RC25454 (WAT1403-067) March 24, 2014
Mathematics

IBM Research Report

Control and Verification of High-Dimensional Systems with DSOS and SDSOS Programming

Anirudha Majumdar[1], Amir Ali Ahmadi[2], Russ Tedrake[1]

[1]MIT

[2]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
USA

Research Division
Almaden – Austin – Beijing – Cambridge – Dublin - Haifa – India – Melbourne - T.J. Watson – Tokyo - Zurich

Control and Verification of High-Dimensional Systems with DSOS and SDSOS Programming

Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake

***Abstract*— In this paper, we consider linear programming (LP) and second order cone programming (SOCP) based alternatives to sum of squares (SOS) programming and apply this framework to high-dimensional problems arising in control applications. Despite the wide acceptance of SOS programming in the control and optimization communities, scalability has been a key challenge due to its reliance on semidefinite programming (SDP) as its main computational engine. While SDPs have many appealing features, current SDP solvers do not approach the scalability or numerical maturity of LP and SOCP solvers. Our approach is based on the recent work of Ahmadi and Majumdar [1], which replaces the positive semidefiniteness constraint inherent in the SOS approach with stronger conditions based on *diagonal dominance* and *scaled diagonal dominance*. This leads to the *DSOS* and *SDSOS* cones of polynomials, which can be optimized over using LP and SOCP respectively. We demonstrate this approach on four high dimensional control problems that are currently well beyond the reach of SOS programming: computing a region of attraction for a 22 dimensional system, analysis of a 50 node network of oscillators, searching for degree 3 controllers and degree 8 Lyapunov functions for an Acrobot system (with the resulting controller validated on a hardware platform), and a balancing controller for a 30 state and 14 control input model of the ATLAS humanoid robot. While there is additional conservatism introduced by our approach, extensive numerical experiments on smaller instances of our problems demonstrate that this conservatism can be small compared to SOS programming.**

I. Introduction

Sum of squares (SOS) programming has had a large impact on the control community since its advent over a decade ago [2]. These techniques have been used to tackle a wide variety of problems including feedback control synthesis, safety verification and computation of regions of attraction, invariant sets, and reachable sets for a broad class of nonlinear and hybrid systems [3]–[8]. The key observation behind the approach is that many questions in control theory can be posed as checks on nonnegativity of functions. However, checking nonnegativity is NP-hard even when the functions are restricted to the set of polynomials [9]. The SOS relaxation relies on the ability to efficiently check if a polynomial can be expressed as a sum of squares of other polynomials. This search for a sum of squares decomposition can be cast as a semidefinite program (SDP) and solved using numerical tools from convex optimization.

Despite the wide acceptance of these approaches in the control and optimization communities, applications of these methods considered in the literature typically involve systems of very modest dimension (approximately 5-10 states) This has limited the use of SOS methods in domains such as robotics, aircraft control, network control and power systems, where the dimensionality of systems are typically much higher. While control systems of higher dimension have been addressed using SOS programming in certain cases, they involve exploiting special structure (e.g., symmetry, sparsity) of the particular problem under consideration [10]–[13]. For many real-world control applications, we would like to be able to handle problems of high dimensionality even when such structure is limited or not available. Further, even for smaller problems, being able to obtain an answer much more quickly than we currently can (perhaps at the cost of conservatism) would be of significant practical utility.

The limited scalability of the SOS approach is due in part to the fact that, in general, SDPs are among the most expensive convex relaxations. At the current state of solver technology, it is not uncommon for the more practically-oriented user to want to avoid SDP-based approaches. For example, in the industry-motivated field of integer programming, the cutting-plane approaches used on real-life problems are almost exclusively based on linear programming (LP) or second order cone programming (SOCP) [14], [15]. Even though semidefinite cuts are known to be stronger, they are too expensive to be used even at the root node of branch-and-bound techniques for integer programming. In the field

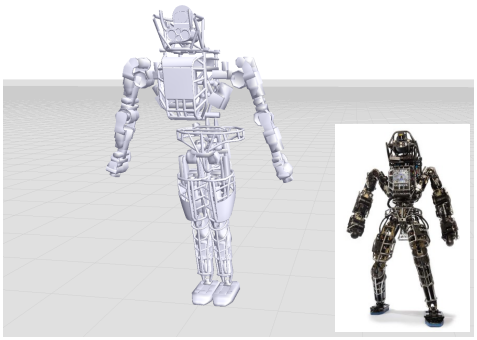


Fig. 1: The methods presented in this paper allow us to handle problems of dimensionality well beyond the reach of current SOS programming based approaches. For example, in Section V-D, we design a balancing controller for a 30 state and 14 control input model of the ATLAS humanoid robot. A visualization of the model is shown in this figure, along with the hardware platform (inset) on which the parameters of the system are based. (Picture of robot reproduced with permission from Boston Dynamics.)

Anirudha Majumdar and Russ Tedrake are with the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology. Email: {anirudha,russt}@mit.edu. Amir Ali Ahmadi is a Goldstine Fellow at the Department of Analytics and Mathematical Sciences of the IBM Watson Research Center; Email: aaa@us.ibm.com.

of SOS optimization, however, a sound alternative to SOS programming that can avoid SDP and take advantage of the existing mature and high-performance LP/SOCP solvers is lacking. This is precisely what we are after here.

