Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

New Developments in Sum of Squares Optimization and SOSTOOLS

Stephen Prajna   Antonis Papachristodoulou   Peter Seiler   Pablo A. Parrilo

***Abstract*— We describe the latest additions to SOSTOOLS, a freely available MATLAB toolbox for formulating and solving sum of squares programs. Among the many improvements, there are native polynomial objects, structure-exploiting techniques for sparse and structured polynomials, new customized functions, and support for alternative SDP solvers. We sketch some of the theory behind the new improvements, and illustrate the new commands using control-oriented examples.**

I. INTRODUCTION

In this paper we describe recent developments of SOSTOOLS [1], [2], a free, third-party MATLAB[1] toolbox for solving sum of squares programs. The functions implemented in SOSTOOLS are based on the sum of squares decomposition of multivariate polynomials [3], which can be efficiently computed using semidefinite programming [4]. SOSTOOLS was the result of the recent interest in sum of squares polynomials [5], [3], [6], [7], [8], [9], [10], partly due to the fact that these techniques provide convex relaxations for many computationally hard problems such as global, constrained, and Boolean optimization.

In addition to the optimization problems mentioned above, sum of squares polynomials (and hence SOSTOOLS) find direct applications in many control theory problems, such as nonlinear stability analysis [7], [11], [12], robustness analysis [7], [11], [12], nonlinear synthesis [13], and model validation [14]. Some other areas in which SOSTOOLS is applicable are geometric theorem proving [15] and quantum information theory [16].

This paper covers in detail recent improvements and additions to SOSTOOLS, the new version of which was released in mid 2004. Besides the implementation of new features, customized special purpose functions and a number of new application examples have been added in the new version. Examples will be presented throughout the paper to illustrate the use of these new features, the motivation and reasons for using them, as well as the advantages they provide.

The structure of the paper is as follows: in Section II we describe the new polynomial objects available in SOSTOOLS. In Sections III and IV, the improvements in the SOS formulation for structured and multipartite polynomials are presented, including the connections with the so-called SOS matrices. We describe next a few additional customized functions, such as a version of `findbound` for multivariate polynomial optimization, followed by Section VI, where we detail other additions such as the possibility of using the SDP solver SDPT3 and the computation of rational solutions. Finally, in Section VII we present our conclusions and outline future developments.

II. POLYNOMIAL OBJECTS

In the original SOSTOOLS release, polynomials were represented in MATLAB solely as symbolic objects, using the Symbolic Math Toolbox (which is essentially an interface to the Maple kernel). While on the one hand this gives the user the possibility of utilizing all the powerful routines in MATLAB's Extended Symbolic Toolbox and its Maple library, at the same time it prohibits those without access to the Symbolic Toolbox (such as those using the student edition of MATLAB) from using SOSTOOLS. In the new SOSTOOLS release, the user now has the option of using an alternative custom-built polynomial object, along with some basic polynomial manipulation methods to represent and manipulate polynomials.

For this, we have integrated the Multivariate Polynomial Toolbox, a freely available toolbox for constructing and manipulating multivariate polynomials. In the remainder of the section, we give a brief introduction to the new polynomial objects in SOSTOOLS.

Polynomial variables are created with the `pvar` command. For example, the following command creates three variables:

```
>> pvar x1 x2 x3
```

New polynomial objects can now be created from these variables, and manipulated using standard addition, multiplication, and integer exponentiation functions:

```
>> p = x3ˆ4+5*x2+x1ˆ2
```

S. Prajna and A. Papachristodoulou are with the Control and Dynamical Systems Dept., California Institute of Technology, Pasadena, CA 91125, USA.
P. Seiler is with the Mechanical and Industrial Engineering Dept., University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.
P. A. Parrilo is with the Automatic Control Laboratory, Swiss Federal Institute of Technology, CH-8092 Zürich, Switzerland.

