Alternating Direction Method of Multipliers for Real and Complex Polynomial Optimization Models

Bo JIANG * Shiqian MA † Shuzhong ZHANG ‡

March 21, 2013

Abstract

In this paper, we propose a new method for polynomial optimization with real or complex decision variables. The main ingredient of the approach is to apply the classical alternating direction method of multipliers (ADMM) based on the augmented Lagrangian function. In this particular case, this allows us to fully exploit the multi-block structure of the polynomial functions, even though the optimization model encountered is highly non-linear and non-convex. The new method is shown to be convergent under some conditions, and the numerical results show that the algorithm returns high quality solutions and runs much faster than the two other competing algorithms.

Keywords: polynomial optimization, optimization with complex variables, alternating direction method of multipliers.

Mathematics Subject Classification 2010: 90C30, 90C26, 15A69, 65F30

This paper is dedicated to Professor Minyi Yue in celebration of his 95th birthday.

*Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55455.

†Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong.

‡Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55455. Research of this author was supported in part by the NSF Grant CMMI-1161242.

1 Introduction

In this paper, we study the problem of optimizing a polynomial function in real domain or complex domain. More precisely, the real polynomial optimization problem under consideration is as follows:

$$\begin{array}{ll} \min & f(x) := \displaystyle\sum_{1\le i_1\le i_2\le\cdots\le i_d\le n} a_{i_1 i_2\cdots i_d} x_{i_1} x_{i_2}\cdots x_{i_d} \\ \text{s.t.} & x\in S\subseteq\mathbb{R}^n, \end{array} \tag{1}$$

and the complex polynomial optimization problem is:

$$\begin{array}{ll} \min & g(\bar{x},x) := \displaystyle\sum_{\substack{1\le i_1\cdots\le i_d\le n\\ 1\le i_{d+1}\le\cdots\le i_{2d}\le n}} b_{i_1\cdots i_d i_{d+1}\cdots i_{2d}} \overline{x_{i_1}\cdots x_{i_d}} x_{i_{d+1}}\cdots x_{i_{2d}} \\ \text{s.t.} & x\in S\subseteq\mathbb{C}^n, \end{array} \tag{2}$$

where $\bar{x}$ is to denote the conjugate of x, and we assume that $g(\bar{x},x)$ is always real-valued, although x is a complex vector.

There is a growing interest to study polynomial optimization problems, primarily driven by many emerging applications of such models in science and engineering such as biomedical engineering, financial engineering, material science, graph theory, quantum mechanics, signal processing, speech recognition, and so on (see e.g., [16] for references). In particular, the conjugate (complex) polynomial optimization (2) was found to be very useful in radar waveform design [1].

In terms of solution methods, Lasserre [14] and Parrilo [18] initiated the so-called sum of squares (SOS) approach for generic polynomial optimization, which has since become a widely accepted general purposed tool for polynomial optimization. As for specific polynomial optimization models (most of them are NP-hard), various approximation algorithms have been proposed to find a solution with worst case performance guarantee; see e.g., the recent monograph of Li et al. [16] and the references therein. Due to the non-convex nature of polynomial optimization, another line of research is to design efficient algorithms to find good KKT solutions for the problem. The so-called maximum block improvement (MBI) method studied by Chen et al. [4] is one of such methods, which guarantees that any cluster point of the sequence generated by the algorithm is a stationary solution (hence a KKT point). Moreover, this approach was shown to be very efficient in practice; see e.g. [4] and [1].

In this paper, we propose the alternating direction method of multipliers (ADMM) to solve problems (1) and (2), which also aims at finding the KKT point. ADMM has been shown to be equivalent to the Douglas-Rachford operator-splitting method, which was proposed in 1950s for solving variational problems arising from PDEs (see [5]). ADMM was reinvigorated recently as it was found to be very efficient for some convex problems arising from compressed sensing [25], compressive imaging [22, 8], robust/sparse Principal Component Analysis [21, 17], and Semidefinite Programming [23]. We refer the interested reader to the recent survey by Boyd et al. [2] for a more complete discussion and a list of references on ADMM.

Inspired by the success of ADMM for convex optimization, there are some recent efforts to extend the implementation of ADMM to solve non-convex problem as well. For instance, Xu et al. [24] proposed an ADMM algorithm to the problem of factorizing a nonnegative matrix into two nonnegative matrices. For non-convex problems, additional conditions are needed to guarantee the convergence of ADMM, but in many ways it works very well for some practical problems. For example, it was reported in [24] that ADMM can generate solutions with better qualities than that of other two algorithms for matrix completion with nonnegative factorization. Inspired by the work of Xu et al., in this paper we propose to apply the ADMM to solve another very important class of non-convex problems: polynomial optimization. Our numerical results show that the overall performance of ADMM is superior to the other two alternative algorithms for polynomial optimization.

The rest of the paper is organized as follows. To apply ADMM, we first need to reformulate problems (1) and (2) in a form where the objective function has a multi-block structure. This will be discussed in Section 2. The same section also presents several ADMM algorithms according to the different structures of the polynomial optimization models. Section 3 is devoted to the convergence analysis of the algorithms presented in Sections 2.2, 2.3 and 2.4. Finally, Section 4 presents the numerical simulation results for both the sphere constrained polynomial optimization problems and the unimodularly constrained complex polynomial optimization problems.

2 Alternating Direction Method of Multipliers

2.1 Tensor Formulation of Polynomial Function

It is well known that the alternating direction type method is designed to solve optimization problems with an objective with two-block or multi-block structure. Therefore, to apply ADMM for polynomial optimization, we need to convert the polynomial functions into an equivalent form involving tensor formulations, which can be viewed as a function with a multi-block structure.

For the real polynomial $f(x)$, we can construct a real super-symmetric tensor $\mathcal{F} = (\mathcal{F}_{i_1 i_2 \cdots i_d})$ with

$$\mathcal{F}_{j_1 j_2 \cdots j_d} = \frac{a_{i_1 i_2 \cdots i_d}}{|\pi(i_1 i_2 \cdots i_d)|} \ \forall\ (j_1 j_2 \cdots j_d) \in \pi(i_1 i_2 \cdots i_d),$$

such that

$$f(x) = \mathcal{F}(\underbrace{x, x, \cdots, x}_{d}) := \sum_{1 \le i_1, i_2, \cdots, i_d \le n} \mathcal{F}_{i_1 i_2 \cdots i_d} x_{i_1} x_{i_2} \cdots x_{i_d},$$

where $\pi(i_1 i_2 \cdots i_d)$ is the set of all distinct permutations of the indices $\{i_1, i_2, \cdots, i_d\}$. In the complex case, if $g(\bar{x}, x)$ always takes real value, we introduce a complex tensor $\mathcal{G} = (\mathcal{G}_{i_1 i_2 \cdots i_{2d}})$ with

$$\mathcal{G}_{j_1 \cdots j_{2d}} = \frac{b_{i_1 \cdots i_{2d}}}{|\pi(i_1 \cdots i_d)| \cdot |\pi(i_{d+1} \cdots i_{2d})|},\ \forall\ (j_1 \cdots j_d) \in \pi(i_1 \cdots i_d) \text{ and } (j_{d+1} \cdots j_{2d}) \in \pi(i_{d+1} \cdots i_{2d}).$$

