Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

Efficient On-Line Computation of Constrained Optimal Control

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

†Automatic Control Laboratory, ETH Zentrum - ETL, CH-8092 Zürich, Switzerland
tel: +41-1-632 4158, fax: +41-1-632 1211, borrelli,baotic,bemporad,morari@aut.ee.ethz.ch

‡Dip. Ingegneria dell'Informazione, Università di Siena, Via Roma 56, I-53100 Siena, Italy
tel: +39-0577-23 46 31, fax: +39-0577-23 46 32, bemporad@dii.unisi.it

http://control.ethz.ch/~hybrid

Abstract

For discrete-time linear time-invariant systems with constraints on inputs and outputs, the constrained finite time optimal controller can be obtained explicitly as a piecewise affine function of the initial state via multiparametric programming [2, 1].

Exploiting the properties of the value function, we present two algorithms that efficiently perform the on-line evaluation of the explicit optimal control law both in terms of storage demands and computational complexity. The algorithms are particularly effective when used for *Model Predictive Control* (MPC) where an open loop constrained finite time optimal control problem has to be solved at each sampling time.

1 Introduction

Recently in [2] and in [1] the authors have shown how to compute the solution of the constrained finite time optimal control (CFTOC) problem as a explicit piecewise affine function of the initial state. Such a function is computed off-line by using a multiparametric programming solver [2, 4], which divides the state space into polyhedral regions, and for each region determines the linear gain which produces the optimal control action.

This method reveals its effectiveness when applied to *Model Predictive Control* (MPC) [8, 6]. MPC requires to solve at each sampling time an open-loop CFTOC problem. The optimal command signal is applied to the process only during the following sampling interval. At the next time step a new optimal control problem based on new measurements of the state is solved over a shifted horizon. Clearly the explicit solution reduces the on-line computation of the MPC control law to the evaluation of a piecewise affine function instead of the on-line solution of a quadratic or linear program.

The only drawback of such explicit optimal control laws is that the number of polyhedral regions could grow exponentially with the size of the optimal control problem. In this paper we focus on efficient on-line methods for the evaluation of this piecewise affine control law. The simplest algorithm would require: (i) the storage of the list of polyhedral regions and of the corresponding linear control laws, (ii) a sequential search through the list of polyhedra for the i-th polyhedron that contains the current state in order to implement the i-th control law. By exploiting the properties of the value function, for CFTOC based on LP and 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 of the evaluation of the explicit CFTOC control law and consequently of the explicit MPC control law.

2 CFTOC, MPC and Their Explicit Solution

2.1 CFTOC problem formulation

Consider the linear time-invariant system

$$\begin{cases} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t \end{cases} \tag{1}$$

subject to the constraints

$$y_{\min} \le y_t \le y_{\max},\ u_{\min} \le u_t \le u_{\max}{}^{1} \tag{2}$$

at all time instants $t \ge 0$.

In (1)–(2), $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$, and $y_t \in \mathbb{R}^p$ are the state, input, and output vector respectively, $y_{\min} \le y_{\max}$ ($u_{\min} \le u_{\max}$) are bounds on outputs (inputs), and the pair (A, B) is stabilizable.

Let $x(t)$ be the state at the generic time t and consider the constrained finite time optimal control problem

$$\begin{aligned} J^*(x(t)) = \min_U\ & \|x_{t+N}\|_p^P + \sum_{k=0}^{N-1} \|(x'_{t+k})\|_p^Q + \|u'_{t+k}\|_p^R \\ \text{subj. to}\quad & y_{\min} \le y_{t+k} \le y_{\max},\ k = 1, \dots, N \\ & u_{\min} \le u_{t+k} \le u_{\max},\ k = 0, \dots, N-1 \\ & x_{t+k+1} = Ax_{t+k} + Bu_{t+k},\ k \ge 0 \end{aligned} \tag{3}$$

In (3) we denote with $U \triangleq [u'_t, \dots, u'_{t+N-1}]'$ the optimization vector and with $\|x\|_p^M$ the p-norm of the vector x weighted with the matrix M, in particular $\|x\|_p^M = x'Mx$ for $p = 2$, $\|x\|_p^M = \|Mx\|_p$ for $p = 1, \infty$.

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 $p = 1, \infty$.

Depending on the norm p used in the cost function, the optimization problem (3) can be translated into an LP

[1] Although the form (2) is very common in practical implementations of CFTOC, the results of this paper also hold for the more general mixed constraints $Ex_t + Lu_t \le M$ arising, for example, from constraints on the input rate $\Delta u_t \triangleq u_t - u_{t-1}$.