[1]A registered trademark of The MathWorks, Inc.

```
p =
  x3^4 + 5*x2 + x1^2
```

Matrices of polynomials can be created from polynomials using horizontal/vertical concatenation and block diagonal augmentation, e.g.:

```
>> M1 = blkdiag(p,2*x2)
M1 =
  [ x3^4 + 5*x2 + x1^2 ,    0 ]
  [                  0 , 2*x2 ]
```

Naturally, it is also possible to build new polynomial matrices from already constructed submatrices. Elements of a polynomial matrix can be referenced and assigned using the standard MATLAB referencing notation:

```
>> M1(1,2)=x1*x2
M1 =
  [ x3^4 + 5*x2 + x1^2 , x1*x2 ]
  [                  0 ,  2*x2 ]
```

The internal data structure for an $N \times M$ polynomial matrix of V variables and T terms consists of a $T \times NM$ sparse coefficient matrix, a $T \times V$ degree matrix, and a $V \times 1$ cell array of variable names. This information can be easily accessed through the MATLAB field accessing operators: `p.coefficient`, `p.degmat`, and `p.varname`. The access to fields uses a case insensitive, partial-match. Thus abbreviations, such as `p.coef`, can also be used to obtain the coefficients, degrees, and variable names. A few additional operations exist in this initial version of the toolbox such as trace, transpose, determinant, differentiation, logical equal and logical not equal.

The input to the SOSTOOLS commands can be specified using either the Maple objects or the new MPT objects. There are some minor variations in performance depending on the degree/number of variables of the polynomials, due the fact that the new implementation always keeps an expanded internal representation, but for most reasonable-sized problems the difference is minimal.

III. Polynomial structure

For a polynomial $p(x)$, the complexity of computing the sum of squares decomposition $p(x) = \sum_i p_i^2(x)$ (or equivalently, $p(x) = Z(x)^T Q Z(x)$, where $Z(x)$ is a vector of monomials — see [7] for details) depends on two factors: the number of variables and the degree of the polynomial. However when $p(x)$ has special structural properties, the computation effort can be notably simplified through the reduction of the size of the semidefinite program, removal of degeneracies, and better numerical conditioning. Since the initial version of SOSTOOLS, Newton polytopes techniques have been available via the optional argument `'sparse'` to the function `sosineq`. In the new release, we have improved our support for structured polynomials, with two kinds of structures being automatically exploited: sparsity and bipartite structure.

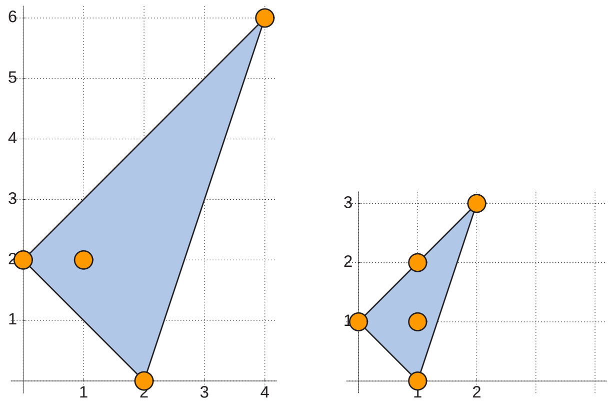

Fig. 1. Newton polytope for the polynomial in Example 1 (left), and possible monomials in its SOS decomposition (right).

The first type of simplification can be performed when $p(x)$ is sparse. The notion of sparseness for multivariate polynomials is stronger than the one commonly used for matrices. While in the matrix case this word usually means that many coefficients are zero, in the polynomial case the specific vanishing pattern is also taken into account. This idea is formalized by using the *Newton polytope* [17], defined as the convex hull of the set of exponents, considered as vectors in $\mathbb{R}^n$. It was shown by Reznick in [18] that $Z(x)$ need only contain monomials whose squared degrees are contained in the convex hull of the degrees of monomials in $p(x)$. Consequently, for sparse $p(x)$ the size of the vector $Z(x)$ and matrix Q appearing in the sum of squares decomposition can be reduced which results in a decrease of the size of the semidefinite program.

Example 1: This example is taken from [19]. Consider the polynomial $p(x, y) = 4x^4y^6 + x^2 - xy^2 + y^2$. Its Newton polytope is a triangle, being the convex hull of the points $(4, 6), (2, 0), (1, 2), (2, 0)$; see Figure 1. By the result mentioned above, we can always find a SOS decomposition that contains only the monomials $(1, 0), (0, 1), (1, 1), (1, 2), (2, 3)$. By exploiting sparsity, non-negativity of $p(x, y)$ can thus be verified by solving a semidefinite program of size 5×5 with 13 constraints. On the other hand, when sparsity is not exploited, we need to solve a 11×11 semidefinite program with 32 constraints.