Then it holds that

$$g(\bar{x},x)=\mathcal{G}(\underbrace{\bar{x},\cdots,\bar{x}}_{d},\underbrace{x,\cdots,x}_{d}):=\sum_{\substack{1\le i_1,\cdots,i_d\le n\\ 1\le i_{d+1},\cdots,i_{2d}\le n}}\mathcal{G}_{i_1\cdots i_d i_{d+1}\cdots i_{2d}}\overline{x_{i_1}\cdots x_{i_d}}x_{i_{d+1}}\cdots x_{i_{2d}}.$$

Consequently, the tensor $\mathcal{G}$ is no longer super-symmetric. Instead, it is conjugate partial-symmetric as introduced in [13], and it satisfies the following two properties:

(1) $\mathcal{G}_{i_1\cdots i_d\, i_{d+1}\cdots i_{2d}}=\overline{\mathcal{G}_{i_{d+1}\cdots i_{2d}\, i_1\cdots i_d}}$;

(2) $\mathcal{G}_{i_1\cdots i_d\, i_{d+1}\cdots i_{2d}}=\mathcal{G}_{j_1\cdots j_d\, j_{d+1}\cdots j_{2d}}\quad\forall\,(j_1\cdots j_d)\in\pi(i_1\cdots i_d),\ (j_{d+1}\cdots j_{2d})\in\pi(i_{d+1}\cdots i_{2d})$.

Observe now that the problems (1) and (2) can be equivalently written as

$$\begin{array}{ll}\min & \mathcal{F}(x^1,x^2,\cdots,x^d)\\ \text{s.t.} & x^0=x^i, i=1,2,\ldots,d\\ & x^i\in S, i=0,1,2,\ldots,d\end{array}\tag{3}$$

and

$$\begin{array}{ll}\min & \mathcal{G}(\bar{x}^1,\cdots,\bar{x}^d,x^{d+1},\cdots,x^{2d})\\ \text{s.t.} & x^0=x^i, i=1,2,\ldots,2d\\ & x^i\in S, i=0,1,2,\ldots,2d\end{array}\tag{4}$$

respectively.

2.2 General iteration scheme

A classical method for solving (3) and (4) is the augmented Lagrangian method. The augmented Lagrangian functions associated with only the linear constraints $x^0=x^i, i=1,\ldots,d$ in (3) and (4) are given by

$$\mathcal{L}_{\mathcal{F},\mu}(x^0,x^1,\ldots,x^d;\lambda^1,\ldots,\lambda^d):=\mathcal{F}(x^1,\ldots,x^d)-\sum_{i=1}^{d}\langle\lambda^i,x^i-x^0\rangle+\frac{1}{2\mu}\sum_{i=1}^{d}\|x^i-x^0\|_2^2,\tag{5}$$

and

$$\begin{array}{rl}&\mathcal{L}_{\mathcal{G},\mu}(x^0,x^1,\ldots,x^{2d};\lambda^1,\ldots,\lambda^{2d})\\ :=&\ \mathrm{Re}\left(\mathcal{G}(\bar{x}^1,\ldots,\bar{x}^d,x^{d+1},\ldots,x^{2d})-\sum_{i=1}^{2d}\langle\lambda^i,x^i-x^0\rangle+\frac{1}{2\mu}\sum_{i=1}^{2d}\|x^i-x^0\|_2^2\right)\end{array}\tag{6}$$

respectively, where λ^i are the Lagrange multipliers and $\mu>0$ is the penalty parameter. A typical iteration of the augmented Lagrangian method for solving (3) is

$$\begin{cases}\left((x^0)^{(k+1)},(x^1)^{(k+1)},\ldots,(x^d)^{(k+1)}\right) & :=\arg\min\limits_{\{x^i\in S,i=0,1,\ldots,d\}}\mathcal{L}_{\mathcal{F},\mu}(x^0,x^1,\ldots,x^d;(\lambda^1)^{(k)},\ldots,(\lambda^d)^{(k)})\\ (\lambda^i)^{(k+1)} & :=(\lambda^i)^{(k)}-((x^i)^{(k+1)}-(x^0)^{(k+1)})/\mu, i=1,\ldots,d.\end{cases}\tag{7}$$

In fact, minimizing the augmented Lagrangian function in (7) is not any easier than solving the original problem (3). Because the variables $x^i, i = 0, 1, \ldots, d$ are coupled together in the function $\mathcal{F}(x^1, x^2, \ldots, x^d)$ and the penalty term, it is usually hard to minimize the augmented Lagrangian function with respect to variables $x^i, i = 0, 1, \ldots, d$ simultaneously. We will encounter the same difficulty if we directly apply the augmented Lagrangian method to Problem (4). However, if we minimize the augmented Lagrangian function with variables x^i alternatingly, then we have several subproblems in each iteration and these subproblems are usually easy to solve. This leads to the alternating direction method of multipliers (ADMM) for solving (3) and (4).

The standard iteration schemes of ADMM for solving (3) and (4) are given by:

$$\begin{cases} (x^i)^{(k+1)} := \arg\min\limits_{\{x^i \in S\}} \mathcal{L}_{\mathcal{F},\mu}\left((x^0)^{(k+1)}, \ldots, (x^{i-1})^{(k+1)}, x^i, (x^{i+1})^{(k)} \ldots, (x^d)^{(k)}; (\lambda^1)^{(k)}, \ldots, (\lambda^d)^{(k)}\right) \\ (\lambda^i)^{(k+1)} := (\lambda^i)^{(k)} - \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu, i = 1, \ldots, d, \end{cases} \tag{8}$$

$$\begin{cases} (x^i)^{(k+1)} := \arg\min\limits_{\{x^i \in S\}} \mathcal{L}_{\mathcal{G},\mu}\left((x^0)^{(k+1)}, \ldots, (x^{i-1})^{(k+1)}, x^i, (x^{i+1})^{(k)} \ldots, (x^{2d})^{(k)}; (\lambda^1)^{(k)}, \ldots, (\lambda^{2d})^{(k)}\right) \\ (\lambda^i)^{(k+1)} := (\lambda^i)^{(k)} - \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu, i = 1, \ldots, 2d, \end{cases} \tag{9}$$

respectively. The preliminary convergence properties of these two iteration schemes will be presented in Section 3. In the following, we apply algorithms (8) and (9) to two special polynomial optimization problems and derive specific iteration schemes.

2.3 Sphere Constrained Polynomial Optimization

One immediate application of the ADMM algorithm (8) is to solve the sphere constrained polynomial optimization problem:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & \|x\|_2^2 = 1, x \in \mathbb{R}^n. \end{array} \tag{10}$$

This problem has quite a few synonyms: it is sometimes known as the maximum Z-eigenvalue problem, or the best rank-one approximation for tensor $\mathcal{F}$, or the Tensor Principle Component Analysis problem. The eigenvalues and eigenvectors of a real symmetric tensor are first introduced by Lim [15] and Qi [19] independently in 2005. Since then, various methods have been proposed to find the KKT point of (10) [4, 20, 9, 10]. Most recently, Jiang, Ma and Zhang [12] revisited this problem and proposed an algorithm that can find the global optimum with high probability.

