数理解析研究所講究録 1335 巻 2003 年 97-104

Proving and Solving Semi-definite Programming over Reals

穴井 宏和
HIROKAZU ANAI
(株) 富士通研究所
FUJITSU LABORATORIES LTD *

パブロ・パリロ
PABLO. A. PARRILO
ETH
SWISS FEDERAL INSTITUTE OF TECHNOLOGY †

Abstract

Semidefinite Programming (SDP) is a class of convex optimization problems with a linear objective function and linear matrix inequality (LMI) constraints. SDP problems have many applications in engineering and applied mathematics. We propose a reasonably fast algorithm to prove and solve SDP exactly by exploiting the convexity of the SDP feasibility domain. This is achieved by combining a symbolic algorithm of cylindrical algebraic decomposition (CAD) and a lifting strategy that takes into account the convexity properties of SDP. The effectiveness of our method is examined by applying it to some examples on QEPCAD and maple.

1 Introduction

Semidefinite Programming (SDP) is one of the recent main developments in mathematical programming, with many applications in engineering problems. In particular, a wide variety of questions in systems and control theory, as well as in several other areas, can be cast and solved as SDP problems, that is, optimization problems with a linear objective function and linear matrix inequality (LMI) constraints (see [2],[6]). For these reasons, SDP problems are of great practical and theoretical interest in control theory.

Usually, SDPs are solved by using numerical packages based on an interior point method, hence obtaining an approximate solution with finite precision. In certain applications (for instance, SDP algorithms based on algebraic geometry [9]) or critical situations, such as ill-posed problems, there is a real danger of arriving at an incorrect answer; we may obtain a "numerically" feasible solution for an infeasible problem, or vice versa. Hence it is important to develop methods of computing the exact feasible solution of SDP problems, and that are also able to determine their infeasibility exactly. This can be accomplished by a symbolic optimization method based on quantifier elimination (QE). The downsides of this approach are the bad computational complexity properties of generic QE algorithms.

Therefore, in this paper we propose a reasonably faster algorithm to prove and solve SDP problems *exactly* based on a symbolic method of *cylindrical algebraic decomposition (CAD)*[4], and the careful exploitation of the convexity of the SDP domain. Moreover, we also assume that we have a feasible interior point as a numerical interior point method requires one. We examine the performance of our

*anai@jp.fujitsu.com
†parrilo@aut.ee.ethz.ch

method by solving some examples by using QEPCAD[1)] and maple. Our method can be regarded as a specialized CAD algorithm for SDP exploiting convexity and an initial interior point. This could be generalized to other classes of convex programming.

2 Semidefinite programming

SDP problems. We briefly show the definitions of LMIs and SDP problems. A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is *positive (semi) definite* if and only if quadratic forms $x^TAx > 0$ (≥ 0) for all $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$ s.t. $x \neq 0$, where x^T stands for the transpose of x. In the sequel, when A is positive (semi) definite, we denote it by $A >_d 0$ ($\geq_d 0$). For a real symmetric matrix A, $A >_d 0$ ($\geq_d 0$) if and only if all eigenvalues of A are positive (non negative). A *linear matrix inequality (LMI)* is a matrix inequality of the form

$$M(x) = M_0 + \sum_{i=1}^{m} x_i M_i >_d 0 \ (\geq_d 0) \tag{1}$$

where $x \in \mathbb{R}^m$ is the variable vector and $M_i = M_i^T \in \mathbb{R}^{n\times n}$, $i = 0, \ldots, m$, are given symmetric matrices.

In general, there are three types of generic LMI problems; *Feasibility problem*, *Linear objective minimization problem under LMI constraints* and *Generalized eigenvalue minimization problem* (see [6]). Among them we consider the problem of minimizing a linear objective function in a vector variable $x \in \mathbb{R}^m$ subject to a linear matrix $M(x)$,

$$\begin{array}{ll} \textit{minimize} & c^Tx \\ \textit{subject to} & M(x) \geq_d 0, \end{array} \tag{2}$$

where $c \in \mathbb{R}^m$. This problem is called *Semidefinite Programming (SDP)*. For a vector x_0, if $M(x_0) \geq_d 0$, x_0 is called *feasible*. If there is no feasible solution, we say that the problem (2) is *infeasible*. Notice in particular that the optimal solution is on the boundary of the (convex) feasible set. Also, SDP includes many important optimization problems such as linear programming, as special cases.