SOSTOOLS takes the sparse structure into account, and computes an appropriate set of monomials for the sum of squares decompositions. The convex hulls are computed using either the native MATLAB command `convhulln` (which is based on the software QHULL), or the specialized external package CDD [20], developed by K. Fukuda. Special care is taken with the case when the set of exponents has lower affine dimension than the number of variables (this case occurs for instance for homogeneous polynomials, where the sum of the degrees is equal to a constant), in which case a projection to a lower dimensional space is performed prior to the convex hull computation.

IV. Multipartite polynomials

Exploiting the structure of polynomials for which SOS decompositions are being sought can reduce significantly the computational burden and result into better conditioned SDPs. In this section we concentrate on a particular structure of polynomials that appears frequently in robust control theory when considering, for instance, Lyapunov function analysis for linear systems with parametric uncertainty (see Example 6). In this case, the indeterminates that appear in the Lyapunov conditions are Kronecker products of parameters (zeroth order and higher) and state variables (second order). This special structure should be taken into account when constructing the vector $Z(x)$ used in the sum of squares decomposition $p(x) = Z(x)^T Q Z(x)$. Let us first define what we mean by a *multipartite* polynomial.

Definition 2: A polynomial $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ in $\sum_{i=1}^{n} m_i$ indeterminates, where $\mathbf{x}_i = [x_{i1}, \ldots, x_{im_i}]$ given by

$$p(\mathbf{x}) = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}_1^{\boldsymbol{\alpha}_1} \mathbf{x}_2^{\boldsymbol{\alpha}_2} \cdots \mathbf{x}_n^{\boldsymbol{\alpha}_n}$$

is termed *multipartite* if for all $i \geq 2$, $\sum_{k=1}^{m_i} \boldsymbol{\alpha}_{ik}$ is constant, i.e. the monomials in all but one partition are of *homogeneous* order.

In other words, a multipartite polynomial is homogenous when fixing any $(n-1)$ blocks of variables, always including the first block. This special structure of $p(\mathbf{x})$ can be taken into account through its Newton polytope. It has been argued in an earlier section that when considering the SOS decomposition of a sparse polynomial (in which many of the coefficients $c_{\boldsymbol{\alpha}}$ are zero), the nonzero monomials in $Z(\mathbf{x}) = [\mathbf{x}^{\boldsymbol{\beta}}]$ are the ones for which $2\boldsymbol{\beta}$ belongs to the convex hull of the degrees $\boldsymbol{\alpha}$ [18]. What distinguishes this case from the general one, is that the Newton polytope of $p(\mathbf{x})$ is the *Cartesian product* of the individual Newton polytopes corresponding to the blocks of variables. Hence, the convex hull should only be computed for the individual $\boldsymbol{\alpha}_i$, which significantly reduces the complexity and avoids ill-conditioning in the computation of a degenerate convex hull in a higher dimensional space.

A specific kind of multipartite polynomials important in practice is the one that appears when considering *sum of squares matrices*. These are matrices with polynomial entries that are positive semi-definite for every value of the indeterminates.

Definition 3: Let $S \in \mathbb{R}[\mathbf{x}]^{m \times m}$ be a symmetric matrix, and $\mathbf{y} = [y_1, \ldots, y_m]$ be new indeterminates. The matrix S is a *sum of squares (SOS) matrix* if the bipartite scalar polynomial $\mathbf{y}^T S \mathbf{y}$ is a sum of squares in $\mathbb{R}[\mathbf{x}, \mathbf{y}]$.

Let us give an example.

Example 4: [21] Consider the matrix $S \in \mathbb{R}[x]^{2\times 2}$ given by:

$$S = \begin{bmatrix} x^2 - 2x + 2 & x \\ x & x^2 \end{bmatrix}.$$

This is a SOS matrix, since

$$\begin{aligned} \mathbf{y}^T S \mathbf{y} &= \begin{bmatrix} y_1 \\ xy_1 \\ y_2 \\ xy_2 \end{bmatrix}^T \begin{bmatrix} 2 & -1 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ xy_1 \\ y_2 \\ xy_2 \end{bmatrix} \\ &= (y_1 + xy_2)^2 + (xy_1 - y_1)^2. \end{aligned}$$

Note that SOS matrices for which $n = 1$, i.e. $S \in \mathbb{R}[x]^{m\times m}$, are positive semidefinite for all real x if and only if they are SOS matrices; this is because the resulting polynomial will be second order in $\mathbf{y}$ and will only contain one variable x; the resulting positive semidefinite biform is always a sum of squares [22].

