Explicit Predictive Control with Non-Convex Polyhedral Constraints

Emilio Pérez [a] Carlos Ariño [a] F.Xavier Blasco [b] Miguel A. Martínez [b]

[a] *Departamento de Ingeniería de Sistemas Industriales y Diseño, Universitat Jaume I, Avenida Vicent Sos Baynat, s/n. 12071 Castelló de la Plana, Spain.*

[b] *Instituto Universitario de Automática e Informática Industrial. Universidad Politécnica de Valencia. Camino de Vera S/N 46022 - Valencia, Spain.*

Abstract

This paper proposes an explicit solution to the model predictive control of linear systems subject to non-convex polyhedral constraints. These constraints are modeled as the union of a finite number of convex polyhedra. The algorithm is based on calculating the explicit solution to a modified problem with linear constraints defined as the convex hull of the original ones and classifying its regions by their relation with the regions of the explicit solution to the original problem. Some of the regions are divided and a procedure based on sum-of-squares programming is designed to determine which of the possible solutions are in fact optimal. Finally, the online algorithm is shown to be better in terms of computational cost and memory requirements than an algorithm based on obtaining and comparing the solutions of the problem using as constraints the polyhedra which union form the non-convex regions, both theoretically and by the results of an example.

Key words: Model Predictive Control, Multiparametric Programming, Non-convex Constraints, Sum-of-squares

1 Introduction

A special class of non-convex constraints are the so called non-convex polyhedral constraints, defined as the non-convex union of a finite number of polyhedra. Non-convex polyhedral constraints arise naturally in problems such as obstacle avoidance, which is inherently non-convex. Model predictive control (MPC) approaches for this kind of systems exist, but they require solving a mixed integer quadratic program online, which may be excessively time consuming.

Another field of application of non-convex polyhedral constraints comes from linear systems with non-linear constraints. These problems usually arise when controlling Hammerstein-Wiener systems by inversion of static non-linearities and an appropriate controller, such as MPC, for the linear part [3]. They also appear in feedback linearization control with linear constraints [12]. In both cases, constraints on the system inputs have to be transformed by the non-linearity in order to be coped with by a linear MPC. If these non-linear constraints are non-convex, precise approximation requires the use of several polyhedra with a non-convex union.

Linear systems with non-convex polyhedral constraints can also be treated as piecewise affine (PWA) systems. For this kind of models, finite-time constrained optimal control (FTCOC) and MPC have been active research fields in the recent years.

A completely explicit solution of the FTCOC problem with a quadratic cost was first obtained in [2], where authors show that the partition of the state-space in the general case is not polyhedral, but defined by linear and quadratic inequalities. The procedure is based on dynamic programming, and in each iteration it requires the solving of a number of multiparametric quadratic programs (mpQP). However, the comparison of quadratic objective functions to possibly eliminate regions is not treated in a systematic way. A different approach was proposed in [8] where all the possible switching sequences are enumerated and, for each of them, the PWA dynam-

⋆ This paper was not presented at any IFAC meeting. Corresponding author E.Pérez Tel. +34 964 728179 Fax +34 964 728170

Email addresses: pereze@esid.uji.es (Emilio Pérez), arino@esid.uji.es (Carlos Ariño), xblasco@isa.upv.es (F.Xavier Blasco), mmiranzo@isa.upv.es (Miguel A. Martínez).

ics are treated as a time-varying system. Several FTCOC are solved then via mpQP and objective functions compared online, which in general increases the CPU time spent.

Although the system with non-convex polyhedral constraints can be modeled as PWA, fully exploiting the structure of the initial problem presents several advantages. First, the calculation of the regions of the explicit solution can be simplified by solving an intermediate linear constrained problem for the convex hull of the original constraints. Furthermore, the systematic elimination of non-optimal solutions allows us to reduce the number of regions of the explicit solution. Lastly, from the detailed explicit solution we develop efficient online algorithms based on binary search tree strategies [14].