Reducing SDP to QE problems Optimization problems of minimizing an objective function $h(x)$ subject to a constraint that is a first-order formula $\phi(x)$ are solved by using QE as follows: First introduce a new indeterminate z assigned to the objective function h and consider the new first-order formula $\phi'(x, z) = \phi \ \wedge (z - h \geq 0)$. We call the polynomial $z - h$ a *objective polynomial*. Then the problem minimizing h subject to ϕ is formulated as a QE problem $\Phi \equiv \ ^\exists x_1 \cdots \ ^\exists x_n(\phi')$. Next eliminate all quantified variables $x_1, \ldots, x_n$ to have the resulting quantifier-free formula Φ' in z. Then Φ' gives a finite union M of intervals for z, which shows a possible range of z. If M is empty, ψ is unsolvable (*i.e.* infeasible); if M is unbounded from below, h has no minimum w.r.t. ϕ; if $m \in M$ is a lowest endpoint of M, then m is the minimum value of z w.r.t. ϕ (for details [10]).

Now we show how SDP problems are reduced to QE problems and solved by using QE techniques. Determining (semi)definiteness for a real symmetric matrix can be achieved without computing eigenvalues matrix by using the following well-known Sylvester's criterion:

Theorem 1 (Sylvester's criterion)
Let $A = (a_{ij}) \in \mathbb{C}^{n\times n}$ be a Hermitian matrix. Then A is positive semidefinite if and only if all principal

[1)] See `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`

minors of A are non negative i.e.

$$\det A \begin{pmatrix} i_1 & i_2 & \cdots & i_r \\ i_1 & i_2 & \cdots & i_r \end{pmatrix} \geq 0, \tag{3}$$

for $1 \leq i_1 < i_2 < \cdots < i_r \leq n$, $r = 1, 2, \cdots, n$, *where*

$$A \begin{pmatrix} i_1 & i_2 & \cdots & i_r \\ j_1 & j_2 & \cdots & j_r \end{pmatrix}$$

denotes the $r \times r$ submatrix of A which consists of (i_k, j_l)-entries of A, where $1 \leq i_1 < i_2 < \cdots < i_r \leq n$ and $1 \leq j_1 < j_2 < \cdots < j_r \leq n$.

By this criterion, $A(x) \geq_d 0$ can be reduced to an equivalent formula which is the conjunction of $2^n - 1 (\equiv \sum_{r=1}^{n} \binom{n}{r})$ inequalities.

3 CAD algorithm

We briefly sketch the basic ideas of cylindrical algebraic decomposition, see [4] for details. Assume that we are given an input formula

$$\varphi(u_1, \ldots, u_m) \equiv \mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n \; \psi(u_1, \ldots, u_m, x_1 \ldots x_n), \quad \mathsf{Q}_i \in \{\exists, \forall\}.$$

Let F be the set of polynomials appearing in ψ as left hand sides of atomic formulas. We say that $C \subseteq \mathbb{R}^{m+n}$ is *sign-invariant* for F if every polynomial in F has a constant sign on all points in C. Then $\psi(c)$ is either "true" or "false" for all $c \in C$.

Suppose we have a finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$ for F which has the following properties:

1. Each $\mathcal{D}_i$ is a finite partition of $\mathbb{R}^i$ into connected semi-algebraic cells. For $1 \leq j \leq n$ each $\mathcal{D}_{m+j}$ is labeled with Q_j
2. $\mathcal{D}_{i-1}$ for $1 < i \leq m+n$ consists exactly of the projections of all cells in $\mathcal{D}_i$ along the coordinate of the i-th variable in $(u_1, \ldots, u_m, x_1 \ldots x_n)$. For each cell $C \in \mathcal{D}_{i-1}$ we can determine the preimage $S(C) \subseteq \mathcal{D}_i$ under the projection.
3. For each cell $C \in \mathcal{D}_m$ we know a quantifier-free formula $\delta_C(u_1, \ldots, u_m)$ describing this cell.
4. Each cell $C \in \mathcal{D}_{m+n}$ is sign-invariant for F. Moreover for each cell $C \in \mathcal{D}_{m+n}$ we are given a *test point* t_C in such a form that we can determine the sign of $f(t_C)$ for each $f \in F$ and thus evaluate $\varphi(t_C)$.

