Robust Stabilization of Matrix Polytopes with the Cone Complementarity Linearization Algorithm: Numerical Issues

Didier Henrion[1,2,3]
Denis Arzelier[2]
Dimitry Peaucelle[2]

Abstract

The paper describes an application of the cone complementarity linearization algorithm to the robust stabilization by state-feedback of a polytopic linear system. The emphasis is put onto practical implementation issues and numerical properties.

1 Introduction

It is now widely recognized that most of the problems in robust control cannot be solved in a reasonable amount of time when problem dimensions are reasonably large, see the recent survey [2]. Several approaches have been pursued to get through this fundamental bottleneck. Some researchers decided to rely on probabilities and statistics, some others are further improving global optimization strategies to cope with larger problems at a lower computational cost, some others are rather concentrating on developing heuristics that prove efficient in practice. The present work belongs to the latter category.

We aim at studying the numerical behavior of the heuristic cone complementarity linearization (CCL) algorithm when applied to the notoriously difficult problem of state-feedback robust stabilization of a polytope of matrices. The reader is referred to [3] for an introduction to the CCL algorithm and its application to the static output feedback synthesis problem, as well as to our companion paper [1] for a brief overview of the robust synthesis problem for matrix polytopes and its relevance in control.

After a brief statement in Section 2 of the problem to be solved, Sections 3 and 4 describe in great detail the application of the CCL method when combined with recent theoretical

[1]Corresponding author. FAX: +33 5 61 33 69 69. E-mail: `henrion@laas.fr`

[2]Laboratoire d'Analyse et d'Architecture des Systèmes, Centre National de la Recherche Scientifique, 7 Avenue du Colonel Roche, 31 077 Toulouse, cedex 4, France.

[3]Also with the Institute of Information Theory and Automation, Czech Academy of Sciences, Pod vodárenskou věží 4, 182 08 Praha, Czech Republic.

achievements in robust stability analysis [8]. Intensive numerical experiments are reported in Section 5 that illustrate the relevance of the CCL algorithm but also some limitations of the approach. Possible extensions are eventually pointed out in Section 6. The MATLAB code of an implementation of the CCL algorithm is given in the Appendix. The corresponding m-file is available on request.

2 Problem Statement

We are given a linear system whose state and input matrices are assumed to belong to given convex polytopes whose matrix vertices are $\{A_1, A_2, \ldots, A_N\}$ and $\{B_1, B_2, \ldots, B_N\}$, respectively. We are seeking a state-feedback matrix K of size $n \times m$ such that every closed-loop matrix in the convex polytope

$$\mathcal{P} = \{A_1 + B_1 K, A_2 + B_2 K, \ldots, A_N + B_N K\}$$

is stable, i.e. its eigenvalues belong to some given region $\mathcal{D}$ of the complex plane $\mathbb{C}$.

The stability regions are described by

$$\mathcal{D} = \{s \in \mathbb{C} \;:\; \begin{bmatrix} 1 \\ s \end{bmatrix}^{\star} \begin{bmatrix} a & b^{\star} \\ b & c \end{bmatrix} \begin{bmatrix} 1 \\ s \end{bmatrix} < 0\}$$

where the Hermitian matrix with components (a, b, c) has inertia $(1, 0, 1)$. For example, $(a, b, c) = (0, 1, 0)$ corresponds to the open left half-plane (continuous-time systems) and $(a, b, c) = (-1, 0, 1)$ corresponds to the open unit disk (discrete-time systems).

3 Problem Formulation

Following [8], a sufficient condition for matrix K to robustly stabilize matrix polytope $\mathcal{P}$ is that there exists some matrices F, G and $P_1, P_2, \ldots, P_N$ such that

$$\begin{aligned} &\begin{bmatrix} A_i + B_i K \\ -I_n \end{bmatrix} \begin{bmatrix} F & G \end{bmatrix} + \left(\begin{bmatrix} A_i + B_i K \\ -I_n \end{bmatrix} \begin{bmatrix} F & G \end{bmatrix} \right)^{\star} + \begin{bmatrix} aP_i & b^{\star} P_i \\ bP_i & cP_i \end{bmatrix} \prec 0 \\ &P_i = P_i^{\star} \succ 0, \quad i = 1, 2, \ldots, N. \end{aligned} \tag{1}$$

Note that this robust stability analysis condition was shown to be significantly less conservative than the widely used quadratic stability condition, see [8] for details.

Enforcing the bilinear constraint

$$M = \begin{bmatrix} K \\ I_n \end{bmatrix} \begin{bmatrix} F & G \end{bmatrix} = \begin{bmatrix} X & Y \\ F & G \end{bmatrix}, \tag{2}$$

inequality (1) can be written as a linear matrix inequality (LMI) in matrix variables X,Y,F,G and P_i, namely

$$\begin{aligned} &\begin{bmatrix} A_i F + B_i X + (A_i F + B_i X)^{\star} + aP_i & A_i G + B_i Y - F^{\star} + b^{\star} P_i \\ (A_i G + B_i Y)^{\star} - F + bP_i & -G - G^{\star} + cP_i \end{bmatrix} \prec 0 \\ &P_i = P_i^{\star} \succ 0, \quad i = 1, 2, \ldots, N. \end{aligned} \tag{3}$$

