Available online at www.sciencedirect.com
ScienceDirect
Automatica 43 (2007) 2022–2033
automatica
www.elsevier.com/locate/automatica

A probabilistic analytic center cutting plane method for feasibility of uncertain LMIs☆

Giuseppe C. Calafiore[a,*], Fabrizio Dabbene[b]

[a]*Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy*
[b]*IEIIT-CNR, Politecnico di Torino, Italy*

Received 1 March 2006; received in revised form 4 October 2006; accepted 2 April 2007
Available online 13 August 2007

Abstract

Many robust control problems can be formulated in abstract form as convex feasibility programs, where one seeks a solution x that satisfies a set of inequalities of the form $\mathscr{F} \doteq \{f(x,\delta) \leqslant 0, \delta \in \mathscr{D}\}$. This set typically contains an infinite and uncountable number of inequalities, and it has been proved that the related *robust feasibility* problem is numerically hard to solve in general.

In this paper, we discuss a family of cutting plane methods that solve efficiently a probabilistically relaxed version of the problem. Specifically, under suitable hypotheses, we show that an Analytic Center Cutting Plane scheme based on a *probabilistic oracle* returns in a finite and pre-specified number of iterations a solution x which is feasible for most of the members of $\mathscr{F}$, except possibly for a subset having arbitrarily small probability measure.

Keywords: Randomized algorithms; Uncertain linear matrix inequalities; Robust control

1. Introduction

Recently, there has been a widespread interest in the literature towards probabilistic methods for solving robust control problems that are hard to attack using deterministic techniques. An overview of this research area, along with many pointers to the related literature is available for instance in the books (Calafiore & Dabbene, 2006; Tempo, Calafiore, & Dabbene, 2004).

The focus of this paper is on analysis and design problems that can be cast in the format of a feasibility problem over a possibly infinite intersection of linear matrix inequalities (LMIs), see Section 2 for a precise statement. Such robust feasibility problems are known to be computationally hard in general (Ben-Tal & Nemirovski, 1998; El Ghaoui, Oustry, & Lebret, 1998). Deterministic methods based on LMI relaxations exist that may either solve exactly the problem—when the dependence of the LMI parameters on the uncertainty is of simple affine form, see Ben-Tal and Nemirovski (2002)—or produce a family of LMI relaxations having decreasing conservatism—when the uncertainty dependence is polynomial (Bliman, 2004) or rational (Scherer, 2005). However, no deterministic method is known that can solve exactly the problem under generic uncertainty dependence.

To fix ideas, consider a paradigmatic problem of quadratic stability. Let $A(\delta) \in \mathbb{R}^{n,n}$ be a generic function of a vector of uncertain parameters $\delta \in \mathscr{D} \subseteq \mathbb{R}^{\ell}$. The set of matrices $\{A(\delta),\ \delta \in \mathscr{D}\}$ is said to be quadratically stable if there exist a matrix $P \succ 0$ such that

$$A^{\mathrm{T}}(\delta)P + PA(\delta) \prec 0, \quad \forall \delta \in \mathscr{D}, \tag{1}$$

where the notation $X \succ 0$ (resp. $X \prec 0$) indicates that X is a symmetric positive (resp. negative) definite matrix. In this general setting, no algorithm is known that solves the problem in an exact and efficient way. There are, however, efficient *probabilistic* algorithms that are guaranteed with high probability

☆ This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Minyue Fu under the direction of Editor Roberto Tempo. This work was supported by MIUR under the FIRB project "Learning, randomization and guaranteed predictive inference for complex uncertain systems."

* Corresponding author. Tel.: +39 011 5647071; fax: +39 011 5647099.
E-mail addresses: giuseppe.calafiore@polito.it (G.C. Calafiore), fabrizio.dabbene@polito.it (F. Dabbene).

0005-1098/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2007.04.003

to return a solution $P \succ 0$ that may fail to satisfy the matrix inequalities (1) at most on a subset of $\mathscr{D}$ having arbitrarily small probability measure, see for instance Calafiore and Campi (2005), Calafiore and Polyak (2001), Kanev, De Schutter, and Verhaegen (2003), Oishi (2003) and Oishi and Kimura (2003).

The first of such algorithms has been proposed in Calafiore and Polyak (2001) for general uncertain LMIs, and in Polyak and Tempo (2001) in the context of robust linear quadratic control. These algorithms were based on an iterative scheme in which the current solution is updated towards a descent direction obtained by a random gradient of a suitable feasibility violation function. Later, improved algorithms for probabilistic feasibility, based on the use of the ellipsoid method (Nemirovski & Yudin, 1983; Shor, 1970), have been developed in Kanev et al. (2003) and further analyzed in Oishi (2003).

The two mentioned classes of probabilistic algorithms have iterative nature and are suited for the solution of robust convex feasibility problems. Also along the probabilistic approach, a different method called the *scenario optimization method* has been recently proposed in Calafiore and Campi (2005, 2006). This method, which is mainly suited for optimization, works by solving in one-shot a convex optimization problem subject to a finite number of LMI constraints, sampled at random from the original infinite constraint set.

The contribution of this paper is on the lines of the iterative probabilistic methods of Calafiore and Polyak (2001), Kanev et al. (2003), Oishi (2003), Oishi and Kimura (2003) and Tempo et al. (2004). Its main purpose is to introduce a suitable probabilistic version of an analytic center cutting plane (ACCP) method, that can be employed in order to compute efficiently a probabilistically feasible solution for an infinite set of linear matrix inequalities.

Cutting plane methods (Goffin & Vial, 2002; Mitchell, 2003; Péton, 2002) belong to the family of *localization methods* that, at each iteration, keep an updated set that is guaranteed to contain the original feasible set of the problem. In this sense, the ellipsoid algorithm (EA) (Khachiyan, 1979; Lovász, Grötschel, & Schrijver, 1993; Nemirovski & Yudin, 1983) can be interpreted as a particular cutting plane method that uses ellipsoids to inscribe the feasible set. The EA is important for its ease of implementation and has a better theoretical worst-case complexity than the algorithm presented in this paper. However, it is well known that in practice its numerical performance is poor (it always attains its worst-case complexity, Lovász et al., 1993; Nemirovski & Yudin, 1983), and that ACCP schemes provide better practical computational performances. Moreover, the EA may suffer from numerical problems related to the degradation of the condition number of the ellipsoid shape matrix with the number of iterations.

This paper is organized as follows. In Section 2 we formally introduce the problem under study. Section 3 describes the general probabilistic ACCP (P-ACCP) scheme and states a first result on the functioning of the probabilistic oracle, which constitutes the inner part of the algorithm. Section 4 discusses the outer centering and update phase of the algorithm, while Section 5 contains the overall probabilistic convergence results (one lengthy proof is relegated to the Appendix). A comparative example is presented in Section 6, and conclusions are finally drawn in Section 7.

2. Problem statement

Consider an uncertain LMI of the form

$$F(x,\delta) \preccurlyeq 0, \tag{2}$$

where $F(x,\delta) = F_0(\delta) + \sum_{i=1}^{n} x_i F_i(\delta)$ and where $F_i(\delta)$, $i = 0,\ldots,n$ are symmetric $m \times m$ real matrices, that depend in a generic and possibly nonlinear way on the uncertainty vector $\delta \in \mathscr{D}$, being $\mathscr{D}$ a subset of $\mathbb{R}^\ell$. For a given $\delta \in \mathscr{D}$, a point $x \in \mathbb{R}^n$ is feasible for (2) if and only if $\lambda_{\max}(F(x,\delta)) \leqslant 0$, where $\lambda_{\max}(X)$ denotes the largest eigenvalue of a symmetric matrix X. Let us define

$$f(x,\delta) \doteq \lambda_{\max}(F(x,\delta)) \tag{3}$$

and the collection of inequalities $\mathscr{F} \doteq \{f(x,\delta) \leqslant 0,\ \delta \in \mathscr{D}\}$. Notice that $f(x,\delta)$ is a convex function in x, for any fixed δ. This follows immediately from the variational characterization of the largest eigenvalue

$$\begin{aligned}\lambda_{\max}(F(x,\delta)) &= \sup_{\|\xi\|=1} \xi^{\mathrm{T}} F(x,\delta)\xi \\ &= \sup_{\|\xi\|=1} \xi^{\mathrm{T}} F_0(\delta)\xi + \sum_{i=1}^{n} x_i \xi^{\mathrm{T}} F_i(\delta)\xi \end{aligned} \tag{4}$$

and recalling that the supremum of affine functions is convex. For fixed $\delta \in \mathscr{D}$, define the set

$$\mathscr{X}_\delta \doteq \{x :\ f(x,\delta) \leqslant 0\}. \tag{5}$$

A point $x \in \mathbb{R}^n$ is said to be *robustly feasible* for $\mathscr{F}$ if it satisfies all members of $\mathscr{F}$, i.e. if it belongs to the convex set

$$\mathscr{X} \doteq \{x :\ f(x,\delta) \leqslant 0,\ \forall \delta \in \mathscr{D}\} = \bigcap_{\delta\in\mathscr{D}} \mathscr{X}_\delta. \tag{6}$$

Robust feasible solutions are in general hard to determine (Ben-Tal & Nemirovski, 1998; El Ghaoui et al., 1998), and we shall not insist here on their computation.

Let instead a probability measure "Prob" be defined over $\mathscr{D}$. Our goal is to devise an algorithm that, with high probability, returns a candidate solution x with the property that

$$V(x) \doteq \mathrm{Prob}\{f(x,\delta) > 0\} \leqslant \varepsilon, \tag{7}$$

where $\varepsilon \in (0,1)$ is an a priori specified small number. Such a solution is called *probabilistically robust*, since it satisfies "almost all" the inequalities in $\mathscr{F}$. In other words, the event that the inequality $f(x,\delta) \leqslant 0$ is violated has probability smaller than the specified ε. The quantity $V(x)$ is the *violation probability* of a candidate solution x.

We show in the following sections that a probabilistically feasible solution may be found in a numerical efficient way by using an ACCP method based on a probabilistic oracle.