For Problem (10), the ADMM algorithm (8) is reduced to

$$\begin{cases} (x^0)^{(k+1)} := \arg\min\limits_{\|x^0\|_2^2=1} \mathcal{L}_{\mathcal{F},\mu}\left(x^0, (x^1)^{(k)}, \ldots, (x^d)^{(k)}; (\lambda^1)^{(k)}, \ldots, (\lambda^d)^{(k)}\right) \\ (x^i)^{(k+1)} := \arg\min\limits_{\|x^i\|_2^2=1} \mathcal{L}_{\mathcal{F},\mu}\left((x^0)^{(k+1)}, \ldots, (x^{i-1})^{(k+1)}, x^i, (x^{i+1})^{(k)}, \ldots, (x^d)^{(k)}; (\lambda^1)^{(k)}, \ldots, (\lambda^d)^{(k)}\right) \\ (\lambda^i)^{(k+1)} := (\lambda^i)^{(k)} - \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu. \end{cases} \tag{11}$$

Note that with other variables fixed, the augmented Lagrangian function $\mathcal{L}_{\mathcal{F},\mu}$ with respect to x^i is a simple quadratic function in the form of minimizing $\|x^j - z\|_2^2$ with a given z. Thus, each of the subproblems in (11) corresponds to a projection onto the Euclidean ball $\|x^j\| = 1$, which has a closed form solution given by

$$\text{Normalize}(z) := z/\|z\|_2.$$

Therefore, algorithm (11) can be reduced to:

$$\begin{cases} (x^0)^{(k+1)} & := & \text{Normalize}\left(\sum_{j=1}^d (x^j)^{(k)} - \mu(\lambda^j)^{(k)}\right) \\ (v^i)^{(k+1)} & := & \mathcal{F}\left((x^1)^{(k+1)}, \ldots, (x^{i-1})^{(k+1)}, \bullet, (x^{i+1})^{(k)}, \ldots, (x^d)^{(k)}\right) \\ (x^i)^{(k+1)} & := & \text{Normalize}\left((x^0)^{(k+1)} - \mu((v^i)^{(k+1)} - (\lambda^i)^{(k)})\right) \\ (\lambda^i)^{(k+1)} & := & (\lambda^i)^{(k)} - \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu. \end{cases} \tag{12}$$

2.4 Unimodularly Constrained Complex Polynomial Optimization

Now let us consider another special polynomial optimization problem: unimodularly constrained complex polynomial optimization. The problem is in the form of:

$$\begin{array}{ll} \min & g(\bar{x}, x) \\ \text{s.t.} & |x_j| = 1,\ \forall\ j,\ x \in \mathbb{C}^n. \end{array} \tag{13}$$

This problem turns out to be important in the field of signal processing. In particular, Aubry et al. [1] found that the design of phase-only modulated Radar waveforms sharing a desired range-Doppler response can be formulated in the form of (13).

When applying algorithm (9) to this problem, the exact procedure reads as follow:

$$\begin{cases} (x^0)^{(k+1)} := \arg\min\limits_{|x_j^0|=1,\ \forall j} \mathcal{L}_{\mathcal{G},\mu}\left(x^0, (x^1)^{(k)}, \ldots, (x^{2d})^{(k)}; (\lambda^1)^{(k)}, \ldots, (\lambda^{2d})^{(k)}\right) \\ (x^i)^{(k+1)} := \arg\min\limits_{|x_j^i|=1\ \forall j} \mathcal{L}_{\mathcal{G},\mu}\left((x^0)^{(k+1)}, \ldots, (x^{i-1})^{(k+1)}, x^i, (x^{i+1})^{(k)}, \ldots, (x^{2d})^{(k)}; (\lambda^1)^{(k)}, \ldots, (\lambda^{2d})^{(k)}\right) \\ (\lambda^i)^{(k+1)} := (\lambda^i)^{(k)} - \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu. \end{cases} \tag{14}$$

Notice the augmented Lagrangian function $\mathcal{L}_{\mathcal{G},\mu}$ can be rewritten as follows,

$$\begin{aligned} & \text{Re}\left(\mathcal{G}(\bar{x}^1, \ldots, \bar{x}^d, x^{d+1}, \ldots, x^{2d}) - \sum_{i=1}^{2d} \langle \lambda^i, x^i - x^0\rangle + \frac{1}{2\mu}\sum_{i=1}^{2d} \|x^i - x^0\|_2^2\right) \\ = & \frac{d}{\mu}\left\|x^0 - \sum_{i=1}^{2d} \frac{x^i - \mu\lambda^i}{2d}\right\|_2^2 + \frac{1}{2\mu}\sum_{i=1}^{2d}\|x^i\|_2^2 - \frac{d}{\mu}\left\|\sum_{i=1}^{2d}\frac{x^i - \mu\lambda^i}{2d}\right\|_2^2 \\ & + \text{Re}\left(\mathcal{G}(\bar{x}^1, \ldots, \bar{x}^d, x^{d+1}, \ldots, x^{2d}) - \sum_{i=1}^{2d}\langle \lambda^i, x^i\rangle\right). \end{aligned}$$

So the subproblem for solving x_0 has a closed form solution

$$(x^0)_j^{(k+1)} = e^{\boldsymbol{i}\theta},\ \text{with}\ \theta = \arg\left(\sum_{i=1}^{2d}\frac{(x^i)^{(k)} - \mu(\lambda^i)^{(k)}}{2d}\right)_j,\ \forall 1 \le j \le n,$$

where the notation $\arg(z)$ represents the argument of complex number z. To derive the iteration scheme for x^i with $i \neq 0$, we fix other variables and extract expression only involving x^i from the augmented Lagrangian function $\mathcal{L}_{\mathcal{G},\mu}$:

$$\begin{aligned}
& \text{Re}\left(\mathcal{G}(\bar{x}^1,\ldots,\bar{x}^d,x^{d+1},\ldots,x^{2d}) - \langle\lambda^i,x^i\rangle + \frac{1}{2\mu}\|x^i\|_2^2 - \frac{1}{\mu}\langle x^0,x^i\rangle\right) \\
= & \text{Re}\left(\bar{\mathcal{G}}(x^1,\ldots,x^d,\bar{x}^{d+1},\ldots,\bar{x}^{2d}) - \langle\lambda^i,x^i\rangle + \frac{1}{2\mu}\|x^i\|_2^2 - \frac{1}{\mu}\langle x^0,x^i\rangle\right) \\
= & \begin{cases} \frac{1}{2\mu}\left\|x^i - x^0 + \mu\left(\bar{\mathcal{G}}(x^1,\ldots,x^{i-1},\bullet,x^{i+1},\ldots,x^d,\bar{x}^{d+1},\ldots,\bar{x}^{2d}) - \lambda^i\right)\right\|_2^2 + h_1(x^{-i}), & \text{if } 1\le i\le d \\ \frac{1}{2\mu}\left\|x^i - x^0 + \mu\left(\mathcal{G}(\bar{x}^1,\ldots,\bar{x}^d,x^{d+1},\ldots,x^{i-1},\bullet,x^{i+1},\ldots,x^{2d}) - \lambda^i\right)\right\|_2^2 + h_2(x^{-i}), & \text{if } d+1\le i\le 2d, \end{cases}
\end{aligned}$$

where $h_1(x^{-i})$ and $h_2(x^{-i})$ are the functions not involving x^i. So the solution of subproblem for solving x_i is in the form of $(x^i)_j^{(k+1)} = e^{\boldsymbol{i}\theta}$ with $\theta = \arg\left((x^0)^{(k+1)} - \mu((v^i)^{(k+1)} - (\lambda^i)^{(k)})\right)_j$, where

$$(v^i)^{(k+1)} = \begin{cases} \bar{\mathcal{G}}((x^1)^{(k+1)},\ldots,(x^{i-1})^{(k+1)},\bullet,(x^{i+1})^{(k)},\ldots,(x^d)^{(k)},(\bar{x}^{d+1})^{(k)},\ldots,(\bar{x}^{2d})^{(k)}), & \text{if } 1\le i\le d \\ \mathcal{G}((\bar{x}^1)^{(k+1)},\ldots,(\bar{x}^d)^{(k+1)},(x^{d+1})^{(k+1)},\ldots,(x^{i-1})^{(k+1)},\bullet,(x^{i+1})^{(k)},\ldots,(x^{2d})^{(k)}), & \text{if } d+1\le i\le 2d \end{cases}$$

Finally, a standard rule is followed to update the Lagrange multipliers:

$$(\lambda^i)^{(k+1)} := (\lambda^i)^{(k)} - \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu,\ \forall\, i = 1,\ldots,2d.$$

3 Convergence Results

For convex problems with two blocks, the global convergence of ADMM was given in [7, 6] under mild conditions. Since problems (3) and (4) are highly non-convex, the convergence analysis for the ADMM algorithms requires additional conditions.