Note that inequality (3) imply that $G + G^\star \succ 0$ hence that G is a non-singular matrix. The bilinear constraint (2) can thus be written equivalently

$$X = YG^{-1}F \tag{4}$$

and we have proved the following

Lemma 1 *A sufficient condition for robust stabilization by state-feedback of matrix polytope $\mathcal{P}$ is that there exists matrices X, Y, F, G and P_i satisfying both LMI (3) and non-linear matrix inequality (4). The state-feedback matrix is then $K = YG^{-1}$.*

There exists efficient (polynomial time) algorithms to solve convex LMI (3), see [6]. Unfortunately, equality (4) is non-convex in problem unknowns and no efficient algorithm has been proposed so far to address it. The remainder of the paper will focus on a heuristic for solving problem (3-4), i.e. an efficient numerical procedure that may find the solution at a low computational cost, but without guarantee to do so. In other words, it may fail sometimes, the trade-off being between the amount of computation involved and the overall failure rate.

4 CCL Algorithm

The heuristic is based on a idea proposed in [3], an extension to LMIs of a linearization technique based on complementarity. Before exposing the algorithm, we need the following

Lemma 2 *Problem (3-4) is feasible if and only if zero is the global optimum of the optimization problem*

$$\begin{array}{ll} \min & \operatorname{trace} \begin{bmatrix} T_1 & T_3^\star \\ T_3 & T_2 \end{bmatrix} \begin{bmatrix} Z_1 & Z_3^\star \\ Z_3 & Z_2 \end{bmatrix} \\ \text{s.t.} & \text{LMI (3)} \\ & \left[\begin{array}{cc|cc} Z_1 & Z_3^\star & X^\star & F^\star \\ Z_3 & Z_2 & Y^\star & G^\star \\ \hline X & Y & I_m & 0 \\ F & G & 0 & I_n \end{array}\right] \succeq 0 \\ & \begin{bmatrix} T_1 & T_3^\star \\ T_3 & T_2 \end{bmatrix} \succeq 0, \quad T_1 \succeq I_n. \end{array} \tag{5}$$

Proof: Since both

$$T = \begin{bmatrix} T_1 & T_3^\star \\ T_3 & T_2 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} Z_1 & Z_3^\star \\ Z_3 & Z_2 \end{bmatrix}$$

are positive semi-definite matrices, $\operatorname{trace} TZ = 0$ implies $TZ = 0$ thus $T_1Z_1 + T_3^\star Z_3 = 0$ and $T_1Z_3^\star + T_3^\star Z_2 = 0$. The second LMI above implies $Z_2 \succeq YY^\star + GG^\star$ and LMI (3) implies that G is non-singular, therefore Z_2 is non-singular. Since T_1 is also non-singular, $T_3^\star = -T_1Z_3^\star Z_2^{-1}$ and $T_1(Z_1 - Z_3^\star Z_2^{-1}Z_3) = 0$ imply $Z_1 = Z_3^\star Z_2^{-1}Z_3$. It follows that Z has rank n and so has M

because $Z \succeq M^\star M$. Equality (4) then readily follows. To show the converse implication, just choose $Z = M^\star M$ and $T = [I_n \quad -(G^{-1}F)^\star][I_n \quad -(G^{-1}F)^\star]^\star$. $\Box$

In the above optimization problem, the non-convexity is concentrated in the criterion only. The idea is then to convexify the criterion for example using the first order linearization

$$\begin{array}{rcll} T_{k+1}, Z_{k+1} & = & \arg \min & \operatorname{trace}(TZ_k + T_k Z) \\ & & \text{s.t.} & \text{LMI}(5). \end{array} \tag{6}$$

The above problem is a convex LMI optimization problem that can be addressed efficiently, see [6]. It is used in the following

---

CCL Algorithm:

Input: Polytope vertex matrices $\{A_1, A_2, \ldots, A_N\}$ and $\{B_1, B_2, \ldots, B_N\}$. Maximum number of iterations $k_{\max}$. Relative accuracy for rank evaluation ϵ. Relative accuracy for criterion variation μ.

Output: A robustly stabilizing state-feedback matrix K if the algorithm finds one. Otherwise, we cannot conclude about existence of such a matrix.

Step 1: Let $k = 0$. Find a feasible point T_0, Z_0, M for LMI problem (5). If there is no solution, stop. Problem (3-4) is not feasible and matrix K cannot be found.

Step 2: Solve the LMI problem (6) for T_{k+1}, Z_{k+1}, M.

Step 3: Let $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_{n+m}$ denote the singular values of matrix M. If $\sigma_{n+1} \leq \epsilon \sigma_n$ then a solution to problem (3-4) is found and the state-feedback matrix is $K = YG^{-1}$.

Step 4: Let $k = k + 1$. If $k > k_{\max}$, then matrix K was be found. Otherwise go to Step 2.

In [3] it is shown that the sequence

$$t_k = \operatorname{trace}(T_{k+1} Z_k + T_k Z_{k+1})$$

is bounded below by zero and decreases, thus the CCL algorithm always converges.

5 Numerical Experiments

5.1 Intensive Tests

We carried out intensive numerical experiments to assess the applicability of the CCL algorithm and to compare its performance with other robust stabilization algorithms. We generated a collection of randomly chosen polytopes $\mathcal{P}$ with vertices $\{A_1, A_2, \ldots, A_N\}$ and $\{B_1, B_2, \ldots, B_N\}$ for different values of vertex number N, state dimension n and input dimension m.