In this manner a SOS matrix in several variables can be converted to a SOS polynomial, whose decomposition is computed using semidefinite programming. Because of the bipartite structure, only monomials in the form $x_i^k y_j$ will appear in the vector Z, as mentioned earlier.

The new release of SOSTOOLS handles this special case of polynomials as follows. When an SOS condition of the above structure is added to an existing `sosprogram`, the various parts of the multipartite polynomial $p(\mathbf{x})$ are declared:

```
>> prog = sosineq(prog,p,...
           'sparsemultipartite',...
           {[x1,x2,x3],[y1,y2,y3]});
```

where inside the curly brackets are the sets of variables that form the multipartite polynomial.

In particular, the polynomial in Example 4 can be handled as follows:

```
>> syms x y1 y2 real;
>> S = [x^2-2*x+2 , x ; x, x^2];
>> y = [y1 ; y2];
>> p = y' * S * y ;
>> prog = sosprogram([x,y1,y2]);
>> prog = sosineq(prog,p,...
      'sparsemultipartite',{[x],[y1,y2]});
>> prog = sossolve(prog);
```

Future versions of `sosineq` will handle directly SOS matrix constraints.

(m,n,d)	Without multipartite option	With multipartite option
$(3,2,2)$	15×15, 90 constraints	9×9, 36 constraints
$(4,2,2)$	21×21, 161 constraints	12×12, 60 constraints
$(3,3,2)$	21×21, 161 constraints	12×12, 60 constraints
$(4,3,2)$	28×28, 266 constraints	16×16, 100 constraints
$(3,2,4)$	35×35, 279 constraints	18×18, 90 constraints
$(4,2,4)$	53×53, 573 constraints	24×24, 150 constraints
$(3,3,4)$	59×59, 647 constraints	30×30, 210 constraints
$(4,3,4)$	84×84, 1210 constraints	40×40, 350 constraints

TABLE I

SIZE OF THE SEMIDEFINITE PROGRAMS IN EXAMPLE 5.

Example 5: To illustrate the benefit of using the multipartite option, consider the problem of checking whether a polynomial matrix inequality

$$F(\mathbf{x}) = F^T(\mathbf{x}) \succeq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n$$

holds, where $F \in \mathbb{R}[\mathbf{x}]^{m\times m}$. A sufficient test for positive semidefiniteness of $F(\mathbf{x})$ is obtained by showing that the bipartite polynomial $\mathbf{y}^T F(\mathbf{x})\mathbf{y}$ is a sum of squares (equivalently, showing that $F(\mathbf{x})$ is a SOS matrix). We denote the degree of F by d. For various values of (m,n,d), the sizes of the resulting semidefinite programs are depicted in Table I.

We conclude this section with a control-oriented example that illustrates an application of multipartite polynomials in stability analysis.

Example 6: Consider the following system:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -p_1 & 1 & -1 \\ 2-2p_2 & 2 & -1 \\ 3 & 1 & -p_1p_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where p_1 and p_2 are parameters. The region in the parameter space (p_1,p_2) for which stability is retained is shown in Figure 2. Nominal operating conditions for this system are $p_1 \in [1.7, 3.7]$ and $p_2 \in [5, 9]$. We capture this parameter set by constructing two inequalities:

$$\begin{aligned} a_1 &\triangleq (p_1 - 2.7 + \kappa_1)(p_1 - 2.7 - \kappa_1) \le 0 \\ a_2 &\triangleq (p_2 - 7 + \kappa_2)(p_1 - 7 - \kappa_2) \le 0, \end{aligned}$$

where κ_1 and κ_2 are parameters. We want to check the stability of this system by constructing a Lyapunov function. We look for a certificate of the form $V(x;p)$ that is bipartite: quadratic in the state x and any order in p. The two Lyapunov conditions can be converted into sufficient SOS conditions as follows:

$$V(x;p) - \epsilon_1\|x\|^2 + \sum_{i=1}^{2} q_{1i}(x;p)a_i \text{ is SOS},$$

$$-\dot{V}(x;p) - \epsilon_2\|x\|^2 + \sum_{i=1}^{2} q_{2i}(x;p)a_i \text{ is SOS},$$

where $\epsilon_i \ge 0.1$ and q_{1i} and q_{2i} are bipartite sums of squares, quadratic in x and of appropriate order in p.