In the case of non-convex optimization, convergence to a KKT solution is the best convergence property that we can hope for. By imposing some conditions, Xu et al. [24] managed to show that the sequence generated by ADMM converges to a KKT point. In this section, along a similar line, we establish the convergence property of the ADMM algorithms for polynomial optimization problems. The key assumptions for our results are (15) and (25), which basically means that the iterates produced by the algorithm do not exhibit any jumping behavior. Remark that the same condition was used in Xu et al. [24] as well.

3.1 Convergence of ADMM for Real Polynomial Optimization

We first present a convergence result for general ADMM algorithm (8) for solving (3).

Theorem 3.1 *Suppose the closed set S in (3) is given by*

$$S = \{x \in \mathbb{R}^n \mid h_j(x) \le 0, j = 1, 2, \ldots, m; e_j(x) = 0, j = 1, 2, \ldots, \ell\},$$

where $h_j(\cdot)$ and $e_j(\cdot)$ are functions from $\mathbb{R}^n \to \mathbb{R}$ with first order continuous derivatives. For the i-th subproblem in (8), i.e., minimizing the augmented Lagrangian function with respect to x^i, we use $(u_j^i)^{(k+1)}$ and $(v_j^i)^{(k+1)}$ to denote the optimal Lagrange multipliers corresponding to the constraints $h_j(x^i) \le 0$ and $e_j(x^i) = 0$, respectively. Denote

$$z^k := \left([(x^i)^{(k)}]_{i=0}^d; [(\lambda^i)^{(k)}]_{i=1}^d; [(u_j^i)^{(k)}]_{i=0,\ldots,d;j=1,\ldots,m}; [(v_j^i)^{(k)}]_{i=0,\ldots,d,j=1,\ldots,\ell}\right).$$

Assume that

$$\lim_{k\to\infty} (z^{k+1} - z^k) = 0. \tag{15}$$

Then any limit point of $\{z^k\}_{k=1}^\infty$ is a KKT point of Problem (3). Consequently, any limit point of $\{(x^0)^{(k)}\}_{k=1}^\infty$ is a KKT point of Problem (1), and whenever $\{z^k\}$ converges, it converges to a KKT point of (3).

Proof. The argument here is basically the same as the one used in [24]. Since $(u_j^i)^{(k+1)}$ and $(v_j^i)^{(k+1)}$ are the optimal Lagrange multipliers for the i-th subproblem in (8), from the KKT conditions of the i-th subproblem, we have,

$$\begin{array}{l} \nabla_{x^i}\mathcal{F}((x^0)^{(k+1)}, (x^1)^{(k+1)}, \ldots, (x^i)^{(k+1)}, (x^{i+1})^{(k)}, \ldots, (x^d)^{(k)}) \\ -(\lambda^i)^{(k)} - ((x^0)^{(k+1)} - (x^i)^{(k+1)})/\mu - \sum_{j=1}^m (u_j^i)^{(k+1)} \nabla h_j((x^i)^{(k+1)}) \\ -\sum_{j=1}^\ell (v_j^i)^{(k+1)} \nabla e_j((x^i)^{(k+1)}) = 0, 1 \le i \le d, \end{array} \tag{16}$$

$$\begin{array}{l} \sum_{i=1}^d (\lambda^i)^{(k)} + \sum_{i=1}^d ((x^0)^{(k+1)} - (x^i)^{(k)})/\mu - \sum_{j=1}^m (u_j^0)^{(k+1)} \nabla h_j((x^0)^{(k+1)}) \\ -\sum_{j=1}^\ell (v_j^0)^{(k+1)} \nabla e_j((x^0)^{(k+1)}) = 0, \end{array} \tag{17}$$

and

$$(x^i)^{(k+1)} \in S, \quad (u_j^i)^{(k+1)} \le 0, \quad (u_j^i)^{(k+1)} h_j((x^i)^{(k+1)}) = 0, \quad i = 0, 1, \ldots, d, j = 1, \ldots, \ell. \tag{18}$$

It follows from $\lim_{k\to\infty}(z^{k+1} - z^k) = 0$ that

$$\lim_{k\to\infty} \left((\lambda^i)^{(k+1)} - (\lambda^i)^{(k)}\right) = 0, \forall i = 1, \ldots, d.$$

Now from the updating formula for λ^i in (8), we get the term

$$\lim_{k\to\infty} \left((x^i)^{(k+1)} - (x^0)^{(k+1)}\right)/\mu = 0, \forall i = 1, \ldots, d. \tag{19}$$

in both (16) and (17). This fact combined with (15) and (16) implies

$$\begin{array}{l} \nabla_{x^i}\mathcal{F}((x^0)^{(k)}, (x^1)^{(k)}, \ldots, (x^i)^{(k)}, (x^{i+1})^{(k)}, \ldots, (x^d)^{(k)}) \\ -(\lambda^i)^{(k)} - \sum_{j=1}^m (u_j^i)^{(k)} \nabla h_j((x^i)^{(k)}) \\ -\sum_{j=1}^\ell (v_j^i)^{(k)} \nabla e_j((x^i)^{(k)}) \to 0, \text{ as } k \to \infty, \forall 1 \le i \le d. \end{array} \tag{20}$$

Moreover, combining (15), (17) and (19), we get

$$\begin{array}{l}\sum_{i=1}^{d}(\lambda^i)^{(k)}-\sum_{j=1}^{m}(u_j^0)^{(k)}\nabla h_j((x^0)^{(k)})\\ -\sum_{j=1}^{\ell}(v_j^0)^{(k)}\nabla e_j((x^0)^{(k)})\to 0,\ \text{as } k\to\infty.\end{array}\tag{21}$$

It is easy to verify from (18) that any limit point $\hat{z}$ of $\{z^k\}_{k=1}^{\infty}$ satisfies

$$\hat{x}^i\in S,\quad \hat{u}_j^i\le 0,\quad \hat{u}_j^i\,h_j(\hat{x}^i)=0,\quad i=0,1,\ldots,d,j=1,\ldots,\ell.\tag{22}$$

Finally, combining (20), (21) and (22) leads to the conclusion that any limit point of $\{z^k\}_{k=1}^{\infty}$ is a KKT solution for (3). □

Now let us consider the sphere constrained problem (10) and its structured updating rule (12). Under similar assumptions as in Theorem 3.1, one can prove the following results, where no conditions on the Lagrangian multipliers for the constraints are required.

