Low Rank Decompositions for Sum of Squares Optimization

by

Jia Li Sun

Submitted to the School of Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2006

Author ....
School of Engineering
Aug 11, 2006

Certified by ....
Pablo A. Parrilo
Associate Professor
Thesis Supervisor

Accepted by ....
Jaime Peraire
Professor of Aeronautics and Astronautics
Codirector, Computation for Design and Optimization Program

Low Rank Decompositions for Sum of Squares Optimization

by

Jia Li Sun

Submitted to the School of Engineering
on Aug 11, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Computation for Design and Optimization

Abstract

In this thesis, we investigate theoretical and numerical advantages of a novel representation for Sum of Squares (SOS) decomposition of univariate and multivariate polynomials. This representation formulates a SOS problem by interpolating a polynomial at a finite set of sampling points. As compared to the conventional coefficient method of SOS, the formulation has a low rank property in its constraints. The low rank property is desirable as it improves computation speed for calculations of barrier gradient and Hessian assembling in many semidefinite programming (SDP) solvers. Currently, SDPT3 solver has a function to store low rank constraints to explore its numerical advantages. Some SOS examples are constructed and tested on SDPT3 to a great extent. The experimental results demonstrate that the computation time decreases significantly. Moreover, the solutions of the interpolation method are verified to be numerically more stable and accurate than the solutions yielded from the coefficient method.

Thesis Supervisor: Pablo A. Parrilo
Title: Associate Professor

Acknowledgments

I would like to thank the following people for their help in the creation of this thesis from the research to the final formulation.

First and foremost, I would like to express my deepest gratitude and appreciation to my thesis advisor Prof. Pablo Parrilo for introducing me to the subject and for sharing with me his vast knowledge. He provided a motivating and enthusiastic atmosphere during the discussions I had with him. It was a great pleasure to conduct this thesis under his supervision.

I am highly indebted to Prof. Toh Kim Chuan, Professor at National University of Singapore. I received valuable suggestions and guidance on SDPT3 solver. This thesis certainly would not be possible without his help.

I am also grateful to Singapore-MIT Alliance for their financial support.

Needless to say, I am thankful to my fellow colleagues in the research group. Their hospitality has made my stay in MIT a pleasant one. I would also like to thank my parents for their love and endless support throughout the year.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Sum of squares (SOS) optimization has wide applications in many areas such as system and control theory, continuous and combinatorial optimization, and real algebraic geometry. The existence of a SOS form of a polynomial is a sufficient condition to prove its non-negativity. SOS problems can be reformulated and solved by using semidefinite programming (SDP) techniques. As there are many numerical solvers available in the field of SDP such as `SeDuMi` and `SDPT3`, solving SOS is more convenient and tractable than proving non-negativity of polynomials directly.

Many researchers have proposed methodologies for SOS decompositions. Currently, the most common technique is the coefficient method. Assume we have a given polynomial $p(x)$ with degree of $2d$. We can use SOS condition to prove its nonnegativity. The SOS condition is then equivalently expressed in terms of a SDP formulation as follows:

$$p(x) = v(x)^T Q v(x), \tag{1.1}$$

where Q is constrained to be a positive semidefinite (PSD) matrix and $v(x)$ spans $\Re^d$. Constraint equations are then formed by comparing coefficients of the monomial form of polynomial $p(x)$ with the corresponding coefficients obtained from $v(x)^T Q v(x)$. As $p(x)$ has $2d + 1$ coefficients, we have $2d + 1$ constraint equations. Finally, the formulations are interfaced with SDP solvers. Repeated simulations show that the computation time for this formulation increases at an increasing rate when the degree grows bigger. When the degree of polynomials becomes large, the computation effort becomes too intensive for computers. This has spurred research into exploring numerical properties of SOS for more efficient algorithms.

The earlier work by Genin *etal.* [1], [2] adopted low displacement rank of Hankel and Toeplitz matrices to reduce computation effort for the gradient and Hessian of the dual barrier function. They also use Chebyshev basis to avoid ill-conditioned numerical systems. However, their work is restricted to the univariate case. T. Roh and L. Vandenberghe [3] have attempted to explore better algorithm for the multivariate case. They parametrize the SDP formulation of SOS representations by discrete transforms. The numerical stability of the resulting formulation improves. Recently, P. Parrilo and J. Löfberg [4] proposed a novel technique for SOS decompositions based on interpolation of polynomials at a finite set of discrete sampling points for both univariate and multivariate polynomials. Instead of comparing coefficients of polynomials, this formulation enforces constraints for equations at the sampling points. This technique has appealing features such as simplicity in representation and linearity in computation time. In this new method, a set of discrete sampling points must be chosen carefully. The choice of sampling points plays an important role in the numerical conditioning of the resulting SDP. The numerical values of polynomial $p(x_i)$ are evaluated explicitly at these sampling points. Simultaneously, the numerical values of $v(x_i)^T Q v(x_i)$ in terms of unknown variables are also computed. After that, constraints are imposed to equate $p(x_i) = v(x_i)^T Q v(x_i),\ i = 1, \ldots, 2d$ at the set of sampling points.

In this research, we exploit the low rank property of the interpolation method. The low rank property has both numerical and computational advantages. It enables fast computation of barrier gradient and Hessian assembling. In addition, the constraints are now simplified as

$$p(x_i) = A_i \bullet Q,\ A = v(x_i)v(x_i)^T,\ i = 1, \ldots, n, \tag{1.2}$$

where $v(x_i)$ is a vector. The formulation is interfaced with the SDPT3 solver [5] to utilize its built-in low rank function. Several examples have been modeled and simulated. The formulations and experimental results will be discussed in detail later.

1.2 Structure of the thesis

This section outlines the structure of the thesis. Firstly, we discuss sum of squares problems in general. Secondly, several polynomial bases are reviewed in detail. Their structure and numerical properties are compared. Thirdly, different techniques for SOS formulations using SDP are introduced. Several examples are used to implement these techniques. After that, the computational results for the examples are summarized. The relative advantages and disadvantages of each method are outlined. Follows that, we describe how to interface the SOS formulations with cvx. Several examples are quoted to illustrate the implementations. Lastly, we conclude our findings and discuss possible future research directions.

Chapter 2

Sum of Squares and Semidefinite Programming

2.1 Sum of squares background

For a polynomial $p(x)$, its SOS form is defined as follows [6]:

Definition 2.1 *A polynomial $p(x)$ is a SOS if there exists $q_1, \cdots, q_m \in \Re[x]$ such that*

$$p(x) = \sum_{k=1}^{m} q_k^2(x). \tag{2.1}$$

It can be observed that if $p(x)$ has a SOS form, $p(x)$ must be nonnegative, i.e. $p(x) \geq 0$ for all $x \in \Re$. Therefore, it implies SOS condition is a sufficient condition for global nonnegativity. For univariate polynomials, we can always write nonnegative univariate polynomials in SOS form.

However, nonnegative multivariate polynomials do not always have SOS forms. In most cases, nonnegativity is not equivalent to SOS condition. It is NP-hard to check nonnegativity for polynomials of degree greater than or equal to four.

B. Reznick described in his paper [7] the classical results obtained by David Hilbert more than a century ago. David Hilbert showed that the SOS condition and the non-negativity are only equivalent in three cases. First case is the univariate polynomials. All nonnegative univariate polynomials satisfy SOS condition. Secondly, this equality also holds for quadratic polynomials. The third class of polynomials is the bivariate quartics. As can be seen from the name, these polynomials have two variables and the degree of polynomials is up to four.

Besides the three cases, it can be shown that there always exists nonnegative polynomials which do not have equivalent SOS form. One classical counterexample is provided by Motzkin [7]. Given a sextic polynomial

$$M(x,y,z) = x^4y^2 + x^2y^4 + z^6 - 3x^2y^2z^2.$$

This polynomial is shown to be nonnegative by applying an arithmetic-geometric inequality as shown below:

As

$$\frac{a+b+c}{3} \geq (abc)^{1/3}, \tag{2.2}$$

Let $a = x^4y^2, b = x^2y^4, c = z^6$, Substitute a, b, c into equation (2.2),

$$\begin{aligned} &\Rightarrow \frac{x^4y^2 + x^2y^4 + z^6}{3} \geq (x^6y^6z^6)^{1/3} \\ &\Leftrightarrow M(x,y) = x^4y^2 + x^2y^4 + z^6 - 3x^2y^2z^2 \geq 0. \end{aligned}$$

At the same time, we are unable to derive the SOS form for $M(x,y)$. It can be shown that

$$\begin{aligned} M(x,y,z) = &\left(\frac{(x^2-y^2)z^3}{(x^2+y^2)}\right)^2 + \left(\frac{x^2y(x^2+y^2-2z^2)}{(x^2+y^2)}\right)^2 \\ &+ \left(\frac{xy^2(x^2+y^2-2z^2)}{(x^2+y^2)}\right)^2 + \left(\frac{xyz(x^2+y^2-2z^2)}{(x^2+y^2)}\right)^2. \end{aligned} \tag{2.3}$$

$M(x,y,z)(x^2+y^2)$ has a SOS form, but $M(x,y,z)$ cannot be expressed in SOS form though it is proved to be nonnegative.