In very recent work [1], Ahmadi and Majumdar provide new sufficient conditions for polynomial nonnegativity based on linear programming (LP) and second-order cone programming (SOCP) relaxations. The key insight is to replace the positive semidefiniteness constraint on the Gram matrix in the SOS approach with stronger conditions: *diagonal dominance* and *scaled diagonal dominance*. Equivalently, the set of polynomials that is being searched over is restricted to a subclass of sum of squares polynomials: the *Diagonally-Dominant SOS* (DSOS) and *Scaled-Diagonally-Dominant SOS* (SDSOS) polynomials (see Definition 3). This results in significantly cheaper optimization problems, though with more conservative solutions in general. By leveraging the scalability of available LP/SOCP solvers, we demonstrate the efficacy of this approach on several high-dimensional problems that are currently well beyond the reach of SOS programming: computing a region of attraction for a 22 dimensional system, network analysis for an oscillator network with 50 nodes, searching for degree 3 controllers and degree 8 Lyapunov functions for an Acrobot system (with the resulting controller validated on a hardware platform), and a balancing controller for a 30 state and 14 input model of the ATLAS humanoid robot shown in Figure 1. We also perform numerical experiments on smaller instances of our problems (where SOS techniques can be implemented) in order to demonstrate that the additional conservatism introduced by our methods can be small compared to SOS approaches.

Our work also gives rise to several interesting theoretical questions related to the "gap" between (S)DSOS and nonnegative polynomials and how this gap may be reduced and even closed with the help of multiplier polynomials. This is analogous to the classical Positivstellensatz theorems for the gap between nonnegative polynomials and SOS polynomials. Since our goal in this paper is to provide a clear practical exposition of the approach along with its control applications, only a highlight of a few theorems in this direction is presented in Section IV and a more comprehensive treatment is to be found in [1].

II. Relevant Work

There have been many contributions to improvements in scalability of SOS programming. One approach has been to develop systematic techniques for taking advantage of problem structure, such as sparsity or symmetry of the underlying polynomials, to reduce the size of the SDPs. Examples include applications to network problems where connectivity information is known *a priori* [11], dynamical systems that can be decomposed and analyzed using smaller subsystems [12], and analysis of delayed linear systems with a low-rank delay coefficient matrix [13].

Another approach which also holds promise has been to design customized solvers for special classes of SDPs and avoid resorting to off-the-shelf interior point solver. Examples in this direction include the work in [16], which proposes a method for solving large scale robust stability problems in a parallel computing environment with a customized interior point solver, and in [17], which offers a customized interior point algorithm for optimizing over the set of nonnegative trigonometric polynomials.

The approach we take in this paper for enhancing scalability is orthogonal to the ones mentioned above (and can potentially later be combined with them). We propose to not work with the SOS decomposition to begin with, but employ computationally cheaper sufficient conditions for polynomial nonnegativity that are perhaps stronger than a SOS decompostion, but still provide useful solutions for various control applications.

A previous approach that has a similar spirit is the work in [18], which tackles the specific task of finding a lower bound on the minimum of a polynomial using geometric programming (GP). However, these GP-based conditions seem to be too strong and we show in [1] that this method is always outperformed by the SDSOS approach.

III. Diagonal Dominance and Scaled Diagonal Dominance

A polynomial $p(x)$ is a *sum of squares* (sos) if it can be written as $p(x) = \sum_{i=1}^{m} q_i^2(x)$ for some polynomials q_i. We will denote the cone of sos polynomials with degree d in n variables by $SOS_{n,d}$. It is well-known (see e.g. [2]) that the condition $p \in SOS_{n,d}$ is equivalent to the existence of a positive semidefinite symmetric matrix Q that satisfies

$$p(x) = v(x)^T Q v(x). \tag{1}$$

The vector $v(x)$ here is the vector of all monomials that have degree less than or equal to half the degree of p. The search for a matrix Q satisfying the linear constraints coming from the above equation can be cast as a semidefinite program and solved using a variety of techniques (e.g. interior point methods).

Denoting the cone of nonnegative polynomials in n variables and degree d by $POS_{n,d}$, it is clear that $SOS_{n,d} \subseteq POS_{n,d}$. The key insight we exploit in this paper is to replace the condition that the symmetric matrix Q is positive semidefinite (psd) with stronger sufficient conditions in order to obtain inner approximations to the cone $SOS_{n,d}$. In particular, we will require Q to be either *diagonally dominant* (dd) or *scaled diagonally dominant* (sdd). We recall these definitions below.

Definition 1: A symmetric matrix A is *diagonally dominant* (dd) if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$ for all i.

We will refer to the set of $n \times n$ dd matrices as DD_n.

Remark 3.1: It is clear from Definition 1 that the set DD_n has a polytopic description and can thus be optimized over using LP.

Definition 2: A symmetric matrix A is *scaled diagonally dominant* (sdd) if there exists an element-wise positive vector y such that:

$$a_{ii} y_i \geq \sum_{j \neq i} |a_{ij}| y_j, \forall i.$$

Equivalently, A is sdd if there exists a positive diagonal matrix D such that AD (or equivalently, DAD) is diagonally dominant.

The set of $n \times n$ sdd matrices will be denoted by SDD_n. We note that sdd matrices are sometimes referred to as *generalized diagonally dominant* matrices [19].

Remark 3.2: The fact that diagonal dominance is a sufficient condition for positive semidefiniteness follows directly from Gershgorin's circle theorem. This also implies that sdd matrices are psd since the eigenvalues of DAD have the same sign as those of A when D is a diagonal matrix with positive entries. Hence, denoting the set of $n \times n$ symmetric positive semidefinite matrices (psd) as S_n^+, we have from the definitions above that:

$$DD_n \subseteq SDD_n \subseteq S_n^+.$$

The next theorem, which is proved in [1], provides an important characterization of the set SDD_n.

