An Accelerated First-Order Method for Solving Unconstrained Polynomial Optimization Problems

Dimitris Bertsimas* Robert M. Freund† Xu Andy Sun‡

March 2011

Abstract

Our interest lies in solving large-scale unconstrained polynomial optimization problems. Because interior-point methods for solving these problems are severely limited by the large-scale, we are motivated to explore efficient implementations of an accelerated first-order method to solve this class of problems. By exploiting special structural properties of this problem class, we greatly reduce the computational cost of the first-order method at each iteration. We report promising computational results as well as a curious observation about the behavior of the first-order method for unconstrained polynomial optimization.

1 Introduction

We are interested in solving the unconstrained polynomial optimization problem:

$$(P) \quad \gamma^* = \min_{x \in \mathbb{R}^n} p(x),$$

where $p(x) : \mathbb{R}^n \to \mathbb{R}$ is a real-valued polynomial in n variables and of even degree $d = 2m$, where m is an integer. Polynomial optimization has connections to many fields such as algebraic geometry [14], the problem of moments [5, 11], and semidefinite programming (SDP) [17]. It has also been a useful tool in control theory [21], optimized moment inequalities in probability theory [4], combinatorial optimization [19], and stochastic modeling [7]. However, despite interesting theoretical properties and rich modeling capabilities, polynomial optimization techniques have not been extensively used in real

*Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, dbertsim@mit.edu.
†Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, rfreund@mit.edu.
‡Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, sunx@mit.edu.

applications. One reason for this is the limitation of current computational schemes for solving large instances of polynomial optimization problems. Using the most successful solution methodology — the hierarchy of semidefinite programming (SDP) relaxations [13][22], the resulting SDP problem involves a (primal) matrix variable of order $\binom{n+m}{n} \times \binom{n+m}{n} = O(n^m \times n^m)$ and a (dual) vector variable of dimension $\binom{n+2m}{n} = O(n^{2m})$, which grows exponentially in the degree of the polynomial. Currently, the most widely used software (such as Gloptipoly [8] and SOSTOOLS [23]) solve the SDP relaxation by interior-point methods, where the exponential growth in the SDP dimension severely limits the size of solvable instances.

In this paper, we explore the possibility of improving the practical solvability of polynomial optimization problems. Instead of using powerful but expensive interior-point methods (IPMs), we take advantage of the recent progress on solving conic optimization problems with first-order algorithms. In particular, we apply a variant of Nesterov's accelerated first-order method [26, 1] to the SDP relaxation of an unconstrained polynomial optimization problem, using a reformulation proposed in Lu [15]. The key to the success of such an approach depends on the ability to exploit the special structure of the reformulation in the context of the projection operators that are used in the accelerated first-order method. To accomplish this, we identify a special property of the SDP reformulation, and we show that this property enables very fast computation of projections onto a certain affine manifold that arises in accelerated first-order method. The reduced computational cost of the first-order algorithm is then demonstrated via numerical experiments, which show that the accelerated first-order method can solve larger instances than the existing interior-point-method based software. For both 4-degree and 6-degree polynomials, the first-order method solves instances of dimensions roughly twice as large (in terms of the number of polynomial variables n) as existing IPM software (namely, the SOS package in SeDuMi) can handle. The first-order method is also faster than the existing software for medium size problems when solved to comparable accuracy.

Over the past several years, research on accelerated first-order methods has grown substantially, especially on the computational frontier of applying such methods to solving large-scale problems (see, e.g., compressed sensing [15, 3], covariance selection [6, 16], and image processing [2]). However, to the best of our knowledge, such methods have not been applied to general unconstrained polynomial optimization. Herein we study the behavior of accelerated first-order methods on solving this important class of problems. We also discuss a curious observation on the convergence behavior of this method that is not derivative of the usual convergence analysis.

The paper is organized as follows: In Section 2, we first briefly review the method of SDP relaxation of unconstrained polynomial optimization problems and reformulations that are suitable for applying the intended accelerated first-order method. We then demonstrate the special structure of the reformulation that allows for fast computation of the affine manifold projection problem at each iteration. In Section 3 we report computational results and provide a comparison with existing interior-point based methods. We also provide some interesting observations on the convergence behavior of the accelerated first-order method.

1.1 Notation and Definitions

Let $\mathbb{S}^n$ denote the space of $n \times n$ real symmetric matrices, and $\mathbb{S}^n_+$ denote the cone of $n \times n$ positive semidefinite matrices. Let $\succeq$ denote the standard ordering on $\mathbb{S}^n$ induced by the cone $\mathbb{S}^n_+$. For $X, Y \in \mathbb{S}^n$, let $X \bullet Y$ denote the Frobenius inner product, and $\|X\|_F$ denote the induced matrix norm, namely $\|X\|_F = \sqrt{X \bullet X} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n X_{ij}^2}$. For $x, y \in \mathbb{R}^n$, let $x^T y$ denote the standard Euclidean inner product, and $\|x\|_2$ denote the Euclidean norm. More generally, for a finite dimensional inner product space $\mathbb{U}$, let $\|\cdot\|$ denote the induced norm. For a compact convex set $\mathcal{C} \in \mathbb{U}$, define a distance function $d_{\mathcal{C}}(u) := \min\{\|u - v\| |\ v \in C\}$ for all $u \in \mathbb{U}$. Define $\Pi_{\mathcal{C}}(u) := \arg\min\{\|u - v\| |\ v \in C\}$. The "prox" or squared distance function $d^2_{\mathcal{C}}(u)$ is a Lipschitz differentiable convex function with Lipschitz constant $L = 2$, whose gradient operator is given by $\nabla_u d^2_{\mathcal{C}}(u) = 2(u - \Pi_{\mathcal{C}}(u))$, see [10].

Let $\mathbb{N}$ denote the set of nonnegative integers, and $\mathbb{N}^n_d := \{\alpha \in \mathbb{N}^n |\ |\alpha| := \sum_{i=1}^n \alpha_i \leq d\}$ for $n, d \in \mathbb{N}$ and $n > 0$. For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, and $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, let x^α denote the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ of degree $|\alpha|$. For $d = 2m$, let

$$v_{2m}(x) := \{1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_1x_n, x_2^2, x_2x_3, \ldots, x_n^{2m}\} \tag{1.1}$$

be a basis for the n-variable and degree $d = 2m$ real-valued polynomial, so that we can write

$$p(x) = \sum_{\alpha \in \mathbb{N}^n_{2m}} p_\alpha x^\alpha = p^T v_{2m}(x) \ . \tag{1.2}$$

Let M denote the number of components of $v_{2m}(x)$, namely $M = \binom{n+2m}{n}$. Let $N := \binom{n+m}{n}$. For each $\alpha \in \mathbb{N}^n_{2m}$, let $A_\alpha \in \mathbb{S}^N$ denote the matrix corresponding the following operator:

$$A_\alpha(X) := A_\alpha \bullet X = \sum_{\delta} \sum_{\nu = \alpha - \delta} X_{\delta\nu}, \quad \text{for all } X \in \mathbb{S}^N, \tag{1.3}$$

where X is indexed by $\delta, \nu \in \mathbb{N}^n_m$. The matrices A_α are called the *basic moment matrices*, which are used to define the moment matrix $M_{2m}(y)$ of the finite moment sequence $y \in \mathbb{R}^M$ as $M_{2m}(y) := \sum_{\alpha \in \mathbb{N}^n_{2m}} y_\alpha A_\alpha$.

A polynomial $p(x)$ is called a *sum of squares* (SOS) polynomial if $p(x)$ can be written as $p(x) = \sum_{i=1}^{s} q_i^2(x)$ for some polynomials $q_1(x), \dots, q_s(x)$.