($p = 1$ or $p = +\infty$) or into a QP ($p = 2$). The optimizer $U^*(t) = \{u_t^*, \dots, u_{t+N-1}^*\}$ of problem (3) is a function of the initial state $x(t)$. It can be computed by solving a QP or an LP once $x(t)$ is fixed or it can be computed explicitly for all $x(t)$ within a given range of values as explained in the following.

2.2 MPC formulation

The solution $U^*(t) = \{u_t^*, \dots, u_{t+N-1}^*\}$ of problem (3) is an open loop optimal control trajectory over a finite horizon. MPC uses it together with a receding horizon strategy to obtain an infinite horizon feedback control law.

Consider the problem of regulating to the origin the discrete-time linear time-invariant system (1) while fulfilling the constraints (2). MPC solves such a constrained regulation problem in the following way. Assume that a full measurement of the state $x(t)$ is available at the current time t. Then, the CFTOC problem (3) is solved at each time t for $t \geq 0$.

Let $U^*(t) = \{u_t^*, \dots, u_{t+N-1}^*\}$ be the optimal solution of (3). Then at time t

$$u(t) = u_t^* \tag{4}$$

is applied as input to system (1).

The two main issues regarding this policy are the feasibility of the optimization problem (3) for all $t \geq 0$ and the stability of the resulting closed-loop system. We will assume that the matrices Q, R, P, the horizon length N and the terminal constraints in (3) have been chosen to guarantee the stability and the feasibility of MPC control law (3)-(4). For a detailed discussion see, e.g. [2, 1, 6].

2.3 Explicit Solution Based on LP

Consider the case of $p = 1$ or $p = +\infty$ in problem (3). Is well known (e.g. [1]) that by introducing the vector $z \triangleq \{\varepsilon_1^x, \dots, \varepsilon_N^x, \varepsilon_1^u, \dots, \varepsilon_N^u, u_t, \dots, u_{t+N-1}\} \in \mathbb{R}^s$, $s \triangleq (m+2)N$, $\varepsilon_k^x \geq \|Qx_{t+k}\|_\infty$, $\varepsilon_N^x \geq \|Px_{t+N}\|_\infty$, $\varepsilon_k^u \geq \|Ru_{t+k}\|_\infty$, and substituting $x_{t+k} = A^k x(t) + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j}$ in (3), the optimization problem (3) can be rewritten in the form

$$\begin{array}{ll} J^*(x) = \min\limits_z & f^T z \\ \text{subj. to} & Gz \leq S + Fx \end{array} \tag{5}$$

where $x = x(t)$, and the matrices $f \in \mathbb{R}^s$, $G \in \mathbb{R}^{q \times s}$, $F \in \mathbb{R}^{q \times n}$, $S \in \mathbb{R}^q$ are easily obtained from Q, R, P and (3), as explained in [1].

Because the problem depends on $x = x(t)$ the implementation of MPC can be performed in two different ways: solving the LP (5) on-line at each time step or by solving problem (5) *off-line* for all x within a given range of values, i.e., by considering (5) as a *multi-parametric Linear Program* (mp-LP).

The mp-LP problem consists of computing the optimizer $z^*(x)$ and the value function $J^*(x)$ *for all* possible vectors x in a given compact set X. The solution of mp-LP problems can be approached as proposed in [4] or [5].

Once the multi-parametric problem (5) has been solved off-line for a polyhedral set $X \subseteq \mathbb{R}^n$ of states, the CFTOC explicit solution $z^*(x)$ of (5) is available as a piecewise affine function of x, and the model predictive controller (3)-(4) is also available explicitly, as the optimal input $u(t)$ consists simply of m components of $z^*(x)$

$$u(t) = [0 \ \dots 0 \ I_m \ 0 \ \dots \ 0] z^*(x). \tag{6}$$

In [5] the following results about the properties of the solution are proved:

Theorem 1 *Consider the multi-parametric linear program (5). Then the set of feasible parameters X_f is convex, the optimizer (if unique) $z^*(x) : X_f \mapsto \mathbb{R}^s$ is continuous and piecewise affine, and the optimal solution $J^*(x) : X_f \mapsto \mathbb{R}$ is continuous, convex and piecewise linear.*

