Katholieke Universiteit Leuven

Departement Elektrotechniek

ESAT-SISTA/TR 04-230

Reduced Complexity Polyhedral Invariant Sets for LPV Systems [1]

B. Pluymers[2] , J.A. Rossiter[3] , J. Suykens[2] and B. De Moor[2]

March 2005

Internal Report 04-230

[1]This report is available by anonymous ftp from *ftp.esat.kuleuven.ac.be* in the directory *pub/sista/pluymers/reports/ECC-CDC05a-redcomp.pdf*

[2]K.U.Leuven, Dept. of Electrical Engineering (ESAT), Research group SCD-SISTA, Kasteelpark 10, 3001 Leuven, Belgium, Tel. 32/16/32 10 35, Fax 32/16/32 19 70, WWW: *http://www.esat.kuleuven.ac.be/scd*. E-mail: {*bert.pluymers,johan.suykens,bart.demoor*}*@esat.kuleuven.ac.be*.

Research supported by Research Council KUL: GOA-Mefisto 666, GOA-Ambiorics, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects, G.0240.99 (multilinear algebra), G.0407.02 (support vector machines), G.0197.02 (power islands), G.0141.03 (Identification and cryptography), G.0491.03 (control for intensive care glycemia), G.0120.03 (QIT), G.0452.04 (QC), G.0499.04 (robust SVM), research communities (ICCoS, ANMMM, MLDM); AWI: Bil. Int. Collaboration Hungary/ Poland; IWT: PhD Grants, GBOU (McKnow) Belgian Federal Government: Belgian Federal Science Policy Office: IUAP V-22 (Dynamical Systems and Control: Computation, Identification and Modelling, 2002-2006), PODO-II (CP/01/40: TMS and Sustainibility); EU: FP5-Quprodis; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE; Contract Research/agreements: ISMC/IPCOS, Data4s, TML, Elia, LMS, IPCOS, Mastercard; Bert Pluymers is a research assistant with the I.W.T. (Flemish Institute for Scientific and Technological Research in Industry). Dr. Johan Suykens is an associate professor at the Katholieke Universiteit Leuven. Dr. Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium.

[3]University of Sheffield, Department of Automatic Control and Systems Engineering, Mappin Street, Sheffield S1 3JD, United Kingdom, E-mail: *j.a.rossiter@sheffield.ac.uk*

Abstract

Recently an algorithm was proposed (Pluymers *et al.*, 2005) for constructing polyhedral invariant sets for linear systems with polytopic uncertainty description. Polyhedral invariant sets have the advantage of being less conservative than ellipsoidal invariant sets but have the disadvantage that the number of constraints describing the invariant set can become impractically large for high-dimensional systems. This paper shows that invariant sets with a reduced number of constraints can be constructed by modifying the algorithm introduced in (Pluymers *et al.*, 2005). A tuning parameter is introduced to control the trade off between the number of constraints and the volume of the invariant set. The efficiency of the algorithm is illustrated on a numerical example.

Reduced Complexity Polyhedral Invariant Sets for LPV Systems

B. Pluymers†, J. A. Rossiter‡, J. A. K. Suykens†, B. De Moor†

†Katholieke Universiteit Leuven
Department of Electrical Engineering, ESAT-SCD-SISTA
E-Mail : {bert.pluymers, johan.suykens, bart.demoor}@esat.kuleuven.ac.be
Internet : http://www.esat.kuleuven.ac.be/scd/

‡University of Sheffield
Department of Automatic Control and Systems Engineering
E-Mail : j.a.rossiter@sheffield.ac.uk
Internet : http://www.shef.ac.uk/acse/

***Abstract*— Recently an algorithm was proposed [7] for constructing polyhedral invariant sets for linear systems with polytopic uncertainty description. Polyhedral invariant sets have the advantage of being less conservative than ellipsoidal invariant sets but have the disadvantage that the number of constraints describing the invariant set can become impractically large for high-dimensional systems. This paper shows that invariant sets with a reduced number of constraints can be constructed by modifying the algorithm introduced in [7]. A tuning parameter is introduced to control the trade off between the number of constraints and the volume of the invariant set. The efficiency of the algorithm is illustrated on a numerical example.**

I. INTRODUCTION

Invariant sets are a powerful tool for stability analysis and control synthesis for constrained systems. Invariant sets act as a basin of attraction and guarantee that if the current state of the system lies within the invariant set, all future states will stay within the set and hence will satisfy the imposed system constraints. for an overview of set invariance we refer to [2].

The construction of maximal size invariant sets (MAS, Maximal Admissable Set) for LTI systems was introduced in [3]. The construction of the MAS for LPV systems, however, remained unsolved until recently. Invariant ellipsoids and parallelotopes for LPV systems as inner approximations of the MAS were discussed in [5] and [6] respectively. However, both can exhibit significant conservatism, e.g. when asymmetric constraints are imposed on the system.

Recently in [7], a new algorithm was introduced for the exact construction of the MAS for LPV systems. The algorithm avoids the expected exponential increase of the number of constraints through elimination of all redundant constraints and is therefore the first algorithm that makes the computation of such sets practically feasible. However, when dealing with higher dimensional systems, the number of non-redundant constraints describing the invariant set can still rise exponentially, prohibiting the practical application of the algorithm in these cases. In this paper we introduce a new algorithm that is able to make a trade-off between the number of constraints and the volume of invariant sets for LPV systems.

This paper is organized as follows. Section II first introduces some general definitions, after which Section III gives an overview of algorithms for the construction of the MAS for LTI and LPV systems. Section IV then describes the branching effect that leads to the exponentially increasing number of constraints describing the MAS of large dimensional systems. A new algorithm for the construction of reduced complexity invariant sets is given in Section V after which a numerical example is given in Section VI. Finally, Sections VII state the conclusions and future work.