2 Problem statement

Let us consider the problem of regulating to the origin a discrete-time linear time invariant system of the form $x_{k+1} = Ax_k + Bu_k$ with input and states constraints defined as the non-convex union of a finite number of convex polyhedra:

$$(x_k, u_k) \in \bar{\Omega} = \bigcup_i \Omega_i$$

The proposed MPC scheme for this system with an horizon N requires to solve an optimization problem with quadratic weighting on the input and the states. Then, MPC uses only the first element of the optimizer in a receding horizon fashion. It can be shown that the optimization problem can be written as

$$\begin{aligned}\mathcal{P}_N(x) : & V_N^{OPT}(x) := \min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T H \mathbf{u} + \mathbf{u}^T F x \\ & \text{subject to:} \\ & (x_k, u_k) \in \bar{\Omega} \text{ for } k = 0, \ldots, N-1, \\ & x_k = f_k(x, u_0, \ldots, u_{k-1}) \text{ for } k = 1, \ldots, N-1, \\ & x_N = f_N(x, \mathbf{u}) \in \bar{\mathbb{X}}_f\end{aligned}$$

where $\mathbf{u} = [u_0^T, u_1^T, \ldots, u_{N-1}^T]^T$ is the optimization vector, f_k reflect the dependence of the state at each sample time on previous control actions and $\bar{\mathbb{X}}_f$ is the terminal constraint set. In order to give stability guarantees, the optimal choice for this set is the maximum input admissible positively invariant set. For our kind of systems, this set is also a non-convex polyhedron. A procedure for its calculation is given in [11].

As all the constraints are now expressed in terms of the optimization variables u_k, they can be rewritten in a single non-convex polyhedron ($\mathbb{T}$) which can be calculated by finding the feasible sets of all possible constraints combinations. Then, the optimization problem can be formulated as:

$$\begin{aligned}\mathcal{P}_{\mathbb{T}}(x) : & V_N^{OPT}(x) := \min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T H \mathbf{u} + \mathbf{u}^T F x, \\ & \text{subject to:} \\ & (\mathbf{u}, x) \in \mathbb{T} = \textstyle\bigcup_{i=0}^{\gamma} T_i\end{aligned} \tag{1}$$

where T_i are convex polyhedra.

The most direct way to solve problem (1) is to rewrite it in the form of (2), solve problems $\mathcal{P}_{T_i}$ and compare each sample time the objective function for each optimum $\mathbf{u}_{T_i}^{opt}$.

$$\begin{aligned}\mathcal{P}_{\mathbb{T}}(x) : & \ V_N^{OPT}(x) := \min_i \left\{ V_{iN}^{OPT}(x) \right\} \\ & \text{where:} \\ \mathcal{P}_{T_i}(x) : & V_{iN}^{OPT}(x) := \min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T H \mathbf{u} + \mathbf{u}^T F x, \\ & \text{subject to:} \\ & (\mathbf{u}, x) \in T_i\end{aligned} \tag{2}$$

This approach is equivalent to the algorithm of [8] for PWA systems which we use as a base case for comparison purposes.

3 Explicit solution

The objective in this section is to provide a piecewise affine solution to problem (1).To do so, we first calculate the convex hull of $\mathbb{T}$, denoted as $\mathbf{conv}(\mathbb{T})$.

