Separating Functions for Complexity Reduction of Explicit Model Predictive Control

Ivana Rauová [*], **Michal Kvasnica** [*], and **Miroslav Fikar** [*]

[*] *Institute of Automation, Information Engineering and Mathematics, Slovak University of Technology, 812 37 Bratislava, Slovakia*
{ivana.rauova, michal.kvasnica, miroslav.fikar}@stuba.sk

Abstract: In this work we propose to reduce memory footprint of explicit MPC controllers by eliminating a significant portion of controller's regions in which the value of the optimal control action attains saturated values. Such regions are then separated by a suitable function, which serves to recover the original control behavior. As a consequence, complexity of explicit MPC feedback laws is reduced considerably without sacrificing optimality.

Keywords: explicit model predictive control, separation, nonlinear optimization

1. INTRODUCTION

Implementing MPC in the Receding Horizon fashion (RHMPC) requires, at each sampling instance, obtaining the optimal control input by solving a suitable optimization problem. Difficulties arise when the sampling time is too short to perform the optimization on-the-fly. One way around this problem is to precompute the optimal control action u^* for all feasible initial conditions x in form of an explicit feedback law $u^* = \kappa(x)$. As shown in Bemporad et al. (2002), for a rich class of MPC problems the feedback function κ takes the form of a piecewise affine (PWA) function, which is defined over a set of polytopic regions. Computing u^* on-line then reduces to a mere function evaluation. However, the number of regions of κ, which is problem-dependent, tends to be large, easily exceeding the storage capacity of a selected implementation platform. Therefore, it is important to keep the number of regions as low as possible.

One way to reduce complexity is to construct a sub-optimal replacement function $\widetilde{\kappa} \approx \kappa$ of substantially lower complexity, see e.g. Bemporad and Filippi (2003); Johansen and Grancharova (2003); Grieder et al. (2004); Cychowski and O'Mahony (2005); Jones and Morari (2009); Scibilia et al. (2009). Another line of research is concerned with finding such a replacement $\widetilde{\kappa}$ which is simpler than the original function, but maintains the equivalence $\kappa(x) = \widetilde{\kappa}(x)$ for all points x of interest, as elaborated in Baotic et al. (2008); Geyer et al. (2008); Wen et al. (2009).

In Kvasnica and Fikar (2010) we have shown how to find a simpler, equivalent feedback law $u^* = \widetilde{\kappa}(x)$ by exploiting geometric properties of explicit MPC solutions. Specifically, we have demonstrated that in majority of controller's regions the optimal control action is saturated either at the allowable maximum or minimum limits. Such regions can subsequently be eliminated and replaced by "extensions" of the regions in which the control action is unsaturated. Such a procedure leads to an equivalent controller $\widetilde{\kappa}$ which is defined over, on average, $1.3N_{\text{unsat}}$ regions, where N_{unsat} is the number of unsaturated regions. In this work we show how to construct the function $\widetilde{\kappa}$ which is *always* defined over N_{unsat} regions. This is achieved by separating the saturated regions by a suitable function, which serves to recover equivalence between $\widetilde{\kappa}$ and the original feedback κ. Two types of separating functions are considered: polynomials and piecewise affine separators encoded as binary search trees.

This paper is structured as follows. Theoretical concepts of explicit model predictive control are outlined in Section 3. The general idea of complexity reduction employing the concept of separating functions is elaborated in Section 4. Polynomial separators are then reviewed in Section 5, which also reviews various approaches to computing the separating polynomial. Construction of piecewise affine separators is reviewed in Section 6. Finally, in Section 7 we demonstrate viability of the presented approach by means of a large case study with the main focus on the states with one input variable.

2. DEFINITIONS

The interior of a set $\mathcal{R}$ is denoted by $\operatorname{int}(\mathcal{R})$. Given a function κ, $\operatorname{dom}(\kappa)$ denotes its domain. A set of n elements $\mathcal{R} := \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ will be denoted as $\{\mathcal{R}_i\}_{i=1}^n$ and its cardinality by $|\mathcal{R}|$. A polytope is the bounded convex intersection of c closed affine half-spaces, i.e. $\mathcal{R} := \{x \in \mathbb{R}^{n_x} \mid Fx \le g\}$. We call the collection of polytopes $\{\mathcal{R}_i\}_{i=1}^R$ the *partition* of a polytope $\mathcal{R}$ if $\mathcal{R} = \bigcup_{i=1}^R \mathcal{R}_i$, and $\operatorname{int}(\mathcal{R}_i) \cap \operatorname{int}(\mathcal{R}_j) = \emptyset$ for all $i \neq j$. Each polytope $\mathcal{R}_i$ will be referred to as the *region* of the partition. Function $\kappa : \mathcal{R} \to \mathbb{R}^{n_z}$ with $\mathcal{R} \subseteq \mathbb{R}^{n_x}$, $\mathcal{R}$ being a polytope, is called piecewise affine over polytopes if $\{\mathcal{R}_i\}_{i=1}^R$ is the partition of $\mathcal{R}$ and

$$\kappa(x) := K_i x + L_i \quad \forall x \in \mathcal{R}_i, \tag{1}$$

with $K_i \in \mathbb{R}^{n_z \times n_x}$, $L_i \in \mathbb{R}^{n_z}$, and $i = 1, \dots, R$. PWA function $\kappa(x)$ is continuous if $K_i x + L_i = K_j x + L_j$ holds $\forall x \in \mathcal{R}_i \cap \mathcal{R}_j$, $i \neq j$.