Theorem 3.1 ([1]): Denote the set of $n \times n$ symmetric matrices as S^n. Let $M_{2\times 2}^{ij} \in S^n$ denote the symmetric matrix with all entries zero except the elements $M_{ii}, M_{ij}, M_{ji}, M_{jj}$. Then, we have the following description of SDD_n:

$$SDD_n = \left\{ A \in S^n : A = \sum_{i,j \neq i}^{i=n} M_{2\times 2}^{ij}, \begin{bmatrix} M_{ii} & M_{ij} \\ M_{ji} & M_{jj} \end{bmatrix} \succeq 0 \right\}.$$

Theorem 3.1 provides us a method to optimize over the set SDD_n using second order cone programming (SOCP), as the following theorem demonstrates.

Theorem 3.2: The set of matrices SDD_n can be optimized over using second order cone programming (SOCP).

Proof: Positive semidefiniteness of the 2×2 matrices in Theorem 3.1 is equivalent to the diagonal elements A_{ii}, A_{jj}, along with the determinant $A_{ii}A_{jj} - A_{ij}^2$, being nonnegative. This is a *rotated quadratic cone* constraint and can be imposed using SOCP [20]. ∎

IV. DSOS and SDSOS Polynomials

We now introduce naturally motivated cones that are inner approximations of $POS_{n,d}$ and that lend themselves to LP and SOCP.

Definition 3 ([1]):

- A polynomial p is *diagonally-dominant-sum-of-squares* (dsos) if it can be written as

  $$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} \beta_{ij}(m_i \pm m_j)^2,$$

  for some monomials m_i, m_j and some nonnegative constants $\alpha_i, \beta_{i,j}$.

- A polynomial p is *scaled-diagonally-dominant-sum-of-squares* (sdsos) if it can be written as

  $$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} (\beta_i m_i \pm \gamma_j m_j)^2,$$

  for some monomials m_i, m_j and some constants $\alpha_i \geq 0, \beta_i, \gamma_i$.

We denote the set of polynomials in n variables and degree d that are dsos and sdsos by $DSOS_{n,d}$ and $SDSOS_{n,d}$ respectively.

The following inclusion relations are straightforward:

$$DSOS_{n,d} \subseteq SDSOS_{n,d} \subseteq SOS_{n,d} \subseteq POS_{n,d}.$$

Our terminology in Definition 3 comes from the following relationship between dsos and sdsos polynomials to the cones of dd and sdd matrices introduced in Section III.

Theorem 4.1 ([1]):

- A polynomial p of degree $2d$ is dsos if and only if it admits a representation as $p(x) = z^T(x)Qz(x)$, where $z(x)$ is the standard monomial vector of degree d, and Q is a dd matrix.

- A polynomial p of degree $2d$ is sdsos if and only if it admits a representation as $p(x) = z^T(x)Qz(x)$, where $z(x)$ is the standard monomial vector of degree d, and Q is a sdd matrix.

Theorem 4.2: The set $DSOS_{n,d}$ is polyhedral and the set $SDSOS_{n,d}$ has a second order cone representation. For any fixed d, optimization over $DSOS_{n,d}$ (resp. $SDSOS_{n,d}$) can be done with linear programming (resp. second order cone programming), of size polynomial in n.

Proof: This follows directly from Remark 3.1 and Theorem 3.2, along with Theorem 4.1. The size of these programs is polynomial in n since the size of the Gram matrix is $\binom{n+d}{d} \times \binom{n+d}{d}$, which scales as n^d. ∎

We will refer to optimization problems with a linear objective posed over the cones $DSOS_{n,d}$ and $SDSOS_{n,d}$ as DSOS programs and SDSOS programs respectively.

A. Asymptotic Guarantees

Next, we briefly discuss how the "gap" between the cones $DSOS_{n,d}$, $SDSOS_{n,d}$ and $POS_{n,d}$ can be reduced and in some cases closed. In particular, we consider the use of "multipliers" similar to the Positivstellensatz multipliers employed in SOS programming.

Definition 4:

- For a positive integer r, a polynomial p is *r-diagonally-dominant-sum-of-squares* (r-dsos) if $p \cdot \left(\sum_i x_i^2\right)^r$ is dsos.

- For a positive integer r, a polynomial p is *r-scaled-diagonally-dominant-sum-of-squares* (r-sdsos) if $p \cdot \left(\sum_i x_i^2\right)^r$ is sdsos.

We denote the set of polynomials in n variables and degree d that are r-dsos and r-sdsos by $r\text{-}DSOS_{n,d}$ and $r\text{-}SDSOS_{n,d}$, respectively.

Note that the sets $r\text{-}DSOS_{n,d}$ and $r\text{-}SDSOS_{n,d}$ can also be optimized over using LP and SOCP respectively. The purpose of the multiplier $(\sum x_i^2)^r$ is to have a knob for trading off speed with accuracy of approximation. By increasing r, we obtain increasingly accurate inner approximations to the set of nonnegative polynomials. The following example shows that the LPs obtained from even small r can outperform the semidefinite programs resulting from SOS.

Example 4.1: Consider the polynomial, $p(x) = x_1^4x_2^2 + x_2^4x_3^2 + x_3^4x_1^2 - 3x_1^2x_2^2x_3^2$. This polynomial is nonnegative but *not* a sum of squares [21]. However, there is an LP-based nonnegativity certificate since one can show that $p \in$ 1-$DSOS$. Hence, 1-$DSOS \nsubseteq SOS$.

The following two theorems provide asymptotic guarantees on r-dsos (and hence r-sdsos) hierarchies. Their proofs rely on Positivstellensatz results from real algebraic geometry.