3. An ACCP scheme with probabilistic oracle

Standard cutting plane methods for determining a feasible point in a convex set $\mathscr{X}$ are based on the availability of a *separation oracle* for the set $\mathscr{X}$, and work by iteratively shrinking a localization set that is guaranteed to contain $\mathscr{X}$, see Goffin and Vial (2002) and Péton (2002) and references therein.

Specifically, suppose that a localization set $\mathscr{L}_k \supseteq \mathscr{X}$ is given at iteration k of the procedure. Suppose that an oracle is available, such that when we query the oracle with a candidate point $x_k \in \mathscr{L}_k$, the oracle either returns a "yes" answer, meaning that $x_k \in \mathscr{X}$, or it returns a "no" answer (meaning that $x_k \notin \mathscr{X}$), along with a *separating hyperplane* $\{\xi \in \mathbb{R}^n : a_k^{\mathrm{T}}\xi = b_k\}$ having the property that $a_k^{\mathrm{T}} x_k \geqslant b_k$ and $a_k^{\mathrm{T}} x \leqslant b_k$, for all $x \in \mathscr{X}$.

When the oracle returns a "yes" answer, we are done, since we found a feasible point. When the oracle returns a "no" answer, we have a separating hyperplane which indicates that the half-space $\{x : a_k^{\mathrm{T}} x > b_k\}$ cannot contain a feasible point and can therefore be eliminated (cut) from our search. In this case, we know that $\mathscr{X} \subseteq \mathscr{L}_k \cap \mathscr{H}_k$, where $\mathscr{H}_k \doteq \{x : a_k^{\mathrm{T}} x \leqslant b_k\}$, and the algorithm constructs an updated localization set $\mathscr{L}_{k+1}$ such that $\mathscr{L}_{k+1} \supseteq \mathscr{L}_k \cap \mathscr{H}_k$. A new query point $x_{k+1} \in \mathscr{L}_{k+1}$ is then determined, and the process is repeated.

Remark 1. Notice that, if an oracle was available for the set $\mathscr{X}$ in (6), then a standard cutting plane method could be employed to determine a robustly feasible solution. However, it is in general not possible to construct such an oracle, since checking whether $x \in \mathscr{X}$ requires checking an infinite (and usually uncountable) number of inequalities. The key point of this paper is to use a *probabilistic oracle* which returns either a "no" answer, meaning that the query point is unfeasible, or a "yes" answer, meaning that, with a probability that can be made arbitrarily close to one, the current query point is probabilistically feasible, that is $V(x_k) \leqslant \varepsilon$. Such an oracle is discussed in Section 3.1.

The probabilistic algorithm that we propose in this paper is based on an ACCP method that uses a probabilistic oracle, and is denoted as P-ACCP. In this method, the localization set $\mathscr{L}_k$ is the polytope given by the intersection of an initial localization set $\mathscr{L}_1$, that is assumed to contain $\mathscr{X}$, and the half-spaces described by all the cuts returned by the oracle up to iteration k: $\mathscr{L}_k = \mathscr{L}_1 \cap \{\bigcap_{i=1}^{k-1} \mathscr{H}_i\}$. In particular, we assume that $\mathscr{L}_1$ is an hypercube described by

$$\mathscr{L}_1 \doteq \{x : e_i^{\mathrm{T}} x \leqslant R + e_i^{\mathrm{T}} x_0,\ -e_i^{\mathrm{T}} x \leqslant R - e_i^{\mathrm{T}} x_0,\ i = 1, \ldots, n\}, \tag{8}$$

where e_i is the ith vector in the standard basis of $\mathbb{R}^n$, and $x_0 \in \mathbb{R}^n$, $R > 0$ are the hypercube center and radius, respectively.

In the P-ACCP method, the query point x_k is computed as the *analytic center* of $\mathscr{L}_k$, which is defined as the unique minimizer of the logarithmic barrier function (see e.g. Boyd & Vandenberghe, 2004)

$$\Phi_k(x) \doteq -\sum_{i=1}^{k-1} \ln(b_i - a_i^{\mathrm{T}} x) - \sum_{i=1}^{n} \ln(R - e_i^{\mathrm{T}}(x - x_0)) - \sum_{i=1}^{n} \ln(R + e_i^{\mathrm{T}}(x - x_0)). \tag{9}$$

We distinguish two phases in the P-ACCP method: an outer phase where the localization set is updated and the new query point is computed, and an inner phase that consists in the operations performed by the probabilistic oracle. To analyze the numerical complexity of such a scheme, we shall take into account three different terms: (a) the number of oracle calls required to obtain a solution, (b) the numerical complexity of computing the query point, and (c) the operations performed by the probabilistic oracle, which we quantify as the maximum number of feasibility checks. We analyze the two phases of the P-ACCP method and each of the complexity terms in the next sections, under the following standard assumption.

Assumption 1. The target set $\mathscr{X}$ is contained in an hypercube $\mathscr{L}_1$.

Remark 2. Notice that the previous assumption requires the set $\mathscr{X}$ to be bounded. In practical problems, if the original set is unbounded, one can include additional constraints on the problem variables (such as limits on the range of variation) so to enforce boundedness.

To determine a bounding hypercube for $\mathscr{X}$ notice first that by definition (6) for any fixed $\delta \in \mathscr{D}$ it holds that $\mathscr{X} \subseteq \mathscr{X}_\delta$. Therefore, it is sufficient to determine an hypercube including $\mathscr{X}_\delta$ (for one given choice of $\delta \in \mathscr{D}$) in order to have a bounding hypercube for $\mathscr{X}$.

To actually find an hypercube containing $\mathscr{X}_\delta$ one may proceed in at least two ways. A first possibility is to compute an ellipsoid containing $\mathscr{X}_\delta$, using the technique proposed in Boyd and El Ghaoui (1993) and also used in Kanev and Verhaegen (2006), Section 8.4, and then determine the minimal hypercube inscribing it. Alternatively, and perhaps with a greater computational burden, one may proceed as proposed in Kanev et al. (2003), and solve a series of $2n$ semidefinite programs

$$\rho_i^- = \min_x e_i^{\mathrm{T}} x \quad \text{s.t. } F(x, \delta) \preccurlyeq 0, \quad i = 1, \ldots, n,$$

$$\rho_i^+ = \max_x e_i^{\mathrm{T}} x \quad \text{s.t. } F(x, \delta) \preccurlyeq 0, \quad i = 1, \ldots, n$$

and then set $x_0 = \frac{1}{2}[(\rho_1^+ + \rho_1^-) \cdots (\rho_n^+ + \rho_n^-)]^{\mathrm{T}}$, $R = \frac{1}{2}\max_{i=1,\ldots,n}(\rho_i^+ - \rho_i^-)$.

It is worth to notice at this point that the initial bounding set does not need to be a hypercube, but it could as well be an hyperrectangle, or a polytope in general. The choice of an hypercube is made with the only purpose of simplifying the convergence proof given in the Appendix.

3.1. Inner iterations: the probabilistic oracle

In this section, we describe a probabilistic oracle that performs a randomized check of feasibility of a query point x_k. When x_k is found unfeasible, we show how to determine a hyperplane that separates $\mathscr{X}$ from x_k. We start by stating the following lemma.

Lemma 1. *Let x_k, δ be such that $f(x_k, \delta) > 0$ and let $g_\delta(x_k)$ be a subgradient of $f(x, \delta)$ at point x_k. Then,*

(1) *The hyperplane $\{x : a_k^{\mathrm{T}} x = b_k\}$, with*

$$a_k = g_\delta(x_k), \quad b_k = g_\delta^{\mathrm{T}}(x_k)x_k - f(x_k, \delta) \tag{10}$$

defines a deep cutting plane between $\mathscr{X}$ and x_k, i.e. $\mathscr{X} \subseteq \{x : a_k^{\mathrm{T}} x \leqslant b_k\}$, and $a_k^{\mathrm{T}} x_k > b_k$.

(2) *The hyperplane $\{x : a_k^{\mathrm{T}} x = b_k\}$, with*

$$a_k = g_\delta(x_k), \quad b_k = g_\delta^{\mathrm{T}}(x_k)x_k \tag{11}$$

defines a neutral cutting plane between $\mathscr{X}$ and x_k, i.e. $\mathscr{X} \subseteq \{x : a_k^{\mathrm{T}} x \leqslant b_k\}$, and $a_k^{\mathrm{T}} x_k = b_k$.

Proof. By convexity of $f(x, \delta)$ and the definition of subgradient, it holds that, for all x, $f(x, \delta) \geqslant f(x_k, \delta) + g_\delta^{\mathrm{T}}(x_k)(x - x_k)$. Hence, for all points in the half-space $\overline{\mathscr{H}}_k \doteq \{x : a_k^{\mathrm{T}} x > b_k\}$ with a_k, b_k given in (10) (i.e. the half-space $\{x : f(x_k, \delta) + g_\delta^{\mathrm{T}}(x_k)(x - x_k) > 0\}$), we have that $f(x, \delta) > 0$. Therefore, the set $\mathscr{X}_\delta$ defined in (5) belongs to the complementary half-space $\mathscr{H}_k = \{x : a_k^{\mathrm{T}} x \leqslant b_k\}$. Since $\mathscr{X} = \bigcap_{\delta \in \mathscr{D}} \mathscr{X}_\delta$, this implies that $\mathscr{X} \subseteq \mathscr{X}_\delta \subset \mathscr{H}_k$, which means that the considered hyperplane separates $\mathscr{X}$ from x_k. Moreover, since $f(x_k, \delta) > 0$, it follows that $a_k^{\mathrm{T}} x_k - b_k = f(x_k, \delta) > 0$ which proves that the cut is deep. The second point of the lemma follows from a similar reasoning. □

3.1.1. Computation of subgradients

Although the function $f(x, \delta)$ defined in (3) is non-differentiable whenever the maximum eigenvalue of $F(x, \delta)$ has multiplicity greater than one, we can easily compute a subgradient for $f(x, \delta)$, using the characterization in (4) as follows. Let $\varphi_\xi(x, \delta) \doteq \xi^{\mathrm{T}} F(x, \delta)\xi = \xi^{\mathrm{T}} F_0(\delta)\xi + \sum_{i=1}^n x_i \xi^{\mathrm{T}} F_i(\delta)\xi$ (this is an affine, hence convex and differentiable, function of x), and let the gradient of $\varphi_\xi(x, \delta)$ be $\nabla_x \varphi_\xi(x, \delta) = [\xi^{\mathrm{T}} F_1(\delta)\xi \ \cdots \ \xi^{\mathrm{T}} F_n(\delta)\xi]^{\mathrm{T}}$. Then, for all unit-norm ξ,

$$f(x, \delta) \geqslant \varphi_\xi(x, \delta) = \varphi_\xi(x_k, \delta) + \nabla_x^{\mathrm{T}} \varphi_\xi(x, \delta)(x - x_k). \tag{12}$$

