2013 International Conference on Process Control (PC)
June 18–21, 2013, Štrbské Pleso, Slovakia

Simplification of Explicit MPC Solutions via Inner and Outer Approximations

Juraj Oravec, Slavomír Blažek, and Michal Kvasnica
Institute of Information Engineering, Automation, and Mathematics
Faculty of Chemical and Food Technology
Slovak University of Technology in Bratislava
Radlinského 9, 812 37 Bratislava, Slovakia
(Tel: +421 259 325 352; e-mail: juraj.oravec@stuba.sk)

***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 on non-convex objects. To reduce the computational requirements, we therefore propose to devise inner and outer convex approximations of non-convex polygons. Such approximations then allow to perform point location more effectively, leading to 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 [1], Explicit Model Predictive Control (MPC) has garnered significant attention among theoreticians and control engineers because it allows to implement MPC on cheap control hardware. Previously, this was not easily attainable. The reason being that 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 abolished by shifting the optimization offline. Several authors (see e.g. [2], [3], [4], [5]) 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 furthermore 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, 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 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 principal 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. Examples of methods in this class include, but are not limited to, relaxation of optimality conditions [6], application of model reduction [7], reduction of number of degrees of freedom by moveblocking [8], approximation via PWA functions defined over simplicies [9], refinement based on Lyapunov functions [10], or by polynomial approximation [11], [12]. In all cases 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 principal line of research is devoted to simplifying the PWA feedback law such while preserving optimality. This can be achieved by merging regions over which the PWA feedback is defined [13], by encoding the PWA function as a set of $\min/\max$ rules [14], by devising a binary search tree [15], or by exploiting saturation properties [16].

Recently, in [17] 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. 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. In this paper we address this deficiency 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 V. Efficacy of the proposed procedure is then illustrated by means of a motivating example.

II. Notation and Definitions

Definition 2.1 (Polyhedron): 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 $h_i^T x \leq k_i$ where h_i^T is the i-th row of H.

Definition 2.2 (Polytope): Bounded polyhedron is called polytope.

Definition 2.3 (Polygon): 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.

978-1-4799-0927-8/13/$31.00

Definition 2.4 (Geometric subdivision): 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 $\text{int}(\mathcal{Q}_i) \cap \text{int}(\mathcal{Q}_j) = \emptyset$, $\forall i \neq j$.

Definition 2.5 (Vertex representation of a polytope): Every polytope $\mathcal{Q} \subset \mathbb{R}^n$ can be equivalently represented by

$$\mathcal{Q} = \text{convh}(v_1, \ldots, v_M), \tag{1}$$

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

$$x(t + \Delta_T) = Ax(t) + Bu(t), \tag{2}$$

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

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U},\ \forall t \geq 0, \tag{3}$$

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 (2) subject to constraints (3), the MPC optimization problem becomes

$$U_N^* = \arg\min \|Q_N x_N\|_p + \sum_{k=0}^{N-1} \|Q_x x_k\|_p + \|Q_u u_k\|_p \tag{4a}$$

$$\text{s.t. } x_0 = x(t), \tag{4b}$$

$$x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1, \tag{4c}$$

$$x_k \in \mathcal{X},\ k = 0, \ldots, N, \tag{4d}$$

$$u_k \in \mathcal{U},\ k = 0, \ldots, N-1, \tag{4e}$$

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 with $Q_N \succeq 0$, $Q_x \succeq 0$, $Q_u \succ 0$, and p denotes a norm in which corresponding quantities should be minimized (if $p = 2$, then we can equivalently minimize $z^T Q z$ instead of $\|Qz\|_2 = \sqrt{z^T Q z}$). Solving the optimal control problem (4) for a particular initial condition $x(t)$ leads to the sequence of optimal control inputs $U_N^* = \{u_0^*, \ldots, u_{N-1}^*\}$, 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 (4) using parametric programming [18], the optimal receding horizon control action u_0^* can be pre-computed [3], [19] for all feasible values of $x(t)$ as a PWA function of the form