Then a finite partition $\mathcal{D}_{m+n}$ of $\mathbb{R}^{m+n}$ for F is called an *F-invariant cylindrical algebraic decomposition* of $\mathbb{R}^{m+n}$. A quantifier-free equivalent formula φ is obtained as the disjunction of all δ_C for which $C \in \mathcal{D}_m$ is *valid* in the following sense:

1. For $m \leq i < m+n$, we have $\mathcal{D}_{i+1}$ that is labeled:
   (a) If $\mathcal{D}_{i+1}$ is labeled "$\exists$", then $C \in \mathcal{D}_i$ is valid if at least one $C' \in S(C)$ is valid.
   (b) If $\mathcal{D}_{i+1}$ is labeled "$\forall$", then $C \in \mathcal{D}_i$ is valid if all $C' \in S(C)$ are valid.
2. A cell $C \in \mathcal{D}_{m+n}$ is valid if $\varphi(t_C)$ is "true".

The algorithm to obtain such a sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$, the quantifier-free formula δ_C, and the test point t_C consists of two phases, the *projection phase* and *construction (lifting) phase.*

Projection phase In the projection phase, one constructs from $F \subseteq \mathbb{R}[u_1, \ldots, u_m, x_1, \ldots, x_n]$ a new finite set $F' \subseteq \mathbb{R}[u_1, \ldots, u_m, x_1, \ldots, x_{n-1}]$ which satisfies the following condition: Consider $a, b \in \mathbb{R}^{m+n-1}$ such that for all $f' \in F'$ the signs of both $f'(a), f'(b) \in \mathbb{R}$ are equal. Then for all $f \in F$ the corresponding univariate polynomials $f(a, x_n), f(b, x_n) \in \mathbb{R}[x_n]$ both have the same number of different real and complex roots. This guarantees the following property called "*delineability*": Let C be a connected subset of $\mathbb{R}^{m+n-1}$ that is sign-invariant for F'. For each $f \in F$ consider the functions $\rho_k : C \to \mathbb{R}$ assigning to $a \in C$ the k-th real root of $f(a, x_n) \in \mathbb{R}[x_n]$. Then all these ρ_k are continuous. Moreover, the graph of the various ρ_k do not intersect. In other words, the order of the real roots does not change as they continuously change their position in the real line.

The step from F to F' is called a *projection* and denoted by $F' := PROJ(F)$. We call polynomials in F' *projection polynomials* and the irreducible factors of projection polynomials of F' *projection factors.* Iterative application of $PROJ$-operator leads to a finite sequence

$$F_{m+n}, \ldots, F_1, \quad \textit{where } F_{m+n} := F,\ F_i := PROJ(F_{i+1}) \ \textit{ for } \ 1 \leq i < m+n.$$

$PROJ$-operator computes certain coefficients, discriminants, resultants, and subresultant coefficients obtained from the polynomials in F_{i+1} and their higher derivatives, regarded as univariate polynomials in their last variable, which is the $(i+1)$-st one in $(u_1, \cdots, u_m, x_1, \ldots, x_n)$. The final set F_1 contains univariate polynomials in u_1.

Construction phase In the construction phase first construct a partition $\mathcal{D}_1$ of the real line $\mathbb{R}^1$ into finitely many intervals that are sign-invariant for F_1: The real zeros of univariate polynomials in F_1 define a sign invariant decomposition of $\mathbb{R}^1$. The partition $\mathcal{D}_1$ consists of cells that are these zeros and the intermediate open intervals. Thus we isolate the above zeros and find test points in each interval. This procedure is called the *base phase.* For an open interval we may choose a rational test point but for a zero in general we need an exact representation of an algebraic number.

For $1 \leq i < m+n$ the partitions $\mathcal{D}_i \subseteq \mathbb{R}^i$ are computed recursively: The roots of all polynomials in F_i as univariate polynomials in their last variable are delineated above each connected cell C in $\mathcal{D}_{i-1}$. Thus we can cut the *cylinder* above C into finitely many connected semi-algebraic cells. Then $\mathcal{D}_i$ is a collection of all these cells arising from all cylinders above the cells of $\mathcal{D}_{i-1}$