Corollary 1 *The MPC control law (6) $u(t) = f(x)$, $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, defined by the optimization problem (3) and (4) is continuous and piecewise affine.*

2.4 Explicit Solution Based on QP

Consider the case $p = 2$ in problem (3). By substituting $x_{t+k|t} = A^k x(t) + \sum_{j=0}^{k-1} A^j B u_{t+k-1-j}$ in (3), the optimization problem (3) can be rewritten as the QP

$$\begin{array}{ll} J^*(x) = \frac{1}{2} x' Y x + & \min\limits_U \ \dfrac{1}{2} U' H U + x' F U \\ & \text{subj. to } GU \leq W + Ex \end{array} \tag{7}$$

where $x = x(t)$, the column vector $U \triangleq [u_t', \dots, u_{t+N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$, is the optimization vector, $H = H' \succ 0$, and H, F, Y, G, W, E are easily obtained from Q, R, P and (3). As only the optimizer U is needed, the term involving Y is usually removed from (7).

As in the 1-norm or ∞-norm case, because the problem depends on $x(t)$, the implementation of MPC can be performed in two different ways: solving the QP (7) on-line at each time step or, as recently proposed in [2], by solving problem (7) *off-line* for all $x(t)$ within a given range of values, i.e., by considering (7) as a *multi-parametric Quadratic Program* (mp-QP).

The mp-QP problem consists of computing the optimizer $U^*(x)$ and the value function $J^*(x)$ *for all* possible vectors x in a given set X. Once the multi-parametric problem (7) has been solved off line, i.e., the CFTOC solution $U^*(x)$ of (7) has been found, the explicit MPC law is simply

$$u(t) = [I_m \ 0 \ \dots \ 0] U^*(x). \tag{8}$$

In [2] the authors prove the following results about the properties of the solution

Theorem 2 *Consider the multi-parametric quadratic program (7) and let $H \succ 0$. Then the set of feasible parameters X_f is convex, the optimizer $U^*(x) : X_f \mapsto \mathbb{R}^s$ is continuous and piecewise affine, and the optimal solution $J^*(x) : X_f \mapsto \mathbb{R}$ is continuous, convex and piecewise quadratic.*

Corollary 2 *The MPC control law (8) $u(t) = f(x)$, $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, defined by the optimization problem (3) and (4) is continuous and piecewise affine.*

Corollary 2 states that by using an mp-QP solver the computational effort for MPC is reduced to a piecewise affine function evaluation. In the next section we propose a method to efficiently evaluate such a piecewise affine function without storing the polyhedral regions of the feasible domain X_f.

3 Efficient On-Line Algorithms

Let the explicit optimal control law be:

$$u^*(x) = F_i x + G_i, \quad \forall x \in \mathcal{P}_i, \quad i = 1, \ldots, N_r \qquad (9)$$

where $F_i \in \mathbb{R}^{m \times n}$, $G_i \in \mathbb{R}^m$, and $\mathcal{P}_i = \left\{ x \in \mathbb{R}^n \mid H_i x \leq K_i,\ H_i \in \mathbb{R}^{N_c^i \times n},\ K_i \in \mathbb{R}^{N_c^i} \right\}$, $i = 1, \ldots, N_r$ is a polyhedral partition of X_f. In the following H_i^j denotes the j-row of the matrix H_i. The on-line implementation of the control law (9) can be simply executed according to the following steps:

Algorithm 1

1. Measure the current state x
2. Search for the j-th polyhedron that contains x, i.e. $H_j x \leq K_j$
3. Implement the j-th control law, i.e. $u^*(x) = F_j(x) + G_j$

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

Algorithm 2

1. $i = 0$, notfound=1;
2. **while** $i \leq N_r$ and notfound
   2.1. $j = 0$, feasible=1
   2.2. **while** $j \leq N_c^i$ and feasible
      2.2.1. **if** $H_i^j x > K_i^j$ **then** feasible=0
      2.2.2. **else** $j = j + 1$
   2.3. **end**
   2.4. **if** feasible=1 **then** notfound=0
3. **end**

Algorithm 2 requires the storage of all polyhedra $\mathcal{P}_i$, i.e., $(n+1)N_C$ real numbers and in the worst case (the state is contained in the last region of the list) it will give a solution after nN_C multiplications, $(n-1)N_C$ sums and N_C comparisons, where $N_C \triangleq \sum_{i=1}^{N_r} N_c^i$.