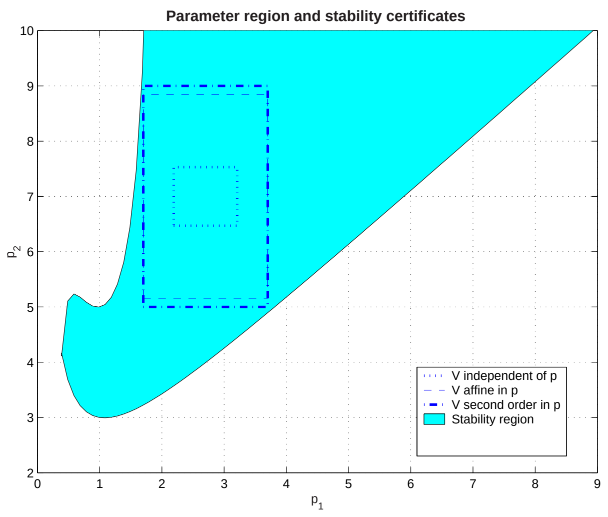

Fig. 2. The full stability region (shaded) and the regions for which stability can be proven by constructing bipartite Lyapunov functions. Bigger regions require higher order certificates, which nonetheless can be easily computed because of their structure.

When the Lyapunov function is not a function of p, we can prove stability for $\kappa_1 = 0.51$ and $\kappa_2 = 0.53$. When V is affine in p, then we can prove stability for $\kappa_1 = 1$ and $\kappa_2 = 1.84$. When it is quadratic in p, we can prove stability for the full region of interest, i.e. $\kappa_1 = 1$ and $\kappa_2 = 2$. In this case if the bipartite structure of the conditions is not taken into account then the dimension of the vector Z required to represent a non-structured $V(x;p)$ is 279; taking into account the bipartite structure this number is reduced to 90.

V. CONSTRAINED OPTIMIZATION

Besides its general facilities for SOS optimization, SOSTOOLS includes several "ready-made" customized functions, that solve specific problems directly, by internally reformulating them as SOS programs. In the new version, these customized functions have been updated and several new capabilities have been added. For instance, the customized function `findbound`, which previously could only handle unconstrained global polynomial optimization problems, can now be used to solve constrained polynomial optimization problems of the form:

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g_i(x) \ge 0, \quad i = 1,...,M \\ &\qquad\qquad h_j(x) = 0, \quad j = 1,...,N. \end{aligned}$$

A lower bound for $f(x)$ can be computed using Positivstellensatz-based relaxations. Assume that there exists a set of sums of squares $\sigma_j(x)$'s, and a set of polynomials $\lambda_i(x)$'s, such that

$$f(x) - \gamma = \sigma_0(x) + \sum_j \lambda_j(x) h_j(x) + \sum_i \sigma_i(x) g_i(x) + \\ + \sum_{i_1,i_2} \sigma_{i_1,i_2}(x) g_{i_1}(x) g_{i_2}(x) + \cdots \quad (1)$$

then it follows that γ is a lower bound for the constrained optimization problem stated above. This specific kind of representation corresponds to Schmüdgen's theorem [23]. By maximizing γ, we can obtain a lower bound that becomes increasingly tighter as the degree of the expression (1) is increased.

As an example, consider the problem of minimizing $x_1 + x_2$, subject to $x_1 \geq 0$, $x_2 \geq 0.5$, $x_1^2 + x_2^2 = 1, x_2 - x_1^2 - 0.5 = 0$. A lower bound for this problem can be computed using SOSTOOLS as follows:

```
>> syms x1 x2;
>> degree = 4;
>> [gam,vars,opt] = findbound(x1+x2,
      [x1, x2-0.5],
      [x1^2+x2^2-1, x2-x1^2-0.5],
      degree);
```

In the above command, `degree` is the desired degree for the expression (1). The function `findbound` will automatically form the products $g_{i_1}(x)g_{i_2}(x)$, $g_{i_1}(x)g_{i_2}(x)g_{i_3}(x)$ and so on; and then construct the sum of squares and polynomial multiplier $\sigma(x)$'s, $\lambda(x)$'s, such that the degree of the whole expression is no greater than `degree`. For this example, a lower bound of the optimization problem is `gam`$= 1.3911$ corresponding to the optimal solution $x_1 = 0.5682$, $x_2 = 0.8229$, which can be extracted from the output argument `opt`.

VI. Additional changes

Several other minor improvements to SOSTOOLS have been added for the current release. Some of them are described below.