2.2 Semidefinite programming background

In this section, we discuss some basic concept of SDP. For this thesis, we use the following notations. We denote $x \bullet y$ as inner product of vectors, $X \bullet Y$ as inner product of two square matrices. If x and y are vectors in $\Re^n$, then $x \bullet y := \sum_{i=1}^{n} x_i y_i$. If X and Y are matrices in $\Re^{n \times n}$, their inner product equals to $trace(XY)$. We use $A \succeq B$ to indicate $A - B$ is PSD. Adopting notations used in [6], let $S^n \subset \Re^{n \times n}$ represent the space of symmetric real matrices of dimension $n \times n$.

SDP is a class of convex optimization problems. We have a linear objective function and subject to an affine combination of PSD matrices. SDP has nonlinear constraints. However, most interior-point methods for linear programming have been generalized to SDP. Moreover, these methods have polynomial worst-case complexity. These advantages motivate us to apply SDP in many areas. More details can be found in [8] and [9]. SDP is closely related to SOS, thus we will use SDP to solve SOS optimization problems.

The standard primal form of SDP is

$$\begin{aligned} \min \quad & C \bullet X \\ \text{subject to} \quad & A_i \bullet X = b_i, \ i = 1, \ldots, m \\ & X \succeq 0, \end{aligned} \tag{2.4}$$

where all $A_i \in S^n$, $b \in \Re^m$, $C \in S^n$ are given, and $X \in S^n$ is the variable.

The corresponding dual form is

$$\begin{aligned} \max \quad & b \bullet y \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i A_i \preceq C, \end{aligned} \tag{2.5}$$

where $y \in \Re^m$.

We can apply various bases on the primal or the dual to formulate our SDP problems.

2.3 Relationship between SOS and SDP

We start to demonstrate the relationship between SOS formulation and SDP by using a univariate case. Assume we have a univariate polynomial $p(x)$ with degree of $2d$, which can be written as the SOS form as shown in equation (2.1). We note that each $q_k(x)$ has degree less than or equal to d. It can be represented as a linear combination of monomials of x:

$$q_k(x) = [a_{0k}\ a_{1k}\ \cdots\ a_{dk}] \begin{bmatrix} 1 \\ x \\ \vdots \\ x^d \end{bmatrix}, \tag{2.6}$$

If we rewrite all $q_k(x)$ in a vector form, we arrive

$$\Rightarrow \begin{bmatrix} q_1(x) \\ q_2(x) \\ \vdots \\ q_m(x) \end{bmatrix} = \underbrace{\begin{bmatrix} a_{01} & a_{11} & \cdots & a_{d1} \\ a_{02} & a_{12} & \cdots & a_{d2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{0m} & a_{1m} & \cdots & a_{dm} \end{bmatrix}}_{V} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^d \end{bmatrix}. \tag{2.7}$$

We use $[x]_d$ to denote $[1, x, \ldots, x^d]^T$. Therefore, we can write $p(x)$ in terms of $[x]_d$ and V.

$$\begin{aligned} p(x) &= (V[x]_d)^T(V[x]_d) \\ &= [x]_d^T \underbrace{V^T V}_{Q} [x]_d \\ &= [x]_d^T Q [x]_d. \end{aligned} \tag{2.8}$$

Let $Q = V^T V$, it is obvious that Q is a PSD matrix. That verifies if $p(x)$ is SOS, then Q must be positive semidefinite. Conversely, if Q is positive semidefinite, we can always decompose Q by the Choleski factorization into $V^T V$. Consequently, we can arrive at a SOS form of $p(x)$.

2.4 Applications of SOS

It is generally NP hard to prove nonnegativity of polynomials. As SOS condition is sufficient to prove nonnegativity, and SOS form can be conveniently converted to SDP formulations, we can use SDP technique to prove nonnegativity globally or locally. Some applications and extensions of SOS are discussed below.

2.4.1 Lower bound problem

We are often interested to know the minimum value of a polynomial $p(x)$. It is computationally intensive to search for the global minimum. Therefore, we try to transform this problem into SOS form, and then the SDP formulation.

It is intuitive that $p(x) \geq \gamma$ for all x where γ is the global minimum. We can write this as:

$$\begin{aligned} & p(x) - \gamma \text{ is SOS,} \\ & \Rightarrow p(x) - \gamma \geq 0, \\ & \Leftrightarrow p(x) \geq \gamma. \end{aligned} \tag{2.9}$$

From the SOS form, we can derive the SDP formulations and solve the problem using SDP solvers.

2.4.2 Nonnegativity on intervals

In many cases, we wish to find an optimal solution within a constrained interval instead of global domain. These restrictions can be implemented by modifying the SOS form with additional terms. Some classical theorems are listed below to illustrate the extensions. The last result is known as the Markov-Lukács theorem [10] [11].

Theorem 2.1 *The polynomial* $p(x)$ *is nonnegative on* $[0, \inf)$, *if and only if it can be written as*

$$p(x) = s(x) + xt(x)$$

where s(x), t(x) are SOS. If deg(p) $= 2d$*, then we have deg(s)* $\leq 2d$*, deg(t)* $\leq 2d - 2$.

Theorem 2.2 *The polynomial* $p(x)$ *is nonnegative on* $[a, b]$*, where* $a < b$*, if and only if it can be written as*

$$\begin{cases} p(x) = s(x) + (x-a)(b-x)t(x), & \text{if } deg(p) \text{ is even}, \\ p(x) = (x-a)s(x) + (b-x)t(x), & \text{if } deg(p) \text{ is odd}, \end{cases} \tag{2.10}$$

where s(x), t(x) are SOS. In the first case, we have deg(p) $= 2d$*, and deg(s)* $\leq 2d$*, deg(t)* $\leq 2d - 2$*. In the second, deg(p)* $= 2d + 1$*, and deg(s)* $\leq 2d$*, deg(t)* $\leq 2d$.

Chapter 3

Review of Polynomial Bases

A polynomial is a mathematical expression involving a sum of powers in one or more variables multiplied by coefficients. A polynomial in one variable is called a univariate polynomial. On the other hand, a polynomial in multiple variables is called a multivariate polynomial.

3.1 Bases of univariate polynomials

In this section, we will discuss three bases for univariate polynomials: the monomial basis, the Chebyshev basis and the Lagrange basis. The concept of these bases will be extended to multivariate polynomials in the next section.

The space of polynomials in one variable with real coefficients is a vector space. A polynomial of degree at most n with real coefficients has a vector space of $\Re^n$. A vector space $\Re^n$ has a constant number of vectors in a basis. The dimension of a vector space $\Re^n$ is the number of vectors of a basis of $\Re^n$, which is n.

3.1.1 The monomial basis

A monomial basis [12] is a way to uniquely describe a polynomial using a linear combination of monomials. The monomial form of a polynomial is often used because of its simple structure. This form is also desirable as the polynomials in monomial form can be evaluated efficiently using the Horner algorithm [12].

The monomial basis for the vector space $\Re_n$ of univariate polynomials with degree n is defined as the sequence of monomials

$$1, x, x^2, \ldots, x^n.$$

The monomial form of a polynomial $p(x)$ in $\Re_n$ is a linear combination of monomials

$$a_0 1 + a_1 x + a_2 x^2 + \ldots + a_n x^n.$$

Alternatively, the shorter notation can be used

$$p(x) = \textstyle\sum_{\nu=0}^{n} a_\nu x^\nu.$$

Without loss of generality, assume we have an even degree univariate polynomial $p(x)$ with highest degree equal to $n = 2d$:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{2d} x^{2d}, \tag{3.1}$$

The monomial basis, $v(x)$ of SDP formulations for $p(x)$ in equation (1.1), has degree d. It spans the vector space $\Re_d$:

$$v(x) = [1, x, x^2, \ldots, x^d]^T.$$

In addition, a polynomial can always be converted into monomial form by calculating its Taylor expansion around zero. Therefore, the monomial basis is popularly adopted on both the primal and dual in the conventional SDP formulations of SOS.