Now, considering $\mathbf{conv}(\mathbb{T})$ as a new constraint set, we can define the following optimization problem:

$$\begin{aligned}\mathcal{P}_C(x) : & V_N^{OPT}(x) := \min_{\mathbf{u}} \frac{1}{2}\mathbf{u}^T H \mathbf{u} + \mathbf{u}^T F x, \\ & \text{subject to:} \\ & (\mathbf{u}, x) \in \mathbf{conv}(\mathbb{T})\end{aligned} \tag{3}$$

As $\mathbf{conv}(\mathbb{T})$ is by definition a convex set, it has been shown in [1] that if matrix H is positive definite, there exists a piecewise affine solution of $\mathcal{P}_C$ of the form:

$$\begin{aligned}\mathbf{u}_C^{opt}(x) = \mathbf{u}_{Ci}^{opt}(x) = G_i^C x + h_i^C, \text{ for } x \in X_i^C \\ \text{where } X_i^C = \{x \in \mathbb{R}^n | L_i^C x \leq W_i^C\}\end{aligned} \tag{4}$$

Our goal is to obtain the explicit solution to $\mathcal{P}_{\mathbb{T}}$ from the already known solution to $\mathcal{P}_C$. To do so, it is useful to notice that if $\mathbf{u}_C^{opt}(x)$ is feasible in $\mathbb{T}$ it will also be the optimal solution of $\mathcal{P}_{\mathbb{T}}$. Therefore we can establish a classification of the regions of the explicit solution to $\mathcal{P}_C$, X_i^C, by their relation with the regions of the explicit solution to $\mathcal{P}_{\mathbb{T}}$, X_j.

3.1 Regions classification

- **Coincident regions:** Regions in which affine expression of the optimum is feasible in $\mathbb{T}$ for every $x \in X_i^C$ and are exactly equal to one of the regions X_j
$$X_i^C = X_j$$
- **Subset regions:** Regions in which affine expression of the optimum is feasible in $\mathbb{T}$ for every $x \in X_i^C$ and are a strict subset of one of the regions X_j
$$X_i^C \subset X_j$$
- **Non-subset regions:** Regions in which affine expression of the optimum is not feasible in $\mathbb{T}$ for some $x \in X_i^C$.

In order to perform the classification of regions, the first step is to identify non-subset regions by determining feasibility of the optimal solution in $\mathbb{T}$. This can be done by solving the linear programs that check if there exists any $x \in X_i^C$ such that $\mathbf{u}_{Ci}^{opt}(x) \in \mathbb{T}^*$, where $\mathbb{T}^* = \mathbf{conv}(\mathbb{T})\backslash\mathbb{T}$.

As non-subset regions have an unknown solution to problem $\mathcal{P}_{\mathbb{T}}$, for some x in these regions the explicit solution can be the same that the one in adjacent regions. Therefore these adjacent regions are subset regions, while the rest are coincident. Further details on regions classification can be found in [10].

3.2 Division of non-subset regions

Once the classification of regions is made we have two types of regions, coincident and subset regions, for which the optimal solution to problem $\mathcal{P}_{\mathbb{T}}$ is known and a third type, non-subset regions, for which it is unknown. In this and the following subsection we present an approach to obtain the solution in this regions too.

Given a non-subset region X_j, the procedure starts by taking one of the constraint sets T_i and dividing X_j in subregions for which the solution to $\mathcal{P}_{T_i}$ is characterized by a different set of active constraints, X_{ji_1} [1]. If this is done for every non-subset regions, different X_{ji_1} can be found to have the same active set. Therefore, it can be checked if their union is convex and be treated from this point as a single region. Once this first step is completed, all the non-subset regions are divided in subregions with different optimal solutions to problem $\mathcal{P}_{T_i}$. At this point, we can consider a new constraint set, T_{i+1}. Following the same philosophy, regions X_{ji_1} can be divided in subregions depending on the optimal solution to $\mathcal{P}_{T_{i+1}}$ they have. This way, we get a new set of regions, $X_{ji_1i_2}$, for which solution to problems $\mathcal{P}_{T_i}$ and $\mathcal{P}_{T_{i+1}}$ have different explicit solutions. In some cases, one of the two possible solutions is the global optimum for the whole region, and therefore it is not necessary to keep the other one.