$$u_0^* = \kappa(x(t)) := \begin{cases} F_1 x(t) + g_1 & \text{if } x(t) \in \mathcal{Q}_1 \\ \vdots & \\ F_M x(t) + g_M & \text{if } x(t) \in \mathcal{Q}_M \end{cases} \tag{5}$$

where $F_i \in \mathbb{R}^{m \times n}$, $g_i \in \mathbb{R}^m$, and $\mathcal{Q}_i = \{x \,|\, H_i x \leq k_i\}$ are polytopes in $\mathbb{R}^n$, for all $i = 1, \ldots, M$, with M denoting the total number of polytopes.

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, as captured by Algorithm 1, whose runtime complexity is $\mathcal{O}(M)$. However, to evaluate $\kappa(\cdot)$

Algorithm 1 Evaluation of $\kappa(\cdot)$ from (5)

INPUT: Polytopes $\mathcal{Q}_i$, feedback laws F_i, g_i, number of polytopes M, state measurement $x(t)$
OUTPUT: Optimal RHMPC control input u_0^*

```
for i = 1, ..., M do
    if x(t) ∈ Q_i then
        u_0^* = F_i x(t) + g_i
        return
    end if
end for
```

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, $i = 1, \ldots, M$) is a linear function of M, the total number of polytopes. More specifically, the memory consumed by the function $\kappa(\cdot)$ is equal to

$$\mathcal{S}(\kappa) = \sum_{i=1}^{M} (c_i + m)(n + 1), \tag{6}$$

where c_i is the number of defining half-spaces of the i-th polytope $\mathcal{Q}_i$, m is the number of control inputs, and n denotes dimension of the state vector.

IV. Polygonic representation of Explicit MPC

The objective is to reduce memory consumption of the explicit feedback law $\kappa(\cdot)$ by exploiting the fact that, in practice, several polytopes of (5) will be associated to identical feedback gains. Denote by M_u the number of unique feedback gains and let $\mathcal{I}_j \subseteq \{1, \ldots, M\}$ be the index set of polytopes Q_i which share the j-th feedback, i.e.

$$\mathcal{I}_j = \{i \,|\, F_i = F_j,\ g_i = g_j,\ i \in \{1, \ldots, M\}\},\ j = 1, \ldots, M_u. \tag{7}$$

Let

$$\mathcal{P}_j = \bigcup_{i \in \mathcal{I}_j} \mathcal{Q}_i \tag{8}$$

be the polygon whose geometric subdivision is given by polytopes $\mathcal{Q}_i$, $\forall i \in \mathcal{I}_j$. Then we can rewrite κ as

$$\tilde{\kappa}(x(t)) := \begin{cases} F_1 x(t) + g_1 & \text{if } x(t) \in \mathcal{P}_1 \\ \vdots & \\ F_{M_u} x(t) + g_{M_u} & \text{if } x(t) \in \mathcal{P}_{M_u} \end{cases} \tag{9}$$

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

Lemma 4.1 ([17]): For all $x(t)$ in the domain of κ we have $\kappa(x(t)) = \tilde{\kappa}(x(t))$. ■

The total memory consumed by the function $\tilde{\kappa}(\cdot)$ in (9) is $\mathcal{O}(M_u)$. More specifically, we have

$$\mathcal{S}(\tilde{\kappa}) = \sum_{i}^{M_u}(b_i + m)(n+1), \tag{10}$$

where b_i is the number of facets defining the boundary of the i-th polygon $\mathcal{P}_i$. Comparing (10) to (6) we see that $\tilde{\kappa}$ will consume less memory than κ if $\sum_i^{M_u} b_i \leq \sum_i^{M} c_i$. Since each polygon $\mathcal{P}_i$ is composed of the geometric subdivision $\{\mathcal{Q}_i\}_{i\in\mathcal{I}_i}$, only the polytopes on the boundary of $\mathcal{P}_i$ need to be stored, therefore the relation always holds. Moreover, $\mathcal{S}(\tilde{\kappa}) \ll \mathcal{S}(\kappa)$ is often the case in practice.