3.1.2 The Chebyshev basis

The Chebyshev basis (also known as the Chebyshev polynomials) is a sequence of orthogonal polynomials. The orthogonal polynomials are related to de Moivre's formula and are easily defined recursively, like Fibonacci or Lucas numbers [13]. There are two kinds of Chebyshev polynomials. The Chebyshev polynomials of the first kind are usually denoted by T_n and Chebyshev polynomials of the second kind are denoted by U_n. The Chebyshev polynomials T_n or U_n are polynomials of degree n.

We consider first use the Chebyshev polynomials of the first kind. The Chebyshev polynomials of the first kind have two equivalent definitions [13]. On one hand, they can be defined by a recurrence relation.

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x). \end{aligned} \tag{3.2}$$

On the other hand, they can also be defined by a trigonometric identity:

$$T_n(\cos(\theta)) = \cos(n\theta). \tag{3.3}$$

In a more explicit form, the identity is written as

$$T_n(x) = \begin{cases} \cos(n\cos^{-1}(x)), & x \in [-1,1] \\ \cosh(n \operatorname{arccosh}(x)), & x \geq 1 \\ (-1)^n \cosh(n \operatorname{arccosh}(-x)), & x \leq -1 \end{cases} \tag{3.4}$$

The Chebyshev form of a polynomial $p(x)$ in $\Re_n$ is a linear combination of Chebyshev polynomials

$$c_0T_0(x) + c_1T_1(x) + c_2T_2(x) + \ldots + c_nT_n(x),$$

or in a shorter form,

$$p(x) = \textstyle\sum_{\nu=0}^{n} c_\nu T_\nu(x).$$

Similarly, for a polynomial $p(x)$ with degree $2d$, the Chebyshev basis, $v(x)$ of SDP formulations for $p(x)$ in equation (1.1), has degree d. It spans the vector space $\Re_d$:

$$v(x) = [T_0(x), T_1(x), T_2(x), \ldots, T_d(x)]^T.$$

The Chebyshev basis has some special properties. Firstly, T_n forms a sequence of orthogonal polynomials. The polynomials of the first kind are orthogonal [13] with respect to the weight $\frac{1}{\sqrt{1-x^2}}$ on the interval [-1,1]. It is illustrated as shown below:

$$\int_{-1}^{1} T_n(x)T_m(x)\frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0, & : n \neq m \\ \pi, & : n = m = 0 \\ \pi/2, & : n = m \neq 0 \end{cases} \quad (3.5)$$

Another property of T_n is the leading coefficient of T_n is directly related to n. If $n \geq 1$, the leading coefficient of T_n is 2^{n-1}. If $n = 0$, the leading coefficient equals one.

In addition, one useful property is the multiplication property:

$$T_i(x)T_j(x) = \frac{1}{2}[T_{i+j}(x) + T_{|i-j|}(x)]. \quad (3.6)$$

The multiplication property enables us to convert the product of two polynomials into a linear combination of two Chebyshev polynomials.

Moreover, the composition identity may also be useful. It is shown below:

$$T_n(T_m(x)) = T_{nm}(x). \quad (3.7)$$

The roots of the Chebyshev polynomials are also called Chebyshev nodes. These nodes are used in polynomial interpolations in our formulations. The Chebyshev nodes of the first kind in the interval of $[a, b]$ are obtained by the formula given below:

$$x_i = 1/2(a+b) + 1/2(b-a)\cos(\frac{2i-1}{2n}\pi), \quad (3.8)$$

where $n = 2d + 1$. The Chebyshev nodes of the second kind in the interval of $[a, b]$ are obtained by the formula given below:

$$x_i = 1/2(a+b) + 1/2(b-a)\cos(\frac{i}{n+1}\pi), \quad (3.9)$$

where $n = 2d + 1$.

3.1.3 The Lagrange basis

The univariate polynomial $p(x)$ is represented in terms of Lagrange polynomials [14] as follows:

$$p(x) = \sum_{i=0}^{n} b_i L_i(x), \tag{3.10}$$

where b_i is a direct evaluation of $p(x_i)$. $L_i(x)$ is defined as

$$L_i(x) = \frac{\prod_{j=0,j\neq i}^{k}(x - \alpha_j)}{\prod_{j=0,j\neq i}^{k}(\alpha_i - \alpha_j)}, \tag{3.11}$$

α_j represents distinct sampling point used in forming the Lagrange basis. For $p(x)$ in $\Re^d$, $L_i(x)$ also has degree d. we need $d+1$ distinct sampling points to form the Lagrange basis.

For a polynomial $p(x)$ with degree $2d$, the Lagrange basis, $v(x)$ of SDP formulations for $p(x)$ in equation (1.1), has degree d. It spans the vector space $\Re_d$:

$$v(x) = [L_0(x), L_1(x), L_2(x), \ldots, L_d(x)]^T.$$

It can be observed that if x is one of the sampling points used in forming the Lagrange basis, i.e. $x = \alpha_k$, then we have

$$L_i(x) \begin{cases} 1, & k = i \\ 0, & k \neq i. \end{cases} \tag{3.12}$$

3.2 Bases of multivariate polynomials

Compared to univariate polynomials, multivariate polynomials of the same degree have much greater number of coefficients. The number of coefficients is determined by the highest degree of the polynomial ($2d$) and the number of variables (n). It can be evaluated by $\binom{n+2d}{n}$.

We adopt two multivariate bases: the monomial basis and the Chebyshev basis. In the following sections, we will illustrate them using bivariate polynomials, a special case of multivariate polynomials. Multivariate polynomials with more than two variables can be derived in a similar manner.

For univariate polynomials, it is clear that the coefficients of a polynomial are arranged in the ascending order with respect to the degree of monomials. However, in the case of multivariate polynomials, the monomial ordering becomes much more complicated. For consistency, we use graded lexicographic ordering [15] in this study.

3.2.1 The monomial basis

Assume now we have an even degree bivariate polynomial $p(x, y)$ with highest degree equal to $2d$:

$$p(x, y) = a_0 + a_1 y + a_2 x + a_3 y^2 + a_4 xy + a_5 x^2 + \cdots + a_k x^{2d}, \tag{3.13}$$

where $a_0, a_1, \ldots, a_k$ are coefficients of monomials following graded lexicographic ordering. Total number of coefficients of a given bivariate polynomial with degree $2d$ is $\binom{2d+2}{2}$. The monomial basis in SOS form for $p(x, y)$ has degree of at most d,

$$v(x) = [1, y, x, y^2, xy, x^2, \ldots, x^d]^T,$$

where the number of elements in $v(x)$ is $\binom{d+2}{2}$. For multivariate polynomials, the dimension of basis is large even for small degree. The complexity of the basis increases noticeably with the increase of degree.

3.2.2 The Chebyshev basis

In Chebyshev basis, the same bivariate polynomial $p(x, y)$ with highest degree equal to $2d$ can be written as:

$$p(x, y) = c_0 T_0 + c_1 T_1(y) + c_2 T_1(x) + c_3 T_2(y) + c_4 T_1(x) T_1(y) + c_5 T_2(x) + \cdots + c_k T_{2d}(x), \tag{3.14}$$

where $c_0, c_1, \ldots, c_k$ are coefficients of monomials following graded lexicographic ordering. Total number of coefficients of a given bivariate polynomial with degree $2d$ is $\binom{2d+2}{2}$. The Chebyshev basis in SOS form for $p(x, y)$ also has degree d,

$$v(x) = [1, T_1(y), T_1(x), T_2(y), T_1(x) T_1(y), T_2(x), \ldots, T_d(x)]^T,$$

where the number of elements in $v(x)$ is $\binom{d+2}{2}$. The Chebyshev basis has exactly the same dimension as the monomial basis. The monomial basis is direct and clear to understand. However, the Chebyshev basis normally is numerically more stable as its trigonometric representation can offset the exponential increase in degree for polynomial evaluations.

3.3 Relationship between bases

We use M, C and L to denote the monomial basis, the Chebyshev basis and the Lagrange basis respectively. For the corresponding transformation matrix, we use F_a^b to indicate the transformation from basis a to basis b. We will derive the transformation matrix relating the bases for univariate polynomials in this section.