3. EXPLICIT MODEL PREDICTIVE CONTROL

We consider the class of discrete-time, stabilizable linear time-invariant systems

$$x_{k+1} = Ax_k + Bu_k, \tag{2}$$

which are subject to polytopic constraints $x \in \mathcal{X} \subset \mathbb{R}^{n_x}$ and $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$. Assume the following constrained finite-time optimal control problem:

$$\min_{U_N} \sum_{k=0}^{N-1} x_{k+1}^T Q_x x_{k+1} + u_k^T Q_u u_k \tag{3a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k,\ x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \tag{3b}$$

where x_k and u_k denote, respectively, state and input predictions over a finite horizon N, given the initial condition x_0. It is assumed in (3a) that $Q_x = Q_x^T \succeq 0$, $Q_u = Q_u^T \succ 0$, i.e. that (3) is a strictly convex quadratic programming (QP). The receding horizon MPC feedback then becomes $u^*(x_0) = [\mathbb{1}\ \mathbb{0}\ \cdots\ \mathbb{0}]U_N^*$, where the optimal control actions $U_N^* := [u_0^T, \dots, u_{N-1}^T]^T$ can be found by solving (3) as a QP for a given value of the initial condition x_0. For problems of a modest size (typically for $n_x < 5$), it is also possible to characterize the optimal feedback $u^*(x_0)$ explicitly as a PWA function of x_0 Bemporad et al. (2002) by solving (3) as a *parametric quadratic program* (pQP).

Theorem 3.1. (Bemporad et al. (2002)). The RHMPC feedback $u^*(x_0)$ for problem (3) is given by $u^*(x_0) = \kappa(x_0)$ where: (i) the set of feasible initial conditions $\Omega := \{x_0 \mid \exists u_0, \dots, u_{N-1} \text{ s.t. (3b) hold}\}$ is a polytope; (ii) $\kappa : \Omega \to \mathcal{U}$ is a continuous PWA function defined over R regions $\mathcal{R}_i$, $i = 1, \dots, R$; (iii) $\mathcal{R}_i$ are full-dimensional polytopes $\mathcal{R}_i = \{x \mid F_i x \leq g_i\}$; and (iv) $\{\mathcal{R}_i\}_{i=1}^R$ is a partition of Ω.

The advantage of such an explicit representation is obvious: obtaining the optimal control action for a given x_0 reduces to a mere evaluation of the function $\kappa(x_0)$, which is henceforth denoted as the *explicit RHMPC feedback law*. The crucial limitation, however, is that the number of regions tends to be large, often above the limits of typical control hardware implementation platforms. Method represented in Kvasnica et al. (2011) deals with the problem how to replace the feedback function κ by a different function $\widetilde{\kappa}$ which requires significantly less memory for its implementation in real-time arrangement and maintains the equivalence $\widetilde{\kappa}(x_0) \equiv \kappa(x_0)\ \forall x \in \Omega$.

4. COMPLEXITY REDUCTION VIA SEPARATION OF REGIONS

Denote by $\overline{\kappa}$ and $\underline{\kappa}$ the maximal and minimal values which the PWA function κ attains over its domain Ω. Since the set of admissible inputs $\mathcal{U}$ in (3) is assumed to be closed and bounded, and since all regions $\mathcal{R}_i$, $i = 1, \dots, R$ are bounded polytopes, $\overline{\kappa}$ and $\underline{\kappa}$ are always finite and can be computed by solving $2R$ linear programs of the form

$$\overline{\kappa}_i = \max\{K_i x + L_i \mid x \in \mathcal{R}_i\},\ i = 1, \dots, R, \tag{4a}$$

$$\underline{\kappa}_i = \min\{K_i x + L_i \mid x \in \mathcal{R}_i\},\ i = 1, \dots, R, \tag{4b}$$

with $\overline{\kappa} = \max\{\overline{\kappa}_1, \dots, \overline{\kappa}_R\}$, $\underline{\kappa} = \min\{\underline{\kappa}_1, \dots, \underline{\kappa}_R\}$. Then the regions of $\kappa(x)$ can be classified as follows.

(1) If $K_i = 0$ and $L_i = \overline{\kappa}$, then region $\mathcal{R}_i$ is *saturated at the maximum*,
(2) if $K_i = 0$ and $L_i = \underline{\kappa}$, then region $\mathcal{R}_i$ is *saturated at the minimum*,
(3) otherwise the i-th region is *unsaturated.*

Denote by $\mathcal{I}_{\max}$ and $\mathcal{I}_{\min}$ the index lists of regions saturated at the maximum and minimum, respectively, and by $\mathcal{I}_{\text{unsat}}$ the index list of unsaturated regions. With this classification, the RHMPC feedback $u^* = \kappa(x)$ can be written as

$$u^* = \kappa(x) = \begin{cases} K_i x + L_i & \text{if } x \in \mathcal{R}_{\mathcal{I}_{\text{unsat}}}, \\ \overline{\kappa} & \text{if } x \in \mathcal{R}_{\mathcal{I}_{\max}}, \\ \underline{\kappa} & \text{if } x \in \mathcal{R}_{\mathcal{I}_{\min}}. \end{cases} \tag{5}$$

Evaluation of $\kappa(x)$ for any $x \in \Omega$ is therefore a two-stage process. First, the index r of region $\mathcal{R}_r$ which contains x needs to be identified. Then, the value of $\kappa(x)$ is either computed by $K_r x + L_r$ if $r \in \mathcal{I}_{\text{unsat}}$, or $\kappa(x) = \overline{\kappa}$ ($\kappa(x) = \underline{\kappa}$) if $r \in \mathcal{I}_{\max}$ ($r \in \mathcal{I}_{\min}$). Identification of the index r can either be done by searching through all regions $\mathcal{R}_i$, $i = 1, \dots, R$ sequentially, or by evaluating a corresponding binary search tree (Tøndel et al., 2003). In either case, the required memory storage is proportional to the total number of regions R.

If the number of saturated regions is non-zero, a simpler representation of κ can be obtained. Notice that, since the regions $\mathcal{R}_i$ are non-overlapping due to Theorem 3.1, for any $x \in \Omega$, $x \notin \mathcal{R}_{\mathcal{I}_{\text{unsat}}}$, $\kappa(x)$ can only take two possible values: either $\kappa(x) = \overline{\kappa}$, or $\kappa(x) = \underline{\kappa}$. This fact can be exploited to derive a new PWA function $\widetilde{\kappa}(x)$ which maintains the equivalence $\widetilde{\kappa}(x) = \kappa(x)$ for all $x \in \Omega$, and requires less memory for its description compared to the memory footprint of $\kappa(x)$.