Evaluation of $\tilde{\kappa}$ from (9) for a given value of $x(t)$ can be done by a straightforward modification of Alg. 1, reported for completeness as Algorithm 2. Its runtime complexity is $\mathcal{O}(M_u)$.

Algorithm 2 Evaluation of $\tilde{\kappa}$ from (9)

INPUT: Polygons $\mathcal{P}_i$, feedback laws F_i, g_i, number of unique feedbacks M_u, state measurement $x(t)$
OUTPUT: Optimal RHMPC control input u_0^*

```
1: for i = 1, ..., M_u do
2:    if x(t) ∈ P_i then
3:       u_0^* = F_i x(t) + g_i
4:       return
5:    end if
6: end for
```

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

Formally, the problem to be solved in Step 2 of Alg. 2 can be stated as follows:

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

In [17], Problem 4.2 was solved using a ray shooting procedure. However, the approach has one crucial downside. Specifically, the ray shooting scheme introduces significant computational overhead, and hence slows down implementation of explicit MPC. An another alternative to solve Problem 4.2 is to exploit the geometric subdivision (8). Specifically, to test whether $x \in \mathcal{P}$, one can instead verify whether $x \in \mathcal{Q}_i$ for some $i \in \{1, \ldots, M_P\}$. However, such a procedure is equivalent, from a computational load of view, to evaluation of the original feedback (5). Therefore in the next section we propose an alternative way to solving Problem 4.2. The idea is based on devising inner and outer approximations of polygons $\mathcal{P}_i$ in (9) such that the point location problem can be performed more effectively.

V. Point Location in Polygons via Approximations

Throughout this section we consider a single non-convex polygon $\mathcal{P} \subset \mathbb{R}^n$, which is represented by a geometric subdivision $\mathcal{P} = \{\mathcal{Q}_i\}$, where $\mathcal{Q}_i$, $i = 1, \ldots, R$ are polytopes. The task is to devise a procedure that solves Problem 4.2, i.e., answers whether $x \in \mathcal{P}$ for an arbitrary $x \in \mathbb{R}^n$.

The central idea of this paper is as follows. Assume that two sets, namely $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$ exist such that

$$\mathcal{P}_{\text{out}} \supseteq \mathcal{P} \tag{11a}$$
$$\mathcal{P}_{\text{in}} \subseteq \mathcal{P} \tag{11b}$$

holds. Then Problem 4.2 can be solved efficiently as follows.

Lemma 5.1: Given is a point x, polygon $\mathcal{P}$, and the sets $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$. If $x \notin \mathcal{P}_{\text{out}}$, then $x \notin \mathcal{P}$. Furthermore, if $x \in \mathcal{P}_{\text{in}}$, then $x \in \mathcal{P}$.

Proof: Follows immediately from (11). ■

The first statement of Lemma (5.1), i.e., $x \notin \mathcal{P}_{\text{out}} \Rightarrow x \notin \mathcal{P}$ represents a necessary condition for validity of the inclusion test $x \in \mathcal{P}$. The second term, i.e., $x \in \mathcal{P}_{\text{in}} \Rightarrow x \in \mathcal{P}$, on the other hand, represents a sufficient condition. With Lemma (5.1) in hand, a procedure to solve Problem 4.2 can be formally stated as Algorithm 3, reported below.

Algorithm 3 Suggested 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 ∈ P_in then
2:    return true
3: else if x ∉ P_out then
4:    return false
5: else
6:    Determine whether x ∈ P using the method of [17].
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 [17]. 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 much easier to solve. Specifically, it is worth noting that the two inclusion tests in Steps 3 and 1 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.