3.3.1 The monomial basis and the Chebyshev basis

Firstly, we are interested in relating the Chebyshev basis to the monomial basis. We define the relation matrix as F_M^C. We relate the monomial basis and the Chebyshev basis as shown below:

$$[T_0\ T_1\ T_2\ \ldots\ T_n]^T = F_M^C[1\ x\ x^2\ \ldots\ x^n]^T. \tag{3.15}$$

The Chebyshev polynomials of the first kind [16], $T_n(x)$, are directly related to the monomial basis by a recursive formula defined in equation (3.2). The transformation matrix varies with the degree of the basis. We illustrate the transformation matrix by using a simple example when degree of the basis is four:

$$\begin{bmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ T_3(x) \\ T_4(x) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ -1 & 0 & 2 & \ddots & \vdots \\ 0 & -3 & 0 & 4 & 0 \\ 1 & 0 & -8 & 0 & 8 \end{bmatrix}}_{F_M^C,\ \text{degree of basis}=4.} \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{bmatrix}. \tag{3.16}$$

We notice that the transformation matrix F_M^C is a lower triangular matrix. It is generally difficult to compute the accurate solution of the inverse of F_M^C. Especially when the degree of polynomial is large, the resulting inverse matrix is often ill-conditioned. We also have

$$[1\ x\ x^2\ \ldots\ x^n]^T = F_C^M [T_0\ T_1\ T_2\ \ldots\ T_n]^T. \tag{3.17}$$

From equation (3.15), we derive

$$[1\ x\ x^2\ \ldots\ x^n]^T = (F_M^C)^{-1} [T_0\ T_1\ T_2\ \ldots\ T_n]^T. \tag{3.18}$$

Thus,

$$F_C^M = (F_M^C)^{-1}. \tag{3.19}$$

Given the transformation matrix relating the monomial basis and the Chebyshev basis, we can also use F_C^M and F_M^C to relate the monomial coefficients to the Chebyshev coefficients. A univariate polynomial $p(x)$ can be represented by the monomial coefficients, $[a_0\ a_1\ \ldots\ a_n]$, as follows:

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \\ &= [a_0\ a_1\ \ldots\ a_n][1\ x\ x^2\ \ldots\ x^n]^T. \end{aligned} \tag{3.20}$$

It can also be represented by the Chebyshev coefficients, $[c_0\ c_1\ \dots\ c_n]$:

$$\begin{aligned} p(x) &= c_0T_0 + c_1T_1 + c_2T_2 + \cdots + c_nT_n \\ &= [c_0\ c_1\ \dots\ c_n][T_0\ T_1\ T_2\ \dots\ T_n]^T. \end{aligned} \tag{3.21}$$

Substitute equation (3.15) into the above equation:

$$\begin{aligned} p(x) &= [c_0\ c_1\ \dots\ c_n][T_0\ T_1\ T_2\ \dots\ T_n]^T \\ &= [c_0\ c_1\ \dots\ c_n]F_M^C[1\ x\ x^2\ \dots\ x^n]^T \\ &= [a_0\ a_1\ \dots\ a_n][1\ x\ x^2\ \dots\ x^n]^T \\ \Rightarrow [a_0\ a_1\ \dots\ a_n] &= [c_0\ c_1\ \dots\ c_n]F_M^C. \end{aligned} \tag{3.22}$$

Thus, we also have

$$[a_0\ a_1\ \dots\ a_n] = [c_0\ c_1\ \dots\ c_n]F_M^C. \tag{3.23}$$

Conversely, it can also be written as

$$[c_0\ c_1\ \dots\ c_n] = [a_0\ a_1\ \dots\ a_n]F_C^M. \tag{3.24}$$

3.3.2 The monomial basis and the Lagrange basis

Similarly, we can derive the relationship between the monomial basis and the Lagrange basis as shown below. The transformation matrix of the monomial basis to the Lagrange basis is denoted as F_M^L. The two bases are related as follows:

$$[L_0\ L_1\ \dots\ L_n]^T = F_M^L[1\ x\ \dots\ x^n]^T, \tag{3.25}$$

and,

$$[1\ x\ \dots\ x^n]^T = F_L^M[L_0\ L_1\ \dots\ L_n]^T. \tag{3.26}$$

Therefore, we also have

$$F_M^L = (F_L^M)^{-1}. \tag{3.27}$$

With F_M^L and F_L^M, we are able to relate the Lagrange coefficients and the monomial coefficients. The polynomial $p(x)$ can be represented by the Lagrange coefficients, $[b_0\ b_1\ \dots\ b_n]$:

$$\begin{aligned} p(x) &= b_0 L_0 + b_1 L_1 + b_2 L_2 + \cdots + b_n L_n \\ &= [b_0\ b_1\ b_2\ \dots\ b_n][L_0\ L_1\ \dots\ L_n]^T \\ &= [b_0\ b_1\ b_2\ \dots\ b_n](F_M^L[1\ x\ \dots\ x^n]^T) \\ &= [a_0\ a_1\ a_2\ \dots\ a_n][1\ x\ \dots\ x^n]^T, \end{aligned} \tag{3.28}$$

$$\begin{aligned} \Rightarrow [a_0\ a_1\ a_2\ \dots\ a_n] &= [b_0\ b_1\ b_2\ \dots\ b_n]F_M^L, \\ \text{and } [b_0\ b_1\ b_2\ \dots\ b_n] &= [a_0\ a_1\ a_2\ \dots\ a_n]F_L^M. \end{aligned}$$

It is well-known that the Lagrange coefficients are also related to the monomial coefficients by the Vandermonde matrix. We denote F_{vand} as the Vandermonde matrix, and

$$\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{\text{Vandermonde matrix}=F_{vand}} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}. \tag{3.29}$$

Therefore, we can relate F_M^L and F_L^M with the Vandermonde matrix as the following:

$$\begin{aligned} [b_0\ b_1\ b_2\ \dots\ b_n]^T &= (F_L^M)^T[a_0\ a_1\ a_2\ \dots\ a_n]^T, \\ \Rightarrow F_{vand} &= (F_L^M)^T, && (3.30) \\ \Leftrightarrow F_L^M &= (F_{vand})^T, && (3.31) \\ F_M^L &= (F_{vand})^{-T}. && (3.32) \end{aligned}$$

3.3.3 The Chebyshev basis and the Lagrange basis

With F_M^C, F_C^M, F_M^L and F_L^M, we can also relate the Lagrange basis with the Chebyshev basis. The transformation matrix is defined as F_C^L and F_L^C respectively:

$$F_C^L = F_M^L F_C^M, \tag{3.33}$$

$$F_L^C = F_M^C F_L^M. \tag{3.34}$$

In general, the transformation matrix becomes ill-conditioned when we take the inverse. The Vandermonde matrix is dense in particular. As the degree of polynomial increases, the converted coefficients may not be very accurate due to ill-conditioning.

Chapter 4

SDP Formulations of SOS Problems

4.1 Overview

It has been shown in previous Chapters that nonnegativity can be relaxed to SOS condition. The SOS condition is then represented by various SDP formulations. We explain our formulations using the most basic problem, finding a SOS form of $p(x)$. Assume $p(x)$ is expressed in terms of monomial coefficients.

$$p(x) \text{ is SOS}, \Leftrightarrow p(x) = v(x)^T Q v(x),\ \ Q \succeq 0.$$

This is a SDP problem. For the primal side, there are many choices for the vector $v(x)$. In this project, I focus on three types of bases: the monomial basis, Chebyshev basis, and Lagrange basis. We have discussed these bases in detail in Chapter 3. Correspondingly, we can choose basis at the dual side. We also use the three types of bases for the dual side. These combinations result in nine possible approaches to solving SOS problems. Each approach is described in the sections below. Their theoretical advantages are also discussed. Modeling and simulations have also been carried out to verify the theoretical properties. These experimental results will be discussed in the next Chapter.