[1] Possibly including several regions with no solution for $\mathcal{P}_{T_i}$ (X^* in Algorithm 1)

A procedure to eliminate these non-optimal solutions is described in Section 3.3. After this elimination, some regions might have the same set of explicit solutions. If these regions have a convex union, they can be replaced by it.

Following this approach with the γ different T_i sets, we will finally have the state space covered by the non-subset regions divided in a number of regions $X_{ji_1i_2\ldots i_\gamma}$ with a set of several explicit solutions each.

Analyzing the proposed methodology, it can be seen that it is based in solving repeatedly three different problems.The first one is, given a region, obtain all the possible subregions in its interior defined by different active sets for a given constraint set. This is a well known problem in explicit MPC with linear constraints and has been solved in different ways, e.g. [13],[5]. The second problem is the elimination of non-optimal solutions in a given region, and is described in Section 3.3. The last problem is the union of regions with the same set of solutions, for which efficient algorithms, such as [4], exist in the literature.

The complete procedure to divide all non-subset region is summarized in Algorithm 1.

It can be noticed that a procedure like the designed one could have been proposed from the beginning without the need of obtaining the convex hull and classificating regions of $\mathcal{P}_C$ (3). Nevertheless, doing that would in general imply a higher number of $X_{ji_1i_2\ldots i_\gamma}$ regions and would be more computationally expensive.

3.3 Elimination of non-optimal solutions

The objective in this subsection is to check if all of the possible optimums for a given region are really the global optimum in a subset of it, or some of them can be discarded from the beginning. That is, given a region $X_r = \{x \in \mathbb{R}^n | L_r x \leq W_r\}$ with λ possible optimums, each of them with a corresponding objective function $V_{Ni}^{OPT}(x)$, we aim to know if every possible optimum gives the minimum objective function somewhere inside X_r.

This can be achieved by checking the feasibility of the following system:

$$\begin{cases} W_r - L_r x \geq 0 \\ V_{Nj}^{OPT}(x) - V_{Ni}^{OPT}(x) \geq 0 \;\; j = \{1 \ldots \lambda\}\backslash i \end{cases} \tag{5}$$

Problem (5) can be solved by means of a sum-of-squares (SOS) program based on the Positivstellensatz theorem [9]. This theorem allows to prove that system (5) is empty if and only if there exist SOS polynomials $s_\alpha(x)$

Algorithm 1 Exploration of non-subset regions

(1) Initialize constraint sets not explored, $\mathbb{T} = \{T_1, \ldots, T_\gamma\}$.
(2) Initialize set of regions to explore with all non-subset regions of $\mathcal{P}_C$: $\mathbb{X} = \{X_j\}$.
(3) While $\mathbb{T} \neq \emptyset$, take $T_i \in \mathbb{T}$:
  (a) Initialize set of new regions to explore: $\mathbb{Y} = \emptyset$.
  (b) Take $X \in \mathbb{X}$:
    (i) Search $x \in X$ and find its active region in problem $\mathcal{P}_{T_i}$: X_1.
    (ii) Initialize set of known regions $\mathbb{R} := \{X_1\}$ and set of unexplored regions $\mathbb{U} := \{X_1\}$.
    (iii) While $\mathbb{U} \neq \emptyset$:
      (A) Take $U \in \mathbb{U}$ and make $\mathbb{U} = \mathbb{U} \backslash U$.
      (B) Search facets of U, $f | f \in \text{int}(X)$.
      (C) For each f:
        - Search regions from explicit solution to $\mathcal{P}_{T_i}$ adjacent to U through f:S_i.
        - $\mathbb{U} = \mathbb{U} \cup \{S_i \backslash \mathbb{R}\}$.
        - $\mathbb{R} = \mathbb{R} \cup S_i$.
    (iv) $X^* = X \backslash \mathbb{R}$.
    (v) $\mathbb{Y} = \mathbb{Y} \cup \mathbb{R} \cup X^*$.
    (vi) $\mathbb{X} = \mathbb{X} \backslash X$.
  (c) If $\mathbb{X} \neq \emptyset$: go back to 3.b.
  (d) If $\mathbb{X} = \emptyset$:
    - For every region in $\mathbb{Y}$, check if any of the possible solutions can be eliminated (Section 3.3).
    - Find regions in $\mathbb{Y}$ with the same set of explicit solutions, check if union is convex and if so, keep only the union in $\mathbb{Y}$ [4].
    - $\mathbb{X} = \mathbb{Y}$, $\mathbb{T} = \mathbb{T} \backslash T_i$, go back to 3.