By using the properties of the value function we show how Algorithm 2 can be replaced by a more efficient algorithm that *avoids storing the polyhedral regions* $\mathcal{P}_i,\ i = 1, \ldots, N_r$ and is *computationally faster*.

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

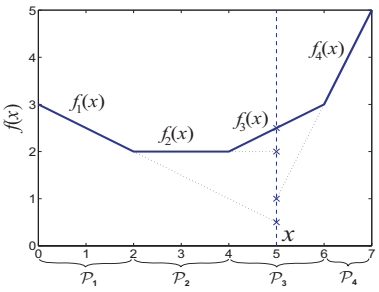

Figure 1: Example for Algorithm 3 in one-dimensional case. For a given point $x \in \mathcal{P}_3$ ($x = 5$) we have $f_3(x) = \max(f_1(x), f_2(x), f_3(x), f_4(x))$.

3.1 Efficient Implementation - LP case

From Theorem 1, the value function $J^*(x)$ corresponding to the explicit solution of CFTOC (3) based on LP is convex and piecewise affine:

$$J^*(x) = T_i' x + V_i, \quad \forall x \in \mathcal{P}_i, \quad i = 1, \ldots, N_r. \qquad (10)$$

The value function can be used to avoid storing the polyhedral regions $\mathcal{P}_i$. From the equivalence of the representations of piecewise linear convex functions [9], the function $J^*(x)$ in equation (10) can be represented alternatively as

$$J^*(x) = \max \{T_i' x + V_i,\ i = 1, \ldots, N\} \text{ for } x \in X_f \quad (11)$$

Exploiting the equivalence of (10) and (11), the polyhedral region P_j containing the state x can simply be identified by searching for the maximum among the list $\{T_i' x + V_i, \}_{i=1}^{N_r}$:

$$x \in \mathcal{P}_j \Leftrightarrow T_j' x + V_j = \max \{T_i' x + V_i,\ i = 1, \ldots, N_r\}. \qquad (12)$$

Therefore, instead of searching for the polyhedron j that contains the point x, we store the expression of the piecewise-linear value function and identify the 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 (see Figure 1).

Algorithm 2 is then substituted by the following:

Algorithm 3

1. Compute the vector $n \in \mathbb{R}^{N_r}$, where $n_i = T_i' x + V_i$
2. Find j such that $n_j = \max(n)$

Algorithm 3 does not require the storage of any polyhedra $\mathcal{P}_i$, but it only requires the storage of the value function, i.e., $(n+1)N_r$ real numbers, and it will give a solution after nN_r multiplications, $(n-1)N_r$ sums, and $N_r - 1$ comparisons. In Table 1 we compare the complexity of Algorithm 2 against Algorithm 3 in terms of storage demand and number of flops.

Table 1: Complexity of Algorithm 2 and Algorithm 3

	Algorithm 2	Algorithm 3
Storage (real numbers)	$(n+1)N_C$	$(n+1)N_r$
Flops (worst case)	$2nN_C$	$2nN_r$

3.2 Efficient Implementation - QP case

Consider the explicit solution of CFTOC (3) based on QP. From Theorem 2, the value function $J^*(x)$ is convex and piecewise quadratic. The simple Algorithm 3 described in the previous subsection cannot be applied. Therefore in a new algorithm is proposed.

Definition 1 *Two polyhedra $\mathcal{P}_i$, $\mathcal{P}_j$ are called neighboring polyhedra if they share a facet.*

Let $\{\mathcal{P}_i\}_{i=1}^{N_r}$ be the polyhedral partition obtained by solving the mp-QP (7) and denote by $C_i = \{j \mid \mathcal{P}_j \text{ is a neighbor of } \mathcal{P}_i,\ j = 1,\dots,N_r, j \neq i\}$ the list of all neighboring polyhedra of $\mathcal{P}_i$.

Theorem 3 *Let $f(x)$ be a continuous real-valued PWA function*

$$f(x) = \{f_i(x) = A_i x + B_i \mid x \in \mathcal{P}_i, \quad i = 1,\dots,N_r\}, \quad (13)$$

with

$$A_i \neq A_j,\ \forall j \in C_i,\ i = 1,\dots,N_r. \quad (14)$$

Define the list $O_i(x) = \{o_j^i(x) \mid j \in C_i\}$, where

$$o_j^i(x) = \begin{cases} +1 & f_i(x) \geqslant f_j(x) \\ -1 & f_i(x) < f_j(x) \end{cases} \quad (15)$$

Then $O_i(x)$ has the following properties:

(i) $O_i(x) = S_i = const, \quad \forall x \in \mathcal{P}_i, \quad i = 1,\dots,N_r.$

(ii) $O_i(x) \neq S_i, \quad \forall x \notin \mathcal{P}_i, \quad i = 1,\dots,N_r.$

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

$$g_j^i(x) = f_i(x) - f_j(x). \quad (16)$$

From the continuity of $f(x)$ it follows that $g_j^i(x) = 0, \forall x \in \mathcal{F}$. From convexity of $\mathcal{P}_i$ and $\mathcal{P}_j$ it follows that $g_j^i(x)$ is the separating hyperplane between $\mathcal{P}_i$ and $\mathcal{P}_j$.

(i) Because $g_j^i(x) = 0$ is a separating hyperplane, the function $g_j^i(x)$ does not change its sign for all $x \in \mathcal{P}_i$, i.e., $o_j^i(x) = s_j^i, \forall x \in \mathcal{P}_i$ with $s_j^i = +1$ or $s_j^i = -1$. The same reasoning can be applied to all neighbors of $\mathcal{P}_i$ to get $S_i = \{s_j^i, \forall j \in C_i,\ x \in \mathcal{P}_i\}$. (ii) $\forall x \notin \mathcal{P}_i,\ \exists j \in C_i$ such that $H_i^j x > K_i^j$. Since $g_j^i(x) = 0$ is a separating hyperplane, $o_j^i(x) = -s_j^i$. ■

Equivalently, Theorem 3 states the following property

$$x \in \mathcal{P}_i \Leftrightarrow O_i(x) = S_i \quad (17)$$

where S_i uniquely characterizes $\mathcal{P}_i$. Therefore to check on-line if the polyhedral region i contains the state x it is sufficient to compute the binary vector $O_i(x)$ and compare it with S_i. List $S_i = \{s_j^i \mid j \in C_i\}$ is calculated off-line, by comparing the values of $f_i(x)$ and $f_j(x)$ for $j \in C_i$ in a point x belonging to $\mathcal{P}_i$, for instance, the Chebychev center of $\mathcal{P}_i$.

Figure 2 illustrates the procedure with 4 regions. The list of neighboring regions C_i and the list S_i can be constructed by simply looking at the figure: $C_1 = \{2\}$, $C_2 = \{1,3\}$, $C_3 = \{2,4\}$, $C_4 = \{3\}$, $S_1 = \{-1\}$, $S_2 = \{-1,1\}$, $S_3 = \{1,-1\}$, $S_4 = \{-1\}$. The point $x = 4$ is in region 2 and we have $O_2(x) = \{-1,1\} = S_2$, while $O_3(x) = \{-1,-1\} \neq S_3$, $O_1(x) = \{1\} \neq S_1$, $O_4(x) = \{1\} \neq S_4$. Obviously, if we are in the wrong region, the checking procedure gives us information about the right search direction(s). The solution can be found by searching in the direction where constraint(s) are violated, i.e., we should check the neighboring region $\mathcal{P}_j$ for which $o_j^i(x) \neq s_j^i$.

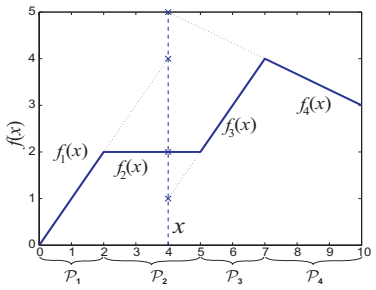

Figure 2: Example for Algorithm 4 in one-dimension.

The overall procedure is composed of two parts:

1. *(off-line)* Construction of the PWA function $f(x)$ in (13) satisfying (14) and computation of the list of neighbors C_i and the list S_i,
2. *(on-line)* Execution of the following algorithm

Algorithm 4

```
1. i = 1, notfound=1;
2. while notfound
  2.1. compute O_i(x)
  2.2. if O_i(x) = S_i then notfound=0
  2.3. else i = q, where o_q^i(x) ≠ s_q^i, q ∈ C_i
3. end
```

Algorithm 4 does not require the storage of the polyhedra $\mathcal{P}_i$, but it only requires the storage of one linear function $f_i(x)$ per polyhedron, i.e., $(n+1)N_r$ real numbers and the list of neighbors C_i which requires N_C integers. In the worst case Algorithm 4 terminates after nN_r multiplications, $(n-1)N_r$ sums and N_C comparisons. In Table 2 we compare the complexity of Algorithm 4 against Algorithm 2 in terms of storage demand and number of flops.