In discrete-time, the state matrix polytope vertices were generated as follows. For $i = 1, \ldots, N$: let A be an $n \times n$ matrix with normally distributed random entries (mean zero and variance one); let ρ be the spectral radius of A; if $\rho > 1$ let $A = A/\rho$.

In continuous-time, the state matrix polytope vertices were generated as follows. For $i = 1, \ldots, N$: let A be an $n \times n$ matrix with normally distributed random entries (mean zero and variance one); let ρ be the maximum real part of the eigenvalues of A; if $\rho > 0$ let $A = A - \rho I_n$; finally, let $A_i = A/\|A\|_2$.

In both continuous-time and discrete-time, the input matrix polytope vertices were generated as follows. For $i = 1, \ldots, N$: let B be an $n \times m$ matrix with normally distributed random entries (mean zero and variance once); let $B_i = B/\|B\|_2$.

In the CCL algorithm we set $\epsilon = 10^{-5}$ as the relative accuracy for rank evaluation, $k_{\max} = 20$ as the maximum number of iterations and $\mu = 10^{-4}$ as the minimum relative variation of the criterion.

To solve the LMI problems we have used the LMI Toolbox for Matlab [6] with relative accuracy 10^{-2} and maximum number of iterations 200.

To improve conditioning of matrices P (resp. P_i) in the LMI problems we systematically enforced the constraint $P \succ I$ (resp. $P_i \succ I$) instead of $P \succ 0$ (resp. $P_i \succ 0$). Note that this is done without loss of generality since the LMI conditions are homogeneous in the decision variables.

In discrete-time we compared the performance of the CCL algorithm with the standard quadratic stabilizability condition [5]

Lemma 3 *A sufficient condition for robust stabilization of matrix polytope $\mathcal{P}$ in discrete-time is that there exists matrices P and Q satisfying the LMI*

$$\begin{aligned} &\begin{bmatrix} -P & A_iP + B_iQ \\ (A_iP + B_iQ)^\star & -P \end{bmatrix} \prec 0 \\ &P = P^\star \succ 0. \end{aligned}$$

A robustly stabilizing state-feedback gain is then $K = QP^{-1}$.

We also compared the CCL algorithm with the less conservative discrete-time condition proposed in [7]

Lemma 4 *A sufficient condition for robust stabilization of matrix polytope $\mathcal{P}$ in discrete-time is that there exists matrices Y, G and P_i satisfying the LMI*

$$\begin{aligned} &\begin{bmatrix} P_i & A_iG + B_iY \\ (A_iG + B_iY)^\star & G + G' - P_i \end{bmatrix} \succ 0 \\ &P_i = P_i^\star \succ 0. \end{aligned}$$

A robustly stabilizing state-feedback gain is then $K = YG^{-1}$.

N	n	m	QS- RS- CCL+	QS- RS- CCL-	QS- RS- CCL>it	QS- RS+ CCL+	QS- RS+ CCl>it	QS+ RS+ CCL+	QS+ RS+ CCL>it	CCL success rate
3	2	1	3	13	14	36	1	33	0	**10**
3	3	1	15	9	20	38	1	17	0	**34**
3	3	2	2	0	15	34	3	46	0	**12**
3	4	1	18	16	24	36	0	6	0	**31**
3	4	2	4	1	15	56	1	23	1	**25**
4	2	1	8	32	28	19	0	13	0	**12**
4	3	1	10	36	36	16	0	12	0	**12**
4	3	2	8	4	32	46	1	9	0	**18**
4	4	1	11	32	39	18	0	0	0	**13**
4	4	2	7	4	35	50	2	2	0	**15**
5	2	1	2	39	31	21	1	6	0	**3**
5	3	1	8	44	41	5	0	2	0	**9**
5	3	2	7	10	47	34	1	1	0	**11**
5	4	1	6	52	37	5	0	0	0	**6**
5	4	2	11	19	54	16	0	0	0	**13**

Table 1: Discrete-time polytopes. Percentages of failure and success of different robust stabilization methods.

In continuous-time we compared the performance of the CCL algorithm with the continuous-time counterpart of lemma 3

Lemma 5 *A sufficient condition for robust stabilization of polytope $\mathcal{P}$ in continuous-time is that there exists matrices P and Q satisfying the LMI*

$$A_iP + B_iQ + (A_iP + B_iQ)^\star \prec 0$$

$$P = P^\star \succ 0.$$

A robustly stabilizing state-feedback gain is then $K = QP^{-1}$.

Note that no continuous-time counterpart of lemma 4 is available up to our knowledge.

In table 1 (resp. table 2) we report for different values of N, n and m the rate of success and failure of different robust stabilization methods in discrete-time (resp. continuous-time) for 100 random polytopes, according to the following legend

QS+ QS-	Quadratic Stabilizability LMI of Lemma 3 (resp. Lemma 5) is feasible Quadratic Stabilizability LMI of Lemma 3 (resp. Lemma 5) is not feasible
RS+ RS-	Robust Stabilizability LMI of Lemma 4 is feasible Robust Stabilizability LMI of Lemma 4 is not feasible
CCL+ CCL-	CCL Algorithm converged successfully CCL Algorithm BMI of Lemma 1 is not feasible.
CCL>it	CCL Algorithm did not converge