4.2 Primal: the monomial basis; Dual: the monomial basis

When the monomial basis is used on the dual form, we compare the monomial coefficients to set our constraint equations. The monomial coefficients of $p(x)$ with degree $2d$ is given. They are $[a_0, a_1, \ldots, a_{2d}]$. These coefficients are arranged in ascending degree of the power of x. The set of coefficients forms one side of the constraint equations. On the other side, we need to extract the corresponding monomial coefficients from $v(x)^T Q v(x)$. This step requires some derivations as follows:

$$\begin{aligned} \text{Let } Y &= v(x)v(x)^T \\ \Rightarrow p(x) &= Y \bullet Q \end{aligned} \tag{4.1}$$

With the monomial basis as $v(x)$,

$$Y = \begin{bmatrix} 1 & x & \cdots & x^d \\ x & x^2 & \cdots & x^{d+1} \\ \vdots & \cdots & \ddots & \vdots \\ x^d & x^{d+1} & \cdots & x^{2d} \end{bmatrix},$$

$$Q = \begin{bmatrix} Q_{00} & Q_{01} & \cdots & Q_{0d} \\ Q_{10} & Q_{12} & \cdots & Q_{1d} \\ \vdots & \cdots & \ddots & \vdots \\ Q_{d0} & Q_{d1} & \cdots & Q_{dd} \end{bmatrix},$$

$$Y \bullet Q = \begin{bmatrix} 1 & x & \cdots & x^d \\ x & x^2 & \cdots & x^{d+1} \\ \vdots & \cdots & \ddots & \vdots \\ x^d & x^{d+1} & \cdots & x^{2d} \end{bmatrix} \bullet \begin{bmatrix} Q_{00} & Q_{01} & \cdots & Q_{0d} \\ Q_{10} & Q_{12} & \cdots & Q_{1d} \\ \vdots & \cdots & \ddots & \vdots \\ Q_{d0} & Q_{d1} & \cdots & Q_{dd} \end{bmatrix}.$$

It can be observed that the monomial coefficients are the sum of respective elements of Q. Thus, we have the following constraints :

$$\Rightarrow \begin{cases} a_0 = Q_{00}, \\ a_1 = Q_{10} + Q_{01}, \\ \vdots \\ a_k = \sum_{i,j:k=i+j} Q_{ij}. \end{cases} \tag{4.2}$$

There are $2d+1$ constraint equations formed as $p(x)$ has $2d+1$ coefficients. To solve using the SDP solvers, we rewrite the series of equations in a matrix form. We construct the standard input form $Ax = b$ explicitly, where A is the constraint matrix, x is the unknown variables rescaled to vector form, and b is the corresponding monomial coefficients. In this approach, A is a sparse matrix with only 1s at appropriate places; b is a vector containing coefficients of monomial basis of the polynomial in ascending degree; x consists of $(d+1)^2$ unknown variables: $Q_{00}, Q_{01}, \cdots, Q_{dd}$. At the same time, Q is constrained to be positive semidefinite. The positive semidefinite constraint enforces symmetry in Q, reducing the number of unknown variables to $\binom{d+1}{2}$.

For the multivariate case, some modifications need to be made. Firstly, the monomial basis is ordered by the graded lexicographic ordering. Secondly, the number of constraint equations increases accordingly. There are $\binom{n+2d}{2d}$ coefficients, and therefore, such number of constraint equations are required.

4.3 Primal: the monomial basis; Dual: the Chebyshev basis

As the Chebyshev basis is used on the dual, we use the Chebyshev coefficients to set our constraints. We first transform the monomial coefficients of polynomials into the Chebyshev coefficients by their transformation matrix. The components in the standard formulation are now described as the following. The left hand side vector b, which consists of the Chebyshev coefficients in ascending degree, is obtained by equation (3.24). Simultaneously, constraint matrix A is also obtained by applying equation (3.24) to the constraint matrice in the previous section. The constraint matrix has a sparse structure, but it is denser than using the monomial basis on the dual. Vector x still consists of $(d+1)^2$ unknown variables: $Q_{00}, Q_{01}, \cdots, Q_{dd}$. Similarly, Q is also constrained to be semidefinite. The positive semidefinite constraint reduces the number of unknown variables to $\binom{d+1}{2}$ because of its symmetry.

4.4 Primal: the monomial basis; Dual: the Lagrange basis

When the Lagrange basis is used on the dual, we compare the coefficients of Lagrange basis, which is a direct evaluation of a set of interpolation points. The interpolation formulation enforces constraints at a finite set of points using scalar values. For any polynomial $p(x)$ with degree $2d$, we need at least $(2d+1)$ points to uniquely define it. Therefore, we also need $(2d+1)$ constraint equations, one constraint formed for each sampling point. In this project, we use a set of the Chebyshev nodes of the first kind [16] located in the range of $[a, b]$ as shown in equation (3.1.2).

For a given sampling point x_i, we evaluate $p(x_i)$ and equate with $v(x_i)^T Q v(x_i)$. The constraint equations are formed in the following way:

$$\Rightarrow \begin{bmatrix} p(x_1) \\ p(x_2) \\ \vdots \\ p(x_n) \end{bmatrix} = \begin{bmatrix} v(x_1)^T Q v(x_1) \\ v(x_2)^T Q v(x_2) \\ \vdots \\ v(x_n)^T Q v(x_n) \end{bmatrix}. \tag{4.3}$$

We define the vector $b = [p(x_1)\ p(x_2)\ \ldots\ p(x_n)]$, and denote $v(x_i)v(x_i)^T$ as matrices A_i. We can formulate the SDP problem in its primal form as follows:

$$Q \succeq 0, \quad b_i = Q \bullet A_i, \quad i = 1, \ldots, n. \tag{4.4}$$

It can be observed that the constraint matrices $A_i = v(x_i)v(x_i)^T$ all have rank one property. This property is desirable as this kind of structure can be exploited to improve computation speed in many ways. Currently, the SDPT3 solver [5] provides an option to store constraint matrices with low rank structure. However, we need to specify the low rank structures explicitly when coding the problem data. The low rank form can also improve speed for searching direction in the interior point method. As observed,

$$v(x_i)^T Q v(x_i) = [v(x_i)v(x_i)^T] \bullet Q. \tag{4.5}$$

Since all our constraint matrices have rank one property, we can simply store constraint matrices as a collection of vectors in SDPT3:

$$[v(x_1)\ v(x_2)\ \dots\ v(x_n)],$$

where $v(x_i) = [1\ x_i\ x_i^2\ \dots\ x_i^{d+1}]^T$, and x_i represents $2d+1$ distinct Chebyshev nodes.

4.5 Primal: the Chebyshev basis; Dual: the Chebyshev basis

With the Chebyshev polynomial as the basis on primal, we have:

$$\begin{aligned} &[T_0\ T_1\ T_2\ \dots\ T_d]^T[T_0\ T_1\ T_2\ \dots\ T_d] \\ &= \begin{bmatrix} T_0T_0 & T_0T_1 & \cdots & T_0T_d \\ T_1T_0 & T_1T_1 & \cdots & T_1T_d \\ \vdots & \cdots & \ddots & \vdots \\ T_dT_0 & T_dT_1 & \cdots & T_dT_d \end{bmatrix}. \end{aligned} \tag{4.6}$$

Applying the multiplication property in equation (3.6) to each entry of the above matrix, we can iteratively form $(2d+1)$ constraint equations with respect to the coefficients of T_i. The lowest possible degree of the resulting Chebyshev polynomial is T_0, while the highest possible degree is T_{2d}. As a result, $2d+1$ constraint equations are formed, equal to the coefficients for T_0 to T_{2d}. The components in the standard formulation are modified as the following. The left hand side vector b consists the

Chebyshev coefficients in ascending degree. The constraint matrix A is obtained by an iterative approach. Here, matrix A still possesses the sparse property. However, it is denser than the resulting matrix from using the monomial basis. It is intuitive as each coefficient will be found by a more complicated linear combination of unknown entries in Q. The vector x still consists of $(d+1)^2$ unknown variables: $Q_{00}, Q_{01}, \cdots, Q_{dd}$. Similarly, Q is also constrained to be semidefinite. The positive semidefinite constraint reduces the number of unknown variables to $\binom{d+1}{2}$.

4.6 Primal: the Chebyshev basis; Dual: the monomial basis

From Section 4.5, we have shown how to construct formulations with the Chebyshev basis on both primal and dual. This formulation is similar to that approach except the monomial basis is used on the dual. In this part, we simply compare the monomial coefficients of $p(x)$. The coefficients of $A \bullet Q$ are transformed into the monomial coefficients by the corresponding transformation matrix. In this way, we can form $(2d+1)$ constraint equations from the monomial coefficients.

4.7 Primal: the Chebyshev basis; Dual: the Lagrange basis

We have discussed the low rank property in constraint matrices when the Lagrange basis is used on the dual in Section 4.4. This approach also inherents the rank one property. However, the Chebyshev basis is used on the primal.