SDPT3 support: Besides SeDuMi [24], SOSTOOLS now additionally supports the SDP software package SDPT3 [25], written by Toh, Tütüncü and Todd. SDPT3 is a robust high-quality solver that runs under MATLAB, and uses an infeasible path-following algorithm. It is also quite fast, and for certain instances it is considerably faster than SeDuMi.

Rational solutions: For certain applications, it is particularly important to ensure that the SOS decomposition found numerically by SDP methods actually corresponds to a true solution, and is not the result of roundoff errors. This is specially true in the case of ill-conditioned problems, since SDP solvers can sometimes produce in this case unreliable results. There are several ways of doing this, for instance using backwards error analysis, or by computing rational solutions, that we can fully verify symbolically. Towards this end, we have incorporated an experimental option to round to rational numbers a candidate floating point SDP solution, in such a way to produce an exact SOS representation of the input polynomial (which should have integer or rational coefficients). The procedure will succeed if the computed solution is "well-centered," far away from the boundary of the feasible set; the details of the rounding procedure will be explained elsewhere.

Currently, this facility is available only through the customized function `findsos`, by giving an additional input argument `'rational'`. On future releases, we may extend this to more general SOS program formulations. We illustrate its usage below.

Example 7: Consider again the polynomial from Example 1. Running

```
>> P = 4*x^4*y^6+x^2-x*y^2+y^2;
>> [Q,Z]=findsos(P,'rational');
```

we obtain the rational represention for $p(x, y)$ given by

$$\begin{bmatrix} y \\ x \\ xy \\ xy^2 \\ x^2y^3 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & -\frac{1}{2} & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 \\ -\frac{1}{2} & 0 & 2 & 0 & 0 \\ 0 & -1 & 0 & 2 & 0 \\ -1 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} y \\ x \\ xy \\ xy^2 \\ x^2y^3 \end{bmatrix}^T,$$

where the matrix is given by `Q`, and `Z` is the vector of monomials. Notice that the monomials in `Z` are exactly the ones computed earlier in Example 1.

VII. Future improvements and conclusions

Future improvements to SOSTOOLS, already partially implemented, will incorporate symmetry reduction and SOS over quotients. These are described briefly below.

Symmetries: Another kind of simplification can be applied when there is a discrete symmetry in the problem, i.e., when the polynomial $p(x)$ is invariant under the action of a finite group on x. In this case, linear representation theory [26] can be used to find a coordinate change which transforms the semidefinite program associated with the computation of a sum of squares decomposition into a set of coupled smaller semidefinite programs [21]. An alternative viewpoint also developed in [21] is based on invariant theory [27]. In this approach, the computation of a sum of squares decomposition is performed using only elements in the invariant ring, which also has a side benefit since in many cases symmetric problems are expressed already in terms of the invariants.

Equality constraints: One application of sum of squares programming is combinatorial optimization, for example 0–1 programming. For such problems and many others, the underlying algebraic variety is zero-dimensional, as the independent variable x has to satisfy n given equalities of the form $h_i(x) = x_i(x_i - 1) = 0$, for $i = 1, ..., n$. Then the sum of squares expressions can be replaced by their remainders modulo the ideal I generated by the $h_i(x)$'s [28]. This also reduces the size of the semidefinite program, as the sum of squares expressions are now elements in the quotient ring $\mathbb{R}[x]/I$, leading to a reduction in the number of decision variables.

Conclusions

All the features described in the previous sections are already implemented in SOSTOOLS. This provides a friendly user interface to SOS programs, allowing the use of native polynomial objects. Numerous examples of the new SOSTOOLS facilities are distributed as demonstration files. The benefits of using the new features are apparent from these examples, such as the notable gains in performance and reliability achieved by taking into account the polynomial structure. More importantly, the resulting significant reduction on the size of the semidefinite programs makes it possible to handle problems that are otherwise too large to solve using current state-of-the-art semidefinite programming solvers.

References

[1] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing SOSTOOLS: A general purpose sum of squares programming solver," in *Proceedings IEEE Conference on Decision and Control*, 2002.

[2] ——, "SOSTOOLS – Sum of Squares Optimization Toolbox, User's Guide," 2002, available at `http://www.cds.caltech.edu/sostools` and `http://control.ee.ethz.ch/~parrilo/sostools`.