Now let $\xi_{\max}$ be a vector such that $\sup_{\|\xi\|=1} \varphi_\xi(x_k, \delta)$ is attained. In the case under consideration, $\xi_{\max}$ is a unit-norm eigenvector of $F(x_k, \delta)$ associated with its largest eigenvalue, hence $\varphi_{\xi_{\max}}(x_k, \delta) = \xi_{\max}^{\mathrm{T}} F(x_k, \delta)\xi_{\max} = \xi_{\max}^{\mathrm{T}} \lambda_{\max}(F(x_k, \delta))\xi_{\max} = \lambda_{\max}(F(x_k, \delta)) = f(x_k, \delta)$. Then, considering (12) for $\xi = \xi_{\max}$ we have $f(x, \delta) \geqslant f(x_k, \delta) + \nabla_x^{\mathrm{T}} \varphi_{\xi_{\max}}(x, \delta)(x - x_k)$, which shows that $\nabla_x \varphi_{\xi_{\max}}$ is a subgradient of $f(x, \delta)$ at the point x_k. This reasoning leads us to the following lemma.

Lemma 2. *A subgradient of $f(x, \delta) = \lambda_{\max}(F(x, \delta))$ in $x = x_k$ is given by*

$$g_\delta(x_k) = [\xi_{\max}^{\mathrm{T}} F_1(\delta)\xi_{\max} \ \cdots \ \xi_{\max}^{\mathrm{T}} F_n(\delta)\xi_{\max}]^{\mathrm{T}},$$

where $\xi_{\max}$ is a unit-norm eigenvector associated with the largest eigenvalue of $F(x_k, \delta)$.

3.1.2. Probabilistic oracle

The probabilistic oracle is based on a randomized check of satisfaction of the inequality $f(x_k, \delta) \leqslant 0$ on a finite number of values of δ chosen at random according to the probability measure Prob.

Let x_k be the query point at the kth outer iteration, let $N(k)$ be an integer (to be specified later), and let $\delta_k^{(1)}, \ldots, \delta_k^{(N(k))}$ be a finite sequence of independent and identically distributed random samples extracted according to Prob. A probabilistic oracle is constructed as follows (to simplify notation, we denote with $g_i(x_k)$ a subgradient of $f(x_k, \delta_k^{(i)})$):

```
[feas, a_k, b_k] = p-oracle(x_k, N(k))
set i = 0, feas = TRUE
1.    while feas = = TRUE & i<N(k)
2.        set i = i+1
3.        if f(x_k, δ_k^(i)) > 0
4.            set feas = FALSE
5.    return a_k = g_i(x_k); b_k = g_i^T(x_k) x_k
6.        end if
7.    end while
```

A call to the probabilistic oracle may have two possible outcomes. If the `while` loop is interrupted at some $i < N(k)$, then the query point x_k is unfeasible: a separating hyperplane is obtained by means of a neutral cut, and control is returned to the outer phase of the cutting plane scheme, see the overall algorithm in Section 4. Otherwise, if the `while` loop is run up to $i = N(k)$, the query point x_k is declared "feasible" (in a probabilistic sense), and x_k is returned as a solution.

Remark 3. Notice that line 5 of the oracle could be modified so to return a deep cut, instead of a neutral cut. We choose to consider neutral cuts in the algorithm, since this choice greatly simplifies the theoretical complexity analysis developed in the Appendix. Moreover, the fast initialization technique discussed in Section 4.1.1 cannot be used with deep cuts. However, since deep cuts "cut off" a larger portion of the localization set at each outer iteration, this set shrinks faster and hence algorithm performance may be significantly improved in practice if deep cuts are used. For an in-depth analysis of ACCP schemes with deep cuts we refer the reader to Goffin and Vial (1999, 2002).

The probabilistic properties of the oracle are described in the following theorem.

Theorem 1 (*Success of the probabilistic oracle*). *Let $\varepsilon \in (0, 1)$ be a given (small) probability level, and let x_k be the query point at outer iteration k of the P-ACCP algorithm. Then, with probability greater than $1 - (1 - \varepsilon)^{N(k)}$ either the current solution is found unfeasible by the probabilistic oracle (and*

hence it has to be updated), *or it holds that* $V(x_k) \leqslant \varepsilon$, *where* $V(\cdot)$ *is the constraint violation probability defined in* (7).

Proof. Notice first that x_k is a random variable that depends on the successive independent random extractions of the sequences $S_j = \{\delta_j^{(1)}, \ldots, \delta_j^{(N(k))}\}$; $j = 1, \ldots$ The probability measure on x_k is therefore defined on this product space, and it is denoted as Prob_∞. Define the following three events:

$$\text{Feas}_k \doteq \{\text{the oracle declares } x_k \text{ "feasible"}\}, \tag{13}$$

$$\text{Bad}_k \doteq \{V(x_k) > \varepsilon\}, \tag{14}$$

$$\text{Iter}_k \doteq \{\text{the } k\text{th outer iteration is reached}\}. \tag{15}$$

Following a reasoning similar to the one proposed in Oishi (2003, 2007), we evaluate the probability that, given that the algorithm reached the kth outer iteration, it exits at this iteration with a "bad" solution. We have

$$\begin{aligned}&\text{Prob}_\infty\{\text{Feas}_k \cap \text{Bad}_k | \text{Iter}_k\}\\ &\quad = \text{Prob}_\infty\{\text{Feas}_k | \text{Bad}_k \cap \text{Iter}_k\}\text{Prob}_\infty\{\text{Bad}_k | \text{Iter}_k\}\\ &\quad \leqslant \text{Prob}_\infty\{\text{Feas}_k | \text{Bad}_k \cap \text{Iter}_k\} < (1-\varepsilon)^{N(k)}.\end{aligned} \tag{16}$$

The last inequality is due to the fact that $V(x_k) = 1 - \text{Prob}\{f(x_k, \delta) \leqslant 0\}$ and that Bad_k is the event $\text{Bad}_k = \{\text{Prob}\{f(x_k, \delta) \leqslant 0\} < 1 - \varepsilon\}$. Therefore, conditioned on the event $\text{Bad}_k \cap \text{Iter}_k$, the probability that the oracle declares x_k feasible is the probability of $N(k)$ successes in $N(k)$ independent trials, each having probability of success smaller than $1 - \varepsilon$, which is indeed smaller than $(1-\varepsilon)^{N(k)}$.

Considering the complementary events $\text{Update}_k \doteq \overline{\text{Feas}_k} = \{\text{the oracle declares } x_k \text{ unfeasible}\}$ and $\text{Good}_k \doteq \overline{\text{Bad}_k} = \{V(x_k) \leqslant \varepsilon\}$, we have that

$$\begin{aligned}&\text{Prob}_\infty\{\text{Update}_k \cup \text{Good}_k | \text{Iter}_k\}\\ &\quad = 1 - \text{Prob}_\infty\{\text{Feas}_k \cap \text{Bad}_k | \text{Iter}_k\} > 1 - (1-\varepsilon)^{N(k)},\end{aligned} \tag{17}$$

which proves the statement. □

Remark 4. Notice that by appropriate choice of the inner iterations limit $N(k)$, we can make the success probability of the oracle as close as desired to one. This can be easily seen from Eq. (17). Indeed, to achieve a desired success probability of at least $1 - \beta$, where $\beta \in (0, 1)$ is a small number, from (17) we impose that $1 - (1-\varepsilon)^{N(k)} \geqslant 1 - \beta$, from which it immediately follows that we need:

$$N(k) \geqslant \frac{\ln 1/\beta}{\ln 1/(1-\varepsilon)}. \tag{18}$$

4. Outer iterations and centering

In this section, we detail the operations required in the outer phase of the P-ACCP method. We first outline the overall scheme of the P-ACCP algorithm. The meaning of the probabilistic parameters ε, β will be clarified in Section 5.

[x] = P − ACCP($\mathscr{L}_1, \varepsilon, \beta$).

1. set outer iteration count $k = 1$
2. centering: compute the analytic center x_k of $\mathscr{L}_k$
3. determine $N(k)$ according to (24), see Section 5
4. oracle call: [feas, a_k, b_k] = p-oracle(x_k, $N(k)$)
5. if feas = = FALSE
6. construct neutral cut $\mathscr{H}_k = \{x : a_k^{\mathrm{T}} x \leqslant b_k\}$
7. update localization set: $\mathscr{L}_{k+1} = \mathscr{L}_k \cap \mathscr{H}_k$
8. set $k = k + 1$
9. goto 2.
10. else return $x = x_k$

At outer iteration k, the new localization set $\mathscr{L}_{k+1}$ is determined as the intersection of the cut $\mathscr{H}_k$ and the current localization set $\mathscr{L}_k$, which in turns consists in the intersection of the initial hypercube $\mathscr{L}_1$ defined in (8) and the set of half-spaces $\mathscr{H}_1, \ldots, \mathscr{H}_{k-1}$.

To analyze the complexity of the outer iterations, we first show how the new analytic center may be computed starting from the previous query point x_{k-1}.

4.1. Analytic center update

The analytic center of the set $\mathscr{L}_{k+1}$ is defined as the minimizer of the barrier function $\Phi_{k+1}(x) = -\sum_{i=1}^{k} \ln(b_i - a_i^{\mathrm{T}} x) - \sum_{i=1}^{n} \ln(R - e_i^{\mathrm{T}}(x - x_0)) - \sum_{i=1}^{n} \ln(R + e_i^{\mathrm{T}}(x - x_0))$. The problem of minimizing $\Phi_{k+1}(x)$ is an unconstrained convex optimization problem, which can be solved by Newton method starting from a point belonging to the open polytope $\{x : \|x - x_0\|_\infty < R, \quad a_i^{\mathrm{T}} x < b_i, \quad i = 1, \ldots, k\}$. In the next two subsections, we discuss the issues of initialization and implementation of a damped Newton algorithm that computes the updated center x_{k+1}.