Basically, we adopt the same methodology, but now we evaluate the basis using the Chebyshev polynomials. This method improves the numerical conditioning of the resulting constraint matrix, and normally yields more accurate solutions. By exploiting the low rank function of SDPT3, we can again store the vectors instead of constraint matrices in the function as shown below:

$$[v(x_1)\ v(x_2)\ \ldots\ v(x_n)],$$

where $v(x_i) = [T_0(x_i)\ T_1(x_i)\ T_2(x_i)\ \ldots\ T_n(x_i)]^T$.

4.8 Primal: the Lagrange basis; Dual: the monomial basis

With the Lagrange basis on the primal, we have

$$\begin{aligned} p(x) &= [L_0(x)\ L_1(x)\ \ldots\ L_d(x)]Q \begin{bmatrix} L_0(x) \\ L_1(x) \\ \vdots \\ L_d(x) \end{bmatrix} \\ &= \left\{ \begin{bmatrix} L_0(x) \\ L_1(x) \\ \vdots \\ L_d(x) \end{bmatrix} [L_0(x)\ L_1(x)\ \ldots\ L_d(x)] \right\} \bullet Q \\ &= \left\{ F_M^L \begin{bmatrix} 1 \\ x \\ \vdots \\ x^n \end{bmatrix} [1\ x\ \ldots\ x^n](F_M^L)^T \right\} \bullet Q \end{aligned} \tag{4.7}$$

The transformation matrix F_M^L has $d+1$ rows in the SOS formulation. We use F_i to represent the ith row, where F_i is a row vector with dimension $d+1$. It is written as below:

$$F_M^L = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_d \end{bmatrix}. \tag{4.8}$$

For each entry Q_{ij}, it corresponds to

$$F_i \begin{bmatrix} 1 \\ x \\ \vdots \\ x^n \end{bmatrix} [1 \ x \ \dots \ x^n] F_j^T.$$

The formula above shows the Q_{ij} is corresponding to the product of two d degree polynomials. We need to extract the coefficients of each monomial from the product and sum up those corresponding to the same monomial. The constraint matrix A is then filled in by proper assignment of each entry. The resulting constraint matrix is dense.

4.9 Primal: the Lagrange basis; Dual: the Chebyshev basis

We have derived the transformation from the Lagrange coefficients to the Chebyshev coefficients in equation (3.34). Again, we compare the coefficients of the Chebyshev basis to form constraints here. We can simply convert the coefficients of $p(x)$ to the Chebyshev coefficients on one side. On the other side, we can transform the constraint matrix in section 4.8 to reflect the Chebyshev coefficients. Following that, we can formulate problems in the same way as described in section 4.3.

4.10 Primal: the Lagrange basis; Dual: the Lagrange basis

The interpolation method using the Lagrange basis is also similar to the two interpolation methods mentioned above. In SDPT3, we can use the low rank function to store constraint matrices in terms of a collection of vectors:

$$[v(x_1) \ v(x_2) \ \dots \ v(x_n)],$$

where $v(x_i) = [L_0(x_i) \ L_1(x_i) \ L_2(x_i) \ \dots \ L_d(x_i)]^T$. We choose $2d+1$ Chebyshev nodes in the range of $[a, b]$. Each point forms one constraint equation. As the Lagrange basis has the dimension $d + 1$, we select $d + 1$ points out of the $2d + 1$ Chebyshev points as basis points, the rest d points are non-basis points. In this formulation, we pick alternating Chebyshev points as the basis points, and extreme points, a and b, are always picked.

The set of vectors corresponding to basis points can be further simplified as shown in equation (3.12). For better numerical conditioning, we group those basis vectors together. It is simplified to an identity matrix with dimension $d + 1$, i.e,

$$[v(x_0)\ v(x_1)\ \ldots\ v(x_d)] = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}.$$

For non-basis points, the constraint vector is formed by evaluating each point at the Lagrange basis, i.e,

$$v(x_i) = \begin{bmatrix} L_0(x_i) \\ L_1(x_i) \\ \vdots \\ L_d(x_i) \end{bmatrix}.$$

The constraint equations of this formulation also have a low rank property. Therefore, they can be implemented together with the SDPT3 solver to input constraints in terms of vectors only. For about half of its constraint equations, the constraints also have sparse structure. Theoretically, it has better numerical conditioning because of its low rank and partial sparse structure. The collection of vectors is now simplified to be:

$$[I\ v(x_{d+2})\ v(x_{d+3})\ \ldots\ v(x_n)],$$

where $v(x_i) = [T_0(x_i)\ T_1(x_i)\ T_2(x_i)\ \ldots\ T_n(x_i)]^T$ for $i = d+2, d+3, \ldots, n$, and I is an identity matrix of dimension $d + 1$.

4.11 Summary

This section summarizes the relative advantages and disadvantages of the various methods.

Dual \Primal	**Monomial basis**	**Chebyshev basis**	**Lagrange basis**
monomial basis	sparse	sparse	dense
Chebyshev basis	sparse	sparse	dense
Lagrange basis	low rank, dense	low rank, dense	low rank, partially sparse

Table 4.1: Comparison of the structure of constraint matrix of various SDP formulations of SOS.

Chapter 5

Some SOS Examples

5.1 Simulation setup

In this chapter, we select several problems to test various SDP relaxations of SOS. `SeDuMi` [17] solver and `SDPT3` [5] solver are used for the simulations. Since the `SDPT3` solver allows us to input low rank constraints directly, we compare our timing profile and error profile based on the results obtained from the `SDPT3` solver version 4.0. The simulations are performed on MATLAB 7.0 on a Windows PC with Intel Pentium 3 processor 797Mhz.

5.2 The constrained lower bound problem

Example 5.1 *Given*

$$p(x) \geq 0, \textit{ for } x \in [-1, 1],$$

find a lower bound of $p(x)$.

This problem can be formulated as a SOS problem as shown below:

$$\begin{gathered} \max \gamma \\ st.\ p(x) - \gamma = s(x) + t(x)(1 - x^2) \\ s(x) \textit{ and } t(x) \textit{ are SOS} \end{gathered}$$

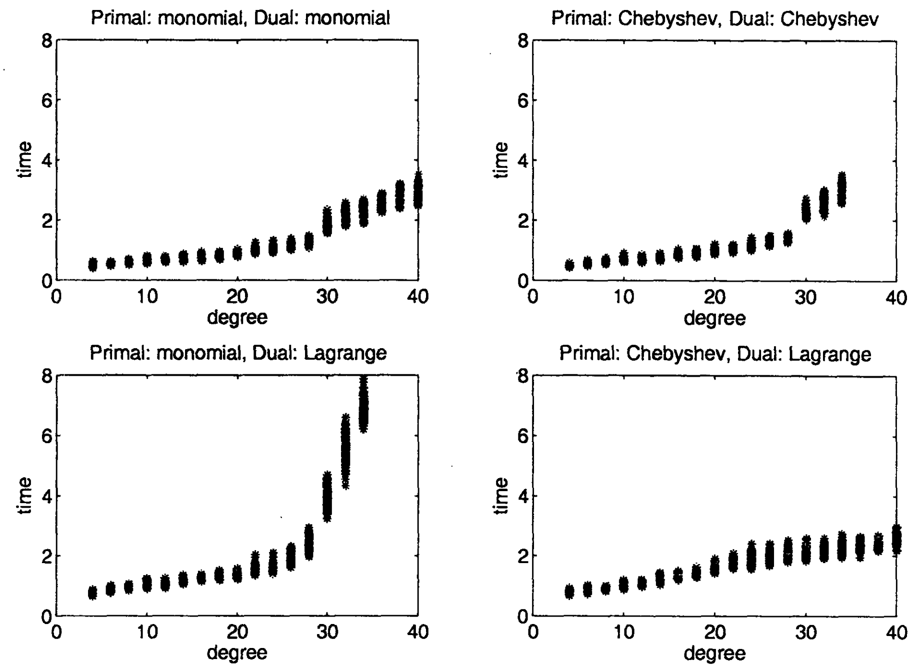

Figure 5-1: Running time against degree for constrained lower bound problem.

We evaluate this problem by using four approaches: the monomial basis on the primal and the monomial basis on the dual; the monomial basis on the primal and the Lagrange basis on the dual; the Chebyshev basis on the primal and the Chebyshev basis on the dual; the Chebyshev basis on the primal and the Lagrange basis on the dual. The timing and error profiles of these approaches are plotted in Figure 5-1 and Figure 5-3.