2 SOS Formulation, Reformulation, and Accelerated First-Order Method

2.1 SOS relaxation

For a given n-variable $d = 2m$-degree polynomial $p(x)$, we consider the unconstrained polynomial optimization problem:

$$(P) \quad \gamma^* = \min_{x \in \mathbb{R}^n} p(x),$$

and there is no loss of generality in assuming that $p(x)$ has zero constant term. Problem (P) admits the following SOS relaxation:

$$\begin{aligned} \gamma^*_{sos} = \max_{\gamma \in \mathbb{R}} \quad & \gamma \\ \text{s.t.} \quad & p(x) - \gamma \text{ is SOS.} \end{aligned} \tag{2.1}$$

In general, we have $\gamma^*_{sos} \le \gamma^*$. When $p(x) - \gamma^*$ is an SOS polynomial, then $\gamma^*_{sos} = \gamma^*$ [13, Theorem 3.2]. The SOS constraint can be written as:

$$p(x) - \gamma = \sum_{i=1}^{s} q_i^2(x) = \sum_{i=1}^{s} v_m(x)^T q_i q_i^T v_m(x) = v_m(x)^T \left(\sum_{i=1}^{s} q_i q_i^T \right) v_m(x) = v_m(x)^T X v_m(x),$$

for some polynomials $q_i(x)$ of degree at most m, and $X \in \mathbb{S}^N_+$. Matching monomial coefficients on both sides above, it follows that X must satisfy:

$$\begin{cases} X_{00} = -\gamma, \\ \sum_\delta \sum_{\nu = \alpha - \delta} X_{\delta\nu} = p_\alpha \qquad \text{for } \alpha \in \mathbb{N}^n_{2m}, |\alpha| > 0, \end{cases} \tag{2.2}$$

which yields the following SDP formulation of (2.1):

$$\begin{aligned} \gamma^*_p = \min_{X \in \mathbb{S}^N} \quad & A_0 \bullet X \\ \text{s.t.} \quad & A_\alpha \bullet X = p_\alpha \ \text{ for } \alpha \in \mathbb{N}^n_{2m}, |\alpha| > 0, \\ & X \succeq 0. \end{aligned} \tag{2.3}$$

The dual of (2.3) is:

$$\begin{aligned}
\gamma_d^* = \max_{y\in\mathbb{R}^{M-1}, S\in\mathbb{S}^N} \quad & \sum_{\alpha\in\mathbb{N}^n_{2m}, |\alpha|>0} p_\alpha y_\alpha, \\
\text{s.t.} \quad & \sum_{\alpha\in\mathbb{N}^n_{2m}, |\alpha|>0} y_\alpha A_\alpha + S = A_0, \\
& S \succeq 0.
\end{aligned} \tag{2.4}$$

We have $\gamma_p^* = -\gamma_{sos}^*$. If (2.3) has a feasible solution, then there is no duality gap between (2.3) and (2.4), i.e., $\gamma_p^* = \gamma_d^*$, see [13, Proposition 3.1].

2.2 Reformulations of the SDP problem

In order to apply an accelerated first-order method to solve (2.3) and (2.4), one needs to reformulate these problems so that the resulting reformulated feasible region is a simple convex set for which computation of projections can be done very efficiently. Several such reformulations have been proposed in the recent literature, see for example [12, 9, 15]. In particular, Lu [15] has derived a detailed worst-case complexity analysis and comparison of the cited proposals, and successfully applied one recommended scheme to solve a class of linear programs, MaxCut SDP relaxation, and the Lovász capacity problem. For completeness, we present these reformulations and choose a scheme (different from that in [15]) here that is most suited for computing solutions of (2.3)-(2.4).

Herein we are concerned with those instances of (P) for which $p(x) - \gamma^*$ is SOS. Hence we presume that (2.3) is feasible, in which case strong duality holds in (2.3) and (2.4). We then write the optimality conditions for the primal-dual pair as follows:

$$A_\alpha \bullet X = p_\alpha, \quad \text{for all } \alpha \in \mathbb{N}^n_{2m}, |\alpha| > 0, \tag{2.5}$$

$$\sum_{\alpha\in\mathbb{N}^n_{2m}, |\alpha|>0} y_\alpha A_\alpha + S = A_0, \tag{2.6}$$

$$A_0 \bullet X = p^T y, \tag{2.7}$$

$$X, S \in \mathbb{S}^N_+, y \in \mathbb{R}^{M-1}. \tag{2.8}$$

Note that any solution of (2.5)-(2.8) is an optimal primal-dual solution of (2.3) and (2.4), and *vice versa.* Define the finite dimensional space $\mathbb{U} := \mathbb{S}^N \times \mathbb{S}^N \times \mathbb{R}^{M-1}$. For $u = (X, S, y), \hat{u} = (\hat{X}, \hat{S}, \hat{y}) \in \mathbb{U}$, define the inner product $u \bullet \hat{u} := X \bullet \hat{X} + S \bullet \hat{S} + y^T \hat{y}$. The solution set of equations (2.5)-(2.7) define an affine subspace $\mathcal{L} \subset \mathbb{U}$, and the inclusion (2.8) defines a convex cone $\mathcal{K} := \mathbb{S}^N_+ \times \mathbb{S}^N_+ \times \mathbb{R}^{M-1} \subset \mathbb{U}$. We represent $\mathcal{L}$ in a more compact form by introducing a linear operator $\mathcal{A} : \mathbb{S}^N \to \mathbb{R}^{M-1}$ and its adjoint $\mathcal{A}^* : \mathbb{R}^{M-1} \to \mathbb{S}^N$ defined as follows:

$$z = \mathcal{A}(X) \quad \text{where} \quad z_\alpha := A_\alpha \bullet X, \quad \text{for all } \alpha \in \mathbb{N}^n_{2m}, |\alpha| > 0, \tag{2.9}$$

$$v = \mathcal{A}^*(y) \quad \text{where} \quad v := \sum_{\alpha \in \mathbb{N}^n_{2m}, |\alpha|>0} y_\alpha A_\alpha \ . \tag{2.10}$$

Here the matrices A_α are the basic moment matrices defined in Section 1.1. Similarly, consider another linear operator $\mathcal{A}_0 : \mathbb{S}^N \to \mathbb{R}$ defined as $\mathcal{A}_0(X) := A_0 \bullet X$ for all $X \in \mathbb{S}^N$. We also denote $\mathcal{I}_1 : \mathbb{S}^N \to \mathbb{S}^N$ as the identity operator. Using all of this notation, the subspace $\mathcal{L}$ defined above as the solution of the equations (2.5)-(2.7) can be conveniently represented as $\mathcal{L} := \{u \in \mathbb{U} : \mathcal{E}u = q\}$ where

$$\mathcal{E} = \begin{bmatrix} \mathcal{A} & 0 & 0 \\ 0 & \mathcal{I}_1 & \mathcal{A}^* \\ \mathcal{A}_0 & 0 & -p^T \end{bmatrix}, u = \begin{bmatrix} X \\ S \\ y \end{bmatrix}, q = \begin{bmatrix} p \\ A_0 \\ 0 \end{bmatrix}. \tag{2.11}$$

Note of course that solving (2.5)-(2.8) (and hence solving (2.3) and (2.4)) is equivalent to finding a point $u \in \mathcal{L} \cap \mathcal{K}$, for which any of the following reformulations are possible candidates for computing a solution via an accelerated first-order method:

$$\min\{f_1(u) := \|\mathcal{E}u - q\|_2^2 : \ u \in \mathcal{K}\}, \tag{2.12}$$

$$\min\{f_2(u) := d^2_{\mathcal{L}}(u) + d^2_{\mathcal{K}}(u) : \ u \in \mathbb{U}\}, \tag{2.13}$$

$$\min\{f_3(u) := d^2_{\mathcal{L}}(u) : \ u \in \mathcal{K}\}, \ \text{and/or} \tag{2.14}$$

$$\min\{f_4(u) := d^2_{\mathcal{K}}(u) : \ u \in \mathcal{L}\}. \tag{2.15}$$

Here $d_{\mathcal{L}}$ and $d_{\mathcal{K}}$ are the distance functions defined in Section 1.1. Formulation (2.12) was first proposed in [9], (2.13) was studied in [12], and (2.14) and (2.15) were proposed in [15]. Scheme (2.15) is the most efficient from the vantage point of worst-case complexity analysis, especially when $\|\mathcal{E}\|$ is large, see [15]. However, when applied to the class of polynomial optimization problems under consideration herein, we have observed that (2.14) converges faster than (2.15), as we look at a converging sequence different from the one in the theoretical analysis (see Section 3.3 for more discussion). We therefore have chosen scheme (2.14) for implementation and extensive computational experiments, which we report in Section 3 with further discussion.