4.1.1. Initialization

Notice that we cannot use the previous query point x_k as the starting point of Newton iterations because, by construction, it violates one of the strict inequalities (the last one). However, since the probabilistic oracle returns neutral cuts, we know that x_k lies on the boundary of $\mathscr{L}_{k+1}$. This fact can be used to find a point $\tilde{x}_k$ in the interior of $\mathscr{L}_{k+1}$ to use as the starting point of the Newton iterations.

The idea is the following: starting from x_k (which lies on the boundary of $\mathscr{H}_k$), we move in the direction of $-a_k$, until we meet the boundary of $\mathscr{L}_{k+1}$, see Fig. 1. The initialization point $\tilde{x}_k$ that we choose is the one that lies halfway between x_k and the intersection with the boundary.

We determine $\tilde{x}_k$ as follows. Let $\tilde{x}(\lambda) \doteq x_k - \lambda a_k$, $\lambda \geqslant 0$ and define $A_{k-1} \doteq [I_n \;\; -I_n \;\; a_1 \;\; \cdots \;\; a_{k-1}]^{\mathrm{T}}$, and $B_{k-1} \doteq [\mathbf{1}R + x_0 \;\; \mathbf{1}R - x_0 \;\; b_1 \;\; \cdots \;\; b_{k-1}]^{\mathrm{T}}$, where $\mathbf{1} \in \mathbb{R}^n$ is a vector of ones. Then, $\tilde{x}(\lambda) \in \mathscr{L}_{k+1}$ if and only if $\lambda \geqslant 0$, $A_{k-1}\tilde{x}(\lambda) \leqslant B_{k-1}$, i.e. if and only if $\lambda \geqslant 0$, $v_k \leqslant \lambda w_k$, where $v_k \doteq A_{k-1}x_k - B_{k-1} < 0$, $w_k \doteq A_{k-1}a_k$. The value of $\lambda \geqslant 0$ that brings $\tilde{x}(\lambda)$ on the boundary of $\mathscr{L}_{k+1}$ is $\bar{\lambda} = \sup\{\lambda \geqslant 0 : A_{k-1}\tilde{x}(\lambda) \leqslant B_{k-1}\}$, which is

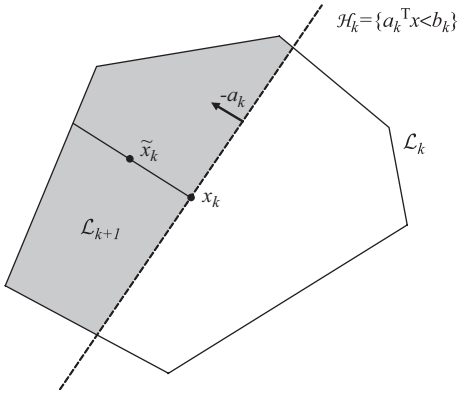

Fig. 1. Initialization for analytic center update.

readily computed as

$$\bar{\lambda} = \min_{i=1,\ldots,n} \bar{\lambda}_i \quad \text{where } \bar{\lambda}_i = \begin{cases} +\infty & \text{if } [w_k]_i \geqslant 0, \\ \dfrac{[v_k]_i}{[w_k]_i} & \text{if } [w_k]_i < 0. \end{cases} \tag{19}$$

This proves the following lemma.

Lemma 3. *The point* $\tilde{x}_k \doteq \tilde{x}(\bar{\lambda}/2) = x_k - (\bar{\lambda}/2)a_k$, *with* $\bar{\lambda}$ *given in* (19), *lies in the interior of* $\mathscr{L}_{k+1}$.

Remark 5. There exist many alternative ways to compute an initial point $\tilde{x}_k$, see for instance Goffin and Vial (2002). A method that can be used also in the presence of deep cuts consists in computing the Chebychev center of $\mathscr{L}_{k+1}$, that is, the center of the largest Euclidean ball inscribed in the polytope. Computing the center and radius of the Chebychev ball requires additional numerical effort, since the solution of a linear program is needed. However, initializing the Newton iterations from the Chebychev center may speed up the computation of the analytic center. Moreover, one can also take advantage of the information about the Chebychev radius in defining a stopping rule for the algorithm, see Remark 7.

4.1.2. Center update

Starting from point $\tilde{x}_k$, we determine the updated analytic center x_{k+1} by using a standard Newton method with backtracking line search, which is analyzed in detail in Boyd and Vandenberghe (2004, Section 9). This procedure is recalled next, where $h_{k+1}(x) = \sum_{i=1}^{k}(b_i - a_i^{\mathrm{T}}x)^{-1}a_i + \sum_{i=1}^{n}(R - e_i^{\mathrm{T}}(x - x_0))^{-1}e_i - \sum_{i=1}^{n}(R + e_i^{\mathrm{T}}(x - x_0))^{-1}e_i$ denotes the gradient of $\Phi_{k+1}(x)$, and

$$\begin{aligned} H_{k+1}(x) = &\sum_{i=1}^{k}(b_i - a_i^{\mathrm{T}}x)^{-2}a_i a_i^{\mathrm{T}} \\ &+ \sum_{i=1}^{n}(R - e_i^{\mathrm{T}}(x - x_0))^{-2}e_i e_i^{\mathrm{T}} \\ &+ \sum_{i=1}^{n}(R + e_i^{\mathrm{T}}(x - x_0))^{-2}e_i e_i^{\mathrm{T}} \end{aligned} \tag{20}$$

denotes the Hessian of $\Phi_{k+1}(x)$.

Newton Iterations.

1. Given $\tilde{x}_k$, set $x = \tilde{x}_k$ and tolerance $\tilde{\varepsilon} > 0$;
2. Compute Newton step: $v_x = -H_{k+1}^{-1}(x)h_{k+1}(x)$;
3. if $h_{k+1}^{\mathrm{T}}(x)H_{k+1}^{-1}(x)h_{k+1}(x) > 2\tilde{\varepsilon}$, then
4. choose stepsize s by backtracking line search;
5. update: $x = x + sv_x$
6. else return $x_{k+1} = x$

Backtracking Line Search.

1. Given descent direction v_x and parameters $\alpha_{\mathrm{bt}} \in (0,\ 0.5)$, $\beta_{\mathrm{bt}} \in (0,\ 1)$;
2. set $s = 1$;
3. while $\Phi_{k+1}(x + sv_x) > \Phi_{k+1}(x) + \alpha_{\mathrm{bt}}sh_{k+1}^{\mathrm{T}}(x)v_x$;
4. $s = \beta_{\mathrm{bt}}s$;
5. end while.

Remark 6. Notice that care should be exerted in the actual numerical implementation of the above algorithm. As a matter of fact, it is well known in the literature that the conditioning of the Hessian may degrade as the number of hyperplanes increases. To partially address this problem, suitable numerical techniques, such as those based on the Cholesky decomposition, should be used to compute the inverse of H_{k+1}. Also, methods based on pruning the polytope from "unimportant" hyperplanes (see, e.g., Atkinson & Vaidya, 1995) may be used to improve the numerical performance of the algorithm.

5. Properties and convergence

5.1. A bound on the number of oracle calls

The convergence of the outer iterations of the P-ACCP method and its complexity can be analyzed using arguments similar to the ones adopted in Goffin, Luo, and Ye (1996). A proof of the next key theorem is reported in the Appendix.

Theorem 2. *Fix a feasibility tolerance* $r \in (0, R)$. *In at most*

$$N_{\mathrm{outer}} \doteq \max\{50n,\ 13.87n^2,\ 8n^2(R/r)^{2.1}\} \tag{21}$$

outer iterations, either the P-ACCP algorithm finds a probabilistically feasible solution (*see Theorem* 3), *or the problem is "unfeasible" up to the given tolerance, in the sense that no n-dimensional sphere of radius r is contained in* $\mathscr{X}$.

In other words, Theorem 2 shows that the number of calls to the probabilistic oracle is of order $O(n^2(R/r)^{2.1})$ in the worst case, and hence the procedure terminates in polynomial time.

Remark 7. A few comments are due regarding the role of the feasibility tolerance r and the outer iterations bound N_{outer}. First, notice that in order to ensure a priori that the algorithm exits after a finite number of outer iterations, it is necessary that the target set $\mathscr{X}$ contains a full-dimensional ball of radius r.

In a specific application, however, it may not be known if these conditions are satisfied. For this reason, Theorem 2 is formulated so to give an explicit bound guaranteeing that if the

number of outer iterations reaches N_{outer}, then the problem under consideration must fail to satisfy the feasibility conditions with the assigned tolerance r. In this way, we demand to the algorithm itself the verification of the feasibility hypothesis. In practice, one may compute at each iteration the Chebychev radius of the localization set $\mathscr{L}_k$. If this radius goes below some pre-specified threshold, one can declare practical infeasibility and stop the algorithm, usually well before reaching the iteration limit N_{outer}.

We also remark that if the algorithm is implemented using deep, instead of neutral, cuts (see Remark 3) then it may happen that the updated polytope turns out to be empty. In this case, the algorithm is terminated with a certificate *proving* that $\mathscr{X}$ is empty and hence that the problem is unfeasible.

Remark 8. Notice that the theoretical bound on the number of outer iterations for the P-ACCP algorithm, given in Theorem 2, is higher than the corresponding bound valid for the probabilistic ellipsoid method, reported in Kanev et al. (2003) and Oishi (2003). In our notation this latter bound is $N_{\text{outer,EA}} = 2n^2 \log(\sqrt{n}R/r)$. Although the theoretical worst-case complexity bound is in favor of the EA, it is known that in practice the ACCP method achieves faster convergence. This fact is confirmed in the numerical tests reported in Section 6. Moreover, our probabilistic scheme can also work with more refined variations of the basic ACCP method. These techniques have been shown to achieve theoretical complexity bounds on the number of outer iterations on the order of $n \log^2(R/r)$, see Atkinson and Vaidya (1995) and Goffin and Vial (2002).

In the next section, we analyze the goodness of the solution returned by the P-ACCP algorithm from a probabilistic viewpoint.