Theorem 3.2 *Denote* $z^k:=\left([(x^i)^{(k)}]_{i=0}^{d};[(\lambda^i)^{(k)}]_{i=1}^{d}\right)$ *is the sequence generated in* (12)*, and assume that*

$$\lim_{k\to\infty}(z^{k+1}-z^k)=0.$$

Then any limit point of $\{z^k\}_{k=1}^{\infty}$ *is a KKT point of Problem* (10)*. Consequently, if* $\{(x^0)^{(k)}\}_{k=1}^{\infty}$ *converges, it converges to a KKT point of Problem* (10)*.*

Proof. The updating rule for x^i in (12) implies there exists a scaler $(\alpha^i)^{(k)}$ such that

$$(x^i)^{(k+1)}=(\alpha^i)^{(k+1)}\left((x^0)^{(k+1)}-\mu((v^i)^{(k+1)}-(\lambda^i)^{(k)})\right)\tag{23}$$

with $(\alpha^i)^{(k+1)}=1/\|(x^0)^{(k+1)}-\mu((v^i)^{(k+1)}-(\lambda^i)^{(k)})\|_2$, for $i=1,2,\ldots,d$, where

$$(v^i)^{(k+1)}=\mathcal{F}\left((x^1)^{(k+1)},\ldots,(x^{i-1})^{(k+1)},\bullet,(x^{i+1})^{(k)},\ldots,(x^d)^{(k)}\right).$$

On the other hand, the KKT condition of i-th subproblem gives us

$$(1+\mu\,(v^i)^{(k+1)})(x^i)^{(k+1)}=(x^0)^{(k+1)}-\mu((v^i)^{(k+1)}-(\lambda^i)^{(k+1)})\text{ and }\|(x^i)^{(k+1)}\|_2^2=1,\tag{24}$$

where $(v^i)^{(k+1)}$ is the Lagrangian multiplier associated with the spherical constraint. Comparing (23) and (24) gives us

$$(v^i)^{(k+1)}=\left(\frac{1}{(\alpha^i)^{(k+1)}}-1\right)/\mu,i=1,\ldots,d.$$

When $i=0$, due to (12) we have

$$(x^0)^{(k+1)}=(\alpha^0)^{(k+1)}\left(\sum_{j=1}^{d}(x^j)^{(k)}-\mu(\lambda^j)^{(k)}\right),$$

with $(\alpha^0)^{(k+1)} = 1/\|\sum_{j=1}^d (x^j)^{(k)} - \mu(\lambda^j)^{(k)}\|_2$. Combining the expression above with the KKT condition of the 0-th subproblem yields

$$(v^0)^{(k+1)} = \left(\frac{1}{(\alpha^0)^{(k+1)}} - d\right)/\mu.$$

Therefore the condition $\lim_{k\to\infty}(z^{k+1} - z^k) = 0$ leads to $\lim_{k\to\infty}((\alpha^i)^{k+1} - (\alpha^i)^k) = 0$, and thus $\lim_{k\to\infty}((v^i)^{k+1} - (v^i)^k) = 0$, for all $i = 0, 1, \ldots, d$. So all the conditions of Theorem 3.1 are satisfied and the conclusion follows. □

3.2 Convergence of ADMM for Complex Polynomial Optimization

In this subsection, we consider the polynomial optimization in the complex domain. First of all, we show the convergence result for general complex polynomial optimization problem (4).

Theorem 3.3 *Suppose the closed set S in (4) is given by*

$$S = \{x \in \mathbb{C}^n \mid h_j(\bar{x}, x) \le 0, j = 1, 2, \ldots, m; e_j(\bar{x}, x) = 0, j = 1, 2, \ldots, \ell\},$$

where $h_j(\cdot)$ and $e_j(\cdot)$ are complex functions but always taking real value and with first order continuous derivatives. For the i-th subproblem in (9), i.e., minimizing the augmented Lagrangian function with respect to x^i, we use $(u_j^i)^{(k+1)}$ and $(v_j^i)^{(k+1)}$ to denote the optimal Lagrange multipliers corresponding to the constraints $h_j(\bar{x}^i, x^i) \le 0$ and $e_j(\bar{x}^i, x^i) = 0$, respectively. Denote

$$z^k := \left([(x^i)^{(k)}]_{i=0}^{2d}; [(\lambda^i)^{(k)}]_{i=1}^{2d}; [(u_j^i)^{(k)}]_{i=0,\ldots,2d;j=1,\ldots,m}; [(v_j^i)^{(k)}]_{i=0,\ldots,2d,j=1,\ldots,\ell}\right).$$

Assume that

$$\lim_{k\to\infty}(z^{k+1} - z^k) = 0. \tag{25}$$

Then any limit point of $\{z^k\}_{k=1}^\infty$ is a KKT point of Problem (3). Consequently, any limit point of $\{(x^0)^{(k)}\}_{k=1}^\infty$ is a KKT point of Problem (2), and whenever $\{z^k\}$ converges, it converges to a KKT point of (4).

Proof. The argument is very similar to the one that we applied in Theorem 3.1, except that the KKT condition is more complicated in this case. Precisely, we have

$$\begin{aligned}
& \mathcal{L}_{\mathcal{G},\mu}(x^0, x^1, \ldots, x^{2d}; \lambda^1, \ldots, \lambda^{2d}) \\
= \; & \mathrm{Re}\left(\mathcal{G}(\bar{x}^1, \ldots, \bar{x}^d, x^{d+1}, \ldots, x^{2d}) - \sum_{i=1}^{2d}\langle\lambda^i, x^i - x^0\rangle + \frac{1}{2\mu}\sum_{i=1}^{2d}\|x^i - x^0\|_2^2\right) \\
= \; & \frac{1}{2}\mathcal{G}(\bar{x}^1, \ldots, \bar{x}^d, x^{d+1}, \ldots, x^{2d}) + \frac{1}{2}\bar{\mathcal{G}}(x^1, \ldots, x^d, \bar{x}^{d+1}, \ldots, \bar{x}^{2d}) \\
& -\frac{1}{2}\sum_{i=1}^{2d}\langle\lambda^i, x^i - x^0\rangle - \frac{1}{2}\sum_{i=1}^{2d}\langle\bar{\lambda}^i, \bar{x}^i - \bar{x}^0\rangle + \frac{1}{2\mu}\sum_{i=1}^{2d}\|x^i - x^0\|_2^2.
\end{aligned}$$

Then, since $(u_j^i)^{(k+1)}$ and $(v_j^i)^{(k+1)}$ are the optimal Lagrange multipliers for the i-th subproblem in (9), from the KKT conditions of the i-th subproblem with $1 \le i \le d$, we have,

$$\begin{aligned}
&\tfrac{1}{2}\nabla_{\bar{x}^i}\mathcal{G}((\bar{x}^1)^{(k+1)},\ldots,(\bar{x}^{i-1})^{(k+1)},(\bar{x}^i)^{(k)},(\bar{x}^{i+1})^{(k)},\ldots,(\bar{x}^d)^{(k)},(x^{d+1})^{(k)},\ldots,(x^{2d})^{(k)})\\
&-\tfrac{1}{2}(\lambda^i)^{(k)}-\tfrac{1}{2\mu}((x^0)^{(k+1)}-(x^i)^{(k+1)})-\textstyle\sum_{j=1}^m(u_j^i)^{(k+1)}\nabla_{\bar{x}^i}h_j((\bar{x}^i)^{(k+1)},(x^i)^{(k+1)})\\
&-\textstyle\sum_{j=1}^\ell(v_j^i)^{(k+1)}\nabla_{\bar{x}^i}e_j((\bar{x}^i)^{(k+1)},(x^i)^{(k+1)})=0, 1\le i\le d,
\end{aligned}$$