II. GENERAL

In this paper autonomous systems of the following form are considered :

$$x(k+1) = \Phi(k)x(k), \tag{1}$$

with $x(k) \in \mathbb{R}^{n_x}$ denoting the state of the system at discrete time k and $n_x \in \mathbb{N}_0$ denoting the number of states. $\Phi(k)$ belongs to a given uncertainty polytope Ω :

$$\Phi(k) \in \Omega \equiv \mathrm{Co}\left\{\Phi_1, \ldots, \Phi_L\right\}, \tag{2}$$

and denotes the time-varying behaviour. Systems with $L = 1$ and $L > 1$ will be called LTI and LPV system respectively. In the former case we denote $\Phi \equiv \Phi(k) \equiv \Phi_1$. We assume that the system is subject to linear inequality constraints :

$$x(k) \in \mathcal{X} \equiv \{x | A_x x(k) \leq \mathbf{1}\}, \qquad k = 0, \ldots, \infty, \tag{3}$$

with $\mathbf{1}$ a column vector of appropriate size containing only ones. Further, we also assume that the system is robustly asymptotically stable :

$$\lim_{n\to\infty} \max_{\substack{\Phi(k)\in\Omega, k=0,\ldots,n-1 \\ \|x_0\|=1}} \|x(n)\|_2 = 0. \tag{4}$$

Given these definitions we can now introduce the concept of invariance.

Definition 1 (Robust Positive Invariance): *Given a system* (1)-(2) *satisfying* (4) *then* $\mathcal{S} \in \mathbb{R}^{n_x}$ *is a robust positive invariant set iff*

$$\Phi x \in \mathcal{S}, \qquad \forall x \in \mathcal{S}, \forall \Phi \in \Omega. \tag{5}$$

Definition 2 (Feasibility): *An invariant set $\mathcal{S}$ for a system* (1)-(2) *is feasible with respect to constraints* (3) *iff*

$$\mathcal{S} \subseteq \mathcal{X}. \tag{6}$$

Remark 1: In what follows we will use the shorter notation *invariant set* instead of *feasible robust positive invariant set*, unless where stated otherwise. The largest invariant set (in the sense that no other invariant set contains points outside this set) is uniquely defined and is called the maximal admissable set (MAS, [3]).

Throughout this paper, Matlab notation is used for matrices :

$A_{(i,j)}$	:	element on row i and column j of A,
$A_{(i,:)}$	:	ith row of A,
$A_{(i_1:i_2,:)}$	:	rows i_1 to i_2 of matrix A.

III. POLYHEDRAL INVARIANT SETS FOR LINEAR SYSTEMS

This section briefly gives an overview of the construction of the MAS for LTI systems ([3]) and LPV systems ([7]).

A. The MAS for LTI systems

It can be shown that, given a system (1)-(2) with $L = 1$ subject to constraints (3), the set $\mathcal{S}_\infty$ is the MAS, with $\mathcal{S}_i$ defined as

$$\mathcal{S}_i = \{x | A_x x \leq \mathbf{1}, A_x \Phi x \leq \mathbf{1}, \ldots, A_x \Phi^i x \leq \mathbf{1}\}. \tag{7}$$

Under certain convergence conditions, roughly speaking equivalent to condition (4) it can be shown ([3]) that

$$\exists i^\star \in \mathbb{N} \quad : \quad \mathcal{S} \equiv \mathcal{S}_\infty = \mathcal{S}_i, \forall i \geq i^\star, \tag{8}$$

indicating that the number of constraints describing $\mathcal{S}$ is finite and that $\mathcal{S}$ is therefore polyhedral. The following algorithm then constructs the MAS $\mathcal{S}$:

Algorithm 1 (MAS for LTI systems, [3]): Given a system (1)-(2) with $L = 1$ subject to constraints (3) and satisfying (4), perform the following steps :

1) set $i := 0$,
2) increase i until $\mathcal{S}_{i+1} = \mathcal{S}_i$,
3) return $\mathcal{S} = \mathcal{S}_i$.

B. The MAS for LPV systems

Theoretically, the MAS for LPV systems can also be calculated using Algorithm 1 combined with a modified definition for $\mathcal{S}_i$ taking into account all possible predictions of the modified system :

$$\mathcal{S}_i = \bigcap_{i=0,\ldots,i} \mathcal{X}_i, \tag{9}$$

with $\mathcal{X}_0 = \mathcal{X}$ and $\mathcal{X}_i, i > 0$ defined as

$$\mathcal{X}_i = \bigcap_{[j_1,\ldots,j_i] \in \{1,\ldots,L\}^i} \left\{ x | A_x \prod_{k=1,\ldots,i} \Phi_{j_k} x \leq \mathbf{1} \right\}. \tag{10}$$

It can still be shown ([7]) that also in this case property (8) holds, but the number of constraints defining $\mathcal{S}_{i^\star}$ increases exponentially as a function of $i^\star$, which makes the application of Algorithm 1 unpractical in the LPV case. Figure 1 illustrates this exponential increase.

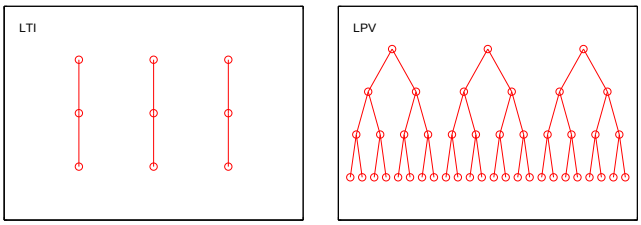


Fig. 1. Schematic comparison of the application of Algorithm 1 to LTI and LPV systems. Each circle represents a constraint defining $\mathcal{S}$. Both examples have 3 constraints imposed on the system (=number of trees) and $i^\star = 3$ (=tree depth). $L = 2$ (=tree branching coefficient) for the LPV system. It is clear that in the LPV case the number of constraints increases exponentially as a function of $i^\star$.