5.2. Probabilistic behavior of the P-ACCP algorithm

We next analyze the overall a priori probability of the P-ACCP algorithm exiting with a "bad" solution, i.e. we assess the probability of the event $\text{ExitBad} \doteq \{\text{P-ACCP exits at some (unspecified) iteration } k \cap V(x_k) > \varepsilon\} = \{\text{P-ACCP exits at some (unspecified) iteration } k \cap \text{Bad}_k\}$, where the event Bad_k is defined in (14). The analysis is similar to the one developed in Oishi (2003, 2007) for the case of the ellipsoid algorithm.

Define the events

$$\text{ExitBad}_k \doteq \{\text{P-ACCP exits at iteration } k \cap \text{Bad}_k\}$$

and notice that

$$\text{ExitBad}_i \cap \text{ExitBad}_j = 0 \quad \text{for } i \neq j \tag{22}$$

since the algorithm may exit only at one specific outer iteration. Notice further that, in order to exit at the kth iteration, the algorithm has to reach the kth outer iteration, and then declare x_k "feasible," i.e.

$$\{\text{P-ACCP exits at iteration } k\} = \{\text{Iter}_k \cap \text{Feas}_k\},$$

where the events Feas_k, Iter_k are defined in (13), (15). Therefore, we have

$$\begin{aligned}
&\text{Prob}_\infty\{\text{ExitBad}_k\} \\
&\quad = \text{Prob}_\infty\{\text{Feas}_k \cap \text{Bad}_k \cap \text{Iter}_k\} \\
&\quad = \text{Prob}_\infty\{\text{Feas}_k \cap \text{Bad}_k | \text{Iter}_k\}\text{Prob}_\infty\{\text{Iter}_k\} \\
&\quad \leqslant \text{Prob}_\infty\{\text{Feas}_k \cap \text{Bad}_k | \text{Iter}_k\} \\
&\quad = \text{Prob}_\infty\{\text{Feas}_k | \text{Bad}_k \cap \text{Iter}_k\}\text{Prob}_\infty\{\text{Bad}_k | \text{Iter}_k\} \\
&\quad \leqslant \text{Prob}_\infty\{\text{Feas}_k | \text{Bad}_k \cap \text{Iter}_k\} < (1-\varepsilon)^{N(k)},
\end{aligned}$$

where the last inequality follows from (16). Now,

$$\begin{aligned}
&\text{Prob}_\infty\{\text{ExitBad}\} \\
&\quad = \text{Prob}_\infty\{\text{ExitBad}_1 \cup \text{ExitBad}_2 \cup \cdots\} \\
&\quad = \text{Prob}_\infty\{\text{ExitBad}_1\} + \text{Prob}_\infty\{\text{ExitBad}_2\} + \cdots \\
&\quad < (1-\varepsilon)^{N(1)} + (1-\varepsilon)^{N(2)} + \cdots = \sum_{k=1}^{\infty} (1-\varepsilon)^{N(k)}.
\end{aligned}$$

The previous sum can be made finite and arbitrarily small by appropriate choice of $N(k)$. This can be obtained in many ways. For instance, by setting $(1-\varepsilon)^{N(k)} = q^k$, for some $q < 1$ we have $\sum_{k=1}^{\infty}(1-\varepsilon)^{N(k)} = \sum_{k=1}^{\infty} q^k = \sum_{k=0}^{\infty} q^k - 1 = q/(1-q)$. Therefore, choosing a small $\beta \in (0, 1)$ and selecting $q = \beta/(1+\beta) < 1$, we have that

$$\text{Prob}_\infty\{\text{ExitBad}\} < \beta \tag{23}$$

provided that

$$N(k) \geqslant \frac{k \log(1 + 1/\beta)}{\log(1/(1-\varepsilon))}.$$

A better bound, in which the outer iteration index k appears under the logarithm, can be derived using the technique proposed in Oishi (2003). This is obtained by letting $(1-\varepsilon)^{N(k)} = (6/\pi^2)\beta 1/k^2$, from which we get

$$\sum_{k=1}^{\infty}(1-\varepsilon)^{N(k)} = \frac{6}{\pi^2}\beta \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{6}{\pi^2}\beta \frac{\pi^2}{6} = \beta.$$

Hence, (23) holds provided that

$$N(k) \geqslant N_{\text{inner}}(k) \doteq \frac{0.5 + 2\log k + \log(1/\beta)}{\log(1/(1-\varepsilon))}. \tag{24}$$

Notice that any $N(k)$ which satisfies (24) also satisfies the weaker bound (18). By the previous reasoning, the following statement holds.

Theorem 3 (*Behavior of a P-ACCP method*). *Let $\varepsilon, \beta \in (0, 1)$ be given (small) probability levels, and let $N(k)$, $k = 1, 2, \ldots,$*

be integers satisfying (24). *Then, the probability of the P-ACCP algorithm returning a bad solution* (*i.e. an* x_k *such that* $V(x_k) > \varepsilon$) *is smaller than* β.

6. Numerical example

We consider the problem of assessing whether a family of interval matrices of the form

$$A(\delta) = A_0 + \sum_{i=1}^{n_a}\sum_{j=1}^{n_a} \delta_{ij} e_i e_j^{\mathrm{T}}, \quad |\delta_{ij}| \leqslant \rho, \quad \rho > 0, \tag{25}$$

with $A(\delta) \in \mathbb{R}^{n_a, n_a}$, shares a common quadratic Lyapunov stability certificate. It is well known, see for instance Barmish (1994), that a common quadratic certificate for the whole family exists if and only if there exists a matrix $P \succ 0$ that simultaneously satisfies $N_v = 2^{n_a^2}$ Lyapunov matrix inequalities corresponding to the vertex matrices:

$$A_v^{\mathrm{T}} P + P A_v \prec 0, \quad k = 1, \ldots, N_v, \tag{26}$$

where A_v represents the kth vertex of the matrix polytope defined by (25). Specifically, we here consider an example previously examined in Calafiore and Polyak (2001), with uncertainty radius $\rho = 0.5$ and 10th order stable nominal matrix A_0 in (27).

Note that a direct application of (26) would require the simultaneous solution of $N_v = 2^{100} \simeq 1.26 \times 10^{30}$ Lyapunov inequalities, which is clearly unaffordable. Very recently, a result has been derived in Alamo, Tempo, Ramirez, and Camacho (2007), which permits to reduce the previous figure to $N_v = 2^{2n_a} \simeq 1.05 \times 10^6$. However, the number of required vertices still remains large and exponential in the matrix dimension.

We hence switch to a probabilistic approach and look for a stability certificate which is common to "most" of the matrices in the uncertain family. In addition, in order to enforce boundedness of the feasibility set, we impose a limit on the condition number of P, of the form $I \preccurlyeq P \preccurlyeq 1000 I$. The problem has $n = n_a(n_a + 1)/2 = 55$ variables.

We first applied the P-ACCP algorithm starting from an initial hyperrectangle computed according to the procedure described in Remark 2, assuming $\delta = 0$. Setting a probability of violation $\varepsilon = 10^{-4}$ and a level of confidence $\beta = 10^{-12}$, the P-ACCP algorithm (with neutral cuts) stopped after 201 outer iterations with the solution P in (27).

The behavior of the P-ACCP algorithm is shown in Fig. 2: for each outer iteration (reported on the abscissae), a number of inner iterations (represented by the bar height in the plot) is performed until either the current solution is found unfeasible (and hence is updated), or the number of random checks reaches the exit level $N_{\mathrm{inner}}(k)$. The solution was returned after being tested and found feasible in the inner loop for $N_{\mathrm{inner}}(201) = 387,357$ random values of the uncertainty.

$$A_0 = \begin{bmatrix} -55 & -23 & 90 & 37 & -16 & -130 & 44 & -49 & 17 & 31 \\ -11 & 40 & -272 & -241 & 250 & 249 & -293 & 103 & -8 & 54 \\ 56 & 56 & -140 & -56 & 58 & 106 & -80 & 82 & -32 & -28 \\ 1 & -54 & 113 & -24 & -111 & -174 & 198 & -46 & -68 & -85 \\ 61 & 61 & -56 & -52 & -12 & 44 & -49 & 76 & -38 & -22 \\ 44 & 107 & -191 & -176 & 197 & 164 & -276 & 111 & -58 & 26 \\ -18 & 96 & -278 & -188 & 283 & 292 & -440 & 107 & 29 & 101 \\ -58 & -134 & 82 & 173 & -77 & -89 & 181 & -135 & 76 & -20 \\ 39 & 133 & -438 & -215 & 273 & 422 & -376 & 165 & -8 & 78 \\ -133 & -113 & 217 & 89 & -80 & -221 & 151 & -108 & 17 & -36 \end{bmatrix},$$

$$P = \begin{bmatrix} 243.4848 & 142.6003 & 121.3552 & -27.8544 & -196.6730 & 85.3411 & -57.8713 & 137.4481 & -32.2335 & -23.1294 \\ 142.6003 & 293.9725 & -63.8313 & 82.0954 & -83.6371 & 129.5405 & -46.5530 & 200.7275 & -101.3757 & -29.3993 \\ 121.3552 & -63.8313 & 463.7165 & 60.7072 & -272.9659 & -158.1790 & -8.9061 & -129.0887 & 78.3220 & 50.3942 \\ -27.8544 & 82.0954 & 60.7072 & 641.6258 & -75.4175 & -62.8078 & 120.3207 & -41.7706 & 168.2345 & 103.8755 \\ -196.6730 & -83.6371 & -272.9659 & -75.4175 & 491.3339 & 55.7253 & -77.7316 & -27.6907 & 51.9761 & 101.8737 \\ 85.3411 & 129.5405 & -158.1790 & -62.8078 & 55.7253 & 337.8017 & -296.9537 & 140.0586 & 75.9091 & 92.2008 \\ -57.8713 & -46.5530 & -8.9061 & 120.3207 & -77.7316 & -296.9537 & 448.8441 & -18.1767 & -186.6656 & -191.7594 \\ 137.4481 & 200.7275 & -129.0887 & -41.7706 & -27.6907 & 140.0586 & -18.1767 & 249.5202 & -110.3031 & -70.0670 \\ -32.2335 & -101.3757 & 78.3220 & 168.2345 & 51.9761 & 75.9091 & -186.6656 & -110.3031 & 270.2053 & 193.2367 \\ -23.1294 & -29.3993 & 50.3942 & 103.8755 & 101.8737 & 92.2008 & -191.7594 & -70.0670 & 193.2367 & 192.3494 \end{bmatrix}. \tag{27}$$

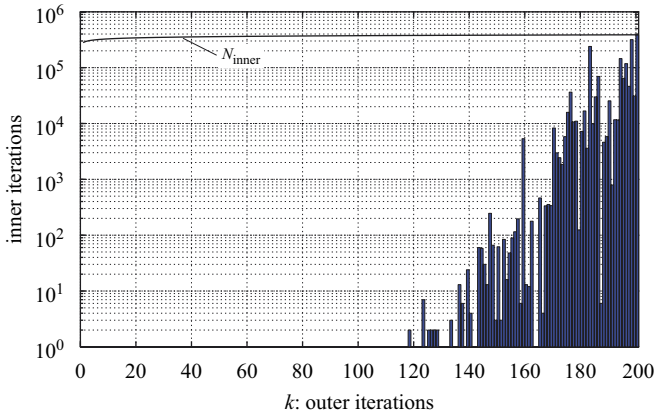

Fig. 2. Number of inner iterations performed at each step by the P-ACCP algorithm.