$$\begin{aligned}
&\tfrac{1}{2}\nabla_{x^i}\bar{\mathcal{G}}((x^1)^{(k+1)},\ldots,(x^{i-1})^{(k+1)},(x^i)^{(k+1)},(x^{i+1})^{(k)},\ldots,(x^d)^{(k)},(\bar{x}^{d+1})^{(k)},\ldots,(\bar{x}^{2d})^{(k)})\\
&-\tfrac{1}{2}(\bar{\lambda}^i)^{(k)}-\tfrac{1}{2\mu}((\bar{x}^0)^{(k+1)}-(\bar{x}^i)^{(k+1)})-\textstyle\sum_{j=1}^m(u_j^i)^{(k+1)}\nabla_{x^i}h_j((\bar{x}^i)^{(k+1)},(x^i)^{(k+1)})\\
&-\textstyle\sum_{j=1}^\ell(v_j^i)^{(k+1)}\nabla_{x^i}e_j((\bar{x}^i)^{(k+1)},(x^i)^{(k+1)})=0, 1\le i\le d,
\end{aligned}$$

and

$$(x^i)^{(k+1)}\in S,\quad (u_j^i)^{(k+1)}\le 0,\quad (u_j^i)^{(k+1)}h_j((\bar{x}^i)^{(k+1)},(x^i)^{(k+1)})=0,\quad i=1,\ldots,d, j=1,\ldots,\ell.$$

The KKT condition for j-th subproblem with $d+1\le j\le 2d$ is similar to the above formula and hence is omitted here. For the 0-th subproblem, the KKT condition is different, which can be described as follows:

$$\begin{aligned}
&\tfrac{1}{2}\textstyle\sum_{i=1}^d(\lambda^i)^{(k)}+\tfrac{1}{2\mu}\sum_{i=1}^d((x^0)^{(k+1)}-(x^i)^{(k)})-\sum_{j=1}^m(u_j^0)^{(k+1)}\nabla_{\bar{x}^0}h_j((\bar{x}^0)^{(k+1)},(x^0)^{(k+1)})\\
&-\textstyle\sum_{j=1}^\ell(v_j^0)^{(k+1)}\nabla_{\bar{x}^0}e_j((\bar{x}^0)^{(k+1)},(x^0)^{(k+1)})=0,
\end{aligned}$$

$$\begin{aligned}
&\tfrac{1}{2}\textstyle\sum_{i=1}^d(\bar{\lambda}^i)^{(k)}+\tfrac{1}{2\mu}\sum_{i=1}^d((\bar{x}^0)^{(k+1)}-(\bar{x}^i)^{(k)})-\sum_{j=1}^m(u_j^0)^{(k+1)}\nabla x^0h_j((\bar{x}^0)^{(k+1)},(x^0)^{(k+1)})\\
&-\textstyle\sum_{j=1}^\ell(v_j^0)^{(k+1)}\nabla_{x^0}e_j((\bar{x}^0)^{(k+1)},(x^0)^{(k+1)})=0,
\end{aligned}$$

and

$$(x^0)^{(k+1)}\in S,\quad (u_j^0)^{(k+1)}\le 0,\quad (u_j^0)^{(k+1)}h_j((\bar{x}^0)^{(k+1)},(x^0)^{(k+1)})=0,\quad 0j=1,\ldots,\ell.$$

By the updating rule for λ^i in (9), we also have the term

$$\lim_{k\to\infty}\frac{1}{2\mu}\left((x^i)^{(k+1)}-(x^0)^{(k+1)}\right)=\lim_{k\to\infty}\frac{1}{2\mu}\left((\bar{x}^i)^{(k+1)}-(\bar{x}^0)^{(k+1)}\right)=0,\forall i=1,\ldots,2d.$$

The rest of the proof is similar to the arguments used in Theorem 3.1. □

For the specific problem: unimodularly constrained complex polynomial problem (13), the convergence does not need to involve the Lagrangian multipliers.

Theorem 3.4 *Denote* $z^k:=\left([(x^i)^{(k)}]_{i=0}^{2d};[(\lambda^i)^{(k)}]_{i=1}^{2d}\right)$ *is the sequence generated in* (12)*, and assume that*

$$\lim_{k\to\infty}(z^{k+1}-z^k)=0.$$

Then any limit point of $\{z^k\}_{k=1}^\infty$ *is a KKT point of Problem* (13)*. Consequently, if* $\{(x^0)^{(k)}\}_{k=1}^\infty$ *converges, it converges to a KKT point of Problem* (13)*.*

The proof of the theorem above is similar to that of Theorem (3.2), and is thus omitted here.

4 Numerical Performance

In this section, we shall compare our algorithms proposed in Section 2 with two competing methods. One of them is the SOS method, based on which Henrion et al. [11] developed a specialized Matlab toolbox known as GloptiPoly 3 and will be used in our test. The other one is the MBI method and the codes in [1] and [4] are also used in our simulations. All the simulations were conducted in an Intel Core i5-2520M 2.5GHz computer with 4GB of RAM. For convenience of the readers, below a list of abbreviations in the tables are provided:

(aver)Time:	(average) cpu seconds to solve one instance;
ADMM1:	run ADMM from 1 starting point;
ADMM5:	run ADMM from 5 random starting points;
MBI1:	run MBI from 1 starting point;
MBI5:	run MBI from 5 random starting points;
Best Val.:	the best solution value among the ones generated by different starting points;
GLP:	run GloptiPoly 3;
status:	1 means GloptiPoly 3 finds the global optimum of the problem.

4.1 Results for Spherical Constraint Polynomial Optimization

In this part, we present some numerical tests on (10). In particular, we consider the fourth order problem

$$\begin{array}{ll} \min & \mathcal{F}(x,x,x,x) \\ \text{s.t.} & x^\top x = 1, x \in \mathbb{R}^n, \end{array} \tag{26}$$

where $\mathcal{F}$ is super-symmetric and generated randomly. In particular, we first generate a tensor $\mathcal{F}'$ such that its n^4 entries follow i.i.d. normal distribution. Then we symmetrize $\mathcal{F}'$ to form a super-symmetric tensor $\mathcal{F}$. Both the ADMM and SOS algorithms can work directly on problem (26). However, direct implementation of MBI to problem (26) will possibly result in a local maximum instead of local minimum. To circumvent this, as suggested in [4], we apply the MBI method to solve the following equivalent problem instead:

$$\begin{array}{ll} \min & \mathcal{F}(x,x,x,x) - 6(x^\top x)^2 \\ \text{s.t.} & x^\top x = 1, x \in \mathbb{R}^n. \end{array}$$

Notice that due to the constraint $x^\top x = 1$, the added term in the objective is a constant, hence the problem is equivalent to the original one. However, the transformation helps to avoid getting trapped in a local maximum.

The comparison is listed in Table 1. From this table, we see that for most problem instances ADMM and MBI can achieve the global optimum if run for 5 times from different starting points. In several instances, ADMM even returned better solutions than MBI did, and these cases are highlighted in bold. In terms of the computational time, we see that GloptiPoly 3 is most time consuming while

ADMM requires the least computational efforts. So we conclude that the overall performance of ADMM is most desirable among the three methods experimented, at least for this set of problem cases.