[3] M. D. Choi, T. Y. Lam, and B. Reznick, "Sum of squares of real polynomials," *Proceedings of Symposia in Pure Mathematics*, vol. 58, no. 2, pp. 103–126, 1995.

[4] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[5] N. Z. Shor, "Class of global minimum bounds of polynomial functions," *Cybernetics*, vol. 23, no. 6, pp. 731–734, 1987.

[6] B. Reznick, "Some concrete aspects of Hilbert's 17th problem," in *Contemporary Mathematics*, vol. 253. American Mathematical Society, 2000, pp. 251–272.

[7] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 2000.

[8] ——, "Semidefinite programming relaxations for semialgebraic problems," *Mathematical Programming Series B*, vol. 96, no. 2, pp. 293–320, 2003.

[9] Y. Nesterov, "Squared functional systems and optimization problems," in *High Performance Optimization*, J. Frenk, C. Roos, T. Terlaky, and S. Zhang, Eds. Kluwer Academic Publishers, 2000, pp. 405–440.

[10] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM J. Optim.*, vol. 11, no. 3, pp. 796–817, 2001.

[11] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in *Proceedings IEEE Conference on Decision and Control*, 2002.

[12] S. Prajna and A. Papachristodoulou, "Analysis of switched and hybrid systems – beyond piecewise quadratic methods," in *Proceedings of the American Control Conference*, 2003.

[13] S. Prajna, P. A. Parrilo, and A. Rantzer, "Nonlinear control synthesis by convex optimization," *IEEE Transactions on Automatic Control*, vol. 49, no. 2, pp. 310–314, 2004.

[14] S. Prajna, "Barrier certificates for nonlinear model validation," in *Proceedings of the 42^{th} IEEE Conference on Decision and Control*, 2003, pp. 2884–2889.

[15] P. A. Parrilo and R. Peretz, "An inequality for circle packings proved by semidefinite programming," *Discrete and Computational Geometry*, vol. 31, no. 3, pp. 357–367, 2004.

[16] A. C. Doherty, P. A. Parrilo, and F. M. Spedalieri, "Distinguishing separable and entangled states," *Physical Review Letters*, vol. 88, no. 18, 2002.

[17] B. Sturmfels, "Polynomial equations and convex polytopes," *American Mathematical Monthly*, vol. 105, no. 10, pp. 907–922, 1998.

[18] B. Reznick, "Extremal PSD forms with few terms," *Duke Mathematical Journal*, vol. 45, no. 2, pp. 363–374, 1978.

[19] P. A. Parrilo and S. Lall, "Semidefinite programming relaxations and algebraic optimization in Control," Dec. 2003, lecture notes for the CDC 2003 workshop. Available at `http://control.ee.ethz.ch/~parrilo/cdc03_workshop/`.

[20] K. Fukuda, *CDD/CDD+ reference manual*, 2003, Institute for Operations Research, Swiss Federal Institute of Technology, Lausanne and Zürich, Switzerland. Program available at `http://www.ifor.math.ethz.ch/staff/fukuda`.

[21] K. Gatermann and P. A. Parrilo, "Symmetry groups, semidefinite programs, and sums of squares," 2002, *J. of Pure and Applied Algebra*, to appear. Available at `http://control.ee.ethz.ch/~parrilo/pubs/`.

[22] M.-D. Choi, T.-Y. Lam, and B. Reznick, "Real zeros of positive semidefinite forms. I," *Math. Z.*, vol. 171, no. 1, pp. 1–26, 1980.

[23] K. Schmüdgen, "The k-moment problem for compact semialgebraic sets," *Math. Ann.*, vol. 289, pp. 203–206, 1991.

[24] J. Sturm, *SeDuMi version 1.05*, Oct. 2001, available from `http://fewcal.uvt.nl/sturm/software/sedumi.html`.

[25] K. C. Toh, R. H. Tütüncü, and M. J. Todd, *SDPT3 - a MATLAB software package for semidefinite-quadratic-linear programming*, available from `http://www.math.cmu.edu/~reha/sdpt3.html`.

[26] A. Fässler and E. Stiefel, *Group Theoretical Methods and Their Applications*. Birkhäuser, 1992.

[27] B. Sturmfels, *Algorithms in Invariant Theory*. Springer-Verlag, 1993.

[28] P. A. Parrilo, "An explicit construction of distinguished representations of polynomials nonnegative over finite sets," 2002, ifA Technical Report AUT02-02, available at `http://control.ee.ethz.ch/~parrilo/pubs/`.