Theorem 4.3 ([1]): Let p be an even form (i.e., a form where individual variables are raised to even degrees), with $p(x) > 0$ for all $x \neq 0$, then there exists an integer r such that $p \in r$-$DSOS$.

If we allow the use of general multipliers (in contrast to $\sum x_i^2$), we can relax the assumption of evenness.

Theorem 4.4 ([1]): For any positive definite form p, there exists a form q such that q and pq are both dsos.

Note that given p, the search for such a q (of a given degree) is a LP. Moreover, a feasible solution to this LP certifies nonnegativity of p.

V. Examples

This section demonstrates the scalability of our approach on four examples relevant to control and verification of dynamical systems. We compare runtimes and optimality of our approach with the SOS approach in cases where this is possible (i.e., smaller instances of problems). An upcoming software package written using the Systems Polynomial Optimization Toolbox (SPOT) [22] includes a complete implementation of the presented methods. The toolbox features very efficient polynomial algebra and allows us to setup the large-scale LPs and SOCPs arising from our examples.

We use MOSEK as our LP and SOCP solver. Runtimes for SDPs are reported for both the recently released MOSEK SDP solver and the very widely used SeDuMi solver. All code was run on a machine with four Intel i7 processors with a clock speed of 3.4 GHz and 16 GB RAM.

A. Regions of Attraction

In our first example, we consider the computation of regions of attraction (ROA), which is known to be a NP-hard problem [23]. The system we examine is the N-link pendulum depicted in Figure 2. This system has $2N$ states $x = [\theta_1, \ldots, \theta_N, \dot{\theta}_1, \ldots, \dot{\theta}_N]$ composed of the joint angles and their derivatives. There are $N-1$ control inputs (the joint closest to the base is not actuated). Each link of the pendulum is assumed to be a uniformly dense cylindrical rod of radius 5 cm with mass $m = 1$ kg and length $l = 1$ m. We take the unstable "upright" position of the system to be the origin of our state space and design a LQR controller in order to stabilize this equilibrium. The cost matrix Q and the action matrix R for the LQR controller are both diagonal, with $Q_{ii} = 10$ for $i = 1, \ldots, N$, $Q_{jj} = 1$ for $j = N+1, \ldots, 2N$, and $R_{ii} = 1$ for $i = 1, \ldots, N-1$.

A polynomial approximation of the dynamics of the closed loop system is obtained by a Taylor expansion of degree 3. Denoting the resulting dynamics by $\dot{x} = f(x)$, if we can find

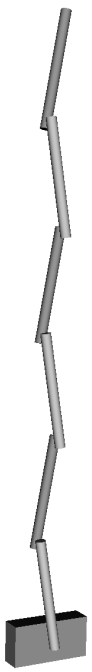
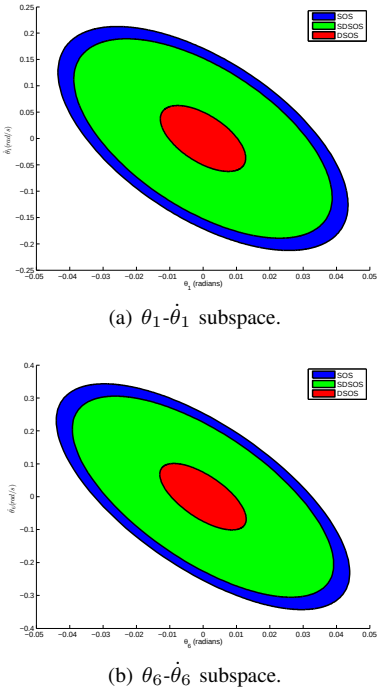


Fig. 2: An illustration of the N-link pendulum system (with N = 6).



(a) θ_1-$\dot{\theta}_1$ subspace.

(b) θ_6-$\dot{\theta}_6$ subspace.

Fig. 3: Comparisons of projections of the ROAs computed for the 6-link pendulum system using DSOS, SDSOS and SOS programming.

a positive definite polynomial $V(x)$ such that the following condition holds:

$$V(x) \leq \rho \implies \dot{V}(x) < 0, \tag{2}$$

then the ρ sublevel set of $V(x)$ is an inner approximation of the ROA. We choose our Lyapunov function to be the cost-to-go function $V(x) = x^TSx$ of the LQR controller and attempt to maximize ρ. As described in [2], under the assumption that the Hessian of $\dot{V}(x)$ is positive definite at the origin, the following is a sufficient condition for (2):

$$(x^Tx)(V(x) - \rho) + L(x)\dot{V}(x) \geq 0. \tag{3}$$

Here, $L(x)$ is a "multiplier" polynomial. By replacing the nonnegativity condition in (3) with a DSOS constraint, we obtain a LP:

$$\begin{aligned} \max_{\rho, L(x)} \quad & \rho \\ \text{s.t.} \quad & (x^Tx)(V(x) - \rho) + L(x)\dot{V}(x) \in DSOS_{2N,6} \end{aligned} \tag{4}$$

Similarly, a SDSOS/SOS constraint yields a SOCP/SDP. The optimization in (4) is over ρ and $L(x)$ (degree 2).