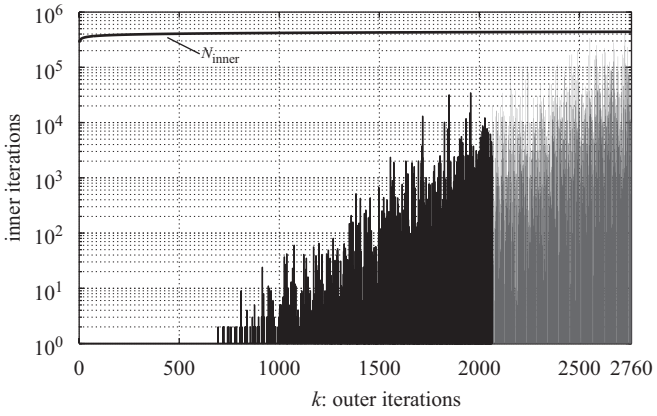

Fig. 3. Number of inner iterations performed at each step by the ellipsoid algorithm.

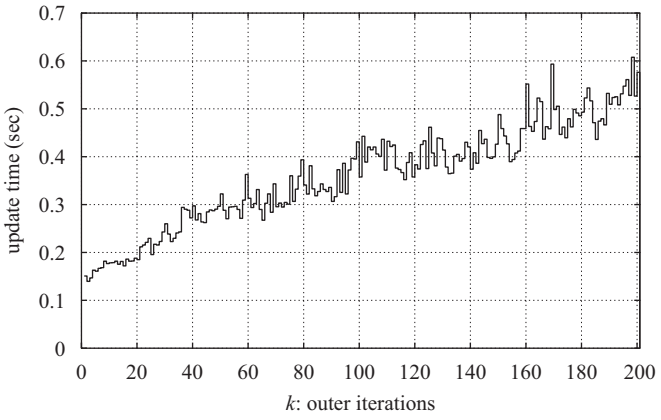

Fig. 4. Computation times for the center update phase of the P-ACCP algorithm.

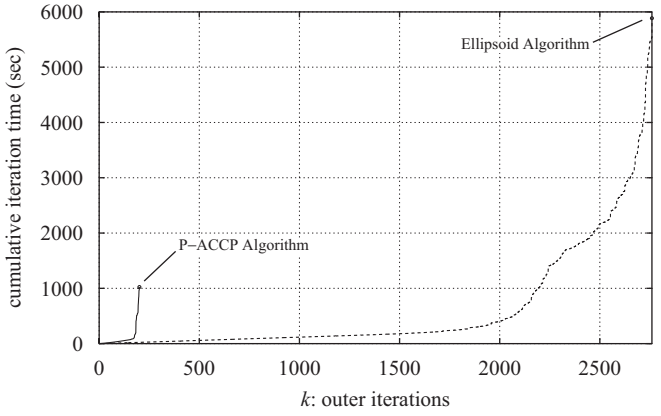

Fig. 5. Total computation time at kth iteration for the P-ACCP and the ellipsoid algorithms.

For the purpose of comparison, we run the probabilistic ellipsoid algorithm of Kanev et al. (2003) on this example, using exactly the same sequence of uncertainty extractions. The EA was initialized as described in Kanev et al. (2003), i.e. by computing a minimal ellipsoid containing the hyperrectangle previously determined. Notice that the same bound (24) applies to the number of inner iterations of the EA; see for instance Oishi (2003). The EA returned a probabilistically feasible solution after 2760 outer iterations. The performance of the EA is shown in Fig. 3.

A few comments are in order. We first remark that the update phase of the EA (see for instance the explicit algorithm description in Kanev et al., 2003, Section 3) only involves matrix multiplications, and that its numerical cost does not depend on the number of outer iterations elapsed before the update phase. Indeed, the center update phase in the EA was executed in a very short and almost constant time of about 0.45 ms. Contrary, the update phase of the P-ACCP method requires the computation of an analytic center of a number of linear inequalities that increases with the number of outer iterations. This results in a computational cost that shows basically linear growth, see Fig. 4. However, the EA needs a much higher number of outer iterations (2760 for the EA versus 201 for P-ACCP) in order to find a probabilistically feasible point. Since each outer iteration requires quite a high number of inner iterations (randomized feasibility checks), the overall computation time resulted to be much higher for the EA compared to that of P-ACCP, see Fig. 5.

Finally, we run a P-ACCP algorithm with deep cuts on the same example. In this case the algorithm found an empty polytope after 53 outer iterations, thus proving that the set $\mathscr{X}$ is actually empty, hence the problem is unfeasible from a deterministically robust point of view.

7. Conclusions

In this paper, we proposed the use of a randomized ACCP technique (P-ACCP) for solving in a probabilistic sense a general class of robust feasibility problems involving LMIs. A solution with arbitrarily small violation level can be found with a computational effort which is polynomial in the problem parameters and in the required probabilistic levels of accuracy.

The proposed method improves upon the available iterative randomized methods for probabilistic feasibility (Calafiore &

Polyak, 2001; Oishi & Kimura, 2003; Polyak & Tempo, 2001; Kanev et al., 2003; Oishi, 2003), due to the better practical convergence properties of the interior-point cutting plane algorithm, which constitutes the kernel of our method. For feasibility problems where very low levels of violation probability are required, the P-ACCP method may also be preferable to the scenario-based technique of Calafiore and Campi (2006), since this latter approach may require the one-shot solution of an SDP problem with too large a number of LMIs, while P-ACCP works iteratively with one sampled LMI constraint at a time. In this respect, notice also that the oracle checks in the P-ACCP method only involve solution of symmetric eigenvalue problems, and are fully parallelizable. It is, however, fair to say that the technique in Calafiore and Campi (2006) is applicable to general robust optimization problems, whereas P-ACCP in its present form only works for feasibility problems.

We also remark that the probabilistic analysis in this paper is not restricted to the basic ACCP method explicitly discussed here, but can actually be applied with minor modifications to *any* type of cutting plane technique, see Calafiore (2006). Changing the way in which the localization set is maintained and updated and the center is computed changes (hopefully in the improving direction) the complexity bound N_{outer} of the outer iterations of the method. For instance, if a suitable pruning is performed on the localization polytope at each outer iteration (as proposed in Atkinson & Vaidya, 1995), then a rather sophisticated analysis in Atkinson and Vaidya (1995) shows that the number of outer iterations grows as $O(n(\ln(R/r)))^2$, thus further improving the theoretical worst-case complexity of the method.

Appendix A. Proof of Theorem 2

First, we state a known property of the barrier function (9) that is instrumental for the proof. For simplicity, and without loss of generality, we shall assume henceforth that $x_0 = 0$ and $\|a_i\| = 1$.

Proposition 1 (*Boyd and Vandenberghe, 2004, Section 8*). *The barrier function* (9) *is a self-concordant function, for which the following inequalities hold for all* $k > 0$ *and for all* x *in its domain*: $\Phi_k(x) \geqslant \Phi_k^* + \sqrt{(x - x_k)^{\mathrm{T}} \tilde{H}_k (x - x_k)} - \ln(1 + \sqrt{(x - x_k)^{\mathrm{T}} \tilde{H}_k (x - x_k)})$, *where* $\tilde{H}_k \doteq H_k(x_k)$ *is the Hessian* (20), *evaluated in the analytic center and* $\Phi_k^* \doteq \Phi_k(x_k)$.

Define preliminarily the clauses: $\mathscr{B} \doteq$ {"an n-dimensional sphere of radius r is contained in $\mathscr{X}$"}; $\mathscr{P} \doteq$ {"the P-ACCP algorithm finds a probabilistically feasible solution in at most N_{outer} iterations"}; $\mathscr{R} \doteq$ {"the P-ACCP algorithm finds a robustly feasible solution, i.e. an $x \in \mathscr{X}$, in at most N_{outer} iterations"}.

Notice that $\mathscr{R} \Rightarrow \mathscr{P}$ since, obviously, a robustly feasible solution is also a probabilistically feasible solution. We prove in the sequel that $\mathscr{B} \Rightarrow \mathscr{R}$, from which it follows that $\mathscr{B} \Rightarrow \mathscr{P}$. This implies that "$\mathscr{P}$ or $\overline{\mathscr{B}}$" is true, which is the statement of Theorem 2.

To prove that $\mathscr{B} \Rightarrow \mathscr{R}$, we follow a reasoning similar to the one in Goffin et al. (1996), and bound from above and below the values of the barrier $\Phi_{k+1}^* \doteq \Phi_{k+1}(x_{k+1})$ with appropriate functions of k.