Proposition 4.1. Let a function $p : \mathbb{R}^{n_x} \to \mathbb{R}$ which satisfies $p(x) > 0$ for all $x \in \mathcal{R}_{\mathcal{I}_{\max}}$ and $p(x) < 0$ for all $x \in \mathcal{R}_{\mathcal{I}_{\min}}$ be given. Define

$$\widetilde{\kappa}(x) = \begin{cases} K_i x + L_i & \text{if } x \in \mathcal{R}_{\mathcal{I}_{\text{unsat}}}, \\ \overline{\kappa} & \text{if } p(x) > 0, \\ \underline{\kappa} & \text{if } p(x) < 0. \end{cases} \tag{6}$$

Then, for all $x \in \Omega$, $\widetilde{\kappa}(x) = \kappa(x)$.

Proof. Follows directly from (5) and from the definition of p.

With a separator p at hand, $u^* = \kappa(x)$ can be evaluated by only looking at the unsaturated regions $\mathcal{R}_{\mathcal{I}_{\text{unsat}}}$. If $x \in \mathcal{R}_r$, $r \in \mathcal{I}_{\text{unsat}}$, then $u^* = K_r x + L_r$. Otherwise, based on the sign of $p(x)$, one either takes $u^* = \overline{\kappa}$ or $u^* = \underline{\kappa}$. The separating function p always exists. Since κ is continuous by Theorem 3.1, regions $\mathcal{R}_j$ and $\mathcal{R}_k$ cannot be adjacent for any $j \in \mathcal{I}_{\max}$, $k \in \mathcal{I}_{\min}$, and therefore they can always be separated by a (possibly discontinuous) function p.

As will be evidenced later, a typical explicit RHMPC feedback laws $u^* = \kappa(x)$ exhibits usually significantly smaller number of unsaturated regions in comparison to the number of saturated ones, i.e. $|\mathcal{I}_{\text{unsat}}| \ll |\mathcal{I}_{\max}| + |\mathcal{I}_{\min}|$. Therefore $\widetilde{\kappa}$ will require significantly less memory than κ, and will be faster to evaluate too, if p is a "simple" separator of the two sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$. Various types of separators can be considered, either continuous (e.g. linear or polynomial), or discontinuous (e.g. piecewise linear or piecewise polynomial). In this work we have opted for the polynomial type of separating functions p and the problem which we aim at solving is formally stated as follows.

Problem 4.2. Given a RHMPC feedback law $u^* = \kappa(x)$ with κ as in (5), construct the replacement feedback (6) by finding the multivariate polynomial

$$p(x) := \sum_{i_1+\cdots+i_n \le \delta} (\alpha_{i_1,\ldots,i_\delta} x_1^{i_1} \cdots x_n^{i_n}), \tag{7}$$

of *minimum degree* $\delta_{\min}$ such that p strictly separates the sets of regions $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$, i.e. $p(x) > 0\ \forall x \in \mathcal{R}_{\mathcal{I}_{\max}}$ and $p(x) < 0\ \forall x \in \mathcal{R}_{\mathcal{I}_{\min}}$.

Solving Problem 4.2 is, however, nontrivial, since the sets

$$\mathcal{R}_{\mathcal{I}_{\max}} = \{x \mid x \in \cup_i \mathcal{R}_i,\ i \in \mathcal{I}_{\max}\}, \tag{8a}$$
$$\mathcal{R}_{\mathcal{I}_{\min}} = \{x \mid x \in \cup_j \mathcal{R}_j,\ j \in \mathcal{I}_{\min}\}, \tag{8b}$$

can in general be non-convex. Even deciding whether they are convex or not is an NP-hard problem (Bemporad et al., 2001).

5. POLYNOMIAL SEPARATION

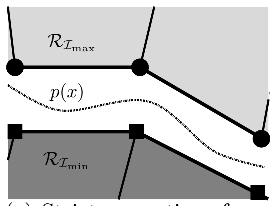
Given are the (non-convex) sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ as in (8), each of which consists of a finite number of polytopes $\mathcal{R}_k$. The knowledge of whether the sets are convex or not is not relevant here. Denote by $\mathcal{V}_k$ the vertices of $\mathcal{R}_k$ and fix some integer $\delta \ge 1$ in (7). Then the necessary condition for the existence of a polynomial p which strictly separates $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ is feasibility of the following optimization problem:

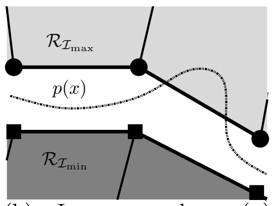
$$\epsilon^* = \max_{\epsilon,\alpha_i} \epsilon \tag{9a}$$
$$\text{s.t. } p(\mathcal{V}_i) \ge \epsilon, \quad \forall i \in \mathcal{I}_{\max}, \tag{9b}$$
$$p(\mathcal{V}_j) \le -\epsilon,\ \forall j \in \mathcal{I}_{\min}. \tag{9c}$$
$$\epsilon \ge 0. \tag{9d}$$

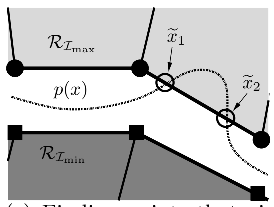
The optimal value ϵ^* then denotes the maximal separation gap between the two sets of points $\mathcal{V}_{\mathcal{I}_{\max}}$ and $\mathcal{V}_{\mathcal{I}_{\min}}$. It is important to notice that (9) is a linear program since, for some fixed argument $x = v_k$, $v_k \in \mathcal{V}_k$, $p(x)$ in (9b)–(9c) are linear functions of the coefficients α_i. If the LP (9) is infeasible, then no strict polynomial separator p of the form of (7) exists for a given degree δ.