We note that while there are different SOS programming based formulations for approximating the ROA that allow one to search over Lyapunov functions in addition to the scaling ρ (see, e.g., [3]), these typically lead to non-convex optimization problems. Algorithms for these formulations generally do not have convergence guarantees, thus making it difficult to distinguish between conservatism caused by the algorithm and conservatism inherent in the (S)DSOS condition. Hence, in order to facilitate a direct comparison

2N (# states)	4	6	8	10	12	14	16	18	20	22
DSOS	< 1	0.44	2.04	3.08	9.67	25.1	74.2	200.5	492.0	823.2
SDSOS	< 1	0.72	6.72	7.78	25.9	92.4	189.0	424.74	846.9	1275.6
SOS (SeDuMi)	< 1	3.97	156.9	1697.5	23676.5	∞	∞	∞	∞	∞
SOS (MOSEK)	< 1	0.84	16.2	149.1	1526.5	∞	∞	∞	∞	∞

TABLE I: Runtime comparisons (in seconds) for ROA computations on N-link system.

of the DSOS, SDSOS and SOS approaches, we use the formulation presented above in (3), which involves solving a single convex optimization problem for each approach.

An important observation is that unlike the sets $POS_{n,d}$ and $SOS_{n,d}$, the sets $DSOS_{n,d}$ and $SDSOS_{n,d}$ are not invariant to coordinate transformations, i.e., a polynomial $p(A(x))$ is not necessarily DSOS (resp. SDSOS) even if $p(x)$ is DSOS (resp. SDSOS). Thus, performing coordinate transformations on the problem data (e.g., on the state variables of a dynamical system) can sometimes have an important effect. We explore this issue in this example by performing a preprocessing step that is intuitive and straightforward to implement. It can be used for problems involving the search for Lyapunov functions, and can potentially be extended to other problems as well. In particular, we find an invertible affine transformation that simultaneously diagonalizes the Hessians of $V(x)$ and $-\dot{V}(x)$ evaluated at the origin (this is always possible for two positive definite matrices). The intuition behind the coordinate change is that the functions $V(x)$ and $-\dot{V}(x)$ locally resemble functions of the form $x^T D x$ (with D diagonal), which are DSOS polynomials that are "far away" from the boundary of the DSOS cone. We solve the optimization problem (3) after performing this coordinate transformation. The transformation is then be inverted to obtain ROAs in the original coordinate frame.

Table I compares the runtimes of the programs obtained using our approach with SOS programming for different values of $2N$ (number of states). The SOS programs obtained for $2N > 12$ are too large to run (due to memory constraints). In contrast, our approach allows us to handle almost twice as many states. Further, for cases where the SOS programs do run, the DSOS and SDSOS programs are significantly faster. In particular for the 12 state system, DSOS is approximately 2500 times faster than SOS using SeDuMi and 150 times faster than SOS using MOSEK, while SDSOS is 900 times faster in comparison to SeDuMi and 60 times faster than MOSEK for SOS.

Table II compares the optimal values of ρ obtained using the different methods. The values obtained using SDSOS programming are within approximately 80 to 90 percent of the values from SOS programming. Note that since the Lyapunov function is fixed in our case and we are only optimizing ρ, the ratio of optimal values of ρ is equal to the ratio of $(\text{Volume})^{\frac{1}{2N}}$ of the ROAs. While the result using DSOS are more conservative, they could still be useful in practice. Figure 3 compares projections of the ROAs obtained using the different methods for the 6-link pendulum.

2N (states)	4	6	8	10	12
ρ_{dsos}/ρ_{sos}	0.38	0.45	0.13	0.12	0.09
ρ_{sdsos}/ρ_{sos}	0.88	0.84	0.81	0.79	0.79

TABLE II: Comparison of optimal values for ROA problem on N-link pendulum.

B. Network Analysis

In this example, we consider Example 3 from [11]. The goal is to analyze the domain on which the Hamiltonian function V for a network of Duffing oscillators is positive definite:

$$V = \sum_{i=1}^{N} a_i (\frac{1}{2} y_i^2 - \frac{1}{4} y_i^4) + \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} b_{ik} \frac{1}{4} (y_k - y_i)^4. \quad (5)$$

Here, N is the number of nodes in the network. The following condition can be used to establish an inner approximation of the domain on which V is positive definite:

$$p(y_1, \ldots, y_N) = V - \sum_{i=1}^{N} \lambda_i y_i^2 (g - y_i^2) \geq 0, \quad (6)$$

where $\lambda_i > 0$ are scalar multipliers. These conditions imply that V is positive definite when $y_i^2 < g$.

In [11], the authors propose an approach that exploits network structure to eliminate monomials from the SOS program that results from replacing the inequality in (6) with a SOS condition. This makes the SOS program smaller (potentially at the cost of conservatism) and allows them to handle large network sizes.

By replacing the nonnegativity in (6) by a (S)DSOS condition, we can apply the techniques presented in this paper. For the comparisons presented below, we do not preprocess the programs using the method presented in [11]. We demonstrate that we are able to handle network sizes considered in [11] *without* explicitly exploiting network structure. Of course, we would expect to scale even better by using the approach in [11] as a preprocessing step.

As in [11], we randomly set $\frac{0.5}{N} \leq b_{ik} \leq \frac{1.5}{N}$, corresponding to a globally coupled network of Duffing oscillators. We also find the largest value of g for which the condition (6) is feasible by performing a bisection search in an outer loop. These results and runtimes are compared for (S)DSOS and SOS programming in Tables IV and III respectively.

We find that our approach works with runtimes comparable to [11], even though we do not exploit the network structure. In contrast, the SDP solvers MOSEK and SeDuMi are significantly slower even for networks with $N = 20$. For N larger than 25, memory (RAM) constraints prevent the SDP from running.

N (nodes)	10	20	30	40	50
DSOS	0.24	0.76	2.86	8.74	21.35
SDSOS	0.26	1.07	5.02	17.81	46.64
SOS (SeDuMi)	3.91	16050.26	∞	∞	∞
SOS (MOSEK)	0.53	173.75	∞	∞	∞

TABLE III: Runtime comparisons (in seconds) for network problem.