A.1. An upper bound on Φ_{k+1}^*

At iteration k, if the algorithm has not terminated, it means that the target set $\mathscr{X}$ is contained in the set $\mathscr{L}_{k+1}$, which is described by the inequalities

$$-R \leqslant x_i \leqslant R,\ i = 1, \ldots, n; \quad a_j^{\mathrm{T}} x \leqslant b_j,\ j = 1, \ldots, k \tag{A.1}$$

with analytic center x_{k+1}. Suppose now, for the purpose of contradiction, that an n-dimensional sphere of radius r and (unknown) center x^o is contained in $\mathscr{X} \subseteq \mathscr{L}_{k+1}$. This implies that, for all directions $\|z\| \leqslant 1$, we have $\|x^o + rz\|_\infty \leqslant R$, $a_j^{\mathrm{T}}(x^o + rz) \leqslant b_j$, $j = 1, \ldots, k$. In particular, this means that

$$(R + x_i^o) \geqslant r, \quad (R - x_i^o) \geqslant r, \quad i = 1, \ldots, n, \tag{A.2}$$

$$b_j - a_j^{\mathrm{T}} x^o \geqslant r, \quad j = 1, \ldots, k. \tag{A.3}$$

The first inequality is trivial and follows from the choice $z = -e_i$, the second inequality follows from the choice $z = a_j$, and from the fact that $\|a_j\| = 1$. Applying the inequalities (A.2)–(A.3) to the quantities in the barrier function, we obtain the following upper bound:

$$\begin{aligned} \Phi_{k+1}^* &\doteq \inf_x \Phi_{k+1}(x) \\ &\leqslant \Phi_{k+1}(x^o) \leqslant (2n + k) \ln(1/r) \doteq \Psi_{\text{UP}}(k). \end{aligned} \tag{A.4}$$

A.2. A lower bound on Φ_{k+1}^*

To obtain a lower bound on Φ_{k+1}^*, we write $\Phi_{k+1}^* = \inf_x \Phi_{k+1}(x) = \inf_x (\Phi_k(x) - \ln(b_k - a_k^{\mathrm{T}} x)) = \inf_x (\Phi_k(x) - \ln(-a_k^{\mathrm{T}}(x - x_k)))$, where the last equality follows from the fact that the cuts are neutral, and hence $a_k^{\mathrm{T}} x_k = b_k$. Applying now Proposition 1 to Φ_k, we get: $\Phi_{k+1}^* \geqslant \inf_x (\Phi_k^* + \sqrt{(x - x_k)^{\mathrm{T}} \tilde{H}_k (x - x_k)}) - \ln(1 + \sqrt{(x - x_k)^{\mathrm{T}} \tilde{H}_k (x - x_k)}) - \ln(-a_k^{\mathrm{T}}(x - x_k)) = \inf_v (\Phi_k^* + \sqrt{v^{\mathrm{T}} \tilde{H}_k v} + \ln(1 + \sqrt{v^{\mathrm{T}} \tilde{H}_k v}) - \ln(-a_k^{\mathrm{T}} v))$, with $v \doteq (x - x_k)$. Taking the gradient with respect to v, it can be shown that the infimum is attained for $v^* = -\eta \tilde{H}_k^{-1} a_k / \sqrt{a_k^{\mathrm{T}} \tilde{H}_k^{-1} a_k^{\mathrm{T}}}$, $\eta \doteq (1 + \sqrt{5})/2$. Substituting in the previous expression, we have

$$\begin{aligned} \Phi_{k+1}^* &\geqslant \Phi_k^* + \eta - \ln(1 + \eta) - \ln(\eta \sqrt{a_{k-1}^{\mathrm{T}} \tilde{H}_k^{-1} a_{k-1}}) \\ &\geqslant \Phi_k^* - \frac{1}{2} \ln(a_k^{\mathrm{T}} \tilde{H}_k^{-1} a_k) \end{aligned} \tag{A.5}$$

$$\geqslant \Phi_1^* - \frac{1}{2} \sum_{j=1}^{k} \ln(a_j^{\mathrm{T}} \tilde{H}_j^{-1} a_j) \tag{A.6}$$

$$= 2n \ln(1/R) - \frac{1}{2} \sum_{j=1}^{k} \ln(a_j^{\mathrm{T}} \tilde{H}_j^{-1} a_j), \tag{A.7}$$

where (A.5) follows yielding the term $\eta - \ln(1+\eta) - \ln(\eta) = 0.1744 > 0$; (A.6) follows from a recursive application of the first inequality; and (A.7) follows from $\Phi_1^* = \Phi_1(x_1) = \Phi_1(0) = -2n\ln(R)$.

We now concentrate on bounding from below the Hessian $\tilde{H}_j$ which appears in the above expression. To this end, we first bound the first term of the Hessian (see (20)) as $\sum_{i=1}^{j}(b_i - a_i^{\mathrm{T}}x_j)^{-2}a_i a_i^{\mathrm{T}} \succcurlyeq (1/4R^2n)\sum_{i=1}^{j} a_i a_i^{\mathrm{T}}$, since, for $i = 1, \ldots, j$, we have $(b_i - a_i^{\mathrm{T}}x_j) =^{(1)} (a_i^{\mathrm{T}}x_i - a_i^{\mathrm{T}}x_j) = a_i^{\mathrm{T}}(x_i - x_j) \leqslant \|a_j\|\|x_i - x_j\| =^{(2)} \|x_i - x_j\| \leqslant^{(3)} 2R\sqrt{n}$, where $^{(1)}$ is a consequence of the fact that all cuts are neutral, $^{(2)}$ follows from the assumption $\|a_j\| = 1$, and $^{(3)}$ follows from $\|x_i - x_j\| \leqslant \sqrt{n}\|x_i - x_j\|_\infty \leqslant (\|x_i\|_\infty + \|x_j\|_\infty)\sqrt{n}$. Similarly, to bound the second term of the Hessian, we write $\sum_{i=1}^{n}(R - e_i^{\mathrm{T}}x_j)^{-2}e_i e_i^{\mathrm{T}} + \sum_{i=1}^{n}(R + e_i^{\mathrm{T}}x_j)^{-2}e_i e_i^{\mathrm{T}} = (1/R^2)\sum_{i=1}^{n}((1-[x_j]_i/R)^{-2} + (1+[x_j]_i/R)^{-2})e_i e_i^{\mathrm{T}} \succcurlyeq 2/R^2 I$, since $|[x_j]_i| \leqslant R$, $i = 1, \ldots, n$, and $(1-\alpha)^{-2} + (1+\alpha)^{-2} \geqslant 2$ for $\alpha \in [-1, 1]$. Hence, we get the inequality

$$\tilde{H}_j \succcurlyeq \frac{2}{R^2}\left(I + \frac{1}{8n}\sum_{i=1}^{j} a_i a_i^{\mathrm{T}}\right) \doteq \frac{2}{R^2}B_j, \tag{A.8}$$

where matrices B_j can be recursively defined, for $j = 1, \ldots, k$, as follows: $B_0 \doteq I$, $B_j \doteq B_{j-1} + (1/8n)a_j a_j^{\mathrm{T}}$. Applying bound (A.8)–(A.7), we get: $\Phi_{k+1}^* \geqslant 2n\ln(1/R) - \frac{1}{2}\sum_{j=1}^{k}\ln((R^2/2)a_j^{\mathrm{T}}B_j^{-1}a_j) = 2n\ln(1/R) + (k/2)\ln(2/R^2) + \frac{1}{2}\sum_{j=1}^{k}(-\ln(a_j^{\mathrm{T}}B_j^{-1}a_j))$. Notice now that $-\ln(x)$ is a convex function, hence applying Jensen's inequality we have: $-\ln((1/k)\sum_{j=1}^{k}\beta_j) \leqslant (1/k)\sum_{j=1}^{k}(-\ln(\beta_j))$, which leads to the inequality $\Phi_{k+1}^* \geqslant 2n\ln(1/R) + (k/2)\ln(2/R^2) - (k/2)\ln((1/k)\sum_{j=1}^{k} a_j^{\mathrm{T}}\tilde{B}_j^{-1}a_j)$. To further bound Φ_{k+1}^*, first notice that, again by Jensen's inequality, $-\ln(\mathrm{Tr}(B_k)/n) = -\ln((1/n)\sum_{i=1}^{n}\lambda_i) \leqslant (1/n)\sum_{i=1}^{n}(-\ln(\lambda_i)) = -(1/n)\ln\det B_k$, where λ_i are the eigenvalues of B_k. Combining this latter expression with the chain of equalities $\mathrm{Tr}(B_k) = \mathrm{Tr}(I + (1/8n)\sum_{i=1}^{k} a_i a_i^{\mathrm{T}}) = n + (1/8n)\mathrm{Tr}(\sum_{i=1}^{k} a_i a_i^{\mathrm{T}}) = n + (1/8n)\sum_{i=1}^{k}\mathrm{Tr}(a_i^{\mathrm{T}}a_i) = n + k/8n$, leads to the inequality $\ln\det B_k \leqslant n\ln(\mathrm{Tr}(B_k)/n) = n\ln(1 + k/8n^2)$. On the other hand, we have

$$\ln\det B_k = \ln\det\left(B_{k-1}\left(I + \frac{1}{8n}B_{k-1}^{-1}a_k a_k^{\mathrm{T}}\right)\right) \tag{A.9}$$

$$= \ln\det B_{k-1} + \ln\left(1 + \frac{1}{8n}a_k^{\mathrm{T}}B_{k-1}^{-1}a_k\right) \tag{A.10}$$

$$\geqslant \ln\det B_{k-1} + \frac{1}{16n}a_k^{\mathrm{T}}B_{k-1}^{-1}a_k \tag{A.11}$$

$$\geqslant \ln\det B_0 + \frac{1}{16n}\sum_{j=1}^{k} a_j^{\mathrm{T}}B_{j-1}^{-1}a_j \tag{A.12}$$

$$\geqslant \frac{1}{16n}\sum_{j=1}^{k} a_j^{\mathrm{T}}B_{j-1}^{-1}a_j, \tag{A.13}$$

where (A.10) follows from the fact that $\det(I + ab^{\mathrm{T}}) = 1 + b^{\mathrm{T}}a$, $\forall a, b \in \mathbb{R}^n$; (A.11) follows from the fact that $a_k^{\mathrm{T}}B_{k-1}^{-1}a_k \leqslant 1$, and that, for $x \in [0, 1]$, $\ln(1+x) \geqslant x\ln(2) \geqslant x/2$, and (A.12) follows from a recursive application of (A.11). Putting things together, we get $\sum_{j=1}^{k} a_j^{\mathrm{T}}B_j^{-1}a_j \leqslant 16n^2\ln(1 + k/8n^2)$ which leads to

$$\begin{aligned}\Phi_{k+1}^* &\geqslant 2n\ln\frac{1}{R} + \frac{k}{2}\ln(2/R^2) - \frac{k}{2}\ln\left(\frac{16n^2}{k}\ln\left(1 + \frac{k}{8n^2}\right)\right)\\ &= 2n\ln\frac{1}{R} + \frac{k}{2}\ln(2/R^2) - \frac{k}{2}\ln\left(2\frac{\ln(1 + k/8n^2)}{k/8n^2}\right)\\ &= 2n\ln\frac{1}{R} + \frac{k}{2}\ln(1/R^2) + \frac{k}{2}\ln\left(\frac{k/8n^2}{\ln(1 + k/8n^2)}\right),\end{aligned}$$

hence $\Phi_{k+1}^* \geqslant \Psi_{\mathrm{LW}}(k)$,

$$\Psi_{\mathrm{LW}}(k) \doteq (2n + k)\ln\frac{1}{R} + \frac{k}{2}\ln\left(\frac{k/8n^2}{\ln(1 + k/8n^2)}\right). \tag{A.14}$$