If $\delta = 1$ in (9) then having $\epsilon^* > 0$ is also sufficient for the linear function $p(x) := \alpha_0 + \alpha_1 x$ to strictly separate the sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ (Boyd and Vandenberghe, 2004). Consider therefore $\delta > 1$. If (9) is feasible with $\epsilon^* > 0$, then one of the two possible scenarios can occur. In an ideal case, solving for coefficients of p from (9) by only considering separation of $\mathcal{V}_{\mathcal{I}_{\max}}$ and $\mathcal{V}_{\mathcal{I}_{\min}}$ will also provide a separator for the sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$, as shown in Fig. 1(a). In a more general case, though, strict separation of vertices is not sufficient for $p(x)$ to separate *all points* from the associated sets.

An additional certification step therefore has to be performed. At this point we remind that all regions of $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ are polytopes described by $\mathcal{R}_i = \{x \mid F_i x \le g_i\}$.


$\mathcal{R}_{\mathcal{I}_{\max}}$
$p(x)$
$\mathcal{R}_{\mathcal{I}_{\min}}$
(a) Strict separation of vertices can sometimes imply strict separation of the associated sets.


$\mathcal{R}_{\mathcal{I}_{\max}}$
$p(x)$
$\mathcal{R}_{\mathcal{I}_{\min}}$
(b) In general, $p(x)$ correctly separating $\mathcal{V}_{\mathcal{I}_{\max}}$ and $\mathcal{V}_{\mathcal{I}_{\min}}$ does not imply strict separation of $\mathcal{R}_{\mathcal{I}_{\max}}$ from $\mathcal{R}_{\mathcal{I}_{\min}}$.


$\mathcal{R}_{\mathcal{I}_{\max}}$
$\widetilde{x}_1$
$\widetilde{x}_2$
$p(x)$
$\mathcal{R}_{\mathcal{I}_{\min}}$
(c) Finding points that violate borders requires solving the problem (10).

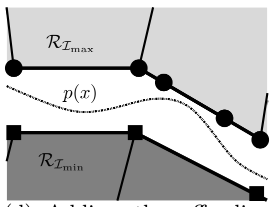

$\mathcal{R}_{\mathcal{I}_{\max}}$
$p(x)$
$\mathcal{R}_{\mathcal{I}_{\min}}$
(d) Adding the offending points to $\mathcal{V}_{\mathcal{I}_{\max}}$ and resolving (9) leads to a new separating polynomial $p(x)$.

Fig. 1. Sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$, vertices $\mathcal{V}_{\mathcal{I}_{\max}}$ (squares), vertices $\mathcal{V}_{\mathcal{I}_{\min}}$ (circles), and the polynomial separator $p(x)$.

Consider the k-th facet of $\mathcal{R}_i$, i.e. $\{x \mid f_{i,k}x - g_{i,k} = 0\}$ where $f_{i,k}$ and $g_{i,k}$ are the k-th rows of the respective matrices F_i and g_i. Denote by $\widetilde{x}_{i,k}$ all solutions to the polynomial equation $p(x) = f_{i,k}x - g_{i,k}$, restricted to $x \in \mathcal{R}_i$:

$$\widetilde{x}_{i,k} = \{x \mid p(x) - f_{i,k}x + g_{i,k} = 0,\ x \in \mathcal{R}_i\}. \tag{10}$$

Clearly, if $\widetilde{x}_{i,k} = \emptyset\ \forall i \in \mathcal{I}_{\max} \cup \mathcal{I}_{\min}$ and $\forall k$, then p as a solution to (9) strictly separates $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ (cf. Figure 1(a)). On the other hand, the situation in Figure 1(c) corresponds to the case where there exist some points $\widetilde{x}_{i,k}$ for which the polynomial $p(x)$ intersects the k-th facet of the i-th region, i.e. when $\widetilde{x}_{i,k} \ne \emptyset$ for some i and k. In such a case, the existence of *any* such point $\widetilde{x}_{i,k}$ provides a certificate that $p(x)$ does not separate $\mathcal{R}_{\mathcal{I}_{\max}}$ from $\mathcal{R}_{\mathcal{I}_{\min}}$.

When at least one offending point $\widetilde{x}_{i,k}$ exists, it can be added to the corresponding set of vertices in (9b)–(9c). I.e., if $\widetilde{x}_{i,k} \ne \emptyset$ for some $i \in \mathcal{I}_{\max}$, then $\mathcal{V}_{\mathcal{I}_{\max}} = \mathcal{V}_{\mathcal{I}_{\max}} \cup \widetilde{x}_{i,k}$. Otherwise, if $i \in \mathcal{I}_{\min}$, then $\mathcal{V}_{\mathcal{I}_{\min}} = \mathcal{V}_{\mathcal{I}_{\min}} \cup \widetilde{x}_{i,k}$. Resolving the LP (9) with the updated list of vertices will then give a new polynomial p for which the certification is repeated, cf. Figure 1(d). If more offenders are found, they are added to the list of vertices and the procedure is repeated. Otherwise, an empty solution to (10) provides a certificate that $p(x)$ strictly separates $\mathcal{R}_{\mathcal{I}_{\max}}$ from $\mathcal{R}_{\mathcal{I}_{\min}}$, whereupon the procedure terminates. The discussed mechanism can be formally stated as Algorithm 1, reported next.

Remark 5.1. Vertex enumeration in Step 1 of Algorithm 1 is considered a hard problem in general. However, for the type of small-dimensional problems considered here, enumerating $\mathcal{V}$ does not pose any significant technical difficulty and the vertices can be easily computed e.g. by CDD (Fukuda, 1997) in a matter of seconds.