such that:

$$s_0 + \sum_i s_i g_i + \sum_{i,j} s_{ij} g_i g_j + \textstyle\sum_{i,j,k} s_{ijk} g_i g_j g_k + \ldots = -1 \tag{6}$$

where g_i are the inequalities in (5).

Positivstellensatz provides strong alternatives, that is, if SOS polynomials s_α satisfying (6) are found, there exists a refutation of (5) and $\mathbf{u}_i^{opt}(x)$, the optimum solution corresponding to objective function $V_{Ni}^{OPT}(x)$, can be discarded. However, for a practical implementation, the degree of these polynomials has to be bounded. Therefore, if SOS polynomials satisfying (6) are not found, it still may be possible that higher degree polynomials exist. Then, feasibility of (5) cannot be assured. This means that we cannot guarantee that $\mathbf{u}_i^{opt}(x)$ is not the optimum for some subset of X_r, and therefore it cannot be discarded.

Obviously, once a given solution is discarded for a given X_r it can be omitted in subsequent feasibility checks, making more simple the subsequent SOS problems to be solved.

4 Online algorithm

For our particular problem $\mathcal{P}_\mathbb{T}$, we have proposed in Section 3 a procedure that finds an explicit solution in the form of a polyhedral partition with several optimal solutions for each region.

Our approach for the online algorithm is, given some x, determine which region of the polyhedral partition it belongs to and later finding which of the possible solutions has a minimum cost.

The first step can be done efficiently by generating a binary search tree. In this context, binary trees were introduced in [14] as an alternative to less efficient sequential search.
Binary trees are data structures formed by several nodes, each of them with a corresponding inequality $d(x)$ and two child nodes, in the case of non-leaf nodes, or a set of possible solutions, in the case of leaf nodes. The tree is traversed by starting from the root node and deciding which child node to go to by checking the sign of the inequality. The process is repeated until a leaf node is encountered.

It is interesting to note here that many of the computationally expensive offline operations to construct the binary tree, i.e. finding the regions of the original partition that satisfy a given set of inequalities, will have already been solved for the algorithm for the union of regions of [4].

For the second step, we propose storing all the objective functions and find the minimum, although other strategies, such as storing difference of objective functions or even building binary trees with quadratic curves inside each linear region, are possible.

4.1 Algorithm performance

As seen in Section 2 problem $\mathcal{P}_\mathbb{T}$ can be rewritten as a set of subproblems with convex constraints $\mathcal{P}_{T_i}$ and a comparison of objective functions (2). These subproblems can be solved by obtaining a polyhedral piecewise affine solution, for which an efficient online algorithm can be developed by computing a binary search tree. Our goal is to compare the performance of such an algorithm with the one proposed in this section. This basic algorithm can be divided in the following tasks: traverse γ binary trees, compute γ objective functions, find the minimum and apply the corresponding affine law. The computational cost of these tasks, formulated in terms of elemental operations (additions, products and comparisons) is:

- $(2n+1)D_i$ for the evaluation of D_i hyperplanes in each binary tree (n additions, n products and a comparison), where D_i is the tree depth.

- $(n+1)(2n+1)-1$ for evaluating each quadratic objective function $x^T Mx + Nx + 1$.
- $\gamma - 1$ comparisons for finding the minimum objective function.
- $2nm$ for the computation of the affine expression.