Figure 5-1 shows that the monomial basis as both primal and dual as well as the Chebyshev basis as primal and the Lagrange basis as dual approaches have shown significant advantages in terms of running time. The faster speed is likely due to the followings:

1. Sparse structure in the case of monomial basis as both primal and dual,

2. Low rank structure in the case of Chebyshev basis as primal and Lagrange basis as dual.

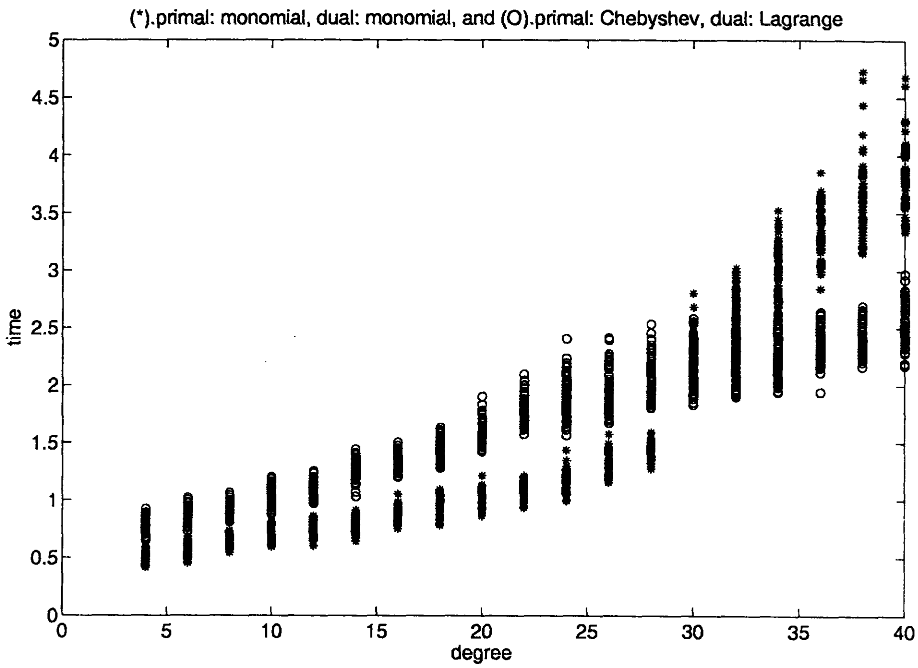

Figure 5-2: Comparison of timing profile against degree for constrained lower bound problem.

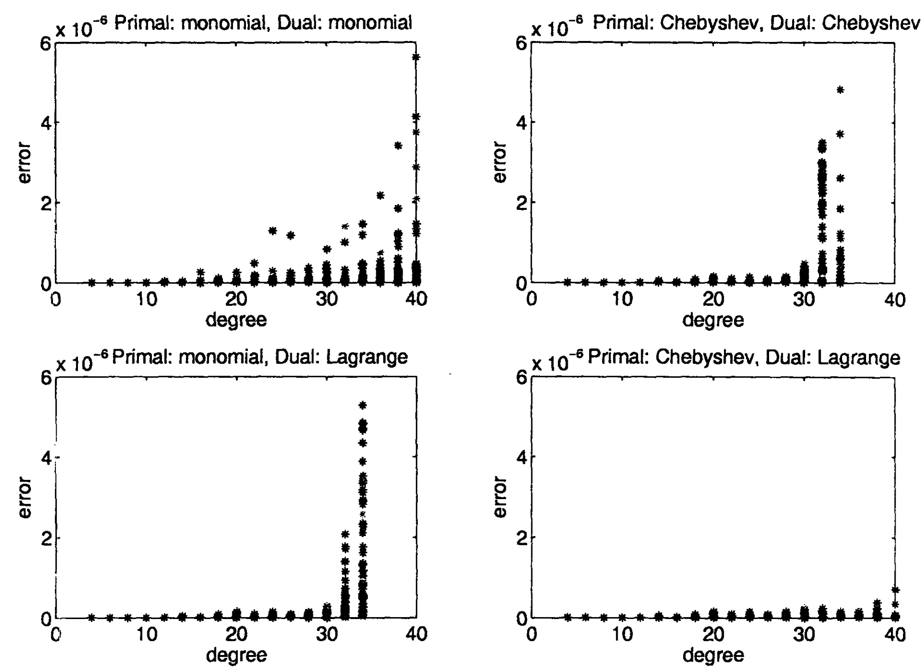

Figure 5-3: Error against degree for constrained lower bound problem.

It is difficult to judge whether the low rank structure or the sparse structure is more desirable based on Figure 5-1. Therefore, we compare timing profiles of the above two methods in detail in Figure 5-2. The figure shows that when the degree of polynomial is small, running time using the two methods is approximately the same. However, as the degree increases, the low rank structure demonstrates noticeable reduction in execution time.

Besides execution time, the method of using the Chebyshev basis as primal and the Lagrange basis as dual is also superior in terms of accuracy of the solutions obtained. Figure 5-3 compares error profiles of the four approaches. Error is calculated by comparing the differences between the computed solution from SDPT3 solver and the optimal solution obtained by evaluating a constrained minimum using *min* function in MATLAB. It is clear from the figure that the Chebyshev-Lagrange method produces much less error, especially for polynomials of higher degrees, as compared to the other three methods. Generally, error increases greatly with the increase of polynomial degrees.

5.3 The Chebyshev bound problem

Example 5.2 *Given a polynomial* $p(x)$*, where*

$$|p(x)| \leq 1, \text{ for } x \in [-1, 1],$$

find the maximum value of the leading coefficient of $p(x)$*, i.e.* p_n.

This problem can be formulated as a SOS problem as shown below:

$$\begin{aligned} &\max\ p_n \\ st.\ 1 + p(x) &= s_0(x)(1+x) + s_1(x)(1-x) \\ 1 - p(x) &= s_2(x)(1+x) + s_3(x)(1-x) \\ &s_0(x), s_1(x), s_2(x) \text{ and } s_3(x) \text{ are } SOS \end{aligned}$$

The above SOS formulation is reformulated by two SDP approaches: the monomial basis on the primal and dual, and the Chebyshev basis on the primal and dual.

An alternative SOS formulation is obtained by using addition and difference of the above two constraints:

$$\begin{aligned} &\max\ p_n \\ where\ p(x) &= 1/2[(s_0(x) - s_2(x))(1+x) + (s_1(x) - s_3(x))(1-x)] \\ st.\ 2 &= (s_0(x) + s_2(x))(1+x) + (s_1(x) + s_3(x))(1-x) \\ &s_0(x), s_1(x), s_2(x) \text{ and } s_3(x) \text{ are } SOS \end{aligned}$$

Two SDP approaches are used for this simplified SOS formulation: the monomial basis as primal and the Lagrange basis as dual, and the Chebyshev basis as primal and the Lagrange basis as dual. These four approaches are simulated and examined.

The exact solution of p_n is related to the degree of the polynomial $p(x)$. It is known to be 2^{n-1}, where n is the highest degree of $p(x)$. Therefore, the exact solution increases exponentially with the increase of polynomial degree. Table 5.1 displays

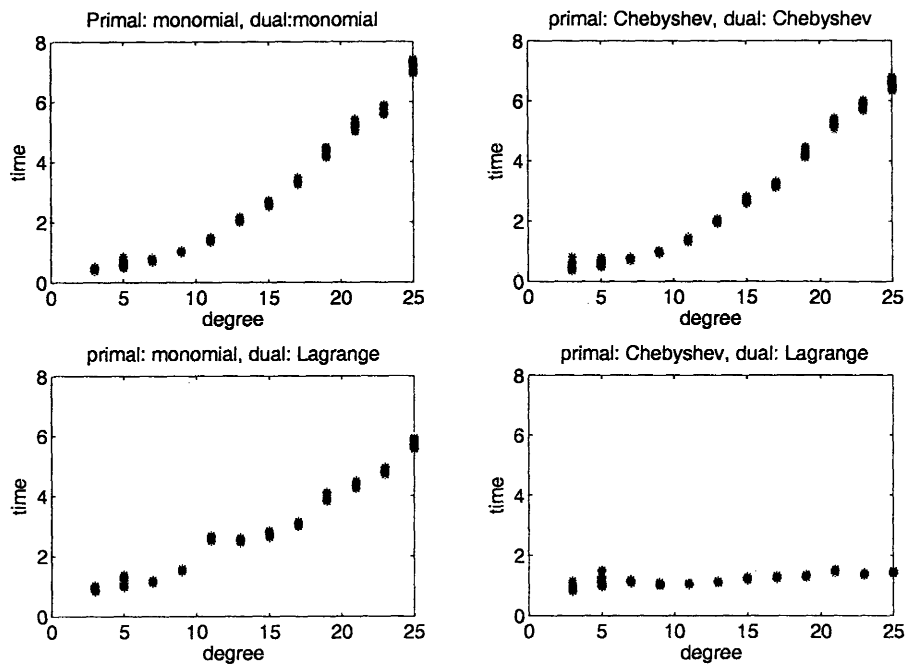

Figure 5-4: Running time against degree for the Chebyshev bound problem.