Algorithm 1 Construction of a polynomial separator

```
INPUT: Sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ as in (8), polynomial degree $\delta$.
OUTPUT: Separating polynomial $p$ as in (7).
 1: Get the lists of vertices $\mathcal{V}_{\mathcal{I}_{\max}}$ and $\mathcal{V}_{\mathcal{I}_{\min}}$.
 2: repeat
 3:   Solve the LP (9) and obtain coefficients $\alpha_0, \dots, \alpha_\delta$.
 4:   if $\epsilon^* > 0$ then
 5:     Compute the list of offending points $\widetilde{x}_{i,k}$ from (10).
 6:     Insert $\widetilde{x}_{i,k}$ to $\mathcal{V}_{\mathcal{I}_{\max}}$ or $\mathcal{V}_{\mathcal{I}_{\min}}$.
 7:   else
 8:     No strict polynomial separator of degree $\delta$ exists, abort.
 9:   end if
10: until $\widetilde{x}_{i,k} \neq \emptyset$.
```

Remark 5.2. There is no theoretical guarantee that the iterations between Steps 2–10 will terminate in finite time. However, for more than 500 random problems reported in Section 7, the number of iterations newer exceeded 10. Even more importantly, in 90% of cases Algorithm 1 terminated after a single iteration.

Solving Problem 4.2 involves finding a strict separator p of the minimum degree $\delta_{\min}$. This can be achieved e.g. by using bisection, i.e. by running Algorithm 1 multiple times for various values of δ until a feasible solution is obtained and δ is minimized.

The list of offending points $\widetilde{x}$ in Step 5 can be obtained by solving (10) in several ways, as reviewed in the sequel.

5.1 Certification via Root Finding

The list of offending points $\widetilde{x}_{i,k}$ in Step 5 of Alg. 1 can be found by interpreting (10) as a problem of finding roots of the polynomial $p(x)-f_{i,k}x-g_{i,k}$, constrained to $x \in \mathcal{R}_i$ for a particular index i. If $n_x = 1$, then the roots can be found conveniently e.g. using the `roots` command of MATLAB.

If $n_x = 2$, then the problem can be solved as follows. Consider the k-th defining hyperplane of polytope $\mathcal{R}_i$, i.e. $f_{i,k}x - g_{i,k} = 0$. Each point on this hyperplane can be expressed as

$$\{x \mid f_{i,k}^T(x - x_{0,k}) = 0\}, \tag{11}$$

where $x_{0,k}$ is any point on such hyperplane, e.g. a suitable vertex of region $\mathcal{R}_i$. This representation can in turn be expressed as

$$\{x \mid x = x_{0,k} + f_{i,k}^{\perp}\}, \tag{12}$$

where $f_{i,k}^{\perp}$ represents the orthogonal complement to $f_{i,k}$, i.e. the set of all vectors orthogonal to it

$$f_{i,k}^{\perp} = \{v \mid f_{i,k}^T v = 0\} \tag{13}$$

Substituting the orthogonal representation (12) into the definition of $p(x)$ and $f_{i,k}x + g_{i,k} = 0$ converts (10) into a problem with only one variable:

$$p(v) - f_{i,k}(x_{0,k} + f_{i,k}^{\perp}v) - g_{i,k} = 0. \tag{14}$$

Consequently, the `roots` command can be used to find all roots $\widetilde{v}_{i,k}$ of (14), from which $\widetilde{x}_{i,k}$ can be recovered from (12). The method is applied on the new polynomial to compute roots v. Note that only real roots need to be considered. Since the procedure does not limit the roots $\widetilde{x}_{i,k}$ to a particular domain ($\mathcal{R}_i$ in our case), it is necessary to obtain *all* roots of (14) and subsequently exclude those which do not belong to $\mathcal{R}_i$.

5.2 Nonlinear Programming Approach to Certification

Although the root finding procedure is easy to implement, it is limited to 1D and 2D situations, only. Another option to find out whether the list of offenders $\widetilde{x}$ is non-empty is to consider (10) as a feasibility problem with linear inequality constraints ($x \in \mathcal{R}_i$) and nonlinear equality constraint ($p(x) = f_{i,k}x - g_{i,k}$). Such an approach is applicable to arbitrary dimensions of x. Recalling that $\mathcal{R}_i = \{x \mid F_i x \le g_i\}$, the list of offenders is non-empty iff there exists a solution to the following problem:

$$\text{find } x \tag{15a}$$
$$\text{s.t. } p(x) = f_{i,k}x - g_{i,k}, \tag{15b}$$
$$F_i x \le g_i, \tag{15c}$$

which can be solved e.g. by `fmincon` of MATLAB. The practical disadvantage of such a formulation lies in the fact that equality constraints are sensitive to numerical noise.

An alternative way is to reformulate (15) as an NLP with nonlinear objective function and linear inequality constraints. Recall that a valid separator has to guarantee that $p(x) > 0$ for all $x \in \mathcal{R}_{\mathcal{I}_{\max}}$. Similarly, $p(x) < 0$ is required for all $x \in \mathcal{R}_{\mathcal{I}_{\min}}$. Let

$$f_{i,\max}^* = \min\ p(x) \tag{16a}$$
$$\text{s.t. } F_i x \le g_i, \tag{16b}$$

and

$$f_{i,\min}^* = \max\ p(x) \tag{17a}$$
$$\text{s.t. } F_i x \le g_i. \tag{17b}$$

The it immediately follows that if $f_{i,\max}^* < 0$ for some $i \in \mathcal{I}_{\max}$ (or if $f_{i,\min}^* > 0$ for some $i \in \mathcal{I}_{\min}$), then the point x_i^* as an optimal solution to (16) or (17) is a valid offending point which shows that $p(x)$ is *not* a strict separating polynomial.