N	n	m	QS- CCL+	QS- CCL-	QS- CCL>it	QS+ CCL+	QS+ CCL>it	CCL success rate
3	2	1	6	34	3	57	0	**14**
3	3	1	16	27	9	47	1	**31**
3	3	2	0	2	3	95	0	**0**
3	4	1	15	27	13	45	0	**27**
3	4	2	4	4	3	87	2	**57**
4	2	1	4	54	11	31	0	**6**
4	3	1	9	67	8	16	0	**11**
4	3	2	8	13	12	67	0	**24**
4	4	1	15	56	10	19	0	**19**
4	4	2	17	12	11	60	0	**42**
5	2	1	4	70	2	24	0	**5**
5	3	1	4	84	6	6	0	**4**
5	3	2	5	34	13	48	0	**10**
5	4	1	10	82	5	3	0	**10**
5	4	2	13	49	9	29	0	**18**

Table 2: Continuous-time polytopes. Percentages of failure and success of different robust stabilization methods.

The first column from the right in the tables contains percentages of polytopes that were successfully robustly stabilized with the CCL algorithm but were not stabilized with other methods. It is meant to illustrate the relevance of the CCL algorithm over existing methods.

Note that in some rare instances the CCL algorithm does not converge whereas the QS or RS LMI is feasible (second and fourth columns from the right in table 1 and second column from the right in table 2). Usually, convergence is then ensured by allowing a greater number of iterations. We are currently investigating the use of scalings to improve the overall convergence of the CCL algorithm. See also paragraph 5.3 for more details about convergence problems of the CCL algorithm.

Our experience reveals that in most of the cases the CCL algorithm does not converge because the maximum number of iterations $k_{\max}$ is exceeded and not because the relative variation of the criterion is less than μ. See also paragraph 5.2 for more details about the number of iterations required by the CCL algorithm.

5.2 Number of Iterations

In the case of continuous-time polytopes with $N = 3$, $n = 3$, $m = 1$ we reported the number of iterations required by the CCL algorithm to converge when successful. The bar plot in figure 1 shows that most of the time the CCL algorithm converges in 3 or 4 iterations. Moreover, it is always necessary to perform at least 3 iterations to reach the required singular value ratio. Our experience reveals that this feature is typical of the CCL algorithm and does not depend on the number of vertices, states or inputs of the polytope.

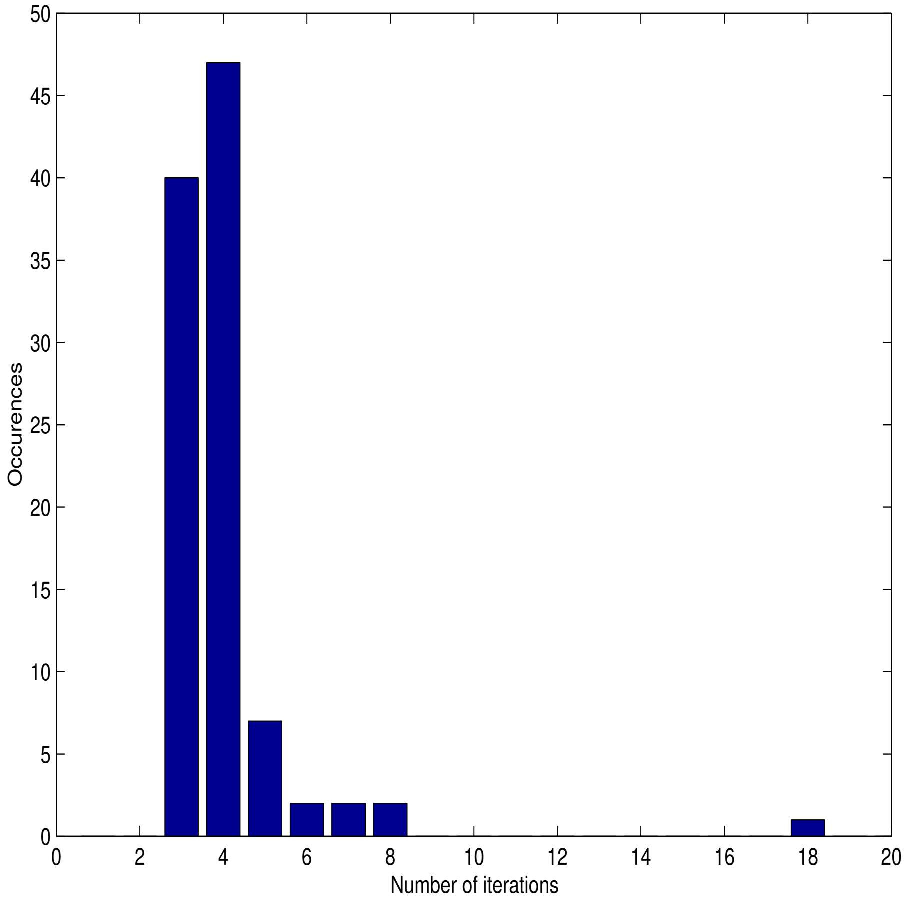

Figure 1: Number of iterations required by the CCL algorithm.

k	σ_{n+1}/σ_n	t_k	$(t_k - t_{k+1})/t_k$
0	0.29		
1	0.54	300	96.44%
2	0.31	2.2	99.27%
3	0.21	0.58	73.41%
4	0.12	0.26	55.85%
5	0.083	0.14	45.69%
...		...	
16	0.059	0.02	13.03%
17	0.0075	0.018	10.68%
18	0.052	0.016	10.08%
19	0.0074	0.014	13.23%
20	0.044	0.012	12.82%

Table 3: Non-converging CCL algorithm for the continuous-time polytope.