2.3 An Accelerated First-Order Method

We wish to solve problem (2.14), namely:

$$\begin{array}{ll} \min & f(u) := d^2_{\mathcal{L}}(u) \\ \text{s.t.} & u \in \mathcal{K} \ , \end{array} \tag{2.16}$$

which, under the presumption that $p(x) - \gamma^*$ is SOS, is an equivalent reformulation of (2.5)-(2.8) and hence of (2.3)-(2.4). Here the objective function is the squared distance function of a point to the affine subspace $\mathcal{L}$. As introduced in Section 1.1, $d^2_{\mathcal{L}}(u)$ has Lipschitz continuous derivative with Lipschitz constant $L = 2$. We use the following accelerated first-order method (AFOM), which was first presented and analyzed in [1] in a more general setting, but is stated here as presented for the specific setting of solving (2.16) treated in [15]:

```
    Accelerated First-Order Method (AFOM):
(0) Choose $\bar{u}_0 = \tilde{u}_0 \in \mathcal{K}$, $k \leftarrow 0$.
(1) Compute $u_k := \frac{2}{k+2}\bar{u}_k + \frac{k}{k+2}\tilde{u}_k$.
(2) Compute $(\bar{u}_{k+1}, \tilde{u}_{k+1}) \in \mathcal{K} \times \mathcal{K}$ as follows:
      $\bar{u}_{k+1} := \Pi_{\mathcal{K}}(\bar{u}_k - \frac{k+2}{2}(u_k - \Pi_{\mathcal{L}}(u_k)))$,
      $\tilde{u}_{k+1} := \frac{2}{k+2}\bar{u}_{k+1} + \frac{k}{k+2}\tilde{u}_k$.
(3) $k \leftarrow k+1$, and go to (1).
```

Note that each iteration of AFOM requires the computation of two projections: $\Pi_{\mathcal{L}}(u)$ which projects u onto the affine subspace $\mathcal{L}$, and $\Pi_{\mathcal{K}}(u)$ which projects $u = (X, S, u)$ onto the cone $\mathcal{K} := \mathbb{S}^N_+ \times \mathbb{S}^N_+ \times \mathbb{R}^{M-1}$. Computing $\Pi_{\mathcal{K}}(u)$ requires computing an eigenvalue decomposition of the two symmetric matrices X and S; fortunately existing numerical packages are quite efficient for this task, see for example the eig function in Matlab, which works very effectively for matrices with several thousand rows/columns. Computing the projection $\Pi_{\mathcal{L}}(u)$ is typically quite costly, but as we show herein in Section 2.4, the special structure of the unconstrained polynomial optimization problem can be exploited to substantially speed up the computation of this projection.

Algorithm AFOM has the following convergence guarantee, originally proved in [1] but stated below from [15] in the context of problem (2.16):

Theorem 1. [1], [15]. *Let $\{\tilde{u}_k\}$ be the sequence generated by algorithm AFOM applied to (2.16) and let u^* be any optimal solution of (2.16). For any given $\varepsilon > 0$, $d_{\mathcal{L}}(\tilde{u}_k) \le \varepsilon$ can be obtained in at most $\left\lceil \frac{2\|u^* - \tilde{u}_0\|}{\varepsilon} \right\rceil$ iterations.*

2.4 Special Structure of (P) and Improved Efficiency of Projection onto $\mathcal{L}$

In this subsection, we identify and then exploit special structure of the SDP formulation of the unconstrained polynomial optimization problem, namely the orthogonality property of the basic moment matrices. This special structure enables a significant speed-up of the projection $\Pi_{\mathcal{L}}$ used in the accelerated first-order method. We demonstrate easily computable formulas for $\Pi_{\mathcal{L}}$ and we demonstrate that this computation uses $O(N^2)$, which represents a speed-up of $O(M^2)$ operations over the traditional cost (namely $O(M^2N^2)$) for computing the projection.

Lemma 1. *For all $\alpha \in \mathbb{N}^n_{2m}$, the basic moment matrices A_α are orthogonal to one another. The linear operator $\mathcal{D} := \mathcal{A}\mathcal{A}^* : \mathbb{R}^{M-1} \to \mathbb{R}^{M-1}$ is diagonal, whose α-th diagonal component is $\mathcal{D}_\alpha = A_\alpha \bullet A_\alpha$ and satisfies $1 \leq \mathcal{D}_\alpha \leq N$ for all $\alpha \in \mathbb{N}^n_{2m}$ satisfying $|\alpha| > 0$. Therefore*

1. $\mathcal{A}(A_0) = 0$, *equivalently* $\mathcal{A}\mathcal{A}_0^* = 0$,
2. $(\mathcal{A}\mathcal{A}^*)^{-1} = \mathcal{D}^{-1}$
3. $(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1} = \mathcal{I}_1 - \mathcal{A}^*(\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A}$,
4. $\mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1} = (\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A}$, *and* $\mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(A_0) = 0$,
5. $(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^* = \mathcal{A}^*(\mathcal{I}_2 + \mathcal{D})^{-1}$,
6. $\mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^* = \mathcal{D}(\mathcal{I}_2 + \mathcal{D})^{-1}$, *and*
7. $\mathcal{I}_2 - \mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^* = (\mathcal{I}_2 + \mathcal{D})^{-1}$,

where the identity operators are defined as $\mathcal{I}_1 : \mathbb{S}^N \to \mathbb{S}^N$ and $\mathcal{I}_2 : \mathbb{R}^{M-1} \to \mathbb{R}^{M-1}$.

Proof. Recall the definition of the basic moment matrix A_α: for all $\delta, \nu \in \mathbb{N}^n_m$, the (δ, ν)-entry of A_α is

$$A_\alpha(\delta, \nu) = \begin{cases} 1 & \text{if } \delta + \nu = \alpha, \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Therefore, the basic moment matrices are orthogonal to each other, i.e., for all $\alpha, \beta \in \mathbb{N}^n_{2m}$,

$$A_\alpha \bullet A_\beta = \begin{cases} 0 & \alpha \neq \beta, \\ n_\alpha \geq 1 & \alpha = \beta, \end{cases} \tag{2.18}$$

where n_α is the number of 1's in the matrix A_α. Let $z := \mathcal{A}\mathcal{A}^*(y)$, then the α-entry of z is given by:

$$z_\alpha = \sum_\nu y_\nu A_\nu \bullet A_\alpha = n_\alpha y_\alpha. \tag{2.19}$$

Therefore $\mathcal{D} := \mathcal{A}\mathcal{A}^*$ is a diagonal operator, in which the α-entry on the diagonal is $\mathcal{D}_\alpha = n_\alpha$. Since $n_\alpha = A_\alpha \bullet A_\alpha$, it follows that n_α is equal to the number of different ways that α can be decomposed as $\alpha = \delta + \nu$ for $\delta, \nu \in \mathbb{N}^n_m$. Since there are in total N different δ's in $\mathbb{N}^n_m$, it also follows that $1 \leq n_\alpha \leq N$.

Item (1.) follows from the orthogonality of the basic moment matrices, and (2.) follows since $\mathcal{D}_\alpha = n_\alpha \geq 1$ for all $\alpha \in \mathbb{N}^n_{2m}$. Item (3.) follows from the Sherman-Morrison-Woodbury identity:

$$(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1} = \mathcal{I}_1 - \mathcal{A}^*(\mathcal{I}_2 + \mathcal{A}\mathcal{A}^*)^{-1}\mathcal{A} \tag{2.20}$$

and then substituting in (2.). To prove the first part of (4.) we premultiply (3.) by $\mathcal{A}$ and derive:

$$\mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1} = \mathcal{A}(\mathcal{I}_1 - \mathcal{A}^*(\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A}) = \mathcal{A} - \mathcal{D}(\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A} = (\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A},$$

and the second part of (4.) follows the first part using $\mathcal{A}\mathcal{A}_0^* = 0$ from (1.). Item (5.) is simply the adjoint of the first identity of (4.). Item (6.) follows from (5.) by premultiplying (5.) by $\mathcal{A}$, and (7.) follows from (6.) by rearranging and simplifying terms. □

Remark 1. *The orthogonality of basic moment matrices is implied by an even stronger property of these matrices involving their support indices. Let S_α denote the* support *of A_α, namely $S_\alpha := \{(\delta, \nu) \in \mathbb{N}^n_m \times \mathbb{N}^n_m \ : \ A_\alpha(\delta, \nu) \neq 0\}$. Then it follows that $S_\alpha \cap S_\beta = \emptyset$ for $\alpha \neq \beta$, which in turn implies the orthogonality of basic moment matrices.*

Lemma 1 allows improved computation of the projection $\Pi_\mathcal{L}$, as we now show.