6. SEPARATION BY BINARY TREES

An another alternative is to separate the sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ by a (possibly discontinuous) piecewise linear function p, as shown in different context by Fuchs et al. (2010). There the authors search for a separator represented as a binary search tree. Each node k of the tree represents one linear separator of the form $p_k(x) := \alpha_{k,1}x - \alpha_{k,0}$. The task then becomes to find the coefficients such that p_k correctly separates as many elements of $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ as possible. The misclassified elements are treated in a recursive fashion while building a tree. The search for p_k at each level of the tree can be cast as a mixed-integer linear program

$$\min \left| \sum R_i - \sum L_j \right| + \sum |R_i + L_j - 1| \tag{18a}$$
$$\text{s.t. } R_i = 1 \Leftrightarrow \{p_k(x) \ge \underline{\epsilon}\ \forall x \in \mathcal{R}_i,\ i \in \mathcal{I}_{\max}\}, \tag{18b}$$
$$L_j = 1 \Leftrightarrow \{p_k(x) \le -\underline{\epsilon}\ \forall x \in \mathcal{R}_j,\ j \in \mathcal{I}_{\min}\}, \tag{18c}$$

where $\underline{\epsilon} > 0$ is a given minimal separation gap introduced to avoid the trivial solution $\alpha_{k,1} = \alpha_{k,0} = 0$. Binary variables R_i (L_j) denote whether or not the corresponding region of $\mathcal{R}_{\mathcal{I}_{\max}}$ ($\mathcal{R}_{\mathcal{I}_{\min}}$) is correctly classified by a linear separator p_k, while minimizing the number of incorrectly separated regions by (18a). The crucial downside of such an approach is that a total of $|\mathcal{I}_{\max}|+|\mathcal{I}_{\min}|$ binaries needs to be introduced. If the number exceeds ~ 700 (which is considered a small case by our standards), the size of the MILP (18) becomes prohibitive to be solved even using state-of-the-art solvers, such as CPLEX.

Therefore we propose a different method of finding the linear separators $p_k(x) := \alpha_{k,1}^T x - \alpha_{k,0}$ at each level of the tree by solving a convex relaxation of (18):

$$\min_{u,v,\alpha} \ 1^T u + 1^T v \tag{19a}$$

$$\text{s.t. } \alpha_{k,1}^T x_i + \alpha_{k,0} \geq 1 - u_i, \quad i = 1, \dots, \mathcal{V}_{max}, \tag{19b}$$

$$\alpha_{k,1}^T y_i + \alpha_{k,0} \leq -(1 - v_i), \ i = 1, \dots, \mathcal{V}_{min}, \tag{19c}$$

$$u \succeq 0, \quad v \succeq 0, \tag{19d}$$

where x_i (y_i) represent extremal vertices of regions saturated at maximum $\mathcal{R}_{max}$ (regions saturated at minimum $\mathcal{R}_{min}$), and u_i and v_i are nonnegative *support vector classifiers* of x_i and y_i, respectively.

When $u = v = 1$ in (19b) and (19c), we recover original constraints $\alpha_{k,1}^T x_i - \alpha_{k,0} \geq 0$ and $\alpha_{k,1}^T y_i - \alpha_{k,0} \leq 0$ as a feasible nonstrict linear discriminant of two sets of points. Think of u_i as a measure how much the constraint $\alpha_{k,1}^T x_i - \alpha_{k,0} \geq 0$ is violated and the same holds for v_i. The goal is to find $\alpha_{k,1}$, $\alpha_{k,0}$ and sparse u and v that satisfy inequalities (19b) – (19c), maximize the slab, and minimize the number of misclassified points. In other words, (19) is a relaxation of the number of points misclassified by the function $\alpha_{k,1}^T z - \alpha_{k,0}$, plus the number of points that are correctly classified but lie in the slab of width $\{z \mid -1 \leq \alpha_{k,1}^T z - \alpha_{k,0} \leq 1\}$ given by $2/\|\alpha_{k,1}\|_2$ (Boyd and Vandenberghe (2004)).

The Algorithm 2 shows a pseudocode of the recursive function used to build a binary tree. Inputs to the function are 2 sets of saturated regions ($\mathcal{R}_{min}, \mathcal{R}_{max}$) which are immediately transformed into 2 sets of extremal vertices ($\mathcal{V}_{min}, \mathcal{V}_{max}$). Acquired PWA function splits points into correctly and incorrectly classified ones, where regions corresponding to the misclassified points are searched. If the sets are empty, algorithm reached a *leaf nodes*, otherwise the solution is referred as *node* and function is recalled with new sets of regions. Steps of the Algorithm 2 are depicted in Fig. 2.

7. EXAMPLES

7.1 Illustrative example

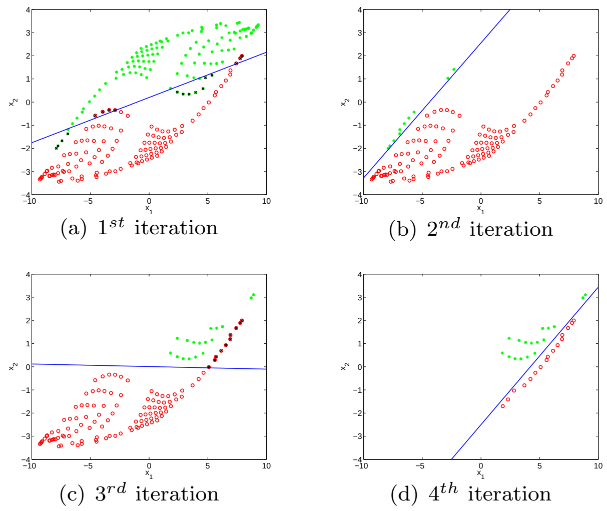
Consider a 2-state 1-input system given by

$$x^+ = \begin{bmatrix} 0.755 & 0.680 \\ 0.651 & -0.902 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0.825 \\ -0.139 \end{bmatrix} u, \tag{20}$$

which is subject to constraints $\mathcal{X} = \{[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}] \mid -10 \leq [\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}] \leq 10\}$ and $\mathcal{U} = \{u \in \mathbb{R} \mid -1 \leq u \leq 1\}$. The MPC problem (3) was formulated with prediction horizon $N = 10$, $Q_x = \mathbb{1}$