N (nodes)	10	20	30	40	50
DSOS	0.94	0.74	0.71	0.69	0.66
SDSOS	1.58	1.49	1.48	1.48	1.48
SOS (SeDuMi)	2.00	2.00	NA	NA	NA
SOS (MOSEK)	2.00	2.00	NA	NA	NA

TABLE IV: Comparison of optimal values (g) for network problem.

In [11], the value of g is 1.8 for all network sizes. Thus, the optimal values we obtain using SDSOS programming are only slightly more conservative.

C. Hardware Experiments on Acrobot

The utility of our method is not restricted to high-dimensional systems. As we show in this example, we can design high degree polynomial feedback controllers using high degree Lyapunov functions for smaller systems with benefits in terms of running time as compared to SOS programming. We consider the Acrobot [24], which is a benchmark for control of underactuated systems. The system is a special case of the N-link pendulum examined in Section V-A (with $N = 2$) and is actuated only at the joint between the two links of the robot (θ_2). The task is to design a polynomial feedback controller in order to stabilize the system about the unstable "upright" position. The hardware platform on which experiments are conducted is shown in Figure 4 balancing in this configuration using the controller designed with SDSOS programming (as described below).

System identification for the hardware platform was performed using the prediction error minimization method in MATLAB's System Identification Toolbox [25] in order to identify parameters of the model presented in [24]. The dynamics were then Taylor expanded around the upright equilibrium to degree 3 in order to obtain a polynomial vector field.

We use the method presented in [26] to design a balancing controller for the system. In particular, we search for a degree 8 Lyapunov function $V(x)$ and a degree 3 feedback controller $u(x)$ in order to maximize the size of the region of attraction (ROA) of the resulting closed-loop system. The DSOS version of the optimization problem is:

$$\begin{aligned} \max_{\rho, L(x), V(x), u(x)} \quad & \rho \qquad (7) \\ \text{s.t.} \quad & V(x) \in DSOS_{4,8} \\ & -\dot{V}(x) + L(x)(V(x) - \rho) \in DSOS_{4,10} \\ & L(x) \in DSOS_{4,4} \\ & \sum_j V(e_j) = 1. \end{aligned}$$

Here, $L(x)$ is a non-negative multiplier term and e_j is the j-th standard basis vector for the state space $\mathbb{R}^n$. It is easy to see that the above conditions are sufficient for establishing $B_\rho = \{x \in \mathbb{R}^4 \mid V(x) \leq \rho\}$ as an inner estimate of the region of attraction for the system. When $x \in B_\rho$, the second constraint implies that $\dot{V}(x) < 0$ (since $L(x)$ is constrained to be non-negative). The last constraint normalizes $V(x)$.

The optimization problem (7) is not convex in general since it involves conditions that are bilinear in the decision variables. However, problems of this nature are common in the SOS programming literature (see e.g. [4]) and are typically solved by iteratively optimizing groups of decision variables. Each step in the iteration is then a DSOS program (or a SDSOS/SOS program if the constraints in (7) are replaced by DSOS/SOS constraints). This iterative procedure is described in more detail in [26] and can be initialized with the Lyapunov function from a LQR controller and a small enough value of ρ. The LQR Lyapunov function can also be used to perform a coordinate transformation in a manner identical to the one described in Section V-A. Finally, we note that in order to use the approach described above on the Acrobot hardware platform, it is also important to take into account the torque limit of the system.The optimization problem (7) can be modified in a straightforward manner to account for torque limits as described in [26, Section IV A].

Figure 5 presents two-dimensional slices of the ROA resulting from the approach described above using SDSOS and SOS programming (DSOS programming yields very conservative results on this example and we do not present the results here). The ROA from SDSOS programming is only slightly conservative on the $\theta_1 - \theta_2$ slice and is in fact slightly larger than the SOS ROA on the $\theta_2 - \dot{\theta}_2$ slice[1] Each iteration of the algorithm takes approximately 40 seconds with SDSOS, while SOS takes 1825 seconds with SeDuMi and 148 seconds with MOSEK. Convergence of the algorithm is typically observed between 5 and 10 iterations for SDSOS and SOS. Thus, we are able to obtain significant gains in running times with very little loss in quality of the solution.

We validated the performance of the balancing controller from SDSOS programming by implementing it on the hardware platform shown in Figure 4. An open-loop controller that swings up the system from the downright configuration to the upright one was designed using the direction collocation trajectory optimization approach [27]. A time-varying LQR controller was then designed to correct for deviations from this nominal trajectory. At the end of the trajectory, the robot switches to our balancing controller. We performed 30 consecutive experimental trials of the robot swinging up and balancing. The balancing controller had a success rate of 100% on these trials. A video of the controller in action is available online at https://www.youtube.com/watch?v=FeCwtvrD76I.

We end this example by noting that attempting to use SOS

[1]Note that the iterative algorithm we employ here is not guaranteed to converge to the global optimum of the problem. Hence, the ROA from SDSOS will not necessarily be a subset of the SOS ROA.

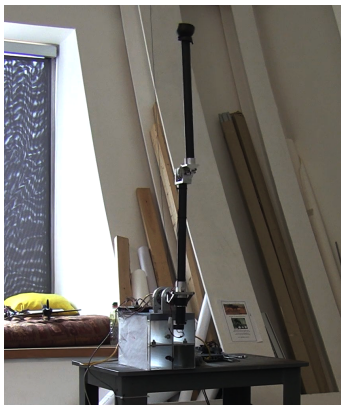


Fig. 4: Picture of the Acrobot balancing in the upright configuration using the controller designed with SDSOS programming. A video of the controller in action is available online at https://www.youtube.com/watch?v=FeCwtvrD76I.