Remark 1 Note that the computation of $O_i(x)$ in Algorithm 4 requires the evaluation of N_c^i linear functions, but the overall computation never exceeds N_r linear function evaluations. Consequently, Algorithm 4 will outperform Algorithm 2, since typically $N_C \gg N_r$.

Table 2: Complexity of Algorithm 2 and Algorithm 4

	Algorithm 2	Algorithm 4
Storage (real n.)	$(n+1)N_C$	$(n+1)N_r$
Flops (worst case)	$2nN_C$	$(2n-1)N_r + N_C$

The PWA function $f(x)$ in (13) satisfying (14) will be referred to as *PWA descriptor function*. In the rest of the section we propose a way to construct a PWA descriptor function from the PWQ value function. In the following we will assume that the mp-QP (7) is not degenerate (refer to [7] for a detailed discussion on the degeneracy of mp-QP).

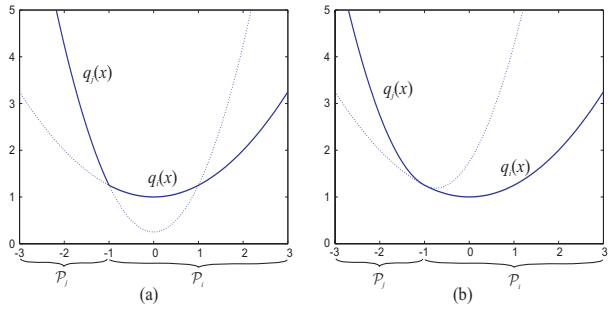

Figure 3: Two convex piecewise quadratic functions: (a) not differentiable one, (b) differentiable one.

Let $J^*(x)$ be the convex and piecewise quadratic (CPWQ) value function corresponding to the explicit solution of CFTOC (3) based on QP:

$$J^*(x) = \left\{ q_i(x) \triangleq x'Q_ix + T_i'x + V_i, \text{ for } x \in \mathcal{P}_i, \ i = 1, \ldots, N_r \right\}, \tag{18}$$

We will prove that the $J^*(x)$ is a $C^{(1)}$ function and we will obtain a PWA descriptor function by differentiating it. Before going further we recall the following result [2, 7]:

Theorem 4 *Consider the set of active constraints at the optimum of QP (7) and assume there are no degeneracies:*

$$\mathcal{A}^*(x) = \{i \mid G^iU^*(x) = W^i + S^ix\}, \tag{19}$$

then

1. *$\mathcal{A}^*(x)$ is constant $\forall x \in \mathcal{P}_i$, $i = 1, \ldots, N_r$, i.e. $\mathcal{A}^*(x) \triangleq \mathcal{A}_i \ \forall x \in \mathcal{P}_i$*
2. *If $\mathcal{P}_i$ and $\mathcal{P}_j$ are neighboring polyhedra then $\mathcal{A}_i \subset A_j$ or $\mathcal{A}_j \subset A_i$.*

Theorem 5 *Consider the value function $J^*(x)$ in (18). Let $\mathcal{P}_i$, $\mathcal{P}_j$ be two neighboring polyhedra then*

$$Q_i - Q_j \preceq 0 \text{ or } Q_i - Q_j \succeq 0 \text{ and } Q_i \neq Q_j \tag{20}$$

and

$$Q_i - Q_j \preceq 0 \text{ iff } \mathcal{A}_i \subset A_j \tag{21}$$

Proof: The proof is given in [3].

We recall a property of convex piecewise quadratic functions [9]:

Proposition 1 *Consider the value function $J^*(x)$ in (18) satisfying (20) and its quadratic expression $q_i(x)$ and $q_j(x)$ on two neighboring polyhedra $\mathcal{P}_i$, $\mathcal{P}_j$ with common boundary $a'x = b$. Then there exist constants $\gamma \in \mathbf{R}/\{0\}$ and $\bar{b}$ such that*

$$q_i(x) = q_j(x) + (a'x - b)(\gamma a'x - \bar{b}) \tag{22}$$

Equation (22) states that the neighboring expressions $q_i(x)$ and $q_j(x)$ of a CPWQ function satisfying (20) either intersect on two parallel hyperplanes $a'x = b$ and $\gamma a'x = \bar{b}$ if $\bar{b} \neq \gamma b$ (see Figure 3(a)), or they are tangent in one hyperplane $a'x = b$ if $\bar{b} = \gamma b$ (see Figure 3(b)).