Lemma 2. *Let $u = (X, S, y) \in \mathbb{U}$ be given, and let $(\bar{X}, \bar{S}, \bar{y}) := \Pi_\mathcal{L}(u)$. Then the following is a valid method for computing $(\bar{X}, \bar{S}, \bar{y})$. First compute the scalars:*

$$\bar{\xi} \leftarrow 1 + p^T(\mathcal{I}_2 + \mathcal{D})^{-1}p, \tag{2.21}$$

$$\bar{\eta} \leftarrow A_0 \bullet X - p^T(\mathcal{I}_2 + \mathcal{D})^{-1}y + p^T(\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A}(S), \tag{2.22}$$

$$\bar{r} \leftarrow \bar{\eta}/\bar{\xi}. \tag{2.23}$$

Then compute $\Pi_\mathcal{L}(u)$ as follows:

$$\bar{X} = X - \mathcal{A}^*\mathcal{D}^{-1}(\mathcal{A}(X) - p) - \bar{r}A_0, \tag{2.24}$$

$$\bar{S} = \mathcal{A}^*(\mathcal{I}_2 + \mathcal{D})^{-1}(\mathcal{A}(S) - y - \bar{r}p) + A_0, \tag{2.25}$$

$$\bar{y} = (\mathcal{I}_2 + \mathcal{D})^{-1}(-\mathcal{A}(S) + y + \bar{r}p). \tag{2.26}$$

Proof. Recalling the definition of $\mathcal{E}$ in (2.11), we note that $\bar{u} = (\bar{X}, \bar{S}, \bar{y})$ along with Lagrange multipliers $\bar{\lambda} = (\bar{w}, \bar{V}, \bar{r}) \in \mathbb{R}^{M-1} \times \mathbb{S}^N \times \mathbb{R}$ are a solution of the following normal equations:

$$\mathcal{E}(\bar{u}) = q \quad \text{and} \quad u - \bar{u} = \mathcal{E}^* \bar{\lambda}. \tag{2.27}$$

Indeed, it is sufficient to determine $\bar{\lambda}$ that solves:

$$\mathcal{E}\mathcal{E}^* \bar{\lambda} = \mathcal{E}u - q, \tag{2.28}$$

and then to assign

$$\bar{u} = u - \mathcal{E}^* \bar{\lambda}. \tag{2.29}$$

Using (2.11) and the results in Lemma 1, we obtain:

$$\mathcal{E}\mathcal{E}^* = \begin{bmatrix} \mathcal{A}\mathcal{A}^* & 0 & \mathcal{A}\mathcal{A}_0^* \\ 0 & \mathcal{I}_1 + \mathcal{A}^*\mathcal{A} & -\mathcal{A}^*(p) \\ \mathcal{A}_0\mathcal{A}^* & -p^T\mathcal{A} & \mathcal{A}_0\mathcal{A}_0^* + p^Tp \end{bmatrix} = \begin{bmatrix} \mathcal{A}\mathcal{A}^* & 0 & 0 \\ 0 & \mathcal{I}_1 + \mathcal{A}^*\mathcal{A} & -\mathcal{A}^*(p) \\ 0 & -p^T\mathcal{A} & 1 + p^Tp \end{bmatrix}, \tag{2.30}$$

therefore the system (2.28) can be written as:

$$\begin{bmatrix} \mathcal{A}\mathcal{A}^* & 0 & 0 \\ 0 & \mathcal{I}_1 + \mathcal{A}^*\mathcal{A} & -\mathcal{A}^*(p) \\ 0 & -p^T\mathcal{A} & 1 + p^Tp \end{bmatrix} \begin{bmatrix} \bar{w} \\ \bar{V} \\ \bar{r} \end{bmatrix} = \begin{bmatrix} \mathcal{A}(X) - p \\ S + \mathcal{A}^*(y) - A_0 \\ A_0 \bullet X - p^Ty \end{bmatrix}. \tag{2.31}$$

It is straightforward to check that the following computations involving auxiliary scalars $\bar{\xi}$ and $\bar{\eta}$ yield a solution to (2.31):

$$\bar{w} \leftarrow (\mathcal{A}\mathcal{A}^*)^{-1}(\mathcal{A}(X) - p) \tag{2.32}$$

$$\bar{\xi} \leftarrow 1 + p^Tp - p^T\mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^*(p) \tag{2.33}$$

$$\bar{\eta} \leftarrow A_0 \bullet X - p^Ty + p^T\mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(S + \mathcal{A}^*(y) - A_0) \tag{2.34}$$

$$\bar{r} \leftarrow \bar{\eta}/\bar{\xi} \tag{2.35}$$

$$\bar{V} \leftarrow (\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(S + \mathcal{A}^*y - A_0 + \mathcal{A}^*(p)\bar{r}). \tag{2.36}$$

We then use (2.29) to compute $\bar{u} = (\bar{X}, \bar{S}, \bar{y})$ as follows:

$$\bar{X} \leftarrow X - \mathcal{A}^*\bar{w} - \bar{r}A_0 = X - \mathcal{A}^*(\mathcal{A}\mathcal{A}^*)^{-1}(\mathcal{A}(X) - p) - \bar{r}A_0 \tag{2.37}$$

$$\bar{S} \leftarrow S - \bar{V} = S - (\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(S - A_0 + \mathcal{A}^*(y + \bar{r}p)) \tag{2.38}$$

$$\bar{y} \leftarrow y - \mathcal{A}\bar{V} + \bar{r}p = y + \bar{r}p - \mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(S + \mathcal{A}^*(y + \bar{r}p) - A_0) \tag{2.39}$$

Summarizing, we have shown that (2.32)-(2.39) solve the normal equations (2.27) and hence (2.37)-(2.39) are valid formulas for the projection $\Pi_{\mathcal{L}}(u)$. We now show that these formulas can alternatively be computed using (2.21)-(2.26). To derive (2.21), we have from (2.33) and (7.) of Lemma 1:

$$\bar{\xi} = 1 + p^T[\mathcal{I}_2 - \mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^*]p = 1 + p^T(\mathcal{I}_2 + \mathcal{D})^{-1}p \ .$$

To derive (2.22), we have from (2.34) and (7.) and (4.) of Lemma 1:

$$\bar{\eta} = A_0\bullet X - p^T[\mathcal{I}_2 - \mathcal{A}(\mathcal{I}_1+\mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^*]y + p^T\mathcal{A}(\mathcal{I}_1+\mathcal{A}^*\mathcal{A})^{-1}(S-A_0) = A_0\bullet X - p^T(\mathcal{I}_2+\mathcal{D})^{-1}y + p^T(\mathcal{I}_2+\mathcal{D})^{-1}\mathcal{A}(S).$$

Equations (2.23) and (2.35) are identical, and to derive (2.24), we have from (2.37) and (2.) of Lemma 1:

$$\bar{X} = X - \mathcal{A}^*\mathcal{D}^{-1}(\mathcal{A}(X) - p) - \bar{r}A_0.$$

To derive (2.25), we have from (2.38) and (5.), (3.), and (1.) of Lemma 1:

$$\begin{aligned} \bar{S} &= S - (\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(S - A_0 + \mathcal{A}^*(y + \bar{r}p)) \\ &= [\mathcal{I}_1 - (\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}](S) - (\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^*(y + \bar{r}p) + (\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(A_0) \\ &= \mathcal{A}^*(\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A}(S) - \mathcal{A}^*(\mathcal{I}_2 + \mathcal{D})^{-1}(y + \bar{r}p) + A_0. \end{aligned}$$