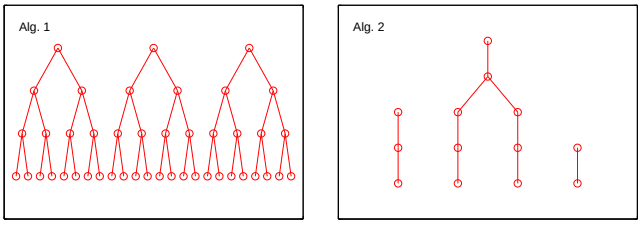


Fig. 2. Schematic comparison of the application of Algorithm 1 and Algorithm 2 to the LPV system of Figure 1. Note that both algorithms produce exactly the same set $\mathcal{S}$, but the result of Algorithm 1 typically contains a significant number of redundant constraints.

In [7] a different (but equivalent) invariance condition is proven :

$$\mathcal{S} \subseteq \mathcal{S}^-, \tag{11}$$

with the $^-$-operator defined as

$$\mathcal{S}^- \equiv \{x | \Phi x \in \mathcal{S}, \forall \Phi \in \Omega\}. \tag{12}$$

Note that $\mathcal{S}^-$ can easily be constructed as

$$\mathcal{S}^- = \{x | A_\mathcal{S} \Phi_1 x \leq \mathbf{1}, \ldots, A_\mathcal{S} \Phi_L x \leq \mathbf{1}\}. \tag{13}$$

The following algorithm can now be introduced ([7]) :

Algorithm 2 (MAS for LPV systems, [7]): Given a system (1)-(2) with $L \geq 1$ subject to constraints (3) and satisfying (4), perform the following steps :

1) set $A_\mathcal{S} := A_x$ and define $\mathcal{S} = \{x | A_\mathcal{S} x \leq \mathbf{1}\}$,
2) set $i := 1$,
3) for all $j = 1, \ldots, L$, perform :
   - if the constraint $a_i^{\mathrm{T}} \Phi_j \leq 1$ (with a_i^{T} defined as the ith rows of $A_\mathcal{S}$) is non-redundant w.r.t. $\mathcal{S}$, then add it to $\mathcal{S}$ by adding a row below to $A_\mathcal{S}$.
4) if $i < \#\text{rows}(A_\mathcal{S})$, then set $i := i + 1$ and continue with step 3), otherwise return $\mathcal{S}$.

Remark 2: The above algorithm can dramatically decrease the number of constraints defining $\mathcal{S}$ by only adding non-redundant constraints to $A_\mathcal{S}, b_\mathcal{S}$. In all observed cases this reduces the exponential number of constraints to a linear number of constraints as a function of $i^\star$ (for constant n_x). Figure 2 schematically illustrates this reduction of the number of constraints.

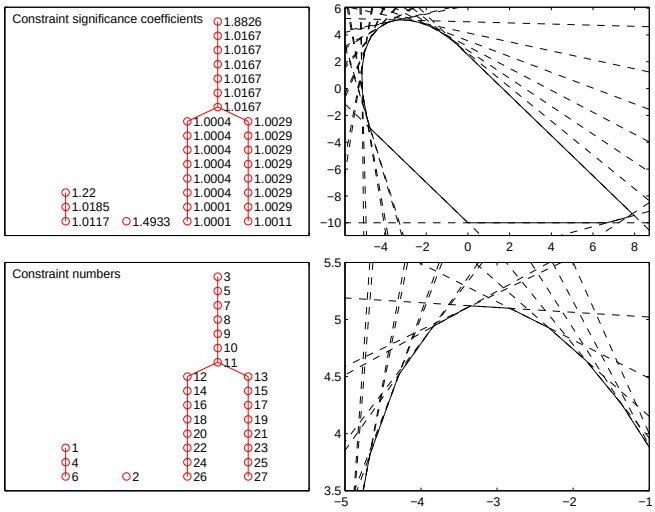


Fig. 3. Structure and shape of the MAS of system (14)-(15) constructed using Algorithm 2. **Upper left :** Structure of the constraints describing the MAS together with the constraint significance coefficients. **Upper right :** Shape of the MAS $\mathcal{S}$ in the state space. **Lower left :** Structure of the MAS together with the constraint numbers. **Lower right :** Close up of part of the MAS where branching can be observed.

Remark 3: Crucial for the efficiency of the above algorithm is the intermediate removal of redundant constraints (called *garbage collection*), since adding new constraints in each iteration can make previously added constraints redundant. See [7] for more details.

IV. BRANCHING

In this section a simple example is used to explain the *branching phenomenon* that leads to an increased number of constraints in the LPV case. A method to avoid this phenomenon is introduced in the final part of this section.

We consider the LPV system described by the following matrices :

$$\Phi_1 = \begin{bmatrix} 1.00 & 0.10 \\ -0.30 & 0.80 \end{bmatrix}, \quad \Phi_2 = \begin{bmatrix} 1.00 & 0.10 \\ -0.33 & 0.78 \end{bmatrix}, \quad (14)$$

subject to constraints ($\forall k \geq 0$) :

$$\begin{aligned} [-10 \ \ -10]^{\mathrm{T}} \leq & \quad x(k) \quad \leq [10 \ 10]^{\mathrm{T}}, \\ -2 \leq & \ [0.3 \ 0.2] \cdot x(k) \ \leq 0.5. \end{aligned} \quad (15)$$

The MAS of system (14)-(15) and its structure are depicted in Figure 3. The structure of the MAS (depicted twice in the left part of Figure 3) can be directly related to the structure of Algorithm 2. In step 1 of the algorithm the invariant set $\mathcal{S}$ is initialized with constraints (15), of which constraints 1, 2 and 3 (Fig. 3, lower left) are remnants in the final result. In each iteration it is checked for one constraint of $\mathcal{S}$ whether its *children* (i.e., its corresponding constraints in $\mathcal{S}^-$) are redundant with respect to $\mathcal{S}$. The non-redundant constraints are then added to $\mathcal{S}$ as *leaves* of the constraint under investigation and as such a tree of constraints is constructed. The final result can be seen in the upper and lower left of Figure 3. This tree structure is not directly relevant when the set $\mathcal{S}$ is eventually used in its intended application, but leads to some interesting insights that are discussed in the next paragraph.