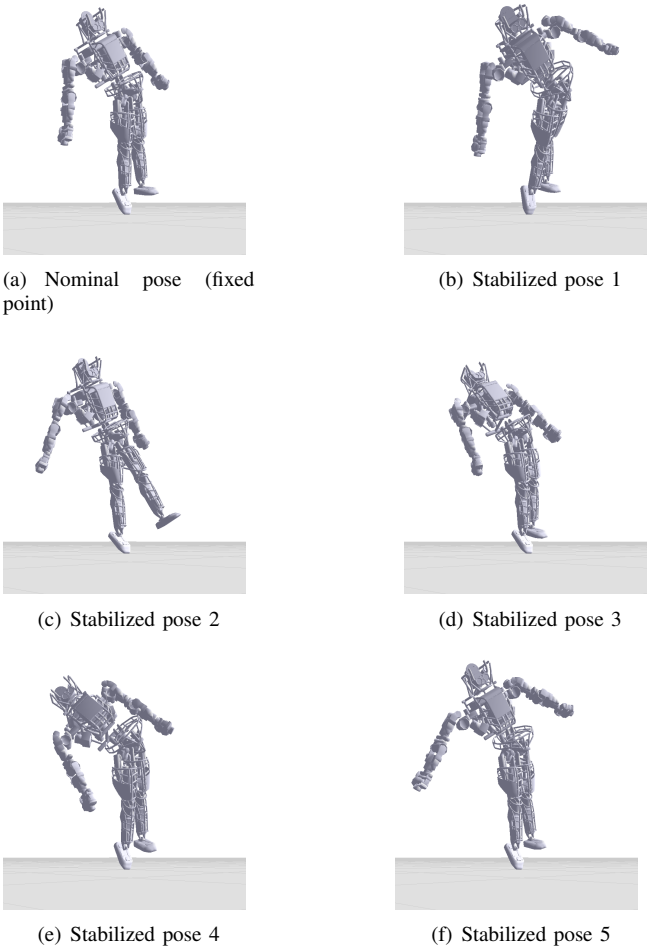
programming to search for a higher degree controller (e.g., degree 5) resulted in numerical errors from the SDP solvers. This prevented us from running more than 2 or 3 iterations of the algorithm. In contrast, we did not run into such numerical issues with SDSOS programming. This highlights another benefit of our approach; in addition to smaller optimization problems, we also obtain programs that seem to be better conditioned numerically and are easier to work with due to the maturity of existing LP and SOCP solvers.

D. Control Synthesis for Humanoid Robot

In our final example, we consider an application in robotics. Control of humanoid robots is an important problem in robotics and presents a significant challenge due to the nonlinear dynamics of the system and high dimensionality of the state space. Here we use the approach described in this paper to design a balancing controller for a model of the ATLAS robot shown in Figure 1. This robot was designed and built by Boston Dynamics Inc. and was used for the 2013 DARPA Robotics Challenge.

Our model of the robot is based on physical parameters of the hardware platform and has 30 states and 14 control inputs. The task we consider here is to balance the robot on its right toe. In order to do this, we make a few simplifying assumptions. First, we assume that the interface of the right foot and the ground is mediated by a pin joint at the toe. This joint is not actuated and its axis is parallel to the ground, constraining the foot to pitch up and down in relation to the ground. Next, we ignore collisions between the different links of the robot. Finally, we do not take into account input saturations (which is a reasonable assumption given the very high torque output capabilities of the hydraulic actuators of the robot).

The balancing controller is constructed using the approach described in Section V-C with SDSOS programming. Each iteration taking approximately 4.5 minutes and convergence occurs in 4-5 iterations. We also used DSOS programming to design a controller, but we do not present these results here due to space constraints. We note that SOS



(a) Nominal pose (fixed point)

(b) Stabilized pose 1

(c) Stabilized pose 2

(d) Stabilized pose 3

(e) Stabilized pose 4

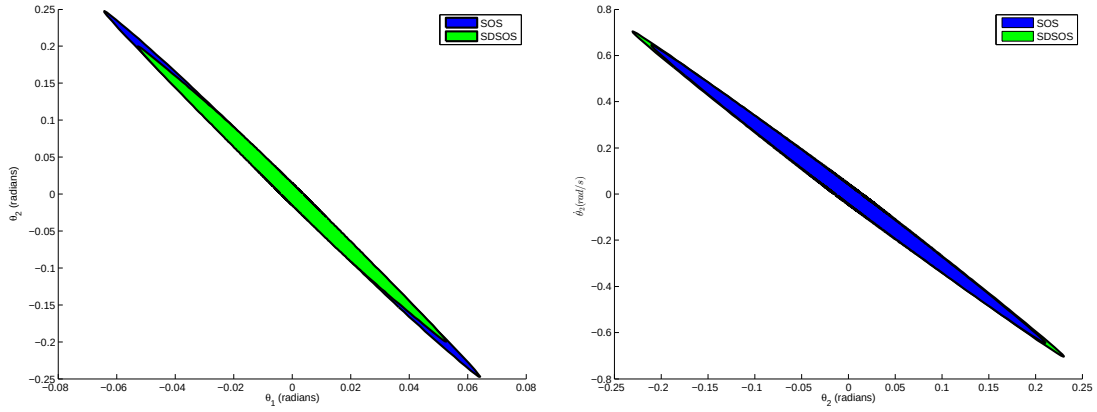
(f) Stabilized pose 5

Fig. 6: Nominal position of the robot, i.e., the fixed point being stabilized (subplot (a)), and configurations of the robot that are stabilized by the controller designed using SDSOS programing (subplots (b)-(f)). A video of simulations of the controller started from different initial conditions is available online at https://www.youtube.com/watch?v=6jhCiuQVOaQ.

programming is unable to handle this system due to memory (RAM) constraints. Figure 6 demonstrates the performance of the resulting controller from SDSOS programming by plotting initial configurations of the robot that are stabilized to the fixed point. As the plot illustrates, the controller is able to stabilize a very wide range of initial conditions. A video of simulations of the closed loop system started from different initial conditions is available online at https://www.youtube.com/watch?v=6jhCiuQVOaQ.