To derive (2.26), we have from (2.39) and (7.) and (4.) of Lemma 1:

$$\begin{aligned} \bar{y} &= [\mathcal{I}_2 - \mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}\mathcal{A}^*](y + \bar{r}p) - \mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(S) + \mathcal{A}(\mathcal{I}_1 + \mathcal{A}^*\mathcal{A})^{-1}(A_0) \\ &= (\mathcal{I}_2 + \mathcal{D})^{-1}(y + \bar{r}p) - (\mathcal{I}_2 + \mathcal{D})^{-1}\mathcal{A}(S). \end{aligned}$$

□

Remark 2. *The above computations are numerically well-behaved. To see this, note that there are no explicit matrix solves, so the issue of solving simultaneous equations is avoided. Second, note that divisors involve either $\bar{\xi}$ or the (diagonal) components $\mathcal{D}_\alpha$ of $\mathcal{D}$. From (2.21) we see that $\bar{\xi}$ is bounded from above and below as*

$$1 \le \bar{\xi} \le 1 + \frac{1}{2}\|p\|^2 \ , \tag{2.40}$$

whereby division by $\bar{\xi}$ is well-behaved relative to the data p. It also follows from Lemma 1 that

$$1 \le \mathcal{D}_\alpha \le N, \tag{2.41}$$

whereby working with $\mathcal{D}^{-1}$ and/or $(\mathcal{I}_2 + \mathcal{D})^{-1}$ are also well-behaved relative to the dimension N of the SDP.

The computational cost of computing $\Pi_{\mathcal{L}}(u)$ using Lemma 2 is bounded from above using the following lemma.

Lemma 3. *The computation of* $\Pi_{\mathcal{L}}(u)$ *requires* $O(N^2)$ *arithmetic operations.*

Proof. From Lemma 2, the computation of $\Pi_{\mathcal{L}}(u)$ involves computations with $\mathcal{A}(\cdot)$, $\mathcal{A}^*(\cdot)$, $\mathcal{D}^{-1}(\cdot)$, $(\mathcal{I}_2 + \mathcal{D})^{-1}(\cdot)$, matrix additions, and vector inner products. These last two involve $O(N^2)$ and $O(M)$ arithmetic operations, respectively. From Remark 1, we observe that $\mathcal{A}(X)$ adds the entries of X and accesses each entry exactly once, so is $O(N^2)$ additions. Similarly, $\mathcal{A}^*(y)$ forms an $N \times N$ matrix by assigning the entries of y to specific locations, and involves $O(N^2)$ operations. Since $\mathcal{D}$ is diagonal and has M positive diagonal elements, $\mathcal{D}^{-1}(\cdot)$ and $(\mathcal{I}_2 + \mathcal{D})^{-1}(\cdot)$ involve M multiplications. Finally, since $N^2 \geq M$ (this can be seen from the fact that $\mathcal{A}^*(y)$ maps exclusive components of y into entries of an $N \times N$ matrix), the computation of $\Pi_{\mathcal{L}}(u)$ requires $O(N^2)$ arithmetic operations. □

Let us now contrast the above bound with the more general case. Consider a general SDP problem with primal-and-dual decision variables in $\mathbb{S}^N \times \mathbb{S}^N \times \mathbb{R}^{M-1}$. From (2.28), computing $\Pi_{\mathcal{L}}$ involves (at a minimum) solving systems of equations with the operator $\mathcal{A}\mathcal{A}^*$, which requires $O(M^2N^2)$ operations to form and $O(M^3)$ operations to solve, so the total computational effort is $O(M^3 + M^2N^2) = O(M^2N^2)$ operations. Here we see that by identifying and exploiting the structure of the SDP for unconstrained polynomial optimization, we reduce the computational cost of computing $\Pi_{\mathcal{L}}(u)$ by a factor of $O(M^2)$. This saving is substantial especially for high degree polynomials, since $M = \binom{n+d}{n} = O(n^d)$ grows exponentially in the degree d of the polynomial.

3 Computational Results, and Discussion

In this section, we present computational results for the accelerated first-order method AFOM and compare these results to those of the SeDuMi `sossolve` interior-point method (IPM) on unconstrained polynomial optimization problems of degree $d = 4$ and $d = 6$. The AFOM is implemented in Matlab R2008, and the SeDuMi package is version 1.1. The computational platform is a laptop with an Intel Core(TM) 2Duo 2.50GHz CPU and 3GB memory.

3.1 Generating instances of polynomials

We generated instances of polynomials $p(x)$ in n variables and of degree $d = 2m$ for $m = 2$ and $m = 3$. Each instance was generated randomly as follows. We first chose a random point x^* uniformly distributed in $[-1,1]^n$, and then generated the coefficients of k polynomials $q_i(x)$ in n variables and degree m for $i=1,2,\ldots,k$, using a uniform distribution on $[0,3]$. We then set $\gamma^* := -\sum_{i=1}^k q_i(x^*)^2$ and $p(x)=\sum_{i=1}^k (q_i(x)-q_i(x^*))^2+\gamma^*$. The resulting polynomial $p(x)$ satisfies $\gamma^*=\min_{x\in\mathbb{R}^n} p(x)$ and $p(x)-\gamma^*$ is an SOS polynomial, and furthermore $p(x)$ has zero constant term since $p(0)=0$. We set $k=4$ for all instances. The resulting instances have fully dense coefficient vectors $p\in\mathbb{R}^{M-1}$. When generating the 6-degree polynomials, we wanted the polynomials to have a density of coefficients of 10% on average. This was accomplished by suitably controlling the density of the coefficients of the polynomials $q_i(x)$ for $i=1,2,\ldots,k$.

3.2 Algorithm Stopping Rules

Both AFOM and the SeDuMi IPM seek to solve the primal/dual SDP instance pair (2.3)-(2.4). We use SeDuMi's stoppping rule, see [25], which we briefly review herein. Using the notation of conic convex optimization, we re-write the primal and dual (2.3)-(2.4) in the standard linear conic optimization format:

$$\begin{array}{lll} (P) & \min\limits_{x} \; c^T x \;\; \text{s.t.} \;\; Ax=b, \; x\in C \; , \\ (D) & \max\limits_{y,z} \; b^T y \;\; \text{s.t.} \;\; A^*y+z=c, \; z\in C^* \; , \end{array}$$

with correspondence $b\equiv p$, $Ax\equiv \mathcal{A}(X)$, $c^Tx\equiv A_0\bullet X$, and $C, C^*\equiv \mathbb{S}^N_+$. SeDuMi solves a homogeneous self-dual (HSD) embedding model until the primal infeasibility, dual infeasibility, and the duality gap of the current trial iterate $(\hat{x},\hat{y},\hat{z})$ (which automatically satisfies the cone inclusions $\hat{x}\in C$, $\hat{z}\in C^*$) are small. More specifically, define:

$$\begin{array}{ll} r_p := b - A\hat{x} & \text{(primal infeasibility gap)} \\ r_d := A^T\hat{y}+\hat{z}-c & \text{(dual infeasibility gap)} \\ r_g := c^T\hat{x}-b^T\hat{y} & \text{(duality gap)} \; . \end{array}$$

SeDuMi's stopping criterion is:

$$2\frac{\|r_p\|_\infty}{1+\|b\|_\infty}+2\frac{\|r_d\|_\infty}{1+\|c\|_\infty}+\frac{(r_g)^+}{\max(|c^T\hat{x}|,|b^T\hat{y}|,0.001\hat{\tau})}\le\varepsilon \; .$$

Here ε is the specified tolerance. The denominator of the third term above depends on the value $\hat{\tau}$ of a scaling variable τ in the HSD model, see [25]. Since the AFOM does not have any such scaling variable, we modify the above stopping rule when applied to AFOM to be:

$$2\frac{\|r_p\|_\infty}{1+\|b\|_\infty}+2\frac{\|r_d\|_\infty}{1+\|c\|_\infty}+\frac{(r_g)^+}{\max(|c^T\hat{x}|,|b^T\hat{y}|)}\le\varepsilon.$$