Remark 4: In what follows we will explicitly differentiate between the terms *child* and *leaf*. *Children* of a constraint of $\mathcal{S}$ denote the corresponding constraints in the related set $\mathcal{S}^-$, whereas *leaves* denote the constraints that are effectively added to the tree structure of the set $\mathcal{S}$. The *branches* of a constraint denote all (possibly indirect) leaves appearing below a constraint.

It can be observed in Figure 3 (lower left) that both children of constraint 11 (i.e., both corresponding constraints in $\mathcal{S}^-$) are non-redundant and are included as leaves of constraint 11 in the set $\mathcal{S}$. This leads to a *branching event* leading to two parallel branches in the tree structure which in turn increases the number of constraints describing $\mathcal{S}$. This can also be observed in the lower right part of the figure, which is a close-up of a part of $\mathcal{S}$. The constraints in the right part of this close-up (from right to left : constraints 3,5,7,8,9,10,11) all appear as separate constraints, while the constraints in the left part of the close-up all appear as couples of almost identical constraints (e.g. constraints 12 and 13, constraints 14 and 15, ...). It are these branching events that occur more frequently with increasing system dimensionality, resulting in a steeply increasing number of constraints for larger dimensional systems. Therefore, it would be favorable if branching could be avoided.

It can be seen in Figure 3 that there is one constraint in each of the above described couples that is only marginally non-redundant as can also be seen in the upper left part of Figure 3. The coefficients stated next to the depicted trees denote the factors with which each constraint has to be relaxed in order to make it redundant. It is clear that all constraints in the left branch below constraint 11 are only marginally non-redundant compared to the constraints in the right branch.

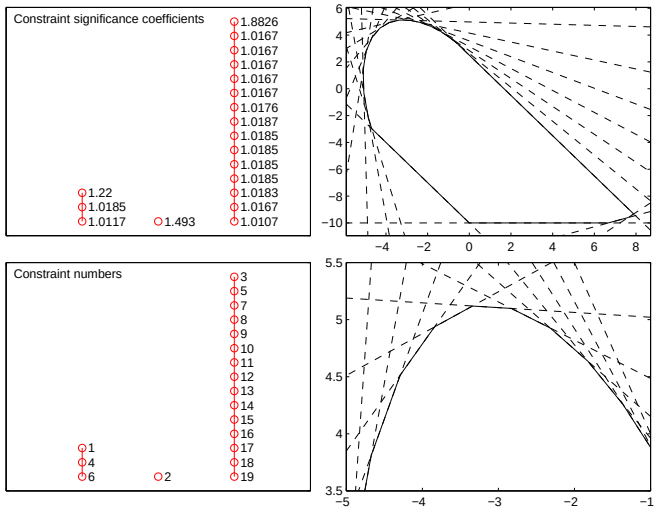


Fig. 4. Invariant set with a reduced number of constraints. By making constraint 13 of the invariant set depicted in Figure 3 stricter, a branching event is avoided and the number of constraints is reduced from 27 to 19, while only slightly reducing the volume of the invariant set.

The idea pursued here is, when a branching event occurs, to slightly modify some of the constraints in order to make the constraints in one of the two branches redundant. However, when relaxing the constraints in one of the branches, the resulting set is no longer invariant. On the other hand, the following theorem suggests that under certain conditions it is allowed to make some constraints of the set stricter without losing the invariance property of the set :

Theorem 1 (Invariance conserving constr. modification): Given an autonomous LPV system (1)-(2) subject to constraints (3) and a set $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$ satisfying the invariance condition (11), a new set $\mathcal{S}'$ can be constructed by making constraint i of set $\mathcal{S}$ stricter with a factor c. The set $\mathcal{S}'$ also satisfies the invariance property (11) if

1) all constraints located in the branches of constraint i are also made stricter with a factor c,
2) all children of constraint i and all children of the constraints in the branch of constraint i that are redundant in set $\mathcal{S}$, remain redundant in their modified form in the modified set $\mathcal{S}'$.

Proof: Invariance of set $\mathcal{S}'$ is proven if condition (11) is satisfied. This condition is satisfied if all constraints of $\mathcal{S}'^{-}$ are redundant with respect to $\mathcal{S}'$. All constraints in $\mathcal{S}'^{-}$ corresponding to unchanged constraints in $\mathcal{S}'$ will remain redundant, since $\mathcal{S}' \subset \mathcal{S}$. All children of constraints that have changed in $\mathcal{S}'$ will also remain redundant. Those that were already included in $\mathcal{S}$ are now also already included in $\mathcal{S}'$ in their modified form (condition 1), while those that were not already included will remain redundant in their modified form due to condition 2. This shows that $\mathcal{S}'$ satisfies condition (11) and is therefore invariant. ■

This theorem is used to eliminate the left branch in the invariant set depicted in Figure 3. Constraint 11 was made stricter with a factor 1.0004 in order to make the left branch redundant. The resulting set is depicted in Figure 4. The number of constraints has been reduced from 27 to 19 with only a minimal volume reduction (smaller than 0.04%).

The ultimate aim, however, would be to eliminate the branching *during* the construction of the invariant set and not *afterwards*. This is discussed in the next section.

V. REDUCED COMPLEXITY INVARIANT SETS

In this section the insights gathered in the previous section are used to modify Algorithm 2 in order to obtain an algorithm that is able to eliminate branching during the construction of the invariant set.