However, in order 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 (11) 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.

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 a polygon $\mathcal{P}$ is given as a geometric subdivision of the form of (8), i.e., $\mathcal{P} = \cup_i \mathcal{Q}_i$.

A. Inner Approximation

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

Problem 5.2: Given is a 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

$$\forall x \in \mathcal{C} \Rightarrow x \notin \mathcal{P}_{\text{in}}. \tag{12}$$

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

$$\mathcal{P}_{\text{in}} = \{x \mid (x - x_c)^T P^{-1} (x - x_c) \leq 1\}, \tag{13}$$

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 (13) is maximized while satisfying (12).

To pose the search for x_c and P in (13) as a convex optimization problem, we exploit the *S-procedure*:

Lemma 5.3 ([20], [21]): Let $f_0(x)$ and $f_i(x)$, $i = 1, \ldots, M$ be quadratic functions. Then

$$f_i(x) \leq 0 \Rightarrow f_0(x) \geq 0 \tag{14}$$

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

$$f_0(x) + \textstyle\sum_{i=1}^{M} \lambda_i f_i(x) \geq 0. \tag{15}$$

■

To see the relation between Lemma 5.3 and Problem 5.2, first note that the complement $\mathcal{C}$ is divided into a finite number of polyhedra $\mathcal{R}_i$. Then the implication in (12) 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 $f_i(x) := C_i x + d_i$. Furthermore, $x \notin \mathcal{P}_{\text{in}}$ can be cast as

$$\underbrace{(x - x_c)^T P^{-1} (x - x_c) - 1 - \epsilon}_{f_0(x)} \geq 0. \tag{16}$$

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

$$g(x) := \begin{bmatrix} x \\ 1 \end{bmatrix}^T \underbrace{\begin{bmatrix} W & w \\ w^T & z \end{bmatrix}}_{\tilde{W}} \begin{bmatrix} x \\ 1 \end{bmatrix}. \tag{17}$$

It is well known (see e.g. [21]) that with $g(x)$ as in (17), $g(x) \geq 0$ holds for all x if and only if the matrix $\tilde{W}$ is positive semi-definite. Using this fact, and by applying the S-procedure of Lemma 5.3 with $f_0(x)$ and $f_i(x)$ defined as above, we can hence formulate the search for P and x_c in (13) as

$$\begin{bmatrix} \tilde{P} & -x_c^T \tilde{P} \\ \tilde{P} x_c & x_c^T \tilde{P} x_c - 1 - \epsilon \end{bmatrix} + \sum_{i=1}^{M_C} \lambda_i \begin{bmatrix} 0 & {}^1\!/_2 C_i \\ {}^1\!/_2 C_i^T & -d_i \end{bmatrix} \succeq 0, \tag{18}$$

where $\tilde{P} = P^{-1}$. When x_c is fixed, the search for an ellipsoid (13) that has maximal volume can be done by searching for $\tilde{P} = \tilde{P}^T \succeq 0$, $\lambda_i \geq 0$, $i = 1, \ldots, M_C$, while minimizing the trace of $\tilde{P}$, i.e.,

$$\min_{\tilde{P},\, \lambda_i} \operatorname{trace}(\tilde{P}) \text{ s.t. } \lambda_i \geq 0, \text{ (18) holds, } i = 1, \ldots, M_C. \tag{19}$$

Note that (19), with x_c being fixed, is an LMI optimization problem that can be solved efficiently using off-the-shelf software (e.g. with [22]). Once $\tilde{P}$ is computed via (19), P in (13) 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 5.4: Given is a 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. □