5.3 Convergence Problems

In this paragraph we provide a simple numerical example illustrating convergence problems that can be faced with the CCL algorithm. Consider the continuous-time polytope $\mathcal{P}$ with $N = 3$ vertices, $n = 3$ states and $m = 1$ inputs whose matrices are given by

$$A_1 = \begin{bmatrix} 0 & -1 & -3 \\ 0 & -2 & -1 \\ -2 & 0 & -1 \end{bmatrix} \quad A_2 = \begin{bmatrix} -2 & -1 & 0 \\ -1 & -1 & -1 \\ 1 & -1 & -3 \end{bmatrix} \quad A_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & -1 & -2 \\ -1 & -1 & -2 \end{bmatrix}$$

and

$$B_1 = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix} \quad B_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad B_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}.$$

The LMI of lemma 5 is feasible and the state-feedback matrix

$$K = \begin{bmatrix} -0.3287 & 0.0609 & 1.6616 \end{bmatrix}$$

is quadratically hence robustly stabilizing for the polytope. When applying the CCL algorithm, we obtain the non-converging sequence reported in table 3. The number of iterations exceeds $k_{\max} = 20$.

If we represent the singular value ratio σ_{n+1}/σ_n as a function of the number of iterations k, we observe a zigzagging phenomenon and a very slow rate of convergence, see figure 2. It is typical of first-order methods. It can be considered as the main limitation of the CCL algorithm.

Now applying the CCL algorithm to the scaled polytope matrices A_i/α and B_i/β with $\alpha = 6$ and $\beta = 3$, we obtain the convergence sequence reported in Table 4.

The scaled state-feedback matrix

$$K = \frac{6}{3} Y G^{-1} = \begin{bmatrix} -0.3016 & 0.0490 & 1.6426 \end{bmatrix}$$

is therefore robustly stabilizing.

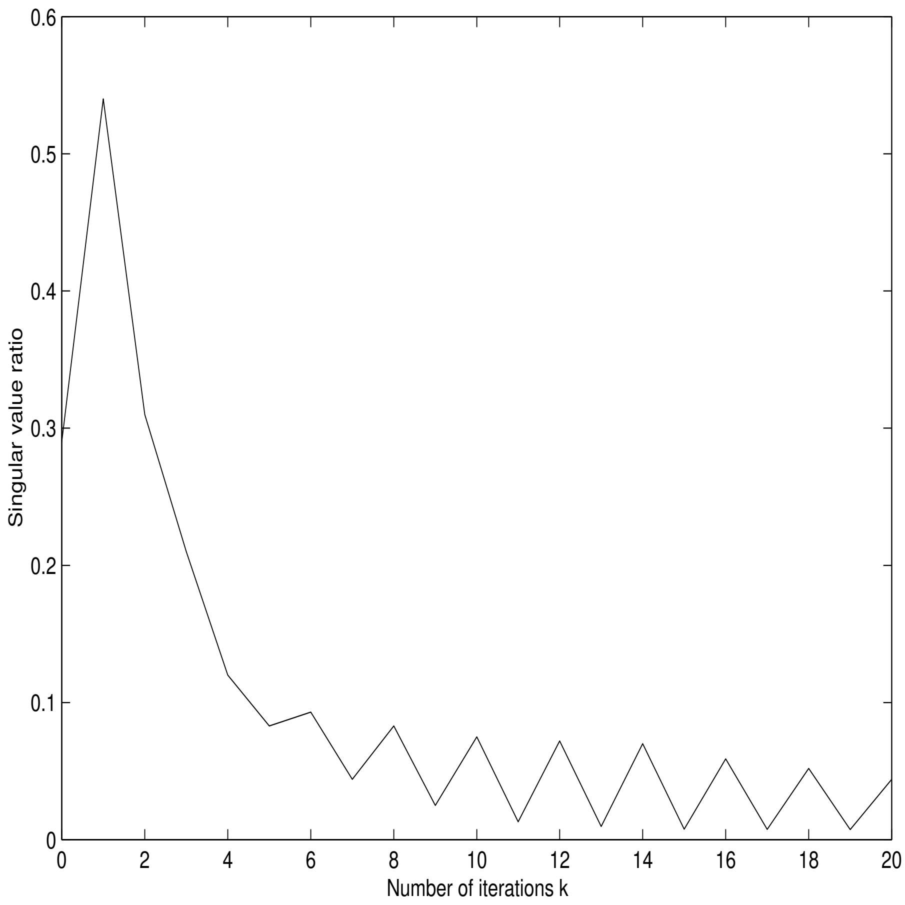

Figure 2: Zigzagging and slow convergence phenomena in the CCL algorithm.

k	σ_{n+1}/σ_n	t_k	$(t_k - t_{k+1})/t_k$
0	0.46		
1	0.91	990	97.00%
2	0.04	2.5	99.74%
3	0.067	0.24	90.70%
4	0.00012	0.029	87.92%
5	0.0017	0.00071	97.56%
6	10^{-7}	$4 \cdot 10^{-5}$	94.18%

Table 4: Converging CCL algorithm for the scaled continuous-time polytope.

