function from the optimizer. In Table 4 we report the comparison between the complexity of Algorithms 2 and 4 for this example.

The average on-line RHC computation for a set of 1000 random points in the state space is 2114 flops (Algorithm 2) and 175 flops (Algorithm 4).

5. Conclusion.

By exploiting properties of the value function and the optimal solution to the CFTOC problem, we presented two algorithms that significantly improved the efficiency of the on-line calculation of the control action (LP-based and QP-based) in terms of storage demand and computational complexity. The following improvements are then achieved:

1. There is no need to store the polyhedral partition of the state space.
2. In the worst case, the optimal control law is computed after the evaluation of one linear function per polyhedron.

Acknowledgments. The authors would like to thank the anonymous reviewers whose careful scrutiny and many useful suggestions considerably improved the quality of this manuscript.

REFERENCES

[1] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer, *Non-Linear Parametric Optimization*, Akademie-Verlag, Berlin, 1982.
[2] M. Baotić, F. J. Christophersen, and M. Morari, *Constrained optimal control of hybrid systems with a linear performance index*, IEEE Trans. Automat. Control, 51 (2006), pp. 1903–1919.
[3] A. Bemporad, F. Borrelli, and M. Morari, *Model predictive control based on linear programming—The explicit solution*, IEEE Trans. Automat. Control, 47 (2002), pp. 1974–1985.
[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, *The explicit linear quadratic regulator for constrained systems*, Automatica J. IFAC, 38 (2002), pp. 3–20.
[5] C. Berge, *Topological Spaces*, Dover Publications, Inc., Mineola, NY, 1997.
[6] A. B. Berkelaar, K. Roos, and T. Terlaky, *The optimal set and optimal partition approach to linear and quadratic programming*, in Advances in Sensitivity Analysis and Parametric Programming, T. Gal and H. J. Greenberg, eds., Internat. Ser. Oper. Res. Management Sci. 6, Kluwer Academic Publishers, Boston, 1997.
[7] F. Borrelli, A. Bemporad, and M. Morari, *Geometric algorithm for multiparametric linear programming*, J. Optim. Theory Appl., 118 (2003), pp. 515–540.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[9] D. Chmielewski and V. Manousiouthakis, *On constrained infinite-time linear quadratic optimal control*, Systems Control Lett., 29 (1996), pp. 121–129.

[10] H. J. Ferreau, H. G. Bock, and M. Diehl, *An online active set strategy to overcome the limitations of explicit MPC*, Internat. J. Robust Nonlinear Control, 18 (2008), pp. 816–830.

[11] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics*, 2nd ed., Walter de Gruyter, Berlin, 1995.

[12] C. Jones, P. Grieder, and S. Raković, *A logarithmic-time solution to the point location problem for parametric linear programming*, Automatica J. IFAC, 42 (2006), pp. 2215–2218.

[13] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari, *Multi-Parametric Toolbox (MPT)*, in Hybrid Systems: Computation and Control, Lecture Notes in Comput. Sci. 2993, Springer-Verlag, Philadelphia, 2004, pp. 448–462.

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

[15] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, *Constrained model predictive control: Stability and optimality*, Automatica J. IFAC, 36 (2000), pp. 789–814.

[16] J. A. Mendez, B. Kouvaritakis, and J. A. Rossiter, *State-space approach to interpolation in MPC*, Internat. J. Robust Nonlinear Control, 10 (2000), pp. 27–38.

[17] M. Schechter, *Polyhedral functions and multiparametric linear programming*, J. Optim. Theory Appl., 53 (1987), pp. 269–280.

[18] P. O. M. Scokaert and J. B. Rawlings, *Constrained linear quadratic regulation*, IEEE Trans. Automat. Control, 43 (1998), pp. 1163–1169.

[19] J. Spjøtvold, E. C. Kerrigan, C. N. Jones, P. Tøndel, and T. A. Johansen, *On the facet-to-facet property of solutions to convex parametric quadratic programs*, Automatica J. IFAC, 42 (2006), pp. 2209–2214.

[20] J. Sun, *On the structure of convex piecewise quadratic functions*, J. Optim. Theory Appl., 72 (1992), pp. 499–510.

[21] M. Sznaier and M. J. Damborg, *Suboptimal control of linear systems with state and control inequality constraints*, in Proceedings of the 26th IEEE Conference on Decision & Control, Los Angeles, CA, 1987, pp. 761–762.

[22] P. Tøndel, T. A. Johansen, and A. Bemporad, *An algorithm for multiparametric quadratic programming and explicit MPC solutions*, Automatica J. IFAC, 39 (2003), pp. 489–497.

[23] P. Tøndel, T. A. Johansen, and A. Bemporad, *Evaluation of piecewise affine control via binary search tree*, Automatica J. IFAC, 39 (2003), pp. 945–950.