Let $\mathcal{V}_i = \{v_{i,1}, \ldots, v_{i,n_{v_i}}\}$ denote the vertices of polytope $\mathcal{Q}_i$. Then

$$\mathcal{P}_{\text{out}} = \operatorname{convh}\{\mathcal{V}_1, \ldots, \mathcal{V}_R\}, \tag{20}$$

is the smallest polytope that contains $\mathcal{P}$, i.e., $\mathcal{P}_{\text{out}} \supseteq \mathcal{P}$. The polytopic nature of $\mathcal{P}_{\text{out}}$ in (20) follows from the definition of a polytope as a convex hull of finitely many points (see [23]). Its minimal volume is a direct consequence of the definition of convex hull as the smallest convex set that contains sets of points $\mathcal{V}_1, \ldots, \mathcal{V}_R$. For formal proofs an interested reader is again referred to [23]. Although $\mathcal{P}_{\text{out}}$ from (20) is the smallest polytopic outer approximation, its construction is not always easy. In particular, computing the convex hull of a set of points is an NP-hard problem 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.,

$$\mathcal{P}_{\text{out}} = \{x \mid (x - x_c)^T W^{-1} (x - x_c) \leq 1\}. \tag{21}$$

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

$$(v - x_c)^T W^{-1} (v - x_c) \leq 1, \ \forall v \in \mathcal{V}_i, \ \forall i \in 1, \ldots, R. \tag{22}$$

Since each polytope $\mathcal{Q}_i$ has only finitely many vertices, (22) is a set of finitely many constraints. For each particular choice of the vertex v, the constraint can be rewritten, using the Schur complement [24], to

$$\begin{bmatrix} 1 & (Sv_j - s)^T \\ (Sv_j - s) & I \end{bmatrix} \succeq 0, \tag{23}$$

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 (21) can then be solved as an LMI problem. To minimize volume of $\mathcal{P}_{\text{out}}$ we need to maximize $\log(\det(S))$, see [20]. Numerically, the LMI (23) can be solved e.g. using SeDuMi [22] and YALMIP [25].

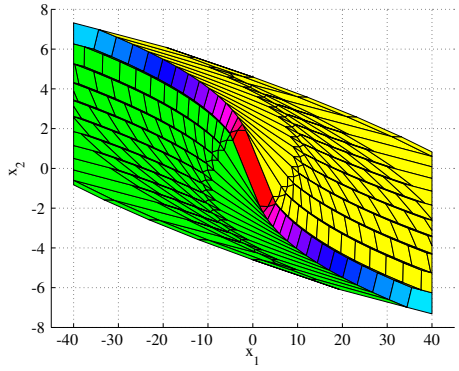

Fig. 1. Polytopes of the same color share the same feedback law. The two principal polygons are constituted by polytopes depicted in the green and yellow color, respectively.

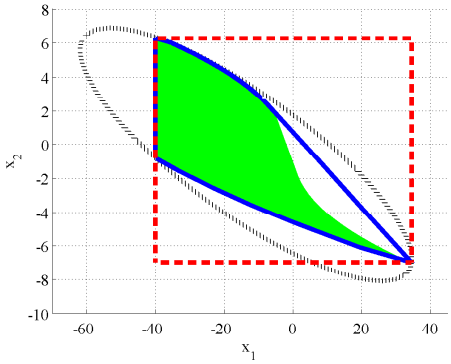

Fig. 2. Outer convex approximation of the green non-convex polygon from Fig. 1. Ellipsoidal approximation (21) is shown by the red dotted line, the convex hull (20) is marked by solid blue color.