6 Conclusions

The success rates in tables 1 and 2 show that the CCL algorithm may be a useful companion to already existing robust stabilization methods: it can provide a robustly stabilizing feedback when other methods fails. Unfortunately there is no guarantee that the CCL algorithm finds a robustly stabilizing feedback when one exists. Similarly, lack of success of the CCL algorithm does not mean automatically that no robustly stabilizing feedback exists. For these reasons and also because the computational cost of solving standard robust stabilization LMIs (Lemmas 3, 4 and 5) is the same as that of performing only one step of the iterative CCL algorithm, we recommend using the CCL algorithm only when standard robust stabilization LMIs fail.

As a first-order approximation to a non-linear programming problem, the CCL algorithm may suffer from slow convergence and zigzagging problems, as illustrated in paragraph 5.3. Scaling of the decision variables can help to overcome these problems, as shown in paragraph 5.3. Another line of attack could consist of using second-order information, as proposed in [4]. This point deserves further study.

Finally, we are now applying the CCL algorithm to more involved control problems such as output-feedback stabilization, mixed $\mathcal{H}_2$ or $\mathcal{H}_\infty$ analysis and design problems, and also robust analysis and design problems formulated in the algebra of polynomials and polynomial matrices.

Appendix

```
function [K,status,P,F,G,nit] = robustab(A,B,options)
%ROBUSTAB Robust state-feedback stabilization of polytopic system.
%
% The command
%
%  K = ROBUSTAB(A,B)
%
% attempts to find a state-feedback gain matrix K that robustly stabilizes
% the polytopic linear system whose state and input matrices vary in the
% convex hulls A = {A1,A2..,AN} and B = {B1,B2..,BN} respectively.
%
% If the macro fails, then K is empty. However, it does not necessarily
% mean that the polytopic linear system is not robustly stabilizable.
%
% The macro is based on an iterative cone complementarity linearization
% algorithm. At each step of the iterative algorithm, an LMI is solved
% with the LMI Control Toolbox.
%
% With the syntax
%
%  [K,STATUS,P] = ROBUSTAB(A,B,OPTIONS)
%
```

```
% an additional vector of control parameters can be specified:
%
%  OPTIONS(1): relative accuracy used by the LMI solver (Default=0.01)
%  OPTIONS(2): max. number of inner iterations used by LMI solver (Default=100)
%  OPTIONS(3): feasibility radius used by LMI solver (Default=1e12)
%  OPTIONS(4): max. number of outer iterations of the cone complementarity
%              linearization algorithm (Default=10)
%  OPTIONS(5): when nonzero, the trace of execution is turned off (Default=0)
%  OPTIONS(6): relative accuracy for rank evaluation (Default=1e-5)
%  OPTIONS(7): when zero, continuous-time polytopic system, otherwise
%              discrete-time polytopic system (Default=0)
%  OPTIONS(8): min. relative variation of the criterion (Default=1e-4)
%
% The output parameter STATUS contains information about convergence
% of the iterative process. When STATUS = 1 the process successfully
% converged and a robustly stabilizing feedback is returned. When STATUS
% is negative no robustly stabilizing feedback was found:
%
%  STATUS = -1: BMI not feasible (see OPTIONS(2) and OPTIONS(3)).
%  STATUS = -2: LMI problem not feasible (see OPTIONS(2) and OPTIONS(3)).
%  STATUS = -3: The maximum number of iterations was exceeded
%               (see OPTIONS(4)).
%  STATUS = -4: No significant relative variation of the criterion
%               (see OPTIONS(8)).
%
% Moreover, a cell array P = {P1,P2,..,PN} of positive definite matrices
% is returned such that the Lyapunov matrix varying in the convex hull of P
% corresponds to a polytopic parameter-dependent quadratic Lyapunov function
% proving robust stability of the polytopic system.

% Authors: D. Henrion, D. Arzelier, D. Peaucelle, March 29, 2000.
% Last modified by D. Henrion on October 12, 2000.

% ------------------------------------------------------------------------
% Parse input arguments
% ------------------------------------------------------------------------

if ~isa(A,'cell')
 A = {A};
elseif ~isa(B, 'cell')
 B = {B};
end;

default = [0.01 100 1e12 10 0 1e-5 0 1e-4 1e-8 0];

if nargin < 2,
 error('Incorrect number of input arguments.');
elseif nargin < 3,
 options = default;
```

```
else
 if length(options) < length(default),
  options(length(default)) = default(length(default));
 end;
 if options(1) <= 0, options(1) = default(1); end;
 if options(2) <= 0, options(2) = default(2); end;
 if options(3) <= 0, options(3) = default(3); end;
 if options(4) <= 0, options(4) = default(4); end;
 if options(6) <= 0, options(6) = default(6); end;
end;

N = length(A); % number of vertices
if length(B) ~= N,
 error('Inconsistent number of vertices in system matrix polytopes.');
end;

n = size(A{1},1); % state dimension
m = size(B{1},2); % input dimension

for i = 1:N,
 if any(size(A{i}) ~= [n n]),
  error(['Incorrect dimension of state matrix vertex #' int2str(i) '.']);
 elseif any(size(B{i}) ~= [n m]),
  error(['Incorrect dimension of input matrix vertex #' int2str(i) '.']);
 end;
end;

% Stability region

if options(7),
 H = [-1 0;0 1]; % discrete-time
else
 H = [0 1;1 0]; % continuous-time
end;

K = []; P = cell(0); F = []; G = [];
status = 0;

% --------------------------------------------------------------------------
% Build LMI
% --------------------------------------------------------------------------

setlmis([]);

varZ1 = lmivar(1,[n 1]);
varZ2 = lmivar(1,[n 1]);
varZ3 = lmivar(2,[n n]);
varT1 = lmivar(1,[n 1]);
varT2 = lmivar(1,[n 1]);
```

```