Algorithm 2 Construction of a PWA form of separator $p(x)$

```
INPUT: Sets $\mathcal{R}_{\mathcal{I}_{\max}}$ and $\mathcal{R}_{\mathcal{I}_{\min}}$ as in (8).
OUTPUT: Separator $p$ encoded as a set of linear functions.
 1: function BINARYTREE($\mathcal{R}_{\mathcal{I}_{\min}}, \mathcal{R}_{\mathcal{I}_{\max}}$)
 2: Get the lists of vertices $\mathcal{V}_{min}$ and $\mathcal{V}_{max}$.
 3: Solve the LP (19) and create a new node in the tree
    defined by $p_k(x) := \alpha_{k,1}^T x - \alpha_{k,0}$.
 4: Find misclassified points $\mathcal{V}_{min}$, $\mathcal{V}_{max}$ and keep corresponding regions $\mathcal{R}_{min}$, $\mathcal{R}_{max}$.
 5: if $\mathcal{R}_{min} \neq \emptyset$ then
 6:     return BINARYTREE($\mathcal{R}_{min}$, $\mathcal{R}_{\mathcal{I}_{\max}}$).
 7: end if
 8: if $\mathcal{R}_{max} \neq \emptyset$ then
 9:     return BINARYTREE($\mathcal{R}_{\mathcal{I}_{\min}}$, $\mathcal{R}_{max}$).
10: end if
11: end function
```



(a) 1st iteration (b) 2nd iteration

(c) 3rd iteration (d) 4th iteration

Fig. 2. Illustration of finding a PWA separator by Algorithm 2. At each iteration a new node is created which contains a linear separator p_k correctly separating as many points as possible. Shown are the vertices $\mathcal{V}_{max}$ (green stars), vertices $\mathcal{V}_{min}$ (red circles), misclassified points (black x-marks), and the linear separator $p_k(x)$ (continuous line) at each iteration.

and $Q_u = 1$ and solved as a parametric QP according to Theorem 3.1. Using the MPT Toolbox (Kvasnica et al., 2004), the explicit RHMPC feedback $u^* = \kappa(x)$ was obtained in 4 seconds[1] as a PWA function defined over 225 regions. The domain of κ consists of 29 unsaturated regions, 98 regions where $\kappa(x) = 1$, and 98 regions where $\kappa(x)$ is saturated at -1.

As can be clearly seen from Figure 2(a), no linear separation between the sets of points $\mathcal{V}_{\mathcal{I}_{\max}}$ and $\mathcal{V}_{\mathcal{I}_{\min}}$ could be found. A polynomial separator $p(x) = -x_1 - x_2 + 0.6103x_1x_2^2 - 0.2076x_1^2x_2 + 0.0458x_1^3 + x_2^3$ of the minimal degree $\delta_{\min} = 3$ was then found by applying bisection in conjunction with Algorithm 1. The algorithm converged within of one iteration. The vertices in Step 1 were com-

[1] On a 2.4 GHz CPU with 4GB of RAM using MATLAB 7.9 and MPT 2.6.3

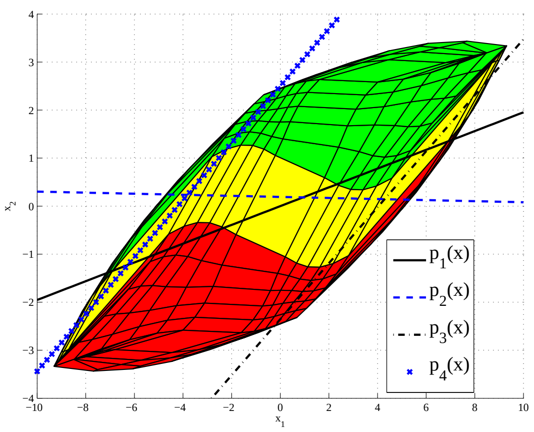


Fig. 3. Final form of a binary tree division. Linear separators $p_k(x)$ represents each of the nodes from binary tree.

puted by CDD in 0.01 seconds. Coefficients of the polynomial were obtained by solving the LP (9), which took 0.7 seconds using CPLEX. Implementing the certification in Step 5 using root finding (cf. Section 5.1) took 1.85 seconds, while the NLP-based certification of Section 5.2 took 6.55 seconds.

A binary separation tree can also be constructed by recursively solving LP problems (19). For the sets of points depicted in Figure 2, the procedure generated a tree consisting of four nodes:

$$p_1(x) = -0.38x_1 + 1.96x_2 \quad (21)$$
$$p_2(x) = 0.019x_1 + 1.75x_2 - 0.34 \quad (22)$$
$$p_3(x) = -2.26x_1 + 3.87x_2 + 9.12 \quad (23)$$
$$p_4(x) = -0.22x_1 + 0.37x_2 - 1.0 \quad (24)$$

The tree is rooted at p_1, with p_2 visited if $p_1(x) < 0$ and p_3 if $p_1(x) > 0$. The next node is rooted at p_2, with p_4 visited if $p_2(x) > 0$ which betrays the unbalanced binary tree. The total runtime of LPs (19) was 4.7 seconds with CDD.

However, as can be seen from Fig. 3, a binary tree with just 3 nodes would be sufficient to correctly separate the corresponding vertices. However, Algorithm 2 finds 4 nodes. This is due to the fact that Alg. 2 attempts to find, at each iteration, a best subdivision of the points by solving the LP (19). No effort is made to minimize the cardinality or the depth of the resulting PWA tree. This can, however, be achieved by an adequate post-processing procedure.

The total memory footprint of κ (which consists of the regions $\mathcal{R}_i$ and the feedback laws $K_i x + L_i$) with 225 regions, is 27 kilobytes. On the other hand, by devising a separator p, the storage requirements of $\widetilde{\kappa}$ drops to a mere 3.5 kilobytes. Here, the unsaturated regions $\mathcal{R}_{\mathcal{I}_{\text{unsat}}}$ contribute by 2.8 kB, the associated feedback laws by 0.7 kB, and the memory footprint is just 24 bytes for the polynomial separator, and 48 bytes for the binary tree. It follows that complexity of the on-line implementation of the RHMPC feedback law can be reduced by a factor of 7.7 by using the modified feedback $u^* = \widetilde{\kappa}(x)$ instead of the original function $u^* = \kappa(x)$.