The new algorithm is based on the following elements :

- **Invariant set construction :** The main algorithm consists of the same iteration steps as Algorithm 2, except for some smaller modifications to be discussed below.
- **Branching detection/elimination :** At certain intermediate steps (e.g., after every 10 iterations, ...) a branching detection / elimination sub-algorithm is invoked, consisting of the following elements :
  - selection of branches to be rendered redundant and related branches to be made stricter based on a heuristic measure using linear programming, making a trade-off between the number of constraints that can be eliminated and the expected volume reduction,
  - calculation of the exact reduction factor c (see Theorem 1) necessary to make the selected constraints redundant.
- **Invariance :** In the above sub-algorithm some constraints are made stricter at certain points in the algorithm, but it is not guaranteed that condition 2 of Theorem 1 is satisfied. Therefore, after execution of the branching elimination algorithm, the redundancy of all children of the constraints that have been made stricter is rechecked.
- **Convergence :** In order to be able to guarantee convergence (despite the fact that certain constraints are made stricter in intermediate steps), it is necessary to impose an upper to limit to the factors c with which constraints are made stricter. This upper limit can be calculated based on $\max_i \|\Phi_i\|$.

These elements are further discussed in the following subsections.

A. Bookkeeping

The above described sub-algorithm to intermediately modify some of the constraints can be seen as a form of backtracking and hence some additional bookkeeping needs to be done. Apart from the matrix $A_{\mathcal{S}} \in \mathbb{R}^{n \times n_x}$, where n is the current number of constraints describing $\mathcal{S}$, the new algorithm also keeps track of the following matrices in each iteration :

$p_{\mathcal{S}} \in \mathbb{R}^{n \times 1}$	:	parent of each constraint
$c_{\mathcal{S}} \in \mathbb{R}^{n \times 1}$	:	multipliers
$t_{\mathcal{S}} \in \mathbb{R}^{n \times 1}$	:	uncertainty polytope node
$d_{\mathcal{S}} \in \mathbb{R}^{n \times 1}$	:	depth of constraint in tree structure

The following two properties, that are always maintained during the execution of the algorithm, further clarify the relationship of the above matrices :

$$A_{\mathcal{S},(i,:)} = c_{\mathcal{S},i} \Phi_{t_{\mathcal{S},i}} A_{\mathcal{S},(p_{\mathcal{S},i},:)}, \qquad \forall i : p_{\mathcal{S},i} \neq 0, \tag{16a}$$

$$d_{\mathcal{S},i} = d_{p_{\mathcal{S},i}} + 1, \qquad \forall i : p_{\mathcal{S},i} \neq 0. \tag{16b}$$

In order to keep the above conditions satisfied after each iteration (apart from the obvious modifications in steps 1) and 3) of Algorithm 2), the following measures are taken into account :

- if a constraint i is made stricter, all its children (constraint $j : p_{\mathcal{S},j} = i$) are to be made stricter (recursively),
- during garbage collection, redundant constraints are only removed if they have no leaves in the tree.

Conditions (16) facilitate the implementation of the algorithms in the following subsections :

B. Branching elimination

The following theorem forms the basis for the elimination of branching :

Theorem 2: Consider sets $\mathcal{S}_1$ and $\mathcal{S}_2$, defined by constraint matrices $A_{\mathcal{S}_1}$ and $A_{\mathcal{S}_2}$ that are identical except for constraints $i_1, \ldots, i_s$:

$$A_{\mathcal{S}_2,(i_j,:)} = \frac{1}{c} A_{\mathcal{S}_1,(i_j,:)}, \qquad j = 1, \ldots, s, \tag{17}$$

with $c < 1$. Furthermore, consider a constraint $a^{\mathrm{T}} x \leq 1$ that is non-redundant with respect to $\mathcal{S}_1$ and $\mathcal{S}_2$ and respective solutions $r_1 > r_2 > 1$ to the following optimization problems :

$$r_1 = \max_{x \in \mathcal{S}_1} a^{\mathrm{T}} x, \tag{18a}$$

$$r_2 = \max_{x \in \mathcal{S}_2} a^{\mathrm{T}} x. \tag{18b}$$

Constraint $a^{\mathrm{T}} x \leq 1$ is then redundant with respect to set $\mathcal{S}_3$, defined by constraint matrix $A_{\mathcal{S}_3}$ that is identical to $A_{\mathcal{S}_1}$ except for constraints $i_1, \ldots, i_s$

$$A_{\mathcal{S}_3,(i_j,:)} = \frac{1}{c'} A_{\mathcal{S}_1,(i_j,:)}, \qquad j = 1, \ldots, s, \tag{19}$$

with

$$c' = \frac{r_2 + (c-1) - c \cdot r_1}{r_2 - r_1}, \tag{20}$$

if $c' \leq c$.

Proof: We have to prove that $r_3 \leq 1$, with r_3 defined as

$$r_3 = \max_x a^{\mathrm{T}} x \qquad \text{s.t.} \qquad x \in \mathcal{S}_3. \tag{21}$$

First we define indices $j_1, \ldots, j_l$ denoting the constraint numbers of the active constraints in optimization problem (18b). Define the sets $\mathcal{S}_1', \mathcal{S}_2', \mathcal{S}_3'$ as the sets defined by constraints $j_1, \ldots, j_l$ of respectively $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ and define r_1', r_2', r_3' as the results of the correspondingly modified optimization problems (18a), (18b) and (21). Due to the fact that $r_1^\star > r_2^\star$ we know that at least one of the constraints $i_1, \ldots, i_s$ is included in the active set $j_1, \ldots, j_l$ and hence $\mathcal{S}_1', \mathcal{S}_2', \mathcal{S}_3'$ will be mutually different.