This way, the total computational cost of the basic algorithm can be expressed as

$$C_1 = (2n+1)\left(\sum_{i=1}^{\gamma} D_i + \gamma(n+1)\right) + 2nm - 1 \quad (7)$$

In our proposed algorithm, we traverse a single binary tree with a depth D, and, in the worst-case, we search the minimum objective function between the maximum number of possible solutions, n_s^{max}

$$C_2 = (2n+1)\left(D + n_s^{max}(n+1)\right) + 2nm - 1 \quad (8)$$

Substracting (8) from (7)

$$C_1 - C_2 = (2n+1)\left(\sum_{i=1}^{\gamma} D_i - D + (\gamma - n_s^{max})(n+1)\right)$$

By definition n_s^{max} is the maximum number of solution in a polyhedral region, and therefore $(\gamma - n_s^{max} \geq 0)$. So

$$C_1 - C_2 \geq (2n+1)\left(\sum_{i=1}^{\gamma} D_i - D\right) \quad (9)$$

On the other hand, from the way the polyhedral partition in our proposed algorithm is constructed, it can be seen that, in the worst case in which every combination of solutions of $\mathcal{P}_{T_i}$ is possible, it has a maximum of $\prod^{\gamma} N_i$ regions, so we can express the total number of regions as $N_t = \eta \prod^{\gamma} N_i$, with $\eta \leq 1$. In [14] it is shown that depth of a binary tree is logarithmic in the number of regions, so we can write

$$D = \left\lceil -\frac{\ln Nt}{\ln \alpha} \right\rceil = \left\lceil \frac{\ln \eta + \sum_{i=1}^{\gamma} \ln N_i}{-\ln \alpha} \right\rceil$$

where $\alpha \in [0.5\ 1)$ is a parameter representing how well the tree is balanced.

From properties of the ceiling function, $\lceil x \rceil + \lceil y \rceil - 1 \leq \lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil$, it follows

$$\left\lceil \frac{\ln \eta}{-\ln \alpha} \right\rceil + \sum_{i=1}^{\gamma} \left\lceil \frac{\ln N_i}{-\ln \alpha} \right\rceil - \gamma \leq D \leq \left\lceil \frac{\ln \eta}{-\ln \alpha} \right\rceil + \sum_{i=1}^{\gamma} \left\lceil \frac{\ln N_i}{-\ln \alpha} \right\rceil \quad (10)$$

As α is unknown until the binary tree is built, authors in [14] propose as a conservative estimate $\alpha = 2/3$. If we consider the same estimate for every tree, we have

$$\left\lceil \frac{\ln N_i}{-\ln \alpha} \right\rceil = D_i$$

Substituting in (10) and taking into account $\lceil -x \rceil = -\lfloor x \rfloor$

$$\left\lfloor \frac{\ln \eta}{\ln \alpha} \right\rfloor \leq \sum_{i=1}^{\gamma} D_i - D \leq \left\lfloor \frac{\ln \eta}{\ln \alpha} \right\rfloor + \gamma$$

As $\eta \leq 1$ and $\alpha \leq 1$, it follows $\left\lfloor \frac{\ln \eta}{\ln \alpha} \right\rfloor \geq 0$.

Substituting in (9) we have

$$C_1 - C_2 \geq (2n+1)\left\lfloor \frac{\ln \eta}{\ln \alpha} \right\rfloor \geq 0$$

i.e., our algorithm has a lower computational cost.

Apart from the computational cost, another fundamental criterion to analyze the algorithm performance is the memory requirement, which is often the critical bottleneck for explicit MPC [6].

For binary trees, an efficient way to store all the necessary data is by means of a table that contains all the hyperplanes, another table containing all the control laws, and pointers to the table of hyperplanes and the child nodes for non-leaf nodes, and to the control law table for leaf nodes as in [14].