Consider the lifting from the partition $\mathcal{D}_1$ of $\mathbb{R}^1$ to a partition $\mathcal{D}_2$ of $\mathbb{R}^2$ since remaining lifting procedures until $\mathbb{R}^{m+n}$ are achieved by repeating the same procedure as the lifting from $\mathbb{R}^1$ to $\mathbb{R}^2$. We show the construction of the test points of each cell of $\mathbb{R}^2$ belonging to the cylinder over a cell $C \in \mathcal{D}_1$ with a test point α. First specialize the polynomials in $F_2 := PROJ^{m+n-2}(F)$ by the test point α of C. We then get a set of univariate polynomials in u_2 and deal with these polynomials in u_2 in the same way as the base phase, i.e., root isolation and choice of test points. The lifting from $\mathbb{R}^1$ to $\mathbb{R}^2$ is regarded as the construction of the second component of the test points of $\mathcal{D}_{m+n}$.

The Construction phase produces a list of (indexed) cells and their test points. We know which cells $S(C)$ in $\mathcal{D}_i$ origin from which cell C in $\mathcal{D}_{i-1}$. This implies that a finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_{m+n}$ for F has a structure of a tree representation. The first level of nodes under the root of the tree corresponds to the cells in $\mathcal{D}_1$. The second level of nodes stands for the cells in $\mathcal{D}_2$, i.e., the cylinders over the cells of $\mathbb{R}^1$. The leaves represent the cells of $\mathcal{D}_{m+n}$ (i.e., CAD of $\mathbb{R}^{m+n}$). A test point of the corresponding cell is stored in each node or leaf. To each level of the tree there are a number of projection polynomials F_i whose signs define a cell when evaluated over a test point.

4 A specialized CAD for SDP

4.1 Improving CAD algorithm

Projection phase It is crucially important for the efficiency of CAD construction that the *PROJ* operator produces as small a set of polynomials as possible, while still ensuring the cylindrical arrangement of cells in resulting decomposition. Several improved projection operators have been proposed so far [8, 7, 3].

The complexity of the projection phase is given by the following: given r irreducible polynomials of degree less than or equal to d in N variables, then after $N-1$ projection steps we have $(r \cdot d)^{2^{O(N)}}$ polynomials of degree at most $d^{2^{O(N)}}$. It is often the case that we cannot decrease the number of variables when we reduce the target problem considered to an equivalent first-order formula. Thus, ideally we should produce an equivalent input formula with fewer polynomials, and smaller degree.

Construction phase There are two devices to improve the efficiency of the construction phase:

(1) Avoiding algebraic computation during lifting processes: Revisit the extension of $\mathcal{D}_1$ to $\mathcal{D}_2$. Let C be a cell of $\mathcal{D}_1$ with a test point α. Consider all polynomials $f(u_2) := f(\alpha, u_2) \in F_2$ that are not identically zero. The real roots of $f(u_2)$'s determine the test points of each cell of $\mathbb{R}^2$ in the cylinder over a cell C. Let β be a root of $f(u_2)$. If the test point α is an algebraic number, we need computations over the algebraic extension field $\mathbb{Q}(\alpha)$ for root isolation. Moreover it is required that the test point of each cell be a vector of algebraic numbers over a simple algebraic extension of $\mathbb{Q}$. Hence we need to compute a primitive element γ for $\mathbb{Q}(\alpha, \beta)$ and represent the test point (α, β) as pairs of elements of $\mathbb{Q}(\gamma)$. Computations over an algebraic extension field are typically more expensive than those over $\mathbb{Q}$. For the efficiency of the construction phase, we should consider the possibility of avoiding the use of algebraic test points.

(2) Pruning unnecessary branches of a CAD tree: In general not every cell in the construction is actually necessary for eliminating quantifiers in a given input formula φ. This observation was first made and generalized to *partial CAD* by H. Hong [5]. Partial CAD systematically exploits the logical structure of the input formula. This greatly reduces the number of cells to be considered. Furthermore we can expect to exploit the special structure of the input formula (e.g., convexity) in order to prune unnecessary branches of a CAD tree.

4.2 Exploiting convexity of SDP

The feasibility domain of an SDP is a convex set, and this implies several important properties: (i) the feasible region is a unique connected region. Then (ii) the boundary of the feasible region of the SDP is defined by the determinant polynomial. Moreover, we also assume that a feasible interior point is available (we can remove this assumption, at a slightly higher computational cost). We will use these properties to improve the CAD algorithms in the sequel. The next theorem guarantees (ii):

Theorem 2
The determinant vanishes on the boundary of the domain of positive semidefiniteness of a real symmetric matrix.