By construction we know that $r_1' \geq r_1, r_2' = r_2$ and that $r_3' \geq r_3$. Due to the linear dependence of r_3' on c' (due to the fact that all constraints of $\mathcal{S}_3$ are active constraints in the calculation of r_3') we can express r_3' as a function of r_1' and r_2'. This leads, after some algebraic manipulation, to

$$r_3' = r_2' - (r_2 - 1)\frac{r_2' - r_1'}{r_2 - r_1}, \tag{22}$$

and since $r_2' = r_2$ we get

$$r_3' = r_2 - (r_2 - 1)\frac{r_2 - r_1'}{r_2 - r_1}. \tag{23}$$

Finally, because $r_1' \geq r_1$, we get

$$r_3' \leq 1. \tag{24}$$

Because of the fact that $r_3' \geq r_3$, this proves that $r_3 \leq 1$. ■

The above theorem provides a way to determine the factor c' with which a set of constraints $i_1, \ldots, i_2$ should be made stricter in order to make another constraint redundant. However, first the initial factor c and the set of constraints $i_1, \ldots, i_2$ has to be selected. This selection is performed in the following algorithm :

Algorithm 3 (branching elimination): Given a set $\mathcal{S}$ (not necessarily invariant), defined by $A_{\mathcal{S}}$, its tree structure defined by $p_{\mathcal{S}}, c_{\mathcal{S}}, t_{\mathcal{S}}, d_{\mathcal{S}}$ and a user-specified constant $\gamma > 0$ determining the desired trade-off between complexity reduction and volume conservation of the set. Iterate top-down over the tree structure (constraints with the largest number of constraints in their branch first) and perform the following steps for each constraint i :

1) Calculate the constraint redundancy coefficient r_i as

$$r_i = \max_x A_{\mathcal{S},(i,:)} x \quad \text{s.t.} \quad A_{\mathcal{S}'} x \leq \mathbf{1}, \tag{25}$$

with $A_{\mathcal{S}'}$ equal to $A_{\mathcal{S}}$ with row i removed. Denote the active constraint numbers as $a_1, \ldots, a_l$.

2) Construct a set $i_1, \ldots, i_s$ containing all constraints $a_1, \ldots, a_l$ (but excluding all parents of constraint i and excluding all branches dependent of constraint i) and all branches dependent of these constraints.
3) If $s > 0$, choose $c = 0.999$ and calculate c' using Theorem 2. Due to the construction of the set $i_1, \ldots, i_s$ we know that $r_1 \geq r_2$. If it turns out that $c' > c$ then choose $c \in [c'\ 1]$, recalculate c' and reiterate until $c' < c$.
4) If $\frac{\gamma \cdot (1+w)}{\frac{1}{c'} - 1} > 1$, with w the number of constraints dependent of constraint i, perform the following steps
   a) Make the constraints $i_1', \ldots, i_s'$ of $\mathcal{S}$, corresponding to constraints $i_1, \ldots, i_s$ of $\mathcal{S}'$ stricter with factor c'

$$A_{\mathcal{S},(i_j',:)} := \frac{1}{c'} A_{\mathcal{S},(i_j',:)}, \qquad j = 1, \ldots, s. \tag{26}$$

   b) For all *root constraints* i_j' in $i_1', \ldots, i_s'$ (i.e., all constraints in $i_1', \ldots, i_s'$ that have no parents in $i_1', \ldots, i_s'$), set $c_{\mathcal{S},i_j'} := c' \cdot c_{\mathcal{S},i_j'}$.
   c) Perform garbage collection in order to remove constraint i and all redundant constraints in its branches from $\mathcal{S}$.
   d) Terminate the algorithm.

The above algorithm looks for branching elimination opportunities, based on a criterion that makes a trade-off between complexity reduction and volume conservation and performs one branching elimination operation (based on Theorem 2).

Remark 5: By excluding certain constraints in step 2), it is guaranteed that 1) constraint i is not made stricter together with constraints $i_1, \ldots, i_s$ by imposing properties (16) and that 2) constraints i does not become non-redundant again by removing its branches in step 4c).

Remark 6: Due to the construction of the set $i_1, \ldots, i_s$ based on the active constraints $a_1, \ldots, a_l$, it is clear that $r_1 > r_2$ and that hence Theorem 2 can be used to calculate c'.

C. Main algorithm

We can now summarize the various elements from the previous sections in the following algorithm :

Algorithm 4 (Red. compl. MAS for LPV systems): Given a scalar $\gamma > 0$ and a system (1)-(2) with $L \geq 1$ subject to constraints (3) and satisfying (4), perform the following steps :

1) set $A_{\mathcal{S}} := A_x, p_{\mathcal{S}} = [0; \ldots; 0], c_{\mathcal{S}} = [1; \ldots; 1], t_{\mathcal{S}} = [1; \ldots; 1], d_{\mathcal{S}} = [1; \ldots; 1]$, and define $\mathcal{S} = \{x | A_{\mathcal{S}} x \leq \mathbf{1}\}$,
2) set $i := 1$,
3) for all $j = 1, \ldots, L$, perform :
   - If the constraint $a_i^{\mathrm{T}} \Phi_j \leq 1$ (with a_i^{T} defined as the ith rows of $A_{\mathcal{S}}$) is non-redundant w.r.t. $\mathcal{S}$, then add it to $\mathcal{S}$ by adding a row below to $A_{\mathcal{S}}$.
   - Update $p_{\mathcal{S}}, c_{\mathcal{S}}, t_{\mathcal{S}}, d_{\mathcal{S}}$ according to (16) for all newly added constraints.
4) If <garbage_collection_trigger> then perform garbage collection by removing all redundant constraints.
5) If <branch_elim_trigger> then call Algorithm 3 and set $i := 1$.
6) if $i < \#\mathrm{rows}(A_{\mathcal{S}})$, then set $i := i + 1$ and continue with step 3), otherwise return $\mathcal{S}$.