For our basic algorithm we need to store γ binary trees, but also all the quadratic objective functions that will have to be compared.

Our proposed algorithm, on the other hand, saves memory due to the following advantages: elimination of suboptimal control laws, elimination of hyperplanes from the explicit solution to problems $\mathcal{P}_{T_i}$ in the process of union of regions and, most importantly, the fact that only a small subset of objective functions need to be stored (inside non-subset regions with more than one possible solution).

5 Example

In this section, the control of an inverted pendulum is proposed. The system is described by the non-linear equations:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{g\sin(x_1) - aml x_2^2 \sin(2x_1)/2}{4l/3 - aml\cos^2(x_1)} + \frac{-a\cos(x_1)}{4l/3 - aml\cos^2(x_1)} u \\
y &= h(x) = x_1
\end{aligned}$$

where x_1 is the pendulum vertical angle, x_2 the angular speed, $g = 9.8\, m/s^2$, $m = 2\, kg$ the pendulum mass, $2l = 4\, m$ the pendulum length, u the force applied to the cart (in Newtons) and $a = 1/(m+M)$, where $M = 8\, kg$ is the mass of the cart. We consider constraints on the applied force of $-300 \leq u \leq 300$.

In order to control the system, input-output feedback linearization is applied. The linearizing control law is defined as:

$$u = \frac{4l/3 - amlcos^2(x_1)}{-acos(x_1)} \left(\frac{amlx_2^2 \sin(2x_1)/2 - g\sin(x_1)}{4l/3 - amlcos^2(x_1)} + v \right)$$

Once the constraints are translated to the new input v, a non-linear function depending on x is obtained. These constraints are then approximated by three convex interior polyhedra (figure 1). The system is sampled with

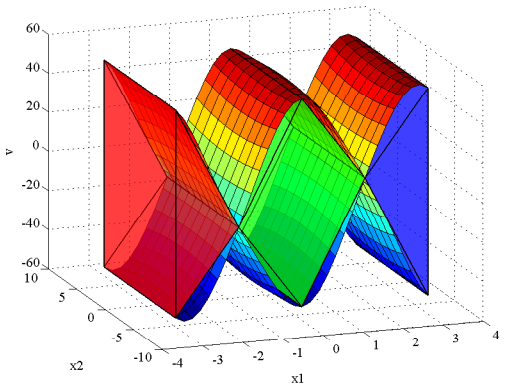

Fig. 1. Non-linear constraints and non-convex polyhedral approximation.

$T_s = 50ms$ for the design of an MPC with parameters $R = 1$, $Q = diag(10, 1)$ and $N = 7$. Next, a terminal region and controller for guaranteeing stability are computed following [11].

At this point, for the selected horizon constraint set $\mathbb{T}$ is computed, defined by the union of 15 convex T_i sets. The proposed methodology is started by computing the convex hull and solving the corresponding mpQP (67 regions). Next, Algorithm 1 is applied for the non-subset regions. After solving 2082 SOS problems, 1011 non-optimal solutions are eliminated. Finally a polyhedral partition of 510 regions is found (441 of these regions with just one solution). This explicit solution is shown in Figure 2.

A simulation of the closed-loop system from the stable equilibrium point $(\pi, 0)$ is performed. The trajectory is shown in Figure 2 (dashed line). It can be seen that the system is correctly driven to the origin.