varT3 = lmivar(2,[n n]);
varF = lmivar(2,[n n]);
varG = lmivar(2,[n n]);
varX = lmivar(2,[m n]);
varY = lmivar(2,[m n]);
varP = cell(N,1);
for i = 1:N,
 varP{i} = lmivar(1,[n 1]);
end;

% Pi > 1
for i = 1:N,
 LMI = newlmi;
 lmiterm([-LMI 1 1 varP{i}], 1, 1);
 lmiterm([LMI 1 1 0], 1);
end;

% [Z1 Z3 X' F'; Z3' Z2 Y' G'; X Y I 0; F G 0 I] >= 0
LMI = newlmi;
lmiterm([-LMI 1 1 varZ1], 1, 1);
lmiterm([-LMI 2 2 varZ2], 1, 1);
lmiterm([-LMI 1 2 varZ3], 1, 1);
lmiterm([-LMI 3 1 varX], 1, 1);
lmiterm([-LMI 3 2 varY], 1, 1);
lmiterm([-LMI 4 1 varF], 1, 1);
lmiterm([-LMI 4 2 varG], 1, 1);
lmiterm([-LMI 3 3 0], 1);
lmiterm([-LMI 4 4 0], 1);

% [T1 T3;T3' T2] >= 0
LMI = newlmi;
lmiterm([-LMI 1 1 varT1], 1, 1);
lmiterm([-LMI 2 2 varT2], 1, 1);
lmiterm([-LMI 1 2 varT3], 1, 1);

% T1 >= 1
LMI = newlmi;
lmiterm([-LMI 1 1 varT1], 1, 1);
lmiterm([LMI 1 1 0], 1);

% [A{i}F+B{i}X+()'+H(1,1)*P{i} -F'+A{i}G+B{i}Y+H(1,2)*P{i};
%  ()' -G-G'+H(2,2)*P{i}] < 0
for i = 1:N,
  LMI = newlmi;
  lmiterm([LMI 1 1 varF], A{i}, 1, 's');
  lmiterm([LMI 1 1 varX], B{i}, 1, 's');
  lmiterm([LMI 1 2 -varF], -1, 1);
  lmiterm([LMI 1 2 varG], A{i}, 1);
  lmiterm([LMI 1 2 varY], B{i}, 1);
```

```
  lmiterm([LMI 2 2 varG], -1, 1, 's');
  if abs(H(1,1)) > 0,
   lmiterm([LMI 1 1 varP{i}], H(1,1), 1);
  end;
  if abs(H(2,1)) > 0,
   lmiterm([LMI 1 2 varP{i}], H(2,1), 1);
  end;
  if abs(H(2,2)) > 0,
   lmiterm([LMI 2 2 varP{i}], H(2,2), 1);
  end;
end;

LMISYS = getlmis;
nvar = decnbr(LMISYS);

x0 = cell(options(4)+3,1);

% ------------------------------------------------------------------
% Initial step
% ------------------------------------------------------------------

nit = 0;

if ~options(5),
 if options(7),
  disp('Discrete-time system.');
 else
  disp('Continuous-time system.');
 end;
 disp('-------------------------------');
 fprintf('N ITER\tRAT SV\tCR\tVAR CR\n');
 disp('-------------------------------');
 fprintf('%g\t', 0);
end;

lmiopt = [options(1) options(2) options(3) 0 1];
[tmin, xopt] = feasp(LMISYS, lmiopt);

if tmin > 0,
 if ~options(5),
  disp('BMI is not feasible.');
 disp('-------------------------------');
  disp('No robustly stabilizing feedback was found.');
 end;
 status = -1;
 return;
end;

Z1 = dec2mat(LMISYS,xopt,varZ1);
```

```
Z2 = dec2mat(LMISYS,xopt,varZ2);
Z3 = dec2mat(LMISYS,xopt,varZ3);
T1 = dec2mat(LMISYS,xopt,varT1);
T2 = dec2mat(LMISYS,xopt,varT2);
T3 = dec2mat(LMISYS,xopt,varT3);
Z = [Z1 Z3;Z3' Z2];
T = [T1 T3;T3' T2];
X = dec2mat(LMISYS,xopt,varX);
Y = dec2mat(LMISYS,xopt,varY);
F = dec2mat(LMISYS,xopt,varF);
G = dec2mat(LMISYS,xopt,varG);
M = [X Y;F G];
x0{1} = xopt;
x0{2} = xopt;

% ------------------------------------------------------------------
% Iterative process
% ------------------------------------------------------------------

S = svd(M);
ratsvM = S(n+1) / S(n);
copt = 2*trace(Z*T);
varcrit = 100;

if ~options(5),
 fprintf('%0.2g\n', ratsvM);
end;

% stop the iterative process as soon as:
% the ratio between the singular values in M is less than OPTIONS(6), or
% the number of iterations exceeds OPTIONS(4), or
% the relative variation of the criterion is less than OPTIONS(8)

while (ratsvM > options(6)) & (nit < options(4)) & (varcrit >= options(8)),

  nit = nit + 1;
  oldcopt = copt;

  % linearized criterion
  c = zeros(nvar, 1); k = 0;
  for i = 1:n, for j = 1:i,
   k = k + 1; if i == j, c(k) = T1(i,i); else c(k) = 2*T1(i,j); end;
  end; end;
  for i = 1:n, for j = 1:i,
   k = k + 1; if i == j, c(k) = T2(i,i); else c(k) = 2*T2(i,j); end;
  end; end;
  for i = 1:n, for j = 1:n,
   k = k + 1; c(k) = 2*T3(i,j);
  end; end;
```

```
for i = 1:n, for j = 1:i,
 k = k + 1; if i == j, c(k) = Z1(i,i); else c(k) = 2*Z1(i,j); end;
end; end;
for i = 1:n, for j = 1:i,
 k = k + 1; if i == j, c(k) = Z2(i,i); else c(k) = 2*Z2(i,j); end;
end; end;
for i = 1:n, for j = 1:n,
 k = k + 1; c(k) = 2*Z3(i,j);
end; end;

if ~options(5),
 fprintf('%g\t', nit);
end;

lmiopt = options(1:5); lmiopt(4:5) = [options(2)/10 1];
[copt, xopt] = mincx(LMISYS, c, lmiopt, x0{nit});

if isempty(copt),

 if ~options(5),
  disp('LMI problem becomes not feasible.');
  disp('No robustly stabilizing feedback was found.');
 end;
 status = -2;
 return;

else

 Z1 = dec2mat(LMISYS,xopt,varZ1);
 Z2 = dec2mat(LMISYS,xopt,varZ2);
 Z3 = dec2mat(LMISYS,xopt,varZ3);
 T1 = dec2mat(LMISYS,xopt,varT1);
 T2 = dec2mat(LMISYS,xopt,varT2);
 T3 = dec2mat(LMISYS,xopt,varT3);
 Z = [Z1 Z3;Z3' Z2];
 T = [T1 T3;T3' T2];
 X = dec2mat(LMISYS,xopt,varX);
 Y = dec2mat(LMISYS,xopt,varY);
 F = dec2mat(LMISYS,xopt,varF);
 G = dec2mat(LMISYS,xopt,varG);
 M = [X Y;F G];
 x0{nit+2} = xopt;

 S = svd(M); ratsvM = S(n+1) / S(n);
 varcrit = 100*(oldcopt-copt)/oldcopt;
 tr = trace(Z*T);

 if ~options(5),
  fprintf('%0.2g\t%0.2g\t%.2f%%\n', ratsvM, copt, varcrit);
```

```
   end;

  end;

end;

if ~options(5),
 disp('-------------------------------');
end;

% -----------------------------------------------------------
% End of iterative process
% -----------------------------------------------------------

if nit == options(4),

 if ~options(5),
  disp('The maximum number of iterations was exceeded.');
  disp('No robustly stabilizing feedback was found.');
 end;
 status = -3;

elseif varcrit < options(8),

 if ~options(5),
  disp('No significant relative variation of the criterion.');
  disp('No robustly stabilizing feedback was found.');
 end;
 status = -4;

else

 if ~options(5),
  disp('The required ratio on singular values was reached.');
  disp('The iterative process successfully converged.');
 end;

 K = Y/G;
 for i = 1:N,
  P{i} = dec2mat(LMISYS,xopt,varP{i});
 end;
 status = 1;

end;

if status < 0,
 K = []; F = []; G = [];
end;
```

```
% END OF MACRO ROBUSTAB.M
```