4.2 Results for Unimodularly Constrained Complex Polynomial Optimization

In this subsection, we report numerical tests on Problem (13). Precisely, the problem has the following specific form

$$\begin{array}{ll} \min & \sum\limits_{r=1}^{R_1}(x^\dagger A^r x)(x^\dagger A^{r\dagger}x) - \sum\limits_{r=1}^{R_2}(x^\dagger B^r x)(x^\dagger B^{r\dagger}x) \\ \text{s.t.} & |x_j| = 1,\ \forall\ j,\ x \in \mathbb{C}^n, \end{array} \tag{27}$$

where A^r and B^r are generated randomly. This problem was first studied in [1] and it was solved by MBI method there. To implement MBI, like in the last part, we actually work on an equivalent problem

$$\begin{array}{ll} \min & \sum\limits_{r=1}^{R_1}(x^\dagger A^r x)(x^\dagger A^{r\dagger}x) - \sum\limits_{r=1}^{R_2}(x^\dagger B^r x)(x^\dagger B^{r\dagger}x) - 6(x^\dagger x)^2 \\ \text{s.t.} & |x_j| = 1,\ \forall\ j,\ x \in \mathbb{C}^n. \end{array}$$

Since GloptiPoly 3 cannot be applied when the model involves complex variables, we shall only compare MBI with ADMM in Table 2.

In this table, we see that the advantage of ADMM is even more pronounced. In particular, ADMM can provide a solution no worse than that of MBI for most problem instances and yielded better solutions in a significant amount of cases, which are marked **bold** in the table. Moreover, ADMM is much faster than MBI for this particular problem. Note that when $n = 15$, ADMM is about 40 times faster than MBI.

References

[1] A. Aubry, A. De Maio, B. Jiang, and S. Zhang, *Cognitive Approach for Ambiguity Function Shaping*, Proceedings of The Seventh IEEE Sensor Array and Multichannel Signal Processing Workshop, 2012. 2, 6, 12, 13

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, 2011. 2

[3] B. Chen, *Optimization with Block Variables: Theory and Applications*, Ph.D. Thesis, The Chinese Univesrity of Hong Kong, June 2012.

[4] B. Chen, S. He, Z. Li and S. Zhang, *Maximum Block Improvement and Polynomial Optimization*, SIAM Journal on Optimization, 22, pp. 87-107, 2012. 2, 5, 12

[5] J. Douglas and H.H. Rachford, em On the numerical solution of the heat conduction problem in 2 and 3 space variables, Transactions of the American Mathematical Society, 82, 421-439, 1956. 2

[6] J. Eckstein and D. P. Bertsekas. *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming, 55, 293-318, 1992. 7

[7] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*, North-Holland Pub. Co., 1983. 7

[8] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2, 323-343, 2009. 2

[9] E. Kofidis and P. A. Regalia, *On the Best Rank-1 Approximation of Higher-Order Supersymmetric Tensors*, SIAM J. Matrix Anal. Appl., 23, 863-884, 2002. 5

[10] T.G. Kolda and J.R. Mayo, *Shifted power method for computing tensor eigenpairs*, SIAM J. Matrix Analysis, 32, 1095-1124, 2011. 5

[11] D. Henrion, J.B. Lasserre, and J. Loefberg, *GloptiPoly 3: Moments, optimization and semidefinite programming*, Optimization Methods and Software, 24, 761-779, 2009. 12

[12] B. Jiang, S. Ma, and S. Zhang, *Tensor Principal Component Analysis via Convex Optimization*, Working Paper, 2012. 5

[13] B. Jiang, Z. Li, and S. Zhang, *Conjugate Symmetric Complex Tensors and Applications*, Working Paper, 2013. 4

[14] J. B. Lasserre, *Global Optimization with Polynomials and the Problem of Moments*, SIAM Journal on Optimization, 11, 796–817, 2001. 2

[15] L.-H. Lim, *Singular values and eigenvalues of tensors: a variational approach*, Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP 05), 129-132, 2005. 5

[16] Z. Li, S. He, and S. Zhang, *Approximation Methods for Polynomial Optimization: Models, Algorithms, and Applications*, SpringerBriefs in Optimization, Springer, New York, NY, 2012. 2

[17] S. Ma, *Alternating direction method of multipliers for sparse principal component analysis*, preprint, 2011. 2

[18] P. A. Parrilo, *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, Ph.D. Dissertation, California Institute of Technology, Pasadena, CA, 2000. 2

[19] L. Qi, *Eigenvalues of a Real Supersymmetric Tensor*, Journal of Symbolic Computation, 40, 1302-1324, 2005. 5

[20] L. Qi, F. Wang and Y. Wang, *Z-eigenvalue Methods for a Global Polynomial Optimization Problem*, Mathematical Programming, Series A, 118, 301-316, 2009. 5

[21] M. Tao and X. Yuan. *Recovering low-rank and sparse components of matrices from incomplete and noisy observations*, SIAM J. Optim., 21, 57-81, 2011. 2

[22] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM Journal on Imaging Sciences, 1, 248-272, 2008. 2

[23] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Mathematical Programming Computation, 2, 203-230, 2010. 2

[24] Y. Xu, W. Yin, Z. Wen and Y. Zhang, *An Alternating Direction Algorithm for Matrix Completion with Nonnegative Factors*, Journal of Frontiers of Mathematics in China, Special Issues on Computational Mathematics, 365-384, 2011. 3, 7, 8

[25] J. Yang and Y. Zhang, *Alternating direction algorithms for ℓ_1 problems in compressive sensing*, SIAM Journal on Scientific Computing, 33, 250-278, 2011. 2