VI. Conclusion

In this paper, we have considered LP and SOCP based alternatives to SOS programming and demonstrated the utility of our approach on several high dimensional control applications that are currently beyond the capabilities of SOS programing. The key idea is to replace the positive semidefiniteness constraint in SOS programming with stronger conditions based on diagonal dominance and scaled diagonal dominance. We refer to the resulting inner approximations



(a) θ_1-θ_2 subspace. (b) θ_2-$\dot{\theta}_2$ subspace.

Fig. 5: Comparisons of slices of ROAs computed for the Acrobot system using SDSOS and SOS programming.

of the SOS cone as the DSOS and SDSOS cones, which can be optimized over using LP and SOCP. We believe that the ability to exploit the scalability and maturity of LP and SOCP solvers in contexts where SOS programming has previously been applied could have a large impact on applications.

In future work, we will consider other inner approximations of the SOS cone that are inexpensive to optimize over. In particular, the set of factor width k matrices [19] may provide a natural extension to DSOS and SDSOS.

VII. Acknowledgements

The author are grateful to Pablo Parrilo for introducing them to sdd matrices and their second order cone characterization, and to Mark Tobenkin for help with implementation of the code that sets up DSOS and SDSOS programs. This work was supported by ONR MURI grant N00014-09-1-1051. Amir Ali Ahmadi is currently supported by a Goldstine Fellowship at IBM.

References

[1] A. A. Ahmadi and A. Majumdar, "DSOS and SDSOS optimization: More tractable alternatives to SOS optimization," *In preparation (http://aaa.lids.mit.edu/publications)*, 2014.

[2] P. A. Parrilo, *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, May 18 2000.

[3] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, no. 10, pp. 2669 – 2675, 2008.

[4] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *42nd IEEE Conference on Decision and Control*, vol. 5, pp. 4676 – 4681, December 2003.

[5] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control* (R. Alur and G. Pappas, eds.), vol. 2993 of *Lecture Notes in Computer Science*, pp. 271–274, Springer Berlin / Heidelberg, 2004.

[6] G. Chesi and D. Henrion, "Guest editorial: Special issue on positive polynomials in control," *IEEE Transactions on Automatic Control*, vol. 54, no. 5, pp. 935–936, 2009.

[7] A. A. Ahmadi, *Algebraic relaxations and hardness results in polynomial optimization and Lyapunov analysis*. PhD thesis, Massachusetts Institute of Technology, September 2011. Available at `http://aaa.lids.mit.edu/publications`.

[8] D. Henrion and M. Korda, "Convex computation of the region of attraction of polynomial control systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 297–312, 2014.

[9] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical Programming*, vol. 39, pp. 117–129, 1987.

[10] P. A. Parrilo, "Exploiting algebraic structure in sum of squares programs," in *Positive polynomials in control*, pp. 181–194, Springer, 2005.

[11] J. Anderson and A. Papachristodoulou, "A network decomposition approach for efficient sum of squares programming based analysis," in *American Control Conference (ACC)*, pp. 4492–4497, IEEE, 2010.

[12] J. Anderson and A. Papachristodoulou, "Dynamical system decomposition for efficient, sparse analysis," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 6565–6570, IEEE, 2010.

[13] Y. Zhang, M. Peet, and K. Gu, "Reducing the complexity of the sum-of-squares test for stability of delayed linear systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 229–234, 2011.

[14] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, vol. 18. Wiley New York, 1988.

[15] M. Jünger, T. M. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*. Springer, 2009.

[16] R. Kamyar, M. Peet, and Y. Peet, "Solving large-scale robust stability problems by exploiting the parallel structure of polya's theorem," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, 2012.

[17] R. Tae, B. Dumitrescu, and L. Vandenberghe, "Interior-point algorithms for sum-of-squares optimization of multidimensional trigonometric polynomials," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, 2007.

[18] M. Ghasemi and M. Marshall, "Lower bounds for polynomials using geometric programming," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 460–473, 2012.

[19] E. G. Boman, D. Chen, O. Parekh, and S. Toledo, "On factor width and symmetric h-matrices," *Linear algebra and its applications*, vol. 405, pp. 239–248, 2005.

[20] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

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

[22] A. Megretski, "Systems polynomial optimization tools (SPOT), available online: http://web.mit.edu/ameg/www." 2010.

[23] A. A. Ahmadi, A. Majumdar, and R. Tedrake, "Complexity of ten decision problems in continuous time dynamical systems," in *Proceedings of the 2013 American Control Conference (ACC)*, 2013.

[24] M. Spong, "The swingup control problem for the acrobot," *IEEE Control Systems Magazine*, vol. 15, pp. 49–55, February 1995.

[25] L. Ljung, "System identification toolbox for use with matlab," 2007.

[26] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[27] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM Advances in Design and Control, Society for Industrial and Applied Mathematics, 2001.