VI. Example

As a motivating example that frequently occurs in practice, consider a double integrator, which models movement of an object in a one-dimensional plane. States of the system represent, respectively, the object's position and velocity, while the control input is equivalent to the force applied to the object. Using sampling time $\Delta_T = 1$ second, the double integrator dynamics can be described by the state-space model

$$x(t+\Delta_T) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u(t). \qquad (24)$$

We assume state constraints $\mathcal{X} = \{x \in \mathbb{R}^2 \mid -40 \leq x \leq 40\}$ and input bounds $\mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$. By solving problem (4) for $N = 15$, $Q_N = I$, $Q_x = I$, $Q_u = 1$, we have obtained the function $\kappa(\cdot)$ in (5) using the Multi-Parametric Toolbox [26]. The function consisted of 493 polytopes in the 2-dimensional state space. The polytopes are shown in Figure 1. The total memory footprint of κ per (6) was $\mathcal{S}(\kappa) = 5961$ real numbers. To assess the amount of computational resources required to implement such an explicit MPC feedback on-line via the standard procedure represented by Alg. 1, we have investigated 6000 equidistantly placed points from the domain of $\kappa(\cdot)$. For each point we have first executed Alg. 1 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 devising a native compiled version of Alg. 1.

Next we investigate how much memory and computational resources can be saved by using the polygonic representation proposed in Sections IV and V. In terms of memory, it is worth noting that among the 493 feedback gains, only 37 were 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 RHMPC feedback function $\tilde{\kappa}$ per (9), the total memory footprint is just $\mathcal{S}(\tilde{\kappa}) = 863$ numbers, a reduction by a factor of 7.

To compare performance of Alg. 3 against Alg. 1, we have designed various outer and inner approximations of non-convex polygons. Two types of outer approximations $\mathcal{P}_{\text{out}}$ were calculated using MPT and YALMIP: the polytopic outer approximation via convex hulls as in (20), and an ellipsoidal outer approximation from (21). For the green-colored polygon shown in Fig. 1, the respective approximations are reported in Fig. 2. As outlined in Section V, the smaller the volume of an outer approximation, the higher the efficacy of Step 3 in Alg. 3. To illustrate this dependence, we again took 6000 equidistantly placed points and investigated how many points are contained *outside* of each respective outer approximation (the higher the number, the more often we can quickly abort Alg. 3 already at Step 4). The results as reported in Table I, along with computational effort needed to construct respective outer approximations. As can be seen, the convex hull outer approximation (20) outperforms the other two alternatives.

Inner ellipsoidal approximations of Section V-A were then computed for each of the principal polygons with an example being shown in Fig. 3. The corresponding LMIs (19) were formulated by YALMIP and solved by SeDuMi. Employing these convex inner approximations in Step 1 significantly improved performance of Alg. 3 upon our benchmark, represented by Alg. 1. Specifically, for each point for which $x \in \mathcal{P}_{\text{in}}$ holds, we can abort Alg. 3 quickly with a positive answer to the query $x \in \mathcal{P}$ already in Step 2.

Employing the described 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 the standard procedure represented by Alg. 1. Needless to say, storing the inner and outer approximation slightly increases the required memory storage. Specifically, the sets have to be represented in the memory by 120 floating point numbers for the two polytopic outer approximations, and by 36 floating point numbers for the six ellipsoidal inner approximations. Combined with the footprint of $\tilde{\kappa}(\cdot)$ reported above, the total amount of memory required to store all data for Algorithm 3 is 1019 floating point numbers. Comparing this to the footprint of the original explicit feedback $\kappa(\cdot)$ (which consumed 5961), we see that the amount of require memory was reduced by a factor of 6, which is significant.

TABLE I. COMPARISON OF OUTER APPROXIMATIONS.

Type of $\mathcal{P}_{\text{out}}$	Percentage of points with $x \notin \mathcal{P}_{\text{out}}$ (higher is better)	Construction time
Convex hull (20)	48 %	0.4 sec
Ellipsoid (21)	39 %	15.3 sec

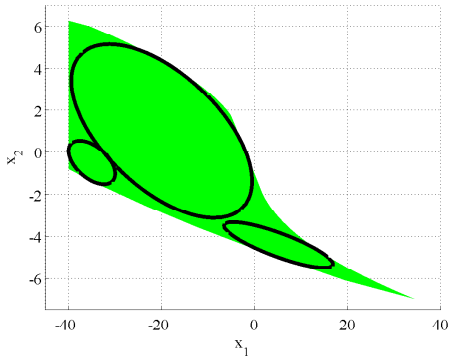

Fig. 3. Inner ellipsoidal approximation of the green non-convex polygon from Fig. 1.