Note that the stopping criterion for AFOM is (marginally) more conservative than that of IPM due to the absence of the component involving $\hat{\tau}$ in the denominator of the third term above. The stopping rule tolerance ε is set to 10^{-4} for all of our computational experiments. Returning to the notation of the SDP primal and dual problems (2.3)-(2.4) and letting $u = (X, S, y) \in \mathcal{K}$ be a current iterate of either AFOM or IPM, then our output of interest is Objective Gap $:= |p^T y + \gamma^*|$, which measures the objective function value gap between the current dual (near-feasible) solution, namely $(-p^T y)$, and the true global optimal value of the polynomial γ^*. For AFOM, according to Theorem 1, the iterate sequence $\tilde{u}_k$ satisfies the convergence bound, which suggests we should compute Objective Gap using $|p^T \tilde{y}_k + \gamma^*|$. However, we have observed that $d_{\mathcal{L}}(\bar{u}_k)$ also consistently obeys the theoretical convergence rate of Theorem 1, and that both $d_{\mathcal{L}}(\bar{u}_k)$ and $|p^T \bar{y}_k + \gamma^*|$ converge to zero much faster than does $d_{\mathcal{L}}(\tilde{u}_k)$ and $|p^T \tilde{y}_k + \gamma^*|$. For this reason we compute Objective Gap using $|p^T \tilde{y}_k + \gamma^*|$ and report our results using the $\bar{u}_k = (\bar{X}_k, \bar{S}_k, \bar{y}_k)$ sequence.

Our computational results for 4-degree and 6-degree polynomials are presented in Tables 1 and Table 2, respectively. In each of these tables, the left-most column displays the dimension n of the optimization variable x, followed by the size dimensions N and M for the SDP instances (2.3)-(2.4). The fourth column shows the number of sample instances that were generated for each value of n, where of necessity we limited the number of sample instances when the dimension n of the problems (and hence the dimensions N and M of the SDP problems (2.3)-(2.4)) were larger.

For small size problems, IPM runs faster than AFOM. However, AFOM computation time dominates IPM computation time for $n \geq 12$ and $n \geq 8$ for 4-degree polynomials and 6-degree polynomials, respectively. More importantly, since AFOM has much lower memory requirements, AFOM is able to solve problems in much larger dimension n than IPM (about twice as large for both 4-degree and 6-degree polynomials). This extends the practical solvability of the SDP representation of the polynomial optimization problems. For 4-degree polynomials, IPM produces slightly better Objective Gap values, but no such difference is discernable for 6-degree polynomials. Figure 1 and Figure 2 illustrate the computation time results in graphical form for 4-degree and 6-degree polynomials, respectively.

One of the generic advantages of interior-point methods is the very low variability in the number of iterations, and hence in computational burden. By contrast, AFOM has higher variability in terms of both computation time and Objective Gap values. Such variability is illustrated in the histogram plots in Figure 3. The histograms in subfigures (a) and (b) show AFOM computation time and Objective Gap values for the $s = 50$ sample instances of 4-degree polynomials in dimension $n = 19$, respectively. The histograms in subfigures (c) and (d) show AFOM computation time and Objective Gap values for

				AFOM Performance						IPM Performance				
	SDP Size			Time (seconds)		Objective Gap		Iterations		Time (seconds)		Objective Gap		Iterations
n	N	M	s	Median	Average	Median	Average	Median	Average	Median	Average	Median	Average	Median
2	6	15	100	2.96E-1	3.9E-1	1.77E-3	1.62E-3	820.0	1068.2	1.13	1.14	1.83E-4	3.98E-4	7
3	10	35	100	1.04	1.66	2.57E-3	2.17E-3	2113.5	3276.9	1.17	1.18	5.70E-4	7.50E-4	8
4	15	70	100	6.35	7.29	4.93E-3	5.15E-3	8416.0	9599.5	1.22	1.22	8.51E-4	1.35E-3	9
5	21	126	100	52.69	36.92	9.10E-3	1.36E-3	19955.5	13661.5	1.40	1.40	4.20E-4	8.63E-4	10
6	28	210	100	11.09	30.22	1.36E-2	8.10E-3	2432.5	6662.8	1.53	1.52	4.45E-4	9.38E-4	10
7	36	330	100	11.31	28.06	8.12E-3	1.18E-2	1624.0	3974.4	1.83	1.84	4.06E-4	9.35E-4	10
8	45	495	100	12.56	23.42	3.09E-3	8.19E-3	1223.5	2311.9	2.57	2.59	6.46E-4	1.09E-3	11
9	55	715	100	14.56	23.22	1.41E-3	4.13E-3	952.0	1151.3	6.86	6.85	6.06E-4	1.36E-3	11
10	66	1001	100	21.42	27.61	1.42E-3	2.93E-3	970.0	1282.1	13.19	13.10	5.21E-4	1.06E-3	11
11	78	1365	100	33.24	40.55	1.74E-3	3.05E-3	1139.5	1401.5	27.40	27.38	5.12E-4	1.28E-3	11
12	91	1820	100	55.17	60.18	1.76E-3	2.76E-3	1240.0	1389.7	64.07	63.54	7.32E-4	1.25E-3	11
13	105	2380	100	82.70	91.12	1.49E-3	2.37E-3	1398.0	1530.0	139.11	138.90	4.09E-4	1.34E-3	12
14	120	3060	100	128.08	141.19	3.90E-3	2.83E-3	1609.0	1781.5	287.70	294.00	8.06E-4	2.53E-3	12
15	136	3876	50	168.99	184.77	3.66E-3	4.63E-3	1693.5	2225.6	575.60	592.70	5.42E-4	2.63E-3	12
16	153	4845	50	263.76	312.94	3.09E-3	6.09E-3	1963.0	2325.8	\	\	\	\	\
17	171	5985	50	385.48	414.21	3.03E-3	5.70E-3	2170.5	2357.6	\	\	\	\	\
18	190	7315	50	586.72	664.94	4.35E-4	6.48E-3	2521.0	2912.0	\	\	\	\	\
19	210	8855	50	833.10	923.29	5.45E-3	9.45E-3	2741.5	3002.7	\	\	\	\	\
20	231	10626	20	1083.90	1125.41	4.60E-3	9.93E-3	3011.0	3097.5	\	\	\	\	\
25	351	23751	15	4682.34	5271.75	8.32E-3	2.13E-2	4595.0	5116.7	\	\	\	\	\
30	496	46376	15	21323.53	21837.43	3.96E-5	5.28E-5	6466.7	6351.5	\	\	\	\	\

Table 1: Computational results for degree-4 polynomials. n is the dimension of the variable x, and s is the number of sample instances. The stopping rule tolerance is $\varepsilon = 10^{-4}$ for both AFOM and IPM. For $n > 15$, IPM fails due to memory requirements.

				AFOM Performance						IPM Performance				
	SDP Size			Time (seconds)		Objective Gap		Iterations		Time (seconds)		Objective Gap		Iterations
n	N	M	s	Median	Average	Median	Average	Median	Average	Median	Average	Median	Average	Median
2	10	28	100	0.47	2.00	3.94E-5	1.14E-4	753.0	3398.9	1.76	1.76	2.14E-5	6.94E-5	7
3	20	84	100	5.09	8.63	8.50E-5	1.72E-4	2384.5	4053.3	1.75	1.75	1.21E-4	2.40E-4	9
4	35	210	100	29.23	37.74	5.41E-4	6.24E-4	5227.5	6719.8	1.93	1.94	4.23E-4	5.98E-4	10
5	56	462	100	78.15	101.38	1.38E-3	1.35E-3	5924.5	7691.5	3.23	3.79	1.89E-4	3.59E-4	10
6	84	924	100	48.76	79.28	1.45E-3	1.36E-3	1529.0	2420.5	10.19	10.13	1.26E-4	2.47E-4	10
7	120	1716	100	48.23	67.14	1.67E-4	8.96E-4	709.5	995.2	49.94	47.90	2.26E-4	3.62E-4	10
8	165	3003	100	67.01	78.35	9.0E-5	3.81E-4	429.5	506.7	231.50	231.53	2.56E-4	5.42E-4	10
9	220	5005	100	123.34	137.42	1.03E-4	3.13E-4	372.5	427.6	1048.44	1063.78	2.64E-4	6.07E-4	11
10	286	8008	60	229.26	250.46	1.50E-4	3.96E-4	404.6	373.0	\	\	\	\	\
11	364	12376	50	471.12	478.16	1.38E-4	3.73E-4	439.5	448.3	\	\	\	\	\
12	455	18564	50	778.63	835.25	1.83E-4	6.49E-4	478.0	516.0	\	\	\	\	\
13	560	27132	10	1380.52	1404.66	9.83E-5	2.16E-4	521.5	538.7	\	\	\	\	\
14	680	38760	10	2944.96	3002.31	8.31E-4	1.01E-3	707.5	725.5	\	\	\	\	\
15	816	54264	10	5393.01	5754.98	5.64E-4	7.91E-4	799.5	859.2	\	\	\	\	\
16	969	74613	10	10041.98	10703.34	6.91E-4	1.40E-3	962.0	1017.3	\	\	\	\	\

Table 2: Computational results for degree-6 polynomials. n is the dimension of the variable x, and s is the number of sample instances. The stopping rule tolerance is $\varepsilon = 10^{-4}$ for both AFOM and IPM. For $n > 9$, IPM fails due to memory requirements.