Remark 7: <garbage_collection_trigger> denotes a condition to be defined by the user determining when garbage collection is invoked. Experiments indicate that garbage collection each time the number of constraints has grown by about 50% is a good rule of thumb.

Remark 8: <branch_elim_trigger> denotes a condition to be defined by the user determining when branching elimination algorithm is invoked. Experiments indicate that branching elimination works best right after garbage collection has been performed. An upper bound n on the number of times branching elimination is invoked should be imposed in order to ensure convergence of the algorithm.

Remark 9: Due to the fact that after branching elimination the variable i is reset to 1, it is still guaranteed that the resulting set $\mathcal{S}$ satisfies (11) and hence is still invariant.

D. Convergence

The proof of convergence of Algorithm 2 (see [7]) is based on the fact that $n_j = \max_{i:d_{\mathcal{S},i}=j} \|A_{\mathcal{S},(i,:)}\|$ is a strictly decreasing function of j and that therefore, for sufficiently large values of j all constraints will become redundant with respect to the constraint defining $\mathcal{X}$, which are used to initialize $A_{\mathcal{S}}$. Convergence of Algorithm 4 is still guaranteed if the factors $c_{\mathcal{S}}$ are sufficiently small. Using recursions (16) we can recursively bound the values of n_j :

$$n_j \leq \left(\max_{i:d_i=j} c_{\mathcal{S},i} \right) \left(\max_{i=1,\ldots,L} \|\Phi_i\| \right) n_{j-1}. \qquad (27)$$

Convergence is hence proven if

$$\max_i c_{\mathcal{S},i} \leq c_{\max} \equiv \left(\max_{i=1,\ldots,L} \|\Phi_i\| \right)^{-1}, \qquad (28)$$

which is satisfied (based on the heuristic used in step 4) of Algorithm 3) if

$$\gamma \leq \frac{1 - \sqrt[n]{c_{\max}}}{\sqrt[n]{c_{\max}}(1 + w_{\max})}, \qquad (29)$$

with n the number of times Algorithm 3 is called and $w_{\max}$ the maximum (over all calls to Alg. 3) number of rows in $A_{\mathcal{S}}$. The above upper bound ensures that at each call to Algorithm 3, the maximum value for c' is equal to $\sqrt[n]{c_{\max}}$, which ensures that no c_i can ever exceed the value $c_{\max}$. It should be noted that (29) can be extremely conservative and that larger values of γ can also lead to convergence of Algorithm 4. This leads to the main conclusion of the above discussion that for 'sufficiently small' values of γ (possibly violating (29)), convergence of Algorithm 4 can be expected.

VI. EXAMPLE

The construction of polyhedral invariant sets for LPV systems has initially been developed with the aim of enabling non-conservative constraint handling in robust MPC algorithms [8], [9], [11]. In this section we examine the performance of Algorithm 4 when applying it to construct an invariant set for use in the robust MPC algorithm introduced in [9]. This algorithm reduces the calculation of the constraints used in the online optimization problem to the construction of an invariant set for a $(N \cdot n_u + n_x)$-dimensional augmented LPV system. The disadvantage of applying Algorithm 2 is that the number of constraints describing the invariant set increases exponentially as a function of N, leading to an impractically large online computational cost. In this section Algorithm 4 is applied in order to obtain reduced complexity invariant sets leading to a reduced computational complexity of the robust MPC algorithm.

We consider randomly generated LPV systems with $L = 2, n_x = 2, n_u = 1$. $[A_1 \ B_1]$ are generated by choosing normally distributed (variance of 1) elements. A_1 is normalized such that the maximum eigenvalue is equal to 0.9, B_1 is normalized such that $\|B_1\| = 0.1$. $[A_2 \ B_2]$ are generated by adding normally distributed (standard deviation 0.01) perturbations to all elements of $[A_1 \ B_1]$ after which $[A_2 \ B_2]$ are normalized in the same way as $[A_1 \ B_1]$. All systems are subject to constraints $[-10 \ -10]^{\mathrm{T}} \leq x \leq [10 \ 10]^{\mathrm{T}}$ and $-1 \leq u \leq 1$ denoted as $A_x x \leq \mathbf{1}$ and $A_u u \leq \mathbf{1}$. As feedback controllers used in the MPC algorithms, we choose $K_1 = K, K_2 = 0.75K$ with K the LQR optimal controller for $[(A_1 + A_2)/2 \ (B_1 + B_2)/2]$ with $Q = I, R = I$.

The robust MPC algorithm described in [9] makes use of polyhedral invariant sets of an augmented system $\tilde{x}(k+1) = \Phi(k)\tilde{x}(k)$ with the vertices Φ_1, Φ_2 of the uncertainty polytope defined as

$$\Phi_i = \begin{bmatrix} A_i + B_i K & B_i & 0 \\ 0 & 0 & I_{((N-1)\cdot n_u,(N-1)\cdot n_u)} \\ 0 & 0 & 0 \end{bmatrix}, \qquad (30)$$

where N is the prediction horizon, subject to constraints $A_{\tilde{x}}\tilde{x}(k) \leq \mathbf{1}, k \geq 0$, with $A_{\tilde{x}}$ defined as

$$A_{\tilde{x}} = \begin{bmatrix} A_x & 0 & 0 \\ A_u K & A_u & 0 \end{bmatrix}. \qquad (31)$$

We generate 10 random systems as discussed in the previous subsection, substitute these system descriptions in (30)-(31) and calculate polyhedral invariant sets for $N = 1, \ldots, 4$. Results are shown in Table I and Figure 5. It is clear that the new

	$\gamma = 0$		$\gamma = 0.005$	
N	#constraints	volume	#constraints	volume
1	12.2	143.95	8.5	143.73
2	22.0	154.02	11.6	152.53
3	38.0	165.23	14.7	160.35
4	68.2	178.05	17.5	165.83

TABLE I

AVERAGE NUMBER OF CONSTRAINTS AND MEAN VOLUMES (OF THE 2D-PROJECTION) OF THE INVARIANT SETS (CALCULATED USING ALGORITHM 4) FOR SYSTEM (30)-(31) FOR 10 RANDOMLY GENERATED SYSTEMS AND DIFFERENT VALUES OF γ.

algorithm (with $\gamma = 0.005$) is able to construct invariant sets with a significantly reduced number of constraints compared to the standard algorithm ($\gamma = 0$). The volume reduction of the feasible region (projection of the invariant onto the first 2 dimensions) is kept acceptably small. It is important to note that the number of constraints is a linear function of N when using the new algorithm, whereas it is an exponential function of N when using Algorithm 2 (i.e., Algorithm 4 with $\gamma = 0$).