The proposed controller is compared with standard MPC approaches with convex constraints. If they are

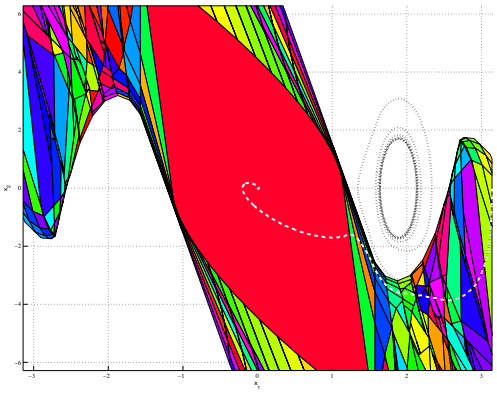

Fig. 2. Final polyhedral partition, closed-trajectory with proposed MPC (dashed line) and closed-loop trajectory with standard MPC (dotted line).

defined as the central region in Figure 1, it does not exist any control action v outside $|x_1| \leq \pi/2$. On the other hand, if they are defined as the convex hull of the initial constraints, stability cannot be guaranteed for saturated control actions, as seen in the trajectory in Figure 2 (dotted line).

Furthermore, the problem can also be formulated as a PWA system and solved by the approach of [2], implemented in the *MATLAB MPT toolbox* [7]. The explicit solution consists of 679 overlapping regions. When implemented online following [2], the affine expressions have to be stored in an ordered way. Therefore, the worst-case online cost is linear in the number of regions instead of logarithmic, requiring the computation of 13453 elemental operations.

For our proposed algorithm, an efficient online performance can be achieved by computing a binary tree of the polyhedral partition, obtaining a depth $D = 13$ and $C_2 = 98$ elemental operations. The algorithm based on solving subproblems $\mathcal{P}_{T_i}$, needs the computation of 15 binary trees leading to $C_1 = 738$ operations. This way, the computational effort is reduced an 85.7%. Memory requirements are 8388 real numbers and 4279 pointers for the basic algorithm, and 2949 real numbers and 3553 pointers for our proposal, reducing a 55.1% when implemented in a 16-bit microcontroller.

6 Conclusions

In this paper, we have presented a procedure to obtain an explicit solution to the problem of MPC with non-convex polyhedral constraints. An explicit solution of the problem which constraints are the convex hull of the non-convex polyhedra is computed. Then, a classification and division of the regions of this explicit solution is performed, and non-optimal solutions in each of the regions are eliminated by means of SOS programming. An

online algorithm that exploits the structure of the obtained explicit solution has also been presented, and its performance compared to an enumeration algorithm. By theoretical analysis and example results, the presented approach has been shown to significantly improve this enumeration algorithm both in terms of computational effort and memory requirements.

Acknowledgment

This work was partially funded by projects DPI2008-02133 and DPI2008-06731-C02-01(and -02)/DPI Ministerio de Ciencia e Innovación - Spanish Government.

References

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

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

[3] K. P. Fruzzetti, A. Palazoglu, and K. A. McDonald. Nonlinear model predictive control using Hammerstein models. *Journal of Process Control*, 7(1):31–41, 1997.

[4] T. Geyer, F. D. Torrisi, and M. Morari. Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, 44(7):1728–1740, 2008.

[5] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40(4):701–708, 2004.

[6] T. A. Johansen, W. Jackson, R. Schreiber, and P. Tøndel. Hardware synthesis of explicit model predictive controllers. *IEEE Transactions on Control Systems Technology*, 15(1):191, 2007.

[7] M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-parametric toolbox (MPT). *Hybrid Systems: Computation and Control*, 2993:448–462, 2004.

[8] D. Q. Mayne and S. V. Rakovic. Optimal control of constrained piecewise affine discrete time systems using reverse transformation. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 2, 2002.

[9] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[10] E. Pérez. *Control Predictivo sujeto a restricciones poliédricas no convexas: Solución explícita y estabilidad*. PhD thesis, Universidad Politécnica de Valencia, 2011.

[11] E. Pérez, C. Ariño, F. X. Blasco, and M. A. Martínez. Maximal closed loop admissible set for linear systems with non-convex polyhedral constraints. *Journal of Process Control*, 21(4):529–537, 2011.

[12] J. J. E. Slotine and W. Li. *Applied nonlinear control*. Prentice-Hall Englewood Cliffs, NJ, 1991.

[13] 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*, 42(12):2209–2214, 2006.

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