Theorem 6 *Under the hypothesis of Theorem 4, the value function $J^*(x)$ in (18) is $C^{(1)}$.*

Proof: We will prove that $\bar{b}$ and b in (22) satisfy $\bar{b} = \gamma b$ by contradiction. Suppose there exists $i \in \{1, \ldots, N_r\}$ and $j \in C_i$ such that $\bar{b} \neq \gamma b$. Without loss of generality we assume that (i) $Q_i - Q_j \preceq 0$ and (ii) $\mathcal{P}_i$ is on the side $a'x \leq b$ of the common boundary. Let $\mathcal{F}_{ij}$ be the common facet between $\mathcal{P}_i$ and $\mathcal{P}_j$ and $\mathcal{F}_{ij}$ its interior.

From (i) and from (22), $\gamma < 0$ and $\gamma = 0$ iff $Q_i - Q_j = 0$. Take $x_0 \in \mathcal{F}_{ij}$, for sufficiently small $\varepsilon \geq 0$, the point $x = x_0 - a\varepsilon$ belongs to $\mathcal{P}_i$.

Let $J^*(\varepsilon) = J^*(x_0 - a\varepsilon)$ then

$$q_i(\varepsilon) = q_j(\varepsilon) + (a'a\varepsilon)(\gamma a'a\varepsilon - (\bar{b} - \gamma b)) \tag{23}$$

From convexity of $J^*(\varepsilon)$, $J^{*-}(\varepsilon) \leq J^{*+}(\varepsilon)$ where $J^{*-}(\varepsilon)$ and $J^{*+}(\varepsilon)$ are the left and right derivatives of $J^*(\varepsilon)$. 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 $q_i(\varepsilon)$. From (23) the latter is true if and only if $(a'a)(\bar{b} - \gamma b)) \leq 2 * \gamma(a'a)^2\varepsilon$, that implies $\bar{b} < \gamma b$ since $\gamma < 0$ and $\varepsilon > 0$.

From (23) $q_i(\varepsilon) < q_j(\varepsilon)$ for all $\varepsilon \in (0, \frac{\bar{b} - \gamma b}{\gamma a'a})$.

Thus there exists $x \in \mathcal{P}_i$ with $q_j(x) < q_i(x)$. This is a contradiction since from Theorem 4, $\mathcal{A}_i \subset A_j$. ■

Note that in case of degeneracy the value function $J^*(x)$ in (18) is not $C^{(1)}$, counterexamples are given in [5].

Combining Theorem 6 and (22), we obtain

$$q_i(x) = q_j(x) + \gamma(a'x - b)^2 \tag{24}$$

Theorem 7 *Consider the value function $J^*(x)$ in (18). Let $m(x) \triangleq \nabla J^*(x)$, be the gradient vector of $J^*(x)$, i.e., $m(x) = \{m_i(x) \triangleq \nabla q_i(x) = 2Q_ix + T_i \mid x \in \mathcal{P}_i, \ i = 1, \ldots, N_r\}$, where $m : \ x \in \mathbb{R}^n \mapsto m(x) \in \mathbb{R}^n$ satisfies the following properties*

(i) $m(x)$ is continuous;

(ii) $\exists w \in \mathbb{R}^n$ such that for each i and j with $\mathcal{P}_i$, $\mathcal{P}_j$ neighboring polyhedra $g_j^i(x) \triangleq w'(m_i(x) - m_j(x)) \neq 0$ for any x which is not an element of their common hyperplane.

Proof: (i) Follows easily from Theorem 6. (ii) Consider two neighboring polyhedra $\mathcal{P}_i$, $\mathcal{P}_j$ and their common hyperplane $a'x = b, a \neq 0$. From equation (24) we get

$$g_j^i(x) = 2\gamma w'a(a'x - b). \tag{25}$$

Since $\gamma \neq 0$, $a \neq 0$, and $a'x - b \neq 0$ it follows that $g_j^i(x) = 0$ if and only if $w'a = 0$ (i.e., vectors w and a are orthogonal). In order to have $g_j^i(x) \neq 0$ for all neighboring polyhedra $\mathcal{P}_i$ and $\mathcal{P}_j$, the vector w must not be parallel to any of the common hyperplanes. As the number of common hyperplanes defining the polyhedral partition is finite, such a vector w exists and can be easily computed. ■

It follows from Theorem 7 that an appropriate PWA descriptor function $f(x)$ can be calculated from the gradient of the value function. It is sufficient to find off-line

a vector w that is not parallel to any hyperplane of the polyhedral partition. Then $f_i(x) = A_i x + B_i$, where $A_i = 2w'Q_i$ and $B_i = w'T_i$.