VII. CONCLUSIONS

In this paper a new algorithm to construct reduced complexity polyhedral invariant sets for LPV systems was introduced. The algorithm makes explicit use of the underlying tree structure of the constraints in the invariant set and reduces *branching* in order to reduced the number of constraints. The new algorithm is based on linear programming and can be guaranteed to converge in a finite number of iterations. Numerical results indicate that the new algorithm is very effective in reducing the number of constraints for examples encountered in MPC applications, while only minimally reducing the volume of the invariant sets.

Future research directions include more thoroughly investigating the performance of the new algorithm when used in MPC algorithms. Another point of interest is reducing the computation time of the new algorithm, which is currently not significantly lower than that of the original algorithm.

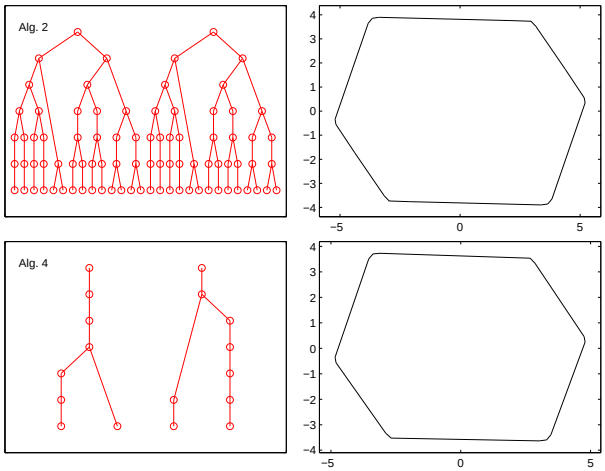


Fig. 5. Structure and projection onto first 2 dimensions (i.e., feasible region of MPC algorithm) of two invariant sets calculated for one system randomly generated as explained above ($N = 4$). **Top :** invariant set computed using Algorithm 2, **Bottom :** invariant set computed using Algorithm 4 ($\gamma = 0.005$), **Left :** tree structure of the calculated invariant sets, **Right :** projection of the invariant sets onto the first 2 dimensions.

VIII. ACKNOWLEDGMENTS

Research supported by: Royal Academy of Engineering and the Royal Society. Research Council KULeuven: GOA-Mefisto 666, several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects, G.0240.99, G.0407.02, G.0197.02, G.0141.03, G.0491.03, G.0120.03, research communities (ICCoS, ANMMM); AWI: Bil. Int. Collaboration Hungary/ Poland; IWT: PhD Grants, Soft4s (softsensors), Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006)), PODO-II (CP/40: TMS and Sustainibility); EU: CAGE; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE; Contract Research/agreements: Data4s, Electrabel, Elia, LMS, IPCOS, VIB. B. Pluymers is a research assistant with the I.W.T. (Flemish Institute for Scientific and Technological Research in Industry) at the Katholieke Universiteit Leuven. Dr. J. Suykens is an associate professor at the Katholieke Universiteit Leuven, Belgium. Dr. B. De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium.

REFERENCES

[1] F. Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions. *IEEE Transactions on Automatic Control*, 39:428–433, 1994.

[2] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.

[3] E. G. Gilbert and Tan. K. T. Linear systems with state and control constraints : The theory and application of maximal output admissable sets. *IEEE Transactions on Automatic Control*, 36(9):1008–1020, 1991.

[4] E. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Cambridge, 2000.

[5] M. V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379, 1996.

[6] Y. I. Lee and B. Kouvaritakis. Robust receding horizon predictive control for systems with uncertain dynamics and input saturation. *Automatica*, 36:1497–1504, 2000.

[7] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty description. *Accepted for publication in proceedings of the American Control Conference (ACC), Portland, USA*, 2005, (`http://www.esat.kuleuven.ac.be/~sistawww/cgi-bin/pub.pl`).

[8] B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. Interpolation based MPC for LPV systems using polyhedral invariant sets. *Accepted for publication in proceedings of the American Control Conference (ACC), Portland, USA*, 2005, (`http://www.esat.kuleuven.ac.be/~sistawww/cgi-bin/pub.pl`).

[9] B. Pluymers, J.A. Rossiter, J.A.K. Suykens, and B. De Moor. A simple algorithm for robust mpc. *Accepted for publication in proceedings of the IFAC World Congress, Prague, Czech Republic*, 2005, (`http://www.esat.kuleuven.ac.be/~sistawww/cgi-bin/pub.pl`).

[10] J.A. Rossiter, Y. Ding, B. Pluymers, J.A.K. Suykens, and B. De Moor. Interpolation based MPC with exact constraint handling: the nominal case. *Internal Report 04-231, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), submitted for publication*, 2005, (`http://www.esat.kuleuven.ac.be/~sistawww/cgi-bin/pub.pl`).

[11] J.A. Rossiter, Y. Ding, B. Pluymers, J.A.K. Suykens, and B. De Moor. Interpolation based MPC with exact constraint handling: the uncertain case. *Internal Report 04-232, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), submitted for publication*, 2005, (`http://www.esat.kuleuven.ac.be/~sistawww/cgi-bin/pub.pl`).