Sketch of the proof: All eigenvalues of a symmetric matrix are real and by the principal axis theorem the matrix is positive semidefinite iff all eigenvalues are greater than or equal to zero. Moreover the eigenvalues, as zeros of the characteristic polynomial, depend continuously on the entries of the matrix. Suppose now the values of x_i are such that the matrix is on the boundary of its positive semidefinite domain. Then in every neighborhood of this point there is a point, where the matrix is not positive semidefinite, and hence has a negative eigenvalue. So by the continuity of eigenvalues, the matrix must have a zero as an eigenvalue at this point, and so the determinant vanishes. ∎

(1) improving projection phase As shown in §2, $M(x) \geq_d 0$ can be reduced to an equivalent formula that is the conjunction of $2^n - 1 (\equiv \sum_{r=1}^{n} \binom{n}{r})$ inequalities. Since the boundary of the feasible region of SDP is a part of the determinant polynomial, it is sufficient that we consider, as an input of CAD, a set consisting of an objective polynomial $z - c^T x$ and the determinant $\det(M)$ of M as an input set of CAD to obtain the test point which provides a minimum of the objective function. This greatly improves the efficiency of the projection phase in contrast to the reduction according to Sylvester's criterion.

(2) improving construction phase We construct a CAD for an input set $\{z - c^T x, det(M(x))\}$ in order to solve an SDP given by (2). After $n-1$ recursive projections we have univariate polynomials in z. We denote the set of test points which are real roots of the univariate polynomials in z by T_R, the set of test points taken from the intervals between the roots by T_I.

Since the SDP feasibility domain is convex, the feasible region of z is a unique isolated interval. The endpoints of the feasible interval of z correspond to the maximum and minimum of z, i.e., the objective function. We call the left endpoint a *truth-boundary cell*, which gives the minimum and is contained in T_R. Suppose that we have a feasible interior point $x_0 = (x_1^0, \ldots, x_n^0)$ of SDP domain. This is the same setting as the numerical interior point method for SDP. Then since we have a feasible value $z_0 = c^T x_0$ of z, the feasible interval of z consists of the connected cells to the cell containing z_0. Here we consider only the case $z_0 \notin T_R$ because if $z_0 \in T_R$ then we can regard the test point of the next left interval of z_0 as z_0 and then we can proceed in the same way as shown below.

The test points larger than z_0 are not needed to find the minimum of z. We can denote the test points smaller than z_0 as follows:

$$-\infty < \cdots < s_\ell < r_\ell < s_{\ell-1} < r_{\ell-1} < \cdots < r_2 < s_1 < r_1 < s_0 < r_0 < z_0,$$

where $r_i \in T_R$, $s_i \in T_I$, and let r_ℓ be the the truth-boundary cell, i.e., the minimum of z. In order to find the truth-boundary cell we start with construction of a CAD from over the cell with a test point s_0. If the cell with a test point s_0 has a "true" leaf then we construct a CAD over the cell s_1. In other word we make *depth-first search* of "true" leaf of the CAD tree over s_i's from the right to the left until a certain s_i has no "true" leaf. We can also employ the order of s_i's to be considered according to bisection. Then we obviously have the following proposition:

Proposition 3
There exists an integer ℓ such that $s_{\ell-1}$ has a "true" leaf but s_ℓ has no "true" leaf. Then r_ℓ is the truth-boundary cell.

The coordinate x_{min} which gives the minimum value of z can be obtained by lifting over the truth boundary cell. Note that we do not have to use test poins in T_R to identify which test point is the

Ex.1	level	1	2	3	4	total	time
	Sylv	152	22	7	3	184	250
	A&P	7	5	3	2	17	10

Ex.2	level	1	2	3	4	total	time
	Sylv	160	18	7	4	189	200
	A&P	37	9	4	3	53	70

Ex.3	level	1	2	3	4	5	6	total	time
	Sylv	–	2879	88	14	8	5	*1 2995	*2 169250
	A&P	57	58	12	5	3	2	137	1240

Table 1: Computational results for projection phase

truth-boundary cell. As for the lifting at the level of the tree corresponding to x_i, we can also prune unnecessary branches since the feasible region consists of the connected cells to the cell with a test point x_i^0. Thus we can ignore outer both sides of the feasible region.