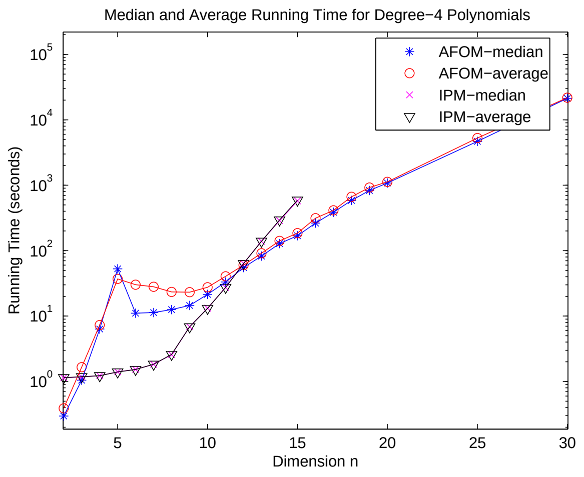


Figure 1: Graph of the median and average running times (in seconds) of AFOM and IPM for degree-4 polynomials in n variables. The values for IPM end in the middle of the graph where IPM fails due to excessive memory requirements.

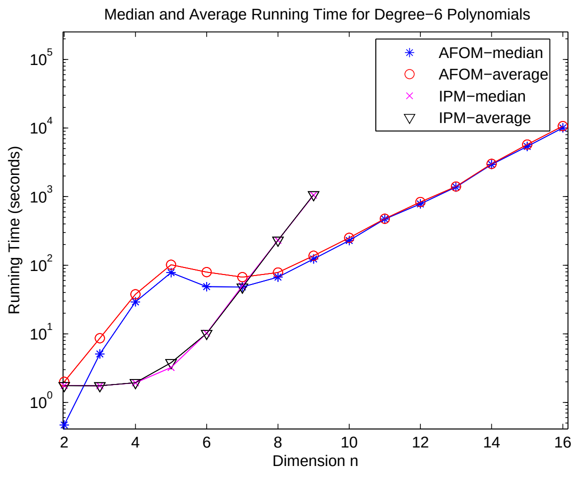
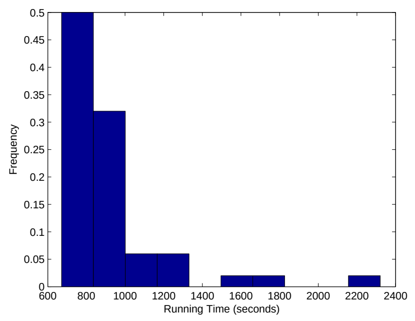
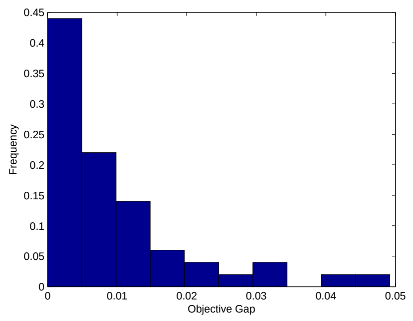


Figure 2: Graph of the median and average running times (in seconds) of AFOM and IPM for degree-6 polynomials in n variables. The values for IPM end in the middle of the graph where IPM fails due to excessive memory requirements.

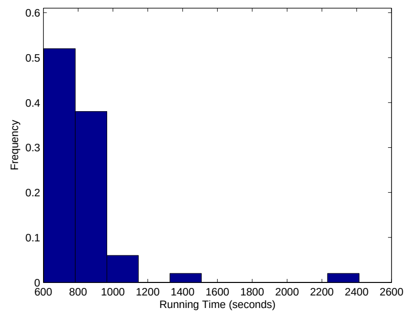
the $s = 50$ sample instances of 6-degree polynomials in dimension $n = 12$, respectively. Observe that tails and outliers contribute to the higher variation in the performance AFOM in these two metrics.



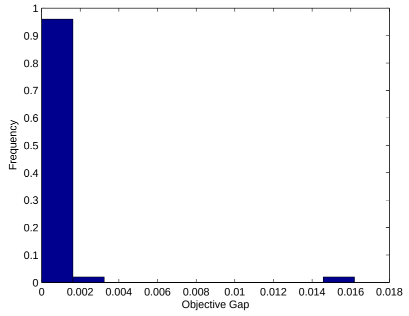
(a) Histogram of AFOM running times for 4-degree polynomials in dimension $n = 19$.



(b) Histogram of AFOM Objective Gap values for 4-degree polynomials in dimension $n = 19$.



(c) Histogram of AFOM running time for 6-degree polynomials in dimension $n = 12$.



(d) Histogram of AFOM Objective Gap values for 6-degree polynomials in dimension $n = 12$.

Figure 3: Histograms illustrating variability in running times (Subfigures (a) and (c)) and Objective Gap values (Subfigures (b) and (d)) of AFOM for 4-degree polynomials and 6-degree polynomials.

3.3 A curious observation

In its simplest form, a traditional first-order (gradient) algorithm produces one sequence of points, where the next point is determined from the current point by taking a step in the direction of the negative of the gradient of the current point, according to some specified step-size rule. A crucial difference between the modern class of accelerated first-order methods and the traditional algorithm is that the accelerated first-order methods involve more than one sequence in the gradient descent scheme. In [27, page 80], Nesterov devised a very simple accelerated first-order method with two sequences. The method used in this paper is a variant of Nesterov's original algorithm, and constructs three sequences–namely $\{u_k\}$, $\{\bar{u}_k\}$, and $\{\tilde{u}_k\}$. The convergence analysis establishes the $O(1/\varepsilon)$ convergence rate of the objective function in terms of the sequence values of $\{\tilde{u}_k\}$. From our computational experiments, we observe that both $d_{\mathcal{L}}(\tilde{u}_k)$ and $d_{\mathcal{L}}(\bar{u}_k)$ closely follow the theoretical rate $O(1/\varepsilon)$.

For the unconstrained polynomial optimization problem, we are ultimately interested in the convergence of $(-p^T\tilde{y}_k)$ to the global optimal value γ^* of the polynomial $p(x)$, and of $d_{\mathcal{L}}(\tilde{u}_k)$ to zero. However, we have consistently observed in our experiments that both $(-p^T\bar{y}_k)$ and $d_{\mathcal{L}}(\bar{u}_k)$ exhibit much faster convergence to their respective limits than does $(-p^T\tilde{y}_k)$ and $d_{\mathcal{L}}(\tilde{u}_k)$, with the difference being more apparent for larger-scale problems (i.e., when n is larger). We illustrate this observation in Figure 4, which shows typical convergence behavior of the two sequences for one of the problem instances of 4-degree polynomials for $n = 20$. We have also observed that when $p^T\bar{y}_k$ converges, it converges to a very high accuracy, which is also illustrated in Figure 4. We do not yet have either a satisfactory theoretical or practical explanation of this phenomenon.