Inst. #	MBI1		ADMM1		MBI5		ADMM5		GLP		
	Val.	Time	Val.	Time	Best Val.	Aver. Time	Best Val.	Aver. Time	Val.	Time	Status
Dimension $n = 6$											
1	-2.40	1.12	**-2.41**	0.23	-2.51	0.15	**-3.13**	0.04	-3.13	2.00	1
2	-2.61	0.06	**-2.77**	0.03	-3.18	0.07	-3.18	0.05	-3.18	0.37	1
3	-3.02	0.44	-2.45	0.02	-3.67	0.14	-3.67	0.03	-3.67	0.31	1
4	-3.03	0.17	-3.03	0.05	-3.03	0.28	-3.03	0.06	-3.03	0.44	1
5	-1.70	0.12	**-3.13**	0.06	-3.13	0.12	-3.13	0.05	-3.13	0.37	1
6	-4.07	0.09	-4.07	0.02	-4.07	0.08	-4.07	0.02	-4.07	0.30	1
7	-3.16	0.26	-2.20	0.13	-3.16	0.17	-3.16	0.06	-3.16	0.33	1
8	-2.89	0.09	-2.89	0.03	-2.89	0.09	-2.89	0.03	-2.89	0.45	1
9	-3.05	0.19	-2.14	0.14	-3.05	0.29	-3.05	0.07	-3.05	0.51	1
10	-2.98	0.17	**-3.75**	0.05	-3.75	0.21	-3.75	0.03	-3.75	0.36	1
Dimension $n = 9$											
1	-4.43	0.28	-4.43	0.09	-4.43	0.22	-4.43	0.05	-4.43	2.15	1
2	-3.92	0.14	-3.92	0.09	-4.45	0.14	-4.42	0.08	-4.45	1.37	1
3	-3.30	0.20	**-3.89**	0.02	-4.11	0.23	-4.11	0.05	-4.11	1.22	1
4	-4.78	0.16	-4.78	0.08	-4.78	0.16	-4.78	0.05	-4.78	1.36	1
5	-4.16	0.45	-3.58	0.04	-4.19	0.15	-3.58	0.06	-4.19	1.28	1
6	-4.60	0.08	-4.26	0.08	-4.60	0.12	-4.60	0.05	-4.60	1.40	1
7	-4.06	0.34	-4.06	0.09	-5.08	0.17	-5.08	0.04	-5.08	1.29	1
8	-4.42	0.23	-3.84	0.04	-4.61	0.17	-4.61	0.04	-4.61	1.25	1
9	-3.83	0.12	-3.83	0.05	-3.83	0.15	-3.83	0.06	-3.83	1.40	1
10	-3.21	0.24	-2.72	0.04	-3.60	0.17	-3.60	0.04	-3.60	1.28	1
Dimension $n = 12$											
1	-5.96	0.22	-4.68	0.06	-5.96	0.27	-5.18	0.03	-5.96	20.89	1
2	-4.73	0.23	-4.73	0.06	-4.73	0.19	-5.70	0.07	-5.70	17.69	1
3	-5.11	0.14	-4.93	0.06	-5.11	0.18	**-5.42**	0.06	-5.42	19.52	1
4	-5.09	0.09	-5.09	0.06	-5.19	0.27	-5.19	0.06	-5.19	21.42	1
5	-4.09	0.10	-4.09	0.10	-5.43	0.23	-5.43	0.20	-5.43	19.56	1
6	-3.76	0.24	-3.32	0.10	-4.74	0.11	-4.74	0.05	-4.81	17.60	1
7	-4.18	0.22	-4.18	0.09	-4.83	0.22	-4.83	0.09	-4.83	23.38	1
8	-5.44	0.35	-4.90	0.05	-5.44	0.28	-5.44	0.12	-5.44	23.37	1
9	-4.50	0.16	-4.50	0.07	-4.76	0.13	-4.76	0.05	-5.58	17.69	1
10	-5.29	0.19	-5.29	0.11	-5.29	0.12	-5.29	0.07	-5.45	19.66	1
Dimension $n = 15$											
1	-5.17	0.41	-4.70	0.32	-5.78	0.34	-5.78	0.08	-5.78	190.34	1
2	-5.77	0.47	-5.77	0.13	-5.77	0.28	-5.77	0.13	-5.92	206.42	1
3	-5.42	0.22	-5.41	0.11	-6.34	0.27	-6.36	0.09	-6.36	229.15	1
4	-5.04	0.30	**-5.54**	0.18	-5.79	0.21	**-7.01**	0.09	-7.01	188.06	1
5	-5.08	0.27	-4.79	0.09	-5.75	0.23	-5.75	0.06	-6.06	201.93	1
6	-5.98	0.10	**-6.37**	0.03	-5.98	0.30	**-6.37**	0.10	-6.37	176.28	1
7	-4.57	0.31	**-5.82**	0.26	-5.97	0.27	-5.97	0.15	-5.97	230.30	1
8	-6.22	0.57	-5.16	0.08	-6.22	0.29	-6.22	0.22	-6.22	203.22	1
9	-5.45	0.29	**-6.14**	0.10	-5.94	0.23	**-6.60**	0.12	-6.60	214.75	1
10	-6.24	0.39	-5.90	0.11	-6.24	0.17	-6.24	0.08	-6.24	201.80	1

Table 1: Comparison of ADMM with $\mu = 0.8$, MBI and Gloptli3 for Problem (26).

Inst. #	MBI1		ADMM1		MBI5		ADMM5	
	Val.	Time	Val.	Time	Best Val.	Aver. Time	Best Val.	Aver. Time
Dimension $n = 6$								
1	-29.68	0.72	**-32.99**	0.19	-45.70	0.55	-45.70	0.45
2	-66.34	0.59	-55.89	0.08	-66.34	0.32	-66.34	0.08
3	-50.46	0.20	-50.46	0.05	-50.46	0.32	-50.46	0.15
4	-40.22	0.33	-40.22	0.13	-40.22	0.43	-40.22	0.40
5	-41.59	0.97	-36.69	1.06	-56.75	0.43	-56.75	0.30
6	-61.82	0.20	-61.82	0.05	-61.82	0.39	-61.82	0.31
7	-58.98	0.64	-58.98	0.43	-58.98	0.57	-58.98	0.49
8	-52.84	0.49	-52.84	0.47	-52.84	0.46	-52.84	0.26
9	-45.27	0.34	-45.27	0.19	-45.27	0.33	-45.27	0.19
10	-26.35	0.23	-26.36	0.07	-41.74	0.32	-41.74	0.26
Dimension $n = 9$								
1	-243.78	1.10	-243.80	0.20	-243.78	0.68	-243.80	0.14
2	-165.69	1.27	-165.73	0.49	-179.56	0.71	**-185.14**	0.34
3	-202.20	0.27	-202.21	0.05	-202.20	0.70	-202.21	0.23
4	-173.96	0.59	-173.96	0.06	-173.96	0.84	-173.96	0.14
5	-156.42	0.61	**-184.01**	0.08	-184.01	0.79	-184.01	0.08
6	-148.86	0.86	-136.02	1.04	-185.53	0.78	-185.53	0.34
7	-125.77	1.01	-125.78	0.14	-160.64	0.67	**-160.65**	0.15
8	-152.74	1.04	-152.75	0.24	-154.83	0.74	-154.83	0.26
9	-247.29	0.74	-247.29	0.43	-247.29	0.86	-247.29	0.44
10	-108.67	0.53	-108.68	0.13	-150.89	0.91	-150.89	0.14
Dimension $n = 12$								
1	-367.27	3.73	**-404.58**	0.13	-404.56	3.43	-404.58	0.40
2	-397.14	1.49	-397.15	0.09	-398.01	2.31	-398.02	0.18
3	-319.20	4.64	**-353.09**	0.19	-353.07	3.26	-353.09	0.16
4	-426.91	0.90	-426.91	0.09	-426.91	1.75	-426.91	0.12
5	-320.66	2.45	**-352.00**	0.08	-344.62	2.27	**-415.99**	0.12
6	-371.18	4.08	-270.43	0.77	-385.45	2.93	-385.46	0.50
7	-436.90	3.05	-251.82	0.10	-436.90	2.34	-436.91	0.10
8	-311.82	5.13	**-372.68**	0.09	-376.70	2.62	-372.68	0.15
9	-426.09	5.86	-348.37	0.09	-450.00	2.71	-450.02	0.16
10	-345.92	2.01	**-345.94**	0.10	-425.70	3.04	-425.74	0.35
Dimension $n = 15$								
1	-648.21	13.37	-619.35	0.30	-753.54	10.22	-753.56	0.53
2	-710.54	9.09	**-712.52**	1.44	-770.17	8.18	**-791.18**	1.07
3	-723.81	7.56	**-723.82**	0.16	-723.81	8.82	**-766.35**	0.17
4	-683.03	13.41	-576.47	0.28	-712.52	6.74	**-712.56**	0.25
5	-758.24	13.58	-514.61	1.43	-759.47	6.01	-758.29	0.45
6	-840.12	7.45	**-840.16**	0.24	-856.72	5.85	**-856.73**	0.41
7	-582.62	12.05	**-583.87**	1.52	-654.34	5.79	-642.03	0.48
8	-570.86	4.58	**-625.53**	0.13	-638.60	6.01	**-668.08**	0.19
9	-658.36	6.25	**-712.45**	0.67	-712.33	3.95	**-712.46**	0.30
10	-628.83	6.40	**-628.86**	0.13	-628.83	9.17	**-651.68**	0.74

Table 2: Comparison of ADMM with $\mu = 0.8$ and MBI for Problem (27).