5 Examples

We have examined how our improvements work for the following SDP problems by using QEPCAD and maple for projection phase and for construction phase, respectively. [2)]

Example 1. A feasible interior point is: $(a, b, c) = (0, 3, 0)$.

$$\text{objective: } a+b+c, \qquad \text{s.t.} \begin{bmatrix} 1 & a & 3-b \\ a & 5 & c \\ 3-b & c & 9 \end{bmatrix} \geq 0.$$

Example 2. A feasible interior point is: $(a, b, c) = (1, -1, 1)$.

$$\text{objective: } a+2b+3c, \qquad \text{s.t.} \begin{bmatrix} a & b \\ b & c+1 \end{bmatrix} \geq 0, \quad \begin{bmatrix} 1 & b+c \\ b+c & 2-a \end{bmatrix} \geq 0.$$

Example 3. This example arises from the minimization of a symmetric quartic polynomial. A feasible interior point is: $(\gamma, a, b, c, d) = (60, -1, 12, 2, 4)$.

$$\text{objective: } \gamma, \qquad \text{s.t.} \begin{bmatrix} \gamma & a & b \\ a & 1 & c \\ b & c & 2d \end{bmatrix} \geq 0, \quad 1-2a \geq 0, \quad \begin{bmatrix} 2b & d \\ d & 1 \end{bmatrix} \geq 0, \quad 2c-3 \geq 0.$$

Projection phase: Table 1 shows the number of projection factors appearing at each level of the CAD tree and total time to accomplish the projection phase for the above three examples. Here, *1 means the time until level 2 and *2 means the time until QEPCAD halted on computing level 1. "Sylv" uses Sylvester's criterion to reduce an SDP to a first-order formula and "A&P" is our approach shown in the previous section. Our approach performs much better than the approach using Sylvester criterion.

Construction phase: We applied QEPCAD for the construction phase of the above SDP problems to get the minimum of objective functions (with the option `+N50000000`; a size for SACLIB's garbage collected array. The default value is `+N2000000`). However, QEPCAD halted due to lack of memory with a message "`Too few cells reclaimed`" for all examples. Unfortunately we have not yet finished the implementation of our proposed method. We manually applied the strategy for choosing test points for the above examples on maple. Then we were able to solve the construction phases for all examples in a few minutes.

[2)] All the computations are executed on a PC with a CPU Pentium III 1GHz and 756 MB memory.

6 Conclusions

We have proposed an efficient algorithm to compute an exact (algebraic) representations of the solution of SDP problems. Our approach can be regarded as a specialized CAD algorithm for partially exploiting the convexity of SDP, and as a successful attempt of fusing symbolic and numeric approaches to achieve efficiency. We hope this work leads to further generalizations of symbolic approaches exploiting convexity to other related problems.

A maple-implementation of the method proposed here on top of the SyNRAC-package [1] is planned.

Acknowledgements

The authors would like to thank C. W. Brown for his invaluable help. The authors would like to thank K. Yokoyama, S. Hara, and V. Weispfenning for fruitful discussions.

References

[1] H. Anai and H. Yanami. SyNRAC: A maple-package for solving real algebraic constraints. In *Proceedings of the International Workshop on Computer Algebra Systems and Their Applications: CASA'2003.* To appear in the series LNCS, Springer-Verlag.

[2] S. Boyd, L. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM Studies in Applied Mathematics, vol 15. SIAM, 1994.

[3] C.W. Brown. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447–465, 2001.

[4] G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition, LNCS 32. Springer Verlag, 1975.

[5] G.E.Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299-328, Sept. 1991.

[6] P. Gahinet, A. Nemirovski, A.J. Laub, and M. Chilali. *LMI Control Toolbox User's Guide, For Use with MATLAB.* The MATH WORKS INC, 1995.

[7] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation*, pages 261–264, 1990.

[8] S. McCallum. An improved projection operation for cylindrical algebraic decomposition of three-dimensional space. *Journal of Symbolic Computation*, 5(1-2):141–161, Feb.–Apr. 1988.

[9] P. A. Parrilo, Semidefinite programming relaxations for semialgebraic problems, *Math. Prog. Ser. B*, to appear, 2003.

[10] V. Weispfenning. Simulation and optimization by quantifier elimination. *J. Symbolic Computation*, 24(2):189–208, 1997.