52nd IEEE Conference on Decision and Control
December 10-13, 2013. Florence, Italy

Polygonic Representation of Explicit Model Predictive Control

Juraj Oravec, Slavomír Blažek, Michal Kvasnica, and Stefano Di Cairano

Abstract— **The paper proposes to reduce complexity of explicit MPC feedback laws by representing regions over which the law is defined as (possibly non-convex) polygons. Each polygon is then represented only by its boundaries, which reduces the memory footprint of the feedback law. Even though significant amount of memory can be saved this way, the price to be paid is increased computational load associated by performing point location tasks in non-convex objects. Therefore we propose to devise inner and outer convex approximations of non-convex polygons to reduce the computational requirements. Such approximations allow to perform point location more effectively, leading to a reduction of the required on-line computational effort. Several ways to design suitable approximations are presented and efficacy of the proposed procedure is evaluated.**

I. INTRODUCTION

Pioneered by [2], Explicit Model Predictive Control (MPC) has gained significant attention among theoreticians and control engineers because it allows to implement MPC on cheap control hardware. Previously, this was not easily achievable, because MPC requires a numerical optimization algorithm to be executed at each sampling instant to obtain optimal control actions. In explicit MPC, the need for repetitive optimization is eliminated by shifting the optimization offline. Several authors (see e.g. [4], [11], [9]) have shown how to pre-compute the solution to a given optimization problem offline using multi-parametric programming, which gives rise to an explicit representation of the MPC feedback law. For a rich class of MPC problems it can be also shown that the feedback law takes a form of a Piecewise Affine (PWA) function which maps measurements onto the optimal control inputs. Hence closed-loop implementation of explicit MPC reduces to a mere function evaluation.

However, for large systems and/or long horizons, complexity of the explicit PWA feedback law often exceeds capabilities of the implementation hardware either in terms of computational load, memory storage, or both. Therefore it is important to keep the complexity of explicit MPC solutions on an acceptable level. The problem of reducing complexity of explicit MPC solutions has thus attracted numerous researchers in the past 10 years. Two main ways to reduce complexity are reported in the literature. The first approach is based on replacing the original PWA feedback law by a simpler function while sacrificing optimality. Hence a simpler, yet sub-optimal feedback law can be obtained. In some practical cases, however, the induced loss of optimality is inacceptable. Therefore the second main line of research is devoted to simplifying the PWA feedback law while preserving optimality. Many approaches have been developed to achieve this goal and the reader is referred to [6] for a concise overview.

Recently, in [3] we have shown how to reduce the memory footprint of an explicit MPC solution by representing regions of the PWA feedback as (possibly non-convex) polygons. Moreover, we have shown that implementation of such a polygonic feedback law only requires storing a subset of regions of the original feedback. Although significant amount of memory can be saved by this approach, it suffers from the increase of on-line computational complexity needed to evaluate the polygonic PWA function. Moreover, application of [3] is limited to 2-dimensional problems. In this paper we address these deficiencies and show how to significantly reduce the on-line computational effort. This is achieved by devising outer and inner convex approximations of polygonic regions. These approximations allow to significantly speed up the task of deciding whether a particular point belongs to a polygon or not. Specifically, the outer approximation serves as a necessary condition for such a test, while the inner approximation constitutes sufficiency. To render these conditions as efficient as possible, the two approximations must be designed such that their volume is either minimized (for outer approximations) or maximized (for inner approximations). Several procedures for designing such approximations are presented in Section VI.

II. NOTATION AND DEFINITIONS

For a set $\mathcal{S}$, $\text{int}(\mathcal{S})$ denotes its interior, $\partial\mathcal{S}$ its boundary and $\text{cl}(\mathcal{S})$ its closure. For a matrix or a vector H, $[H]_i$ denotes its i-th row. A polyhedron $\mathcal{Q} = \{x \in \mathbb{R}^n \mid Hx \leq k\}$ is a convex set that is defined as the intersection of a finite number of affine, closed half-spaces. A bounded polyhedron is called polytope. A polygon $\mathcal{P} \subset \mathbb{R}^n$ is a (possibly non-convex) set that is bounded by a closed path, composed of a finite number of $n-1$ dimensional polytopes. We call the set of polytopes $\{\mathcal{Q}_i\}$, $i = 1, \ldots, R$ the geometric subdivision of a polygon $\mathcal{P}$ if $\mathcal{P} = \cup_i \mathcal{Q}_i$ and $\text{int}(\mathcal{Q}_i) \cap \text{int}(\mathcal{Q}_j) = \emptyset, \forall i \neq j$. Every polytope $\mathcal{Q} \subset \mathbb{R}^n$ can be equivalently represented by $\mathcal{Q} = \text{convh}(v_1, \ldots, v_M)$, where convh is the convex hull operator and $v_i \in \mathbb{R}^n$, $i = 1, \ldots, M$ are vertices of the polytope.