References

[1] D. Arzelier, D. Henrion, D. Peaucelle "Robust $\mathcal{D}$-stabilization of a Polytope of Matrices", LAAS-CNRS Research Report, No. 2000295, June 2000. Submitted for publication.

[2] V. D. Blondel and J. N. Tsitsiklis "A Survey of Computational Complexity Results in Systems and Control", *Automatica*, Vol. 36, No. 9, pp. 1249–1274, 2000.

[3] L. El Ghaoui, F. Oustry and M. Ait Rami "A Cone Complementarity Linearization Algorithm for Static Output Feedback and Related Problems", *IEEE Transactions on Automatic Control*, Vol. 42, No. 8, pp. 1171–1176, 1997.

[4] B. Fares, P. Apkarian, D. Noll "An Augmented Lagrangian Method for a Class of LMI-Constrained Problems in Robust Control Theory", *Proceedings of the American Control Conference*, pp. 3702–3706, Chicago, Illinois, 2000.

[5] J. C. Geromel, P. L. D. Peres and J. Bernussou "On a Convex Parameter Space Method for Linear Control Design of Uncertain Systems", *SIAM Journal on Control and Optimization*, Vol. 29, pp. 381–402, 1991.

[6] The MathWorks, Inc. "LMI Control Toolbox for Matlab", Release 1.0.5, 1998. See the home page `www.mathworks.com`.

[7] M. C. de Oliveira, J. Bernussou and J. C. Geromel, "A New Discrete-time Robust Stability Condition", *Systems and Control Letters*, Vol. 37, No. 4, pp. 261–265, 1999.

[8] D. Peaucelle, D. Arzelier, O. Bachelier and J. Bernussou "A New Robust $\mathcal{D}$-stability Condition for Real Convex Polytopic Uncertainty", *Systems and Control Letters*, Vol. 40, No. 1, pp. 21–30, 2000.