Method	Maximum degree
SOSTOOLS demo7 [18]	13
primal:monomial basis,dual:monomial basis	19
primal:Chebyshev basis,dual:Chebyshev basis	19
primal:monomial basis,dual:Lagrange basis	17
primal:Chebyshev basis,dual:Lagrange basis	> 1000

Table 5.1: Maximum solvable degree of the Chebyshev bound problem using different approaches.

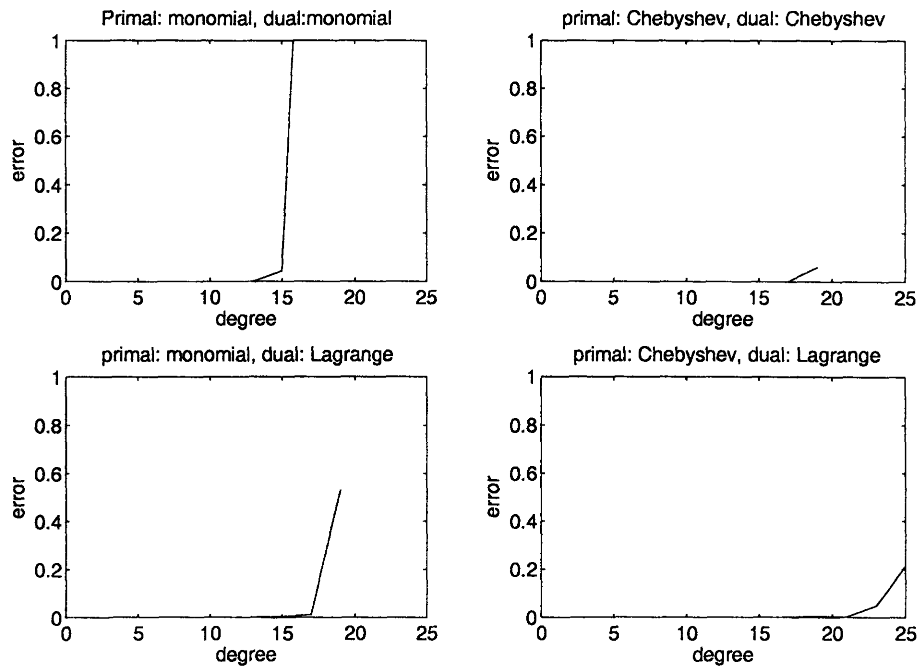

Figure 5-5: Error against degree for the Chebyshev bound problem.

the maximum solvable degree of the polynomials using different approaches. The maximum solvable degree is a key factor to reflect the numerical stability of different approaches. We notice the combination of the Chebyshev basis and the Lagrange basis used for the simplified SOS formulation is able to solve up to amazingly large degree. The good numerical stability is due to the offset of exponential terms in this approach. When the first SOS formulation is used, p_n is set as an unknown variable. Since p_n varies exponentially with n, the solution of resulting SDP problems contains an exponential term. This exponential term is detrimental to the numerical stability. When the simplified SOS formulation is used, p_n is not involved as an variable directly. Instead, we compute p_n from the solutions of $s_0(x), s_1(x), s_2(x)$, and $s_3(x)$. When monomial basis is used as primal, p_n is a linear combination of the corresponding terms in $s_0(x), s_1(x), s_2(x)$, and $s_3(x)$. Exponential terms actually exist in $s_0(x), s_1(x), s_2(x)$, and $s_3(x)$. However, when the Chebyshev basis is used on the primal for the simplified SOS formulation, we obtain the leading coefficient of $p(x)$

in terms of Chebyshev basis first. This value is then scaled to the leading coefficient in terms of monomial basis. The exponential term is contained in the transformation of coefficients but not in the SDP problems. As a result, the combination of the Chebyshev basis and the Lagrange basis used for the simplified SOS formulation has exceptionally good performance as compared to other approaches. This example shows that an appropriate SOS formulation is also an determining factor for numerical stability.

Computation speed of various methods is compared in Figure 5-4. We can observe that the computation time required by using the Chebyshev basis as primal and the Lagrange basis as dual is much shorter (by almost 75 %) than that by other methods. The faster computation speed is likely due to the rank one property of its constraint matrix and its better numerical stability.

We also compare error profiles of each of the four methods. We compute the error by taking the absolute differences between the exact solution and the computed solution. It can be observed in Figure 5-5 that the Chebyshev basis on the primal and the Lagrange basis on the dual method, once again, has the smallest error among the four. However, it appears that absolute error starts to increase rapidly at degree of 25 , even for the best approach. This is because the value of solution p_n increases exponentially, and the MATLAB is only able to provide up to 16 significant figures. This results in truncation errors at large degree. When we evaluate the ratio of error to the exact solution, the percentage of error is actually negligibly small.

5.4 The third degree lower bound problem

Example 5.3 *Given*

$$(1-x^2)^3 \geq 0$$

find a lower bound of $1-x^2$.

This problem can be formulated as a SOS problem as shown below:

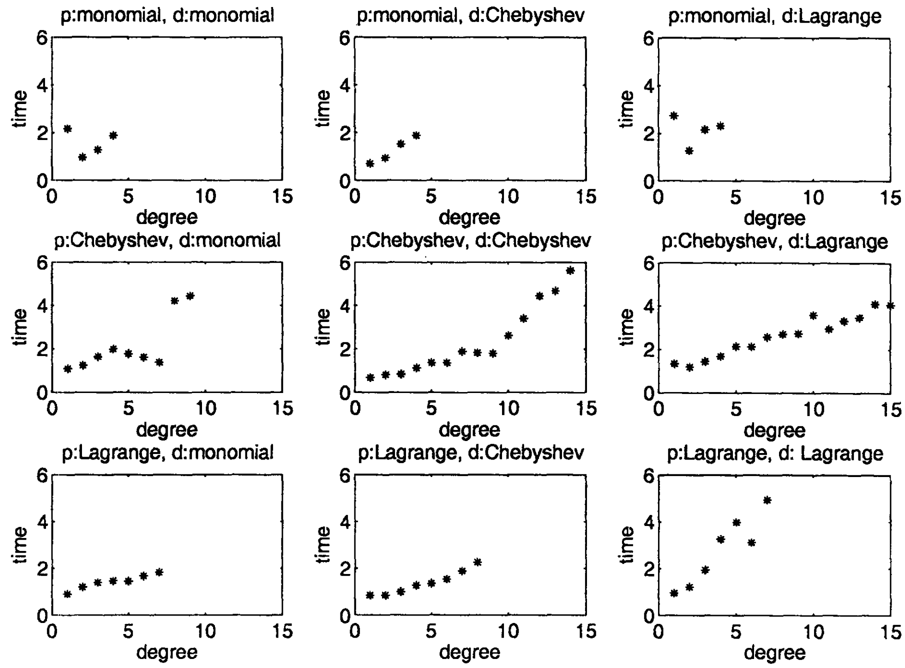

Figure 5-6: Running time against degree for 6th degree polynomial problem.

$$\max \gamma$$
$$st.\ 1 - x^2 - \gamma = S_0(x) + S_1(x)(1 - x^2)^3$$
$$S_0(x) \text{ and } S_1(x) \text{ are SOS}$$

The above problem is formulated by all the nine combinations of basis introduced in the previous Chapter. Their running time, solution and error profiles are plotted in Figure 5-6, Figure 5-8 and Figure 5-9 respectively. Errors are obtained by comparing the computed solutions with the corresponding theoretical values. The exact values of the relaxation are related to the degree of the basis of $S_1(x)$, which is denoted as d, and it is calculated by the formula below:

$$x = \frac{1}{(d+2)^2 - 1}. \tag{5.1}$$

From these graphs, we can observe the performances of each approach. It can be noticed that the choice of the bases plays an integral role in the numerical condition-