A.3. An upper bound on the iterations count k

Combining (A.4) and (A.14) we get the inequality $\Psi_{\mathrm{LW}}(k) \leqslant \Phi_{k+1}^* \leqslant \Psi_{\mathrm{UP}}(k)$ in the variable k. Notice that we can write

$$\begin{aligned}&\Psi_{\mathrm{UP}}(k) - \Psi_{\mathrm{LW}}(k)\\ &\quad= (2n + k)\ln(1/r) - (2n + k)\ln(1/R) - \frac{k}{2}\ln\Upsilon(k)\\ &\quad= (2n + k)\ln(R/r) - \frac{k}{2}\ln\Upsilon(k),\end{aligned} \tag{A.15}$$

where $\Upsilon(k) \doteq (k/(8n^2))/\ln(1 + k/(8n^2))$. We show that for some finite k, the difference $\Psi_{\mathrm{UP}}(k) - \Psi_{\mathrm{LW}}(k)$ is negative, which leads to a contradiction, i.e. a ball of radius r cannot be contained in $\mathscr{L}_{k+1}$. First, notice that, for any $\alpha > 1$, $(2n + k) < \alpha k$ holds for all $k > k_1(\alpha) \doteq 2n/(\alpha - 1)$. Also, for any $\beta < 1$, there exists a $k_2(\beta)$ such that $\Upsilon(k) < (k/8n^2)^{\beta}$ holds for all $k > k_2(\beta)$ where $k_2(\beta)$ is the only positive root of the transcendental equation $\Upsilon(k) = (k/8n^2)^{\beta}$. Hence, substituting the last two inequalities into (A.15), we get

$$\Psi_{\mathrm{UP}}(k) - \Psi_{\mathrm{LW}}(k) \leqslant \alpha k\ln(R/r) - \frac{k}{2}\ln\left(\frac{k}{8n^2}\right)^{\beta},$$

for $k > \max\{k_1(\alpha), k_2(\beta)\}$. The quantity on the right-hand side is negative for

$$k > k_3(\alpha, \beta) \doteq 8n^2(R/r)^{2\alpha/\beta}. \tag{A.16}$$

Therefore, for $k > \max\{k_1(\alpha), k_2(\beta), k_3(\alpha, \beta)\}$ we have that $\Psi_{\mathrm{UP}}(k) - \Psi_{\mathrm{LW}}(k) < 0$ and the contradiction is found. From (A.16) we notice that α/β can be chosen arbitrarily close to one, and therefore k grows as $O\left(n^2R^2/r^2\right)$. In particular, by choosing $\alpha = 1.04$ and $\beta = 0.9905$ we get the explicit bound (21) on the number of oracle calls, thus concluding the proof.

References

Alamo, T., Tempo, R., Ramirez, D. R., & Camacho, E. F. (2007). A new vertex result for robustness problems with interval matrix uncertainty. In *Proceedings of the European control conference*.

Atkinson, D. S., & Vaidya, P. M. (1995). A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming, Series B*, *69*, 1–43.

Barmish, B. R. (1994). *New tools for robustness of linear systems*. New York: MacMillan.

Ben-Tal, A., & Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, *23*, 769–805.

Ben-Tal, A., & Nemirovski, A. (2002). On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. *SIAM Journal on Optimization*, *12*(3), 811–833.

Bliman, P.-A. (2004). A convex approach to robust stability for linear systems with uncertain scalar parameters. *SIAM Journal on Control and Optimization*, *42*(6), 2016–2042.

Boyd, S., & El Ghaoui, L. (1993). Method of centers for minimizing generalized eigenvalues. *Linear Algebra and its Applications (Special Issue on Systems and Control)*, *188*, 63–111.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge.

Calafiore, G. (2006). Cutting plane methods for probabilistically-robust feasibility problems. In *Proceedings of mathematical theory of networks and systems*, Kyoto, Japan.

Calafiore, G., & Campi, M. C. (2005). Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, *102*(1), 25–46.

Calafiore, G., & Campi, M. C. (2006). The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, *51*(5), 742–753.

Calafiore, G., & Dabbene, F. (Eds.) (2006). *Probabilistic and randomized methods for design under uncertainty*. London: Springer.

Calafiore, G., & Polyak, B. T. (2001). Stochastic algorithms for exact and approximate feasibility of robust LMIs. *IEEE Transactions on Automatic Control*, *46*, 1755–1759.

El Ghaoui, L., Oustry, F., & Lebret, H. (1998). Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, *9*, 33–52.

Goffin, J.-L., Luo, Z. Q., & Ye, Y. (1996). Complexity analysis of an interior point cutting plane method for convex feasibility problems. *SIAM Journal on Optimization*, *6*, 638–652.

Goffin, J.-L., & Vial, J.-P. (1999). Shallow, deep and very deep cuts in the analytic center cutting plane method. *Mathematical Programming*, *84*, 89–103.

Goffin, J.-L., & Vial, J.-P. (2002). Convex non-differentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods & Software*, *17*, 805–867.

Kanev, S., De Schutter, B., & Verhaegen, M. (2003). An ellipsoid algorithm for probabilistic robust controller design. *Systems & Control Letters*, *49*, 365–375.

Kanev, S., & Verhaegen, M. (2006). The randomized ellipsoid algorithm for constrained robust least squares problems. *Probabilistic and randomized methods for design under uncertainty* (pp. 223–242). London: Springer.

Khachiyan, L. G. (1979). A polynomial time algorithm for linear programming. *Soviet Math. Doklady*, *20*, 191–194.

Lovász, L., Grötschel, M., & Schrijver, A. (1993). *Geometric algorithms and combinatorial optimization*. New York: Springer.

Mitchell, J. E. (2003). Polynomial interior point cutting plane methods. *Optimization Methods & Software*, *18*, 507–534.

Nemirovski, A. S., & Yudin, D. B. (1983). *Problem complexity and method efficiency in optimization*. New York: Wiley.

Oishi, Y. (2003). Probabilistic design of a robust state-feedback controller based on parameter-dependent Lyapunov functions. In *Proceedings of the IEEE conference on decision and control*.

Oishi, Y. (2007). Polynomial-time algorithms for probabilistic solutions of parameter-dependent linear matrix inequalities. *Automatica*, *43*, 538–545.

Oishi, Y., & Kimura, H. (2003). Computational complexity of randomized algorithms for solving parameter-dependent linear matrix inequalities. *Automatica*, *39*, 2149–2156.

Péton, O. (2002). *The homogeneous analytic center cutting plane method*. Ph.D. Thesis, Université de Genève.

Polyak, B. T., & Tempo, R. (2001). Probabilistic robust design with linear quadratic regulators. *Systems & Control Letters*, *43*, 343–353.

Scherer, C. W. (2005). Relaxations for robust linear matrix inequality problems with verifications for exactness. *SIAM Journal on Matrix Analysis and Applications*, *27*(2), 365–395.

Shor, N. Z. (1970). Utilization of the operation of space dilatation in the minimization of convex functions. *Cybernetics*, *13*, 94–96.

Tempo, R., Calafiore, G., & Dabbene, F. (2004). *Randomized algorithms for analysis and control of uncertain systems. Communications and control engineering series*. London: Springer.

Giuseppe Calafiore received a "Laurea" degree in Electrical Engineering in 1993 and Doctorate in Information and System Theory from the Politecnico di Torino, in 1997.
Dr. Calafiore currently serves as an Associate Professor at the "Dipartimento di Automatica e Informatica," Politecnico di Torino, Italy.
He held visiting positions at the Information Systems Laboratory, Stanford University in 1995, at Ecole Nationale Supérieure de Techniques Avenceés (ENSTA), Paris, in 1998, and at the University of California at Berkeley in 1999 and 2003.
Dr. Calafiore is an Associate Editor for the IEEE Transactions on Systems, Man, and Cybernetics and for the IEEE Transactions on Automation Science and Engineering. His research interests are in the field of analysis and control of uncertain systems, convex optimization, probabilistic algorithms and risk management. He is author or coauthor of six books and over 90 publications on international journals and conferences.

Fabrizio Dabbene received a Laurea degree in Electrical Engineering in 1995 and a Ph.D. in Systems and Computer Engineering in 1999, both from the Politecnico di Torino, Italy. He currently holds a tenured research position at the Institute IEIIT of the National Research Council (CNR) of Italy.
Dr. Dabbene is also in collaboration with Politecnico di Torino, where he teaches several courses in Systems and Control, and with Università degli Studi di Torino, where he does research on modeling of environmental systems. He has been a Visiting Researcher at the Department of Electrical Engineering, University of Iowa. From 2002 he serves as an Associate Editor of the Conference Editorial Board of the IEEE Control System Society. He currently chairs the Action Group on Probabilistic and Randomized Methods in Control (PRMC) of the IEEE Technical Committee on Computer Aided Control System Design (CACSD). His research interests include robust control and identification of uncertain systems, randomized algorithms for systems and control, convex optimization and modeling of environmental systems.