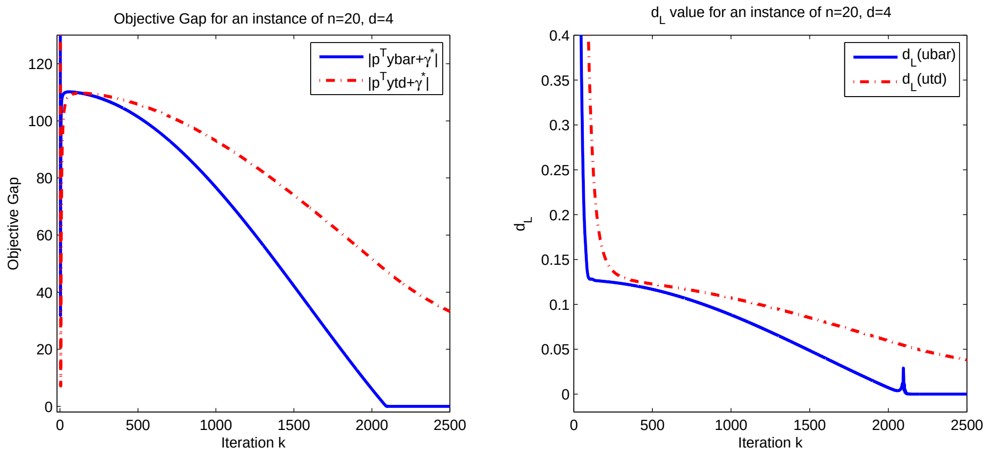


Figure 4: Objective Gap values (left) and $d_{\mathcal{L}}(\cdot)$ values (right) for a 4-degree polynomial instance for $n = 20$. In the left figure, the solid curve plots $|p^T\bar{y}_k + \gamma^*|$ and the dashed curve plots $|p^T\tilde{y}_k + \gamma^*|$. In the right figure, the solid curve plots $d_{\mathcal{L}}(\bar{u}_k)$ and the dashed curve plots $d_{\mathcal{L}}(\tilde{u}_k)$. As shown, $|p^T\bar{y}_k + \gamma^*|$ and $d_{\mathcal{L}}(\bar{u}_k))$ converge to zero much faster than does $|p^T\tilde{y}_k + \gamma^*|$ and $d_{\mathcal{L}}(\tilde{u}_k)$. For example, at iteration $k = 2200$, $|p^T\bar{y}_k + \gamma^*| = 6.03 \times 10^{-10}$ and $d_{\mathcal{L}}(\bar{u}_k) = 3.63 \times 10^{-9}$, while $|p^T\tilde{y}_k + \gamma^*| = 42.93$ and $d_{\mathcal{L}}(\tilde{u}_k) = 0.049$.

4 Discussion

We have applied an accelerated first-order method to solve unconstrained polynomial optimization problems. By exploiting special structural properties of the problem, namely the orthogonality of the moment matrices, the computational cost of the accelerated first-order method is greatly reduced. Numerical results on degree-4 and degree-6 polynomials demonstrate that the accelerated first-order method is capable of solving polynomials with at least twice as many variables as existing interior-point method software (SeDuMi) can handle. We have observed some curious convergence behavior of the iterate sequences of the accelerated first-order method that does not follow from the usual convergence analysis of such methods. This observation bears further study.

In the last several years, several researchers have worked on algorithms for solving large-scale SDPs, see, for example, [18] and [28]. In the recent work of [20], these algorithms were applied to solve large-scale unconstrained polynomial optimization problems. The main algorithmic framework in these recent algorithms is the augmented Lagrangian method, where the linear system at each iteration is solved by a combination of a semismooth Newton method and a conjugate gradient algorithm. The computational experiments recently reported for applying these methods to polynomial optimization in [20] are very promising, for instance they solve 4-degree polynomial optimization problems in up to $n = 100$ variables with high precision. These methods utilize second-order Hessian information but have low memory requirements inherited from the conjugate gradient algorithm. However, the augmented Lagrangian method in general does not have a global convergence rate guarantee; and indeed, the results in [24] and [28] show global convergence and local convergence rates only under regularity conditions. In comparison, the accelerated first-order method has virtually no regularity requirement and admits an explicit global convergence rate. The parameters in the convergence rate can be derived or bounded *a priori* in some cases (for instance, if we know that the optimal solution is contained in a given ball). As a future research direction, it would be very interesting to merge these two types of algorithms, i.e., to use an accelerated first-order method in the initial phase to quickly compute solutions that are in a neighborhood of the optimal solution, and then switch over to a low-memory second-order method to compute highly accurate solution iterates thereafter.

References

[1] A. Auslender and M.Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM J. Optim.*, 16:697–725, 2006.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2:183–202, March 2009.

[3] S. Becker, J. Bobin, and E. J. Cands. Nesta: a fast and accurate first-order method for sparse recovery. *SIAM J. on Imaging Sciences*, 4(1):1–39, 2009.

[4] D. Bertsimas and I. Popescu. Optimal inequalities in probability theory: A convex optimization approach. *SIAM Journal on Optimization*, 15:780–804, 2004.

[5] D. Bertsimas and J. Sethuraman. *Handbook on Semidefinite Programming*, volume 27 of *International Series in Operations Research and Management Science*, chapter Moment problems and semidefinite programming, pages 469–509. Kluwer, 2000.

[6] Alexandre D'aspremont, Onureena Banerjee, and Laurent El Ghaoui. First-order methods for sparse covariance selection. *SIAM J. Matrix Anal. Appl.*, 30(1):56–66, 2008.

[7] D.Bertsimas and K.Natarajan. A semidefinite optimization approach to the steady-state analysis of queueing systems. *Queueing Systems and Applications*, 56(1):27–40, 2007.

[8] D.Henrion and J.B.Lasserre. Gloptipoly: Global optimization over polynomials with matlab and sedumi. *ACM Transactions on Mathematical Software*, 29(2):165–194, June 2003.

[9] Z. Lu G. Lan and R.D.C.Monteiro. Primal-dual first-order methods with $\mathcal{O}(1/\epsilon)$ iteration-complexity for cone programming. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, December 2006.

[10] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*, volume 305 of *Comprehensive Study in Mathematics*. Springer-Verlag, New York, 1993.

[11] I.J.Landau. *Moments in Mathematics*. AMS, Providence, RI, 1987.

[12] F. Jarre and F. Rendl. An augmented primal-dual method for linear conic programs. *SIAM Journal on Optimization*, 19(2):808–823, 2008.

[13] J.B.Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optimization*, 11:796–816, 2001.

[14] M.Coste J.Bochnak and M-F.Roy. *Real Algebraic Geometry.* Springer, 1998.

[15] Z. Lu. Primal-dual first-order methods for a class of cone programming. *submitted*, 2009.

[16] Zhaosong Lu. Smooth optimization approach for sparse covariance selection. *SIAM J. on Optimization*, 19:1807–1827, February 2009.

[17] L.Vandenberghe and S.Boyd. Semidefinite programming. *SIAM Rev.*, 38:49–95, 1996.

[18] J. Malick, J. Povh, F. Rendl, and A. Wiegele. Regularization methods for semidefinite programming. *SIAM J. Optim.*, 20(1):336–356, 2009.

[19] M.Laurent. Semidefinite programming in combinatorial and polynomial optimization. *NAW*, 5/9(4), December 2008.

[20] J. Nie. Regularization methods for sum of squares relaxations in large scale polynomial optimization. Technical report, Department of Mathematics, University of California, 9500 Gilman Drive, La Jolla, CA 92093, 2009.

[21] P. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization.* PhD thesis, California Institute of Technology.

[22] P.Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Prog.*, 96(2, Ser. B):293–320, 2003.

[23] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004. Available from `http://www.cds.caltech.edu/sostools` and `http://www.mit.edu/~parrilo/sostools`.

[24] R. T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Math. Oper. Res.*, 1:97–116, 1976.

[25] J.F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, (11-12):625–653, 1999. Special issue on Interior Point Methods.

[26] Y.Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Doklady*, 269:543–547, 1983. translated as Soviet Math. Docl.

[27] Y.Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

[28] X. Zhao, D. Sun, and K. Toh. A newton-cg augmented lagrangian method for semidefinite programming. *SIAM J. Optim.*, 20(4):1737–1765, 2010.