III. EXPLICIT MODEL PREDICTIVE CONTROL

We consider linear time-invariant systems in the discrete-time domain, described by state-space models of the form

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

J. Oravec, S. Blažek, and M. Kvasnica are with the Slovak University of Technology in Bratislava, Slovakia, {juraj.oravec, slavomir.blazek,michal.kvasnica}@stuba.sk. S. Di Cairano is with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, dicairano@ieee.org.

where $x(t) \in \mathbb{R}^n$ is the state vector at time t and $u(t) \in \mathbb{R}^m$ is the vector of control commands. States and inputs are constrained by

$$x \in \mathcal{X},\ u \in \mathcal{U}, \tag{2}$$

where $\mathcal{X}$ and $\mathcal{U}$ are polytopes of appropriate dimensions, containing the origin in their respective interiors. Furthermore, we assume that the pair (A, B) is controllable and that state measurements $x(t)$ are available at each time instant t.

For system (1) subject to constraints (2), the MPC optimization problem becomes

$$U_N^* = \arg\min \|Q_N x_N\|_p + \sum_{k=0}^{N-1} \|Q_x x_k\|_p + \|Q_u u_k\|_p \tag{3a}$$

$$\text{s.t. } x_0 = x(t), \tag{3b}$$

$$x_{k+1} = Ax_k + Bu_k,\ k = 0, \dots, N-1, \tag{3c}$$

$$x_k \in \mathcal{X},\ k = 0, \dots, N, \tag{3d}$$

$$u_k \in \mathcal{U},\ k = 0, \dots, N-1, \tag{3e}$$

where x_k and u_k denote, respectively, the state and input predictions at time instant $t + k$, initialized by the measurements of the current state $x(t)$. Moreover, Q_N, Q_x and Q_u are penalty matrices of suitable dimensions and p indicates a norm in which corresponding quantities should be minimized. If $p = 2$ then $Q_N \succeq 0$, $Q_x \succeq 0$, $Q_u \succ 0$ is required, and instead of $\|Qz\|_2$ we can equivalently minimize $\|Qx\|_2^2$. Solving the optimal control problem (3) for a particular initial condition $x(t)$ leads to the sequence of optimal control inputs $U_N^* = \{u_0^*, \dots, u_{N-1}^*\}$, defined over the prediction horizon N. In the receding horizon implementation, only the first element of U_N^*, that is, u_0^*, is actually implemented to the controlled plant and the whole procedure is repeated at the next sampling instant for new values of the state measurements.

By solving (3) using parametric programming [2], the optimal receding horizon control action u_0^* can be precomputed [4] for all feasible values of $x(t)$ as a PWA function of the form

$$u_0^* = \kappa(x(t)) := \begin{cases} F_1 x(t) + g_1 & \text{if } x(t) \in \mathcal{Q}_1 \\ & \vdots \\ F_M x(t) + g_M & \text{if } x(t) \in \mathcal{Q}_M \end{cases} \tag{4}$$

where $F_i \in \mathbb{R}^{m \times n}$, $g_i \in \mathbb{R}^m$, and M denotes the total number of polytopes

$$\mathcal{Q}_i = \{x \,|\, H_i x \le k_i\}. \tag{5}$$

Each polytope is constituted by c_i half-spaces $[H_i]_j x \le [k_i]_j$, $j = 1, \dots, c_i$.

For given state measurements $x(t)$, the value of u_0^* can be obtained by evaluating the function $\kappa(\cdot)$ using e.g. the sequential search procedure. Its best-case runtime complexity of $\mathcal{O}(1)$ when $x(t) \in \mathcal{Q}_1$, and the worst case is $\mathcal{O}(M)$ if $x(t) \in \mathcal{Q}_M$ or $x(t) \notin \cup_i \mathcal{Q}_i$.

However, to evaluate $\kappa(\cdot)$ for a given value of $x(t)$, the function first needs to be stored in the memory of the control hardware platform. The memory footprint of $\kappa(\cdot)$ (which consists of polytopes $\mathcal{Q}_i$ and feedback gains F_i, g_i, $i = 1, \dots, M$) is a linear function of M, the total number of polytopes. Clearly, the more polytopes constitute $\kappa(\cdot)$, the more memory needs to be available in the hardware implementation platform, sooner or later hitting the hardware limits.

IV. Polygonic representation of Explicit MPC

The objective is to reduce memory consumption of explicit feedback laws $\kappa(\cdot)$ by exploiting the fact that, in practice, several polytopes of (4) will be associated to identical feedback laws. Denote by M_u the number of unique feedback gains and let $\mathcal{I}_j \subseteq \{1, \dots, M\}$ be the index set of polytopes Q_i which share the j-th feedback, i.e.

$$\mathcal{I}_j = \{i \in \{1, \dots, M\} \mid F_i = F_j,\ g_i = g_j\},\ j = 1, \dots, M_u. \tag{6}$$

Let

$$\mathcal{P}_j = \bigcup_{i \in \mathcal{I}_j} \mathcal{Q}_i \tag{7}$$

be a polygon whose geometric subdivision is given by polytopes $\mathcal{Q}_i$, $\forall i \in \mathcal{I}_j$. Then we can rewrite $\kappa(\cdot)$ in (4) as

$$\tilde{\kappa}(x(t)) := \begin{cases} F_1 x(t) + g_1 & \text{if } x(t) \in \mathcal{P}_1 \\ & \vdots \\ F_{M_u} x(t) + g_{M_u} & \text{if } x(t) \in \mathcal{P}_{M_u} \end{cases} \tag{8}$$

We remark that $\cup_j \mathcal{I}_j = \{1, \dots, M\}$. Naturally, $M_u \le M$ always holds, with $M_u \ll M$ often being the case in practice.

Lemma 4.1: For all x from the domain of $\kappa(\cdot)$ we have $\kappa(x) = \tilde{\kappa}(x)$. ■

Evaluation of $\tilde{\kappa}(\cdot)$ from (8) for a given value of $x(t)$ can be done by a straightforward modification of the sequential search approach, reported for completeness as Algorithm 1.

Algorithm 1 Evaluation of $\tilde{\kappa}$ from (8)

INPUT: Polygons $\mathcal{P}_j$, unique feedback gains F_j, g_j, number of unique gains M_u, state measurement $x(t)$
OUTPUT: Optimal RHMPC control input u_0^*

```
1: for $j = 1, \dots, M_u$ do
2:     if $x(t) \in \mathcal{P}_j$ then
3:         $u_0^* = F_j x(t) + g_j$
4:         return
5:     end if
6: end for
```

To obtain the value of $\tilde{\kappa}(x(t))$ for a particular point $x(t)$, Algorithm 1 needs to determine, in Step 2, whether $x(t) \in \mathcal{P}_j$, a task commonly known as the *point location problem*. Since the polygon $\mathcal{P}_j$ can be a non-convex set, in general, the task of deciding whether $x(t)$ belongs to $\mathcal{P}_j$ becomes challenging. The point location problem is formally stated as follows:

Problem 4.2: Given a polygon $\mathcal{P} \subset \mathbb{R}^n$ and a point $x \in \mathbb{R}^n$, determine whether $x \in \mathcal{P}$. □

V. Memory-Efficient Point Location in Polygons

The point location task of Problem 4.2 can be performed by applying the sequential search approach to the geometric subdivision $\{\mathcal{Q}_i\}$ that constitutes the polygon $\mathcal{P}_j$. Doing so, however, would require storing all polytopes $\mathcal{Q}_i$ in the memory of the control hardware, which does not provide any improvement upon the straightforward implementation of MPC feedback law in (4) via sequential search. Therefore in this section we show how Problem 4.2 can be solved by only considering a subset of polytopes $\mathcal{Q}_i$. Specifically, we illustrate that only the polytopes that touch the boundary of $\mathcal{P}_j$ need to be stored. Let M_j be the number of polytopes $\mathcal{Q}_i$ that constitute $\mathcal{P}_j$ in (7), and let M_b be the number of polytopes touching the boundary $\partial\mathcal{P}_j$. Then clearly $M_b \leq M_j$ always holds in theory, while often, in practice, $M_b \ll M_j$. Therefore the ratio M_j/M_b represents the achievable reduction of memory consumption achievable by the proposed method.

To simplify presentation, we will henceforth consider a single polygon $\mathcal{P}$, composed of M polytopes $\mathcal{Q}_i = \{x \,|\, H_i x \leq k_i\}$, $i = 1, \ldots, M$. Since $\{\mathcal{Q}_i\}$ is assumed to be a geometric subdivision of $\mathcal{P}$, the boundary $\partial\mathcal{P}$ is composed of a finite number of $n-1$ dimensional faces $\mathcal{F}_j$ with

$$\mathcal{F}_j = \{x \mid H_j x \leq k_j,\ a_j^T x = b_j\}, \tag{9}$$

with $j = 1, \ldots, M_f$, where M_f is the total number of boundary faces. We remark that, in the worst case, $M_f = \sum_{i=1}^{M_b} c_i$, where M_b is the number of boundary polytopes $\mathcal{Q}_i$, and c_i is the number of faces of the i-th polytope.

The polytopes $\mathcal{F}_j$ that constitute the boundary of $\mathcal{P}$ can be obtained in a number of ways. In general, one can take $\partial\mathcal{P} = \mathcal{P} \cap \mathrm{cl}(\mathbb{R}^n \setminus \mathcal{P})$. Since $\mathcal{P} = \cup_i \mathcal{Q}_i$, the operation results in computing the difference between two sets of polyhedra. Such a computation can be carried out using version 3.0 of the MPT Toolbox [7], which directly splits $\partial\mathcal{P}$ into polytopes $\mathcal{F}_j$. Alternatively, if polytopes $\mathcal{Q}_i$ of (7) satisfy the facet-to-facet property [9], then the boundary $\partial\mathcal{P}$ is composed of the boundary faces of polytopes for which $\mathcal{Q}_i \cap \partial\mathcal{P} \neq \emptyset$. These can be identified by solving a single linear program per each face.

To solve the point location task of Problem 4.2 using only the boundary of $\mathcal{P}$, we exploit the well-known Jordan-Brouwer theorem, which is a generalization of the Jordan separation theorem to dimensions above 2:

Lemma 5.1 ([1]): Let $\mathcal{P} \subset \mathbb{R}^n$ be a polygon as in (7), $x \in \mathbb{R}^n$ be the query point, and $\gamma \in \mathbb{R}^n$ be an arbitrary vector. Define by

$$\mathcal{R} = \{x + \theta\gamma \mid \theta \in \mathbb{R},\ \theta \geq 0\} \tag{10}$$

a ray emitted from x in the direction of the vector γ. Then $x \in \mathcal{P}$ if and only if the number of intersections between $\partial\mathcal{P}$ and $\mathcal{R}$ is odd. ■

Since $\partial\mathcal{P} = \cup_j \mathcal{F}_j$, using Lemma 5.1 we can thus answer 4.2 by counting the number of intersections between the ray (10) and $\cup_j \mathcal{F}_j$. Since each $\mathcal{F}_j$ is a polytope as in (9), determining whether $\mathcal{R} \cap \mathcal{F}_j = \emptyset$ can be performed efficiently. In particular, note that

$$\mathcal{R} \cap \mathcal{F}_j = \{\theta \,|\, H_j(x + \theta\gamma) \leq k_j,\ a_j^T(x + \theta\gamma) = b_j,\ \theta \geq 0\}. \tag{11}$$

Each admissible θ in (11) has to satisfy $H_j(x + \theta\gamma) \leq k_j$, i.e.,

$$\theta \leq \frac{[k_j]_\ell - [H_j]_\ell x}{[H_j]_\ell \gamma}, \tag{12}$$

where $[\cdot]_\ell$ is the ℓ-th row of the corresponding vector/matrix. Assume that γ is in general position, i.e., for each $\ell \in \{1, \ldots, c_j\}$ we have $[H_j]_\ell \gamma \neq 0$ where c_j is the number of rows of H_j. Then, depending on the sign of the denominator in (12), we can bound feasible values of θ by $\underline{\theta} \leq \theta \leq \overline{\theta}$ with

$$\underline{\theta} = \max_{\ell \in \mathcal{J}_{\text{neg}}} \left\{ \frac{[k_j]_\ell - [H_j]_\ell x}{[H_j]_\ell \gamma} \right\},\ \overline{\theta} = \min_{\ell \in \mathcal{J}_{\text{pos}}} \left\{ \frac{[k_j]_\ell - [H_j]_\ell x}{[H_j]_\ell \gamma} \right\}, \tag{13}$$

where $\mathcal{J}_{\text{pos}}$ is the index set of rows of H_j for which $[H_j]_\ell \gamma$ is positive, and $\mathcal{J}_{\text{neg}}$ is the index set of negative denominators in (12). Note that the maxima and minima in (13) are taken over finite sets, hence no optimization is needed to compute them. Then the number of intersections between the ray in (10) and $\partial\mathcal{P} = \cup_j \mathcal{F}_j$ can be counted, and hence $x \in \mathcal{P}$ be verified, by running Algorithm 2.

Algorithm 2 Test for $x \in \mathcal{P}$ using the boundary of $\mathcal{P}$.

```
INPUT: Point $x$, direction $\gamma$, polytopes $\mathcal{F}_j$ constituting $\partial\mathcal{P}$.
OUTPUT: True if $x \in \mathcal{P}$, false otherwise.
1: $c \leftarrow 0$.
2: for each polytope $\mathcal{F}_j$ in (9) do
3:     Compute $\underline{\theta}$ and $\overline{\theta}$ from (13).
4:     if $\overline{\theta} \geq 0$ and $\overline{\theta} > \underline{\theta}$ and either $a_j^T(x + \underline{\theta}\gamma) = b_j$ or
       $a_j^T(x + \overline{\theta}\gamma) = b_j$ then
5:         $c \leftarrow c + 1$
6:     end if
7: end for
8: return true if $c$ is odd, false otherwise
```

Lemma 5.2: Algorithm 2 returns true if $x \in \mathcal{P}$ and false otherwise. ■

Runtime complexity of Algorithm 2 is proportional to M_f, the number of faces (9) that form $\partial\mathcal{P}$. Since M_f is bounded from above by $\sum_{i=1}^{M_b} c_i$ with M_b denoting the number of polytopes Q_i that touch $\partial\mathcal{P}$, the runtime complexity is $\mathcal{O}(M_b)$. It is worth noting that this is both the best-case and worst-case complexity, regardless of the location of the query point. This is due to the fact that each execution of Alg. 2 has to proceed through all boundary faces before answering the point location query.

VI. Runtime-Efficient Point Location in Polygons via Approximations

In the previous section we have demonstrated how to perform the point location task using only the boundary of a polygon in (7). The advantage is in reduced memory storage, but the downside is that Alg. 2 *always* performs $\mathcal{O}(M_b)$

operations before answering the $x \in \mathcal{P}$ query, regardless of the position of the query point x. In this section we aim at improving the runtime efficiency of the point location task. In particular, would like to answer the query $x \in \mathcal{P}$ more efficiently for as many points x as possible.

Assume that two sets, namely $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$, exist such that

$$\mathcal{P}_{\text{in}} \subseteq \mathcal{P} \subseteq \mathcal{P}_{\text{out}}. \tag{14}$$

Then instead of using Alg. 2 to answer $x \in \mathcal{P}$ in Step 2 of Alg. 1, we can answer the point location query more efficiently as follows. For any $x \in \mathcal{P}_{\text{in}}$ we have $x \in \mathcal{P}$ due to $\mathcal{P}_{\text{in}} \subseteq \mathcal{P}$, and hence the answer to the query in Step 2 of Alg. 1 is positive even without further investigating $\mathcal{P}$. Similarly, if $x \notin \mathcal{P}_{\text{out}}$, then $x \notin \mathcal{P}$ due to $\mathcal{P}_{\text{out}} \supseteq \mathcal{P}$ and therefore the answer to $x \in \mathcal{P}$ is negative. Only for points with $x \in \mathcal{P}_{\text{out}}$ and $x \notin \mathcal{P}_{\text{in}}$ we need to perform the point location per Alg. 2. This reasoning is described in Algorithm 3.

Algorithm 3 More efficient test for $x \in \mathcal{P}$

INPUT: Point x, polygon $\mathcal{P}$, sets $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$.
OUTPUT: True if $x \in \mathcal{P}$, false otherwise.

```
1: if $x \in \mathcal{P}_{\text{in}}$ then
2:     return true
3: else if $x \notin \mathcal{P}_{\text{out}}$ then
4:     return false
5: else
6:     Determine whether $x \in \mathcal{P}$ using Algorithm 2.
7: end if
```

If the sets $\mathcal{P}$, $\mathcal{P}_{\text{in}}$, and $\mathcal{P}_{\text{out}}$ are all non-convex, Algorithm 3 does not provide any improvement upon Alg. 2. However, if $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{out}}$ are both convex (or if they consist of a finite number of convex sets), then Alg. 3 renders Problem 4.2 easier to solve. Specifically, it is worth noting that the two inclusion tests in Steps 1 and 3 can be performed efficiently if $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$ are convex sets. By employing the outer and inner approximations of a (possibly non-convex) polygon $\mathcal{P}$, one can therefore reduce the number of expensive point locations performed in Step 6.

Remark 6.1: It should be noted that the *worst-case* complexity of Alg. 3 is higher compared to runtime complexity of Alg. 2. This is due to the fact that for any x with $x \notin \mathcal{P}_{\text{in}}$ and $x \in \mathcal{P}_{\text{out}}$, Alg. 3 performs two additional tests[1] in Steps 1 and 3. However, Alg. 3 performs better *on average* since the algorithm can often be stopped upon satisfaction of cheap tests in Steps 1 and 3. Average performance is very important in practical applications, as it directly correlates with occupancy of central processing units and hence with thermal load and energy consumption. □

To exploit Algorithm 3, one first needs to design the outer and inner convex approximations $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$ such that (14) holds. Moreover, efficacy of Alg. 3 depends on the volume of $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$. The inner approximation $\mathcal{P}_{\text{in}}$ should be as large as possible, as to maximize the likelihood of terminating Alg. 3 in Step 2 if $x \in \mathcal{P}_{\text{in}}$. On the other hand, the smaller the volume of $\mathcal{P}_{\text{out}}$, the more efficient the test in Step 3 is. This is due to the fact that $\mathcal{P}_{\text{out}}$ is an outer approximation and represents the necessary condition for the inclusion test $x \in \mathcal{P}$. Therefore the smaller $\mathcal{P}_{\text{out}}$ is, the more points x with $x \notin \mathcal{P}_{\text{out}}$ can be ruled out in Step 3.

[1]In practice, the cost of checks in Steps 1 and 3 of Alg. 3 is negligible compared to the cost of Step 6.

Several ways to design inner and outer approximations with these properties are outlined in the sequel. The standing assumption of the remainder of this section is that the polygon $\mathcal{P}$ is represented by its geometric subdivision of the form of (7), i.e., $\mathcal{P} = \cup_i \mathcal{Q}_i$, where $\mathcal{Q}_i$, $i = 1, \ldots, R$ are polytopes.

A. Inner Approximation

The problem of finding a suitable inner convex approximation $\mathcal{P}_{\text{in}}$ can be formally stated as follows.

Problem 6.2: Given a non-convex polygon $\mathcal{P} = \cup_i \mathcal{Q}_i$ where $\mathcal{Q}_i = \{x \mid H_i x \leq k_i\}$, $i = 1, \ldots, R$ are polytopes, find a convex set $\mathcal{P}_{\text{in}}$ that satisfies $\mathcal{P}_{\text{in}} \subseteq \mathcal{P}$ and has a large volume.

The main difficulty is to guarantee that $\mathcal{P}_{\text{in}} \subseteq \mathcal{P}$ (which means that $\forall x \in \mathcal{P}_{\text{in}}$ we have $x \in \mathcal{P}$) when $\mathcal{P}$ is non-convex. Let $\mathcal{C} = \mathbb{R}^n \setminus \mathcal{P}$ denote the complement of the polygon $\mathcal{P}$. Then $\mathcal{P}_{\text{in}} \subseteq \mathcal{P}$ is equivalent to

$$\forall x \in \mathcal{C} \Rightarrow x \notin \mathcal{P}_{\text{in}}. \tag{15}$$

Remark 6.3: Since $\mathcal{P}$ is a geometric subdivision as in (7), its complement $\mathcal{C}$ is an another geometric subdivision, i.e., $\mathcal{C} = \{\mathcal{R}_i\}$, where $\mathcal{R}_i = \{x \mid C_i x \leq d_i\}$, $i = 1, \ldots, M_C$ are polyhedra, see e.g. [4]. □

In what follows we search for an ellipsoidal form of $\mathcal{P}_{\text{in}}$, i.e.,

$$\mathcal{P}_{\text{in}} = \{x \mid (x - x_c)^T P^{-1} (x - x_c) \leq 1\}, \tag{16}$$

where $x_c \in \mathbb{R}^n$ denotes the center of the ellipsoid and $P \in \mathbb{R}^{n \times n}$ is a symmetric, positive definite matrix, that is, $P = P^T \succ 0$. We seek for x_c and P such that the volume of $\mathcal{P}_{\text{in}}$ in (16) is maximized while satisfying $\mathcal{P}_{\text{in}} \subseteq \mathcal{P}$ via (15).

To pose the search for the ellipsoid (16) as a convex optimization problem, we exploit the *S-procedure*:

Lemma 6.4 ([5]): Let $f_0(x)$ and $f_i(x)$, $i = 1, \ldots, M$ be quadratic functions. Then

$$f_i(x) \leq 0 \Rightarrow f_0(x) \geq 0 \tag{17}$$

holds for all $x \in \mathbb{R}^n$ and for all $i = 1, \ldots, M$ if and only if there exist non-negative scalars $\lambda_i \geq 0$, $i = 1, \ldots, M$ such that

$$f_0(x) + \textstyle\sum_{i=1}^{M} \lambda_i f_i(x) \geq 0. \tag{18}$$

■

To see the relation between Lemma 6.4 and Problem 6.2, first note that the complement $\mathcal{C}$ is divided into a finite number of polyhedra $\mathcal{R}_i$. Then the implication in (15) can be expanded to $\forall x \in \mathcal{R}_i \Rightarrow x \notin \mathcal{P}_{\text{in}}$, which has to hold for all $i = 1, \ldots, M_C$. Since $\mathcal{R}_i = \{x \mid C_i x \leq d_i\}$, we can set $f_i(x) := C_i x - d_i$. By reversing the inequality in (16), $x \notin \mathcal{P}_{\text{in}}$ can be cast as

$$\underbrace{(x - x_c)^T P^{-1}(x - x_c) - 1 - \epsilon}_{f_0(x)} \geq 0. \tag{19}$$

Here, $\epsilon > 0$ denotes a small positive tolerance used to rewrite strict inequality $g(x) > 0$ to the non-strict form $g(x) \geq \epsilon$. Finally, note that any quadratic function $g(x) = x^T W x + 2w^T x + z$ can be written in a compact form as

$$g(x) := \begin{bmatrix} x \\ 1 \end{bmatrix}^T \underbrace{\begin{bmatrix} W & w \\ w^T & z \end{bmatrix}}_{\tilde{W}} \begin{bmatrix} x \\ 1 \end{bmatrix}. \tag{20}$$

It is well known that with $g(x)$ as in (20), $g(x) \geq 0$ holds for all x if and only if the matrix $\tilde{W}$ is positive semi-definite. Based on this, and by applying the S-procedure of Lemma 6.4 with $f_0(x)$ and $f_i(x)$ defined as above, we can hence formulate the search for P and x_c in (16) as

$$\begin{bmatrix} \tilde{P} & -x_c^T \tilde{P} \\ \tilde{P} x_c & x_c^T \tilde{P} x_c - 1 - \epsilon \end{bmatrix} + \sum_{i=1}^{M_C} \lambda_i \begin{bmatrix} 0 & 1/2 C_i^T \\ 1/2 C_i & -d_i \end{bmatrix} \succeq 0, \tag{21}$$

where $\tilde{P} = P^{-1}$. When x_c is fixed, the maximum-volume ellipsoid (16) can be found by searching for $\tilde{P} = \tilde{P}^T \succeq 0$, $\lambda_i \geq 0$, $i = 1, \ldots, M_C$, while minimizing the trace of $\tilde{P}$:

$$\min_{\tilde{P},\, \lambda_i} \operatorname{trace}(\tilde{P}) \text{ s.t. } \lambda_i \geq 0, \text{ (21) holds, } i = 1, \ldots, M_C. \tag{22}$$

Note that (22), with x_c being fixed, is a convex LMI optimization problem that can be solved efficiently using off-the-shelf software (e.g. with [10]). Once $\tilde{P}$ is computed via (22), P in (16) is recovered by $P = \tilde{P}^{-1}$.

B. Outer Approximation

To search for a suitable outer approximation $\mathcal{P}_{\text{out}}$, we need to solve the following problem.

Problem 6.5: Given is a non-convex polygon $\mathcal{P} = \cup_i \mathcal{Q}_i$. Find a convex set $\mathcal{P}_{\text{out}}$ that satisfies $\mathcal{P}_{\text{out}} \supseteq \mathcal{P}$ and has a small volume. □

Since $\mathcal{P}_{\text{out}}$ needs to be convex, we restrict ourselves to two classes of convex sets: polytopes and ellipsoids. Let $\mathcal{V}_i = \{v_{i,1}, \ldots, v_{i,n_{v_i}}\}$ denote the vertices of polytope $\mathcal{Q}_i$. Then

$$\mathcal{P}_{\text{out}} = \operatorname{convh}\{\mathcal{V}_1, \ldots, \mathcal{V}_R\}, \tag{23}$$

is the smallest polytope that contains $\mathcal{P}$, i.e., $\mathcal{P}_{\text{out}} \supseteq \mathcal{P}$. Its minimal volume is a direct consequence of the definition of convex hull as the smallest convex set that contains sets of points $\mathcal{V}_1, \ldots, \mathcal{V}_R$. Although $\mathcal{P}_{\text{out}}$ from (23) is the smallest polytopic outer approximation, its construction is NP-hard and therefore applicable only to problems in low dimensions or for low number of points.

As an alternative way, we can also design $\mathcal{P}_{\text{out}}$ as an ellipsoid, i.e.,

$$\mathcal{P}_{\text{out}} = \{x \mid (x - x_c)^T W^{-1} (x - x_c) \leq 1\}. \tag{24}$$

The ellipsoid is parameterized by its center point $x_c \in \mathbb{R}^n$ and the matrix $W \in \mathbb{R}^{n \times n}$ satisfying $W = W^T \succ 0$. To design these parameters such that $\mathcal{P}_{\text{out}} \supseteq \mathcal{P}$ holds, one proceeds as follows. Since $\mathcal{P} = \cup_i \mathcal{Q}_i$, denote again by $\mathcal{V}_i$ the vertices of $\mathcal{Q}_i$ for $i = 1, \ldots, R$. Then $\mathcal{P} \supseteq \cup_i \mathcal{Q}_i$ if and only if $\mathcal{P} \supseteq \mathcal{Q}_i$ for all $i = 1, \ldots, R$. This in turn is equivalent to

$$(v - x_c)^T W^{-1} (v - x_c) \leq 1, \ \forall v \in \mathcal{V}_i, \ \forall i \in 1, \ldots, R. \tag{25}$$

Since each polytope $\mathcal{Q}_i$ has only finitely many vertices, (25) is a set of finitely many constraints. For each vertex v, the constraint can be rewritten using the well-known Schur complement as

$$\begin{bmatrix} 1 & (S v_j - s)^T \\ (S v_j - s) & I \end{bmatrix} \succeq 0, \tag{26}$$

where $S = W^{-1/2}$, $s = S x_c$, and v_j are vertices of polytopes $\mathcal{Q}_i$, $i = 1, \ldots, R$. Together with the constraint $S \succeq 0$, the problem of finding the smallest ellipsoidal outer approximation as in (24) can then be solved as an LMI problem. To minimize volume of $\mathcal{P}_{\text{out}}$ we need to maximize $\log(\det(S))$, see [5]. Numerically, the LMI (26) can be solved for instance using SeDuMi [10] and YALMIP [8].

VII. Example

As a motivating example that is relevant to many practical applications, we consider a double integrator. Using sampling time $\Delta_T = 1$ seconds, the dynamics can be described by the state-space model (1) with $A = \left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right]$, $B = \left[\begin{smallmatrix} 1 \\ 0.5 \end{smallmatrix}\right]$. We assume state constraints $\mathcal{X} = \{x \in \mathbb{R}^2 \mid -40 \preceq x \preceq 40\}$ and input bounds $\mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$. By solving problem (3) for $N = 15$, $Q_N = I$, $Q_x = I$, $Q_u = 1$, and a quadratic form of (3a), we have obtained the function $\kappa(\cdot)$ in (4) using the Multi-Parametric Toolbox [7]. The function consisted of 493 polytopes in the 2-dimensional state space. The polytopes are shown in Figure 1. The total memory footprint of $\kappa(\cdot)$ in (4) was 5961 real numbers. To assess the amount of computational resources required to implement such an explicit MPC feedback on-line via sequential search, we have investigated 6000 equidistantly placed points from the domain of $\kappa(\cdot)$. For each point we have first executed the sequential search procedure and measured the execution time. Using a pure Matlab implementation of the algorithm on a 2.2 GHz CPU, the point location took $2.3 \cdot 10^{-4}$ seconds per point, on average. We remark that the execution times can be significantly reduced by implementing a native compiled version of the sequential search procedure.

Next we investigate how much memory and computational resources can be saved by using the polygonic representation proposed in Sections IV and VI. In terms of memory, it is worth noting that among the 493 feedback gains, only 37 are unique. Two principal polygons consisting of more than one polytope can be identified in Fig. 1. One is composed of 229 polytopes shown in yellow, the other one consists of 229 green-colored polytopes. By employing the polygonic representation of the feedback function $\tilde{\kappa}(\cdot)$ per (8), and by retaining only the boundary polytopes $\mathcal{Q}_i$ to represent the boundary of each polygon, the total memory footprint of $\tilde{\kappa}(\cdot)$ was just 863 floating-point numbers, a reduction by a factor of 7.

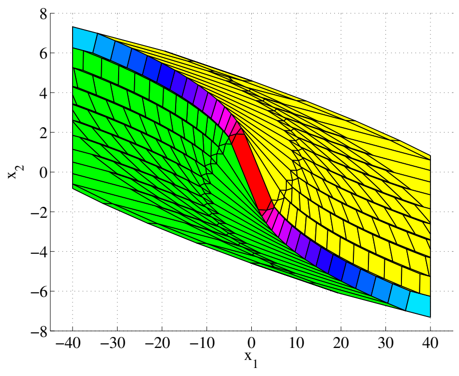

Fig. 1. Polytopes of the same color share the same feedback law. The two principal polygons are constituted by polytopes depicted in the green and yellow color, respectively.

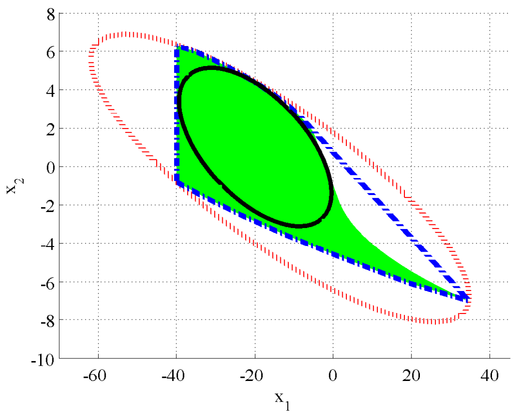

Fig. 2. Outer and inner approximations of the green non-convex polygon from Fig. 1: the red-dotted outer ellipsoidal approximation from (24), the blue-dash-dotted outer polytopic convex hull approximation from (23), and the black ellipsoidal inner approximation of (16).

To reduce the on-line computational effort via Alg. 3, inner and outer approximations of each polygon were subsequently devised using the procedures of Section VI. Results obtained for the green polygon from Fig. 1 are shown in Fig. 2. It is worth noting that the polytopic outer approximation is substantially tighter than its ellipsoidal counterpart, but also consumes more memory (120 floating point numbers for the polytope versus 6 numbers for the ellipsoid). Employing the ellipsoidal inner and outer approximations in Alg. 3 allowed to reduce the average computational effort per point from $2.3 \cdot 10^{-4}$ to $1.4 \cdot 10^{-4}$ seconds, an improvement by $40\,\%$ upon sequential search. The total memory footprint due to all sets required by Algorithm 3 is 905 floating point numbers. Comparing this figure to the footprint of the original explicit feedback $\kappa(\cdot)$ (which consumed 5961 floating point numbers), we see that the amount of required memory storage was reduced by a factor of 6.6, which is significant.

VIII. Conclusion

In this paper we have investigated how to reduce the memory consumption and computational burden of explicit MPC solutions. The required amount of memory was significantly reduced by employing a polygonic representation of regions of the explicit PWA feedback law. Since only the outer boundaries of such polygons need to be stored, significant amount of memory can be saved. This comes at the price of increased computational resources required to perform the point location task. To mitigate such an increase, we have proposed to devise inner and outer approximations of non-convex polygons, which allow to significantly reduce the computational load. The main advantage of the proposed simplification method over competing alternatives is that no special properties of the original feedback law $\kappa(\cdot)$ are assumed. Hence the polygonic representation of explicit MPC can be obtained e.g. for discontinuous explicit solutions, or for cases where the union of controller regions is not convex. The other advantage is that the polygonic representation is equivalent to the original explicit MPC feedback in the sense that it provides the same values of optimal control actions for each admissible value of state measurements.

Acknowledgments

J. Oravec, S. Blažek and M. Kvasnica gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under grants 1/0095/11 and 1/0973/12. J. Oravec was also supported by an internal STU grant no. 1323. This research was supported by Mitsubishi Electric Research Laboratories, under a Collaborative Research Agreement.

References

[1] J.W. Alexander. A proof and extension of the Jordan-Brouwer separation theorem. *Transactions of the American Mathematical Society*, 23(4):333–349, 1922.

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

[3] S. Blažek and M. Kvasnica. Polygonic representation of explicit model predictive control in two dimensions. In *Proceedings of the 10th International Conference - Technical Conference Process Control 2012*, Kouty nad Desnou, Czech Republic, June 2012.

[4] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 2003.

[5] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Studies in Applied Mathematics. SIAM, 1994.

[6] M. Kvasnica and M. Fikar. Clipping-Based Complexity Reduction in Explicit MPC. *IEEE Trans. Automatic Control*, 57(7):1878–1883, July 2012.

[7] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004. Available from http://control.ee.ethz.ch/~mpt/.

[8] J. Löfberg. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proc. of the CACSD Conference*, Taipei, Taiwan, 2004. Available from http://users.isy.liu.se/johanl/yalmip/.

[9] J. Spjøtvold, P. Tøndel, and T. A. Johansen. A Method for Obtaining Continuous Solutions to Multiparametric Linear Programs. In *IFAC World Congress*, Prague, Czech Republic, 2005.

[10] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, pages 625–653, October 1999.

[11] Petter Tøndel, Tor Arne Johansen, and Alberto Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489–497, 2003.