Remark 2 Note that Algorithm 4 applies also to the LP case by using the value function $J^*(x)$ in (10) as descriptor function. Algorithm 4 is less simple than Algorithm 3 but it requires the evaluation of N_r linear functions only in the worst case.

4 Example

We compare the performance of Algorithm 2, Algorithm 3 and Algorithm 4 on an CFTOC problem for the linear system

$$y(t) = \frac{1}{s^4}u(t), \tag{26}$$

or its equivalent discrete-time ($T_s = 1$) state-space representation

$$\begin{aligned} x(t+1) &= \begin{bmatrix} 4 & -1.5 & 0.5 & -0.25 \\ 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 0.083 & 0.22 & 0.11 & 0.02 \end{bmatrix} x(t) \end{aligned} \tag{27}$$

subject to the input constraint

$$-1 \le u(k) \le 1 \tag{28}$$

and the output constraint

$$-10 \le y(k) \le 10 \tag{29}$$

4.1 Explicit MPC based on LP

To regulate (27), we design an MPC controller based on the optimization problem (3) where $p = \infty$, $N = 2$, $Q = diag\{[5\ 10\ 10\ 10]\}$, $R = 0.8$, $P = 0$. The explicit solution of the mp-LP problem consist of a polyhedral partition of the state-space consist in 136 regions. In Table 3 we report the comparison between the complexity of Algorithm 2 and Algorithm 3 for this example.

The average on-line MPC computation for a set of 1000 random points in the state space is 2259 flops (Algorithm 2), and 1088 flops (Algorithm 3). The implicit solution with Matlab Optimization Toolbox LP solver takes 25459 flops on average.

Table 3: Complexity comparison of Algorithm 2 and Algorithm 3 for example in Section 4.1

	Algorithm 2	Algorithm 3
Storage (real numbers)	5690	680
Flops (worst case)	9104	1088

4.2 Explicit MPC based on QP

To regulate (27), we design an MPC controller based on the optimization problem (3) where $p = 2$, $N = 7$, $Q = diag\{[1\ 1\ 1\ 1]\}$, $R = 0.01$, $P = 0$. The explicit solution of the mp-QP problem consists of a polyhedral partition of the state-space consist in 213 regions. For this example the choice of $w = [1\ 0\ 0\ 0]'$ is satisfactory. In Table 4 we report the comparison between Algorithm 2 and Algorithm 4 for this example.

The average on-line MPC computation for a set of 1000 random points in the state space is 2114 flops (Algorithm 2), and 175 flops (Algorithm 4). The implicit solution with Matlab's QP solver takes 25221 flops on average.

Table 4: Complexity comparison of Algorithm 2 and Algorithm 4 for example in Section 4.2

	Algorithm 2	Algorithm 4
Storage (real numbers)	9740	1065
Flops (worst case)	15584	3439

5 Conclusion

By exploiting the properties of the value function of MPC, we presented two algorithms that significantly improve the efficiency of the on-line explicit MPC (both based on LP and QP) in terms of storage demands and computational complexity. The following improvements are achieved

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

References

[1] A. Bemporad, F. Borrelli, and M. Morari. Explicit solution of constrained $1/\infty$-norm model predictive control. In *Proc. 39th IEEE Conf. on Decision and Control*, December 2000.

[2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, to appear.

[3] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. Efficient on-line computation of explicit model predictive control. Technical Report AUT01-15, Automatic Control Laboratory, ETH Zurich, Switzerland, August 2001.

[4] F. Borrelli, A. Bemporad, and M. Morari. A geometric algorithm for multi-parametric linear programming. Technical Report AUT00-06, Automatic Control Laboratory, ETH Zurich, Switzerland, February 2000.

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

[6] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[7] P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *Proc. 40th IEEE Conf. on Decision and Control*, December 2001.

[8] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In *Chemical Process Control - V*, volume 93, no. 316, pages 232–256. AIChE Symposium Series - American Institute of Chemical Engineers, 1997.

[9] M. Schechter. Polyhedral functions and multiparametric linear programming. *Journal of Optimization Theory and Applications*, 53(2):269–280, May 1987.