VII. CONCLUSION

In this paper we have investigated possibilities to reduce the memory consumption and computational burden of explicit MPC solutions. 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 this increase, we have proposed to devise inner and outer approximations of non-convex polygons, which allows to significantly reduce the computational load. Several methods to design suitable inner and outer approximations were presented and efficacy of the procedure was demonstrated on an example.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0973/12 and 1/0095/11. The Authors gratefully acknowledge the contribution of the Slovak Research and Development Agency under the project APVV 0551-11.

REFERENCES

[1] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[2] P. Tøndel, T. A. Johansen, and A. Bemporad, "An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions," *Automatica*, Nov. 2001, preprint submitted.

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

[4] M. Baotić, "Optimal Control of Piecewise Affine Systems – a Multi-parametric Approach," Dr. sc. thesis, ETH Zurich, Zurich, Switzerland, Mar. 2005.

[5] 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.

[6] A. Bemporad and C. Filippi, "Suboptimal explicit RHC via approximate multiparametric quadratic programming," *Journal of Optimization Theory and Applications*, vol. 117, no. 1, pp. 9–38, Apr. 2003.

[7] S. Hovland, K. E. Willcox, and J. T. Gravdahl, "Explicit MPC for large-scale systems via model reduction," *AIAA Journal of Guidance, Control and Dynamics*, vol. 31, no. 4, Jul. 2008.

[8] P. Tøndel and T. Johansen, "Complexity reduction in explicit linear model predictive control," in *Proc. of 15-th IFAC World Congress*, 2002.

[9] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations," *IEEE Trans. Automatic Control*, vol. 56, no. 12, pp. 2883–2897, 2011.

[10] L. Lu, W. Heemels, and A. Bemporad, "Synthesis of low-complexity stabilizing piecewise affine controllers: A control-Lyapunov function approach," in *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 2011, pp. 1227–1232.

[11] G. Valencia-Palomo and J. Rossiter, "Using Laguerre functions to improve efciency of multi-parametric predictive control," in *Proceedings of the American Control Conference*, Baltimore, USA, 2010, pp. 4731–4736.

[12] M. Kvasnica, J. Löfberg, and M. Fikar, "Stabilizing polynomial approximation of explicit MPC," *Automatica*, vol. 47, no. 10, pp. 2292–2297, 2011.

[13] T. Geyer, F. Torrisi, and M. Morari, "Optimal complexity reduction of polyhedral piecewise affine systems," *Automatica*, vol. 44, no. 7, pp. 1728–1740, Jul. 2008.

[14] C. Wen, X. Ma, and B. E. Ydstie, "Analytical expression of explicit MPC solution via lattice piecewise-affine function," *Automatica*, vol. 45, no. 4, pp. 910 – 917, 2009.

[15] P. Tøndel, T. A. Johansen, and A. Bemporad, "Evaluation of Piecewise Affine Control via Binary Search Tree," *Automatica*, vol. 39, no. 5, pp. 945–950, May 2003.

[16] M. Kvasnica and M. Fikar, "Clipping-based complexity reduction in explicit mpc," *IEEE Trans. Automatic Control*, vol. 57, no. 7, pp. 1878–1883, July 2012.

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

[18] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[19] M. Kvasnica, *Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools*. Saarbruecken: VDM Verlag, Jan. 2009.

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

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

[22] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, pp. 625–653, Oct. 1999.

[23] G. M. Ziegler, *Lectures on Polytopes*. Springer, 1994.

[24] J. Löfberg, "Minimax approaches to robust model predictive control," Ph.D. dissertation, Linköping University, Sweden, Apr. 2003.

[25] ——, "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/.

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