Table 1. Comparison of total runtimes for different routines in polynomial separation ($\delta_{min} = 3$), selected problems.

No. of regions	Runtime [sec]	
	`fmincon`	`roots`
189	6.15	2.73
199	5.69	2.85
225	7.82	2.96
257	7.61	3.46
495	14.23	6.36

Table 2. Number of nodes and runtimes necessary for rooting, comparison of LP (19) with MILP algorithm of Fuchs et al. (2010)

No. of regions			LP		MILP	
total	sat	unsat	No. of nodes	total runtime [sec]	No. of nodes	total runtime [sec]
189	172	17	4	4.2	3	8.6
199	154	45	4	6.7	3	7.0
225	196	29	4	6.7	3	9.1
257	208	49	4	5.0	4	17.3
283	244	39	5	6.1	4	23.7
319	260	59	4	5.6	3	12.5
495	460	35	5	13.7	48	1176.9

7.2 Random examples

Next, we have analyzed a large number of random RHMPC feedback laws κ generated by solving problem (3). We have considered 100 random problems with 2 states and 1 input. For each PWA function we have constructed the replacement $\widetilde{\kappa}$ as in (6). Both polynomial and binary tree separation were considered.

The purpose of this study was twofold. First, we have investigated how the root-finding approach to certification compares to the more general NLP-based procedure of Section 5.2. Table 1 shows that the root finding procedure performs twice as fast as the NLP approach. Moreover, it illustrates how the computation scales with increasing complexity of the problem. It should be noted, though, that the NLP procedure is applicable to arbitrary dimensions, while the root finding approach is limited to 2D scenarios only.

Next, we compared the runtime needed to construct a PWA separator, encoded as a binary search tree, using the MILP procedure of Fuchs et al. (2010) and by the convex LP relaxation (19). The results summarized in Table 2 show that the LP relaxation is significantly more efficient, for it being able to construct the PWA separator even for large scenarios. Note also that the number of unsaturated regions (denoted by *unsat*) is significantly smaller in comparison to the number of saturated regions (*sat*).

8. CONCLUSIONS

Given an explicit RHMPC feedback function κ, we have shown how to construct its simpler replacement $\widetilde{\kappa}$ which maintains the equivalence $\kappa(x) = \widetilde{\kappa}(x)$ for all $x \in \operatorname{dom}\kappa(x)$. The mechanism was based on devising a function $p(x)$, which separates regions over which κ attains a saturated value. The replacement $\widetilde{\kappa}$ then requires only the storage of the unsaturated regions of κ, along with the separator p. We have shown how to build such a p by solving a linear optimization problem, followed by a certification step which requires solution to a polynomial equation. Two approaches to such a certification were proposed: one based on finding roots of a polynomial, and second one based on solving a nonlinear programming problem. By means of a case study we have illustrated how different approach to certification influence the total computation time. When a piecewise affine separator is desired, we have proposed to use a convex relaxation of the separation problem, which is significantly more efficient compared to the approach based on solving a mixed integer separation problem.

ACKNOWLEDGMENT

The authors are pleased to acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic under the grant 1/0095/11. This work was supported by the Slovak Research and Development Agency under the contracts No. VV-0029-07 and No. LPP-0092-07. This work was also financed by a grant (No. NIL-I-007-d) from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

Baotic, M., Borrelli, F., Bemporad, A., and Morari, M. (2008). Efficient On-Line Computation of Constrained Optimal Control. *SIAM Journal on Control and Optimization*, 47(5), 2470–2489.

Bemporad, A. and Filippi, C. (2003). Suboptimal explicit RHC via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications*, 117(1), 9–38.

Bemporad, A., Fukuda, K., and Torrisi, F.D. (2001). Convexity Recognition of the Union of Polyhedra. *Computational Geometry*, 18, 141–154.

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

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Cychowski, M. and O'Mahony, T. (2005). Efficient off-line solutions to robust model predictive control using orthogonal partitioning. In *Proceedings of the 16th IFAC world congress*.

Fuchs, A., Jones, C., and Morari, M. (2010). Optimized Decision Trees for Point Location in Polytopic Data Sets - Application to Explicit MPC. In *American Control Conference*. Baltimore, USA.

Fukuda, K. (1997). *Cdd/cdd+ Reference Manual*. Available from www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html.

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

Grieder, P., Wan, Z., Kothare, M., and Morari, M. (2004). Two level model predictive control for the maximum control invariant set. In *American Control Conference*. Boston, Massachusetts.

Johansen, T. and Grancharova, A. (2003). Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Trans. on Automatic Control*, 48, 810–815.

Jones, C. and Morari, M. (2009). Approximate Explicit MPC using Bilevel Optimization. In *European Control Conference*. Budapest, Hungary.

Kvasnica, M. and Fikar, M. (2010). Performance-lossless complexity reduction in explicit MPC. In *Conference on Decision and Control, CDC*, (accepted). Atlanta, USA.

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

Kvasnica, M., Rauová, I., and Fikar, M. (2011). Simplification of Explicit MPC Feedback Laws via Separation Functions. In *IFAC World Congress*. Milano, Italy.

Scibilia, F., Olaru, S., and Hovd, M. (2009). Approximate explicit linear MPC via delaunay tessellation. In *Proceedings of the 10th European Control Conference*. Budapest, Hungary.

Tøndel, P., Johansen, T.A., and Bemporad, A. (2003). Evaluation of Piecewise Affine Control via Binary Search Tree. *Automatica*, 39(5), 945–950.

Wen, C., Ma, X., and Ydstie, B.E. (2009). Analytical expression of explicit MPC solution via lattice piecewise-affine function. *Automatica*, 45(4), 910 – 917. doi: DOI:10.1016/j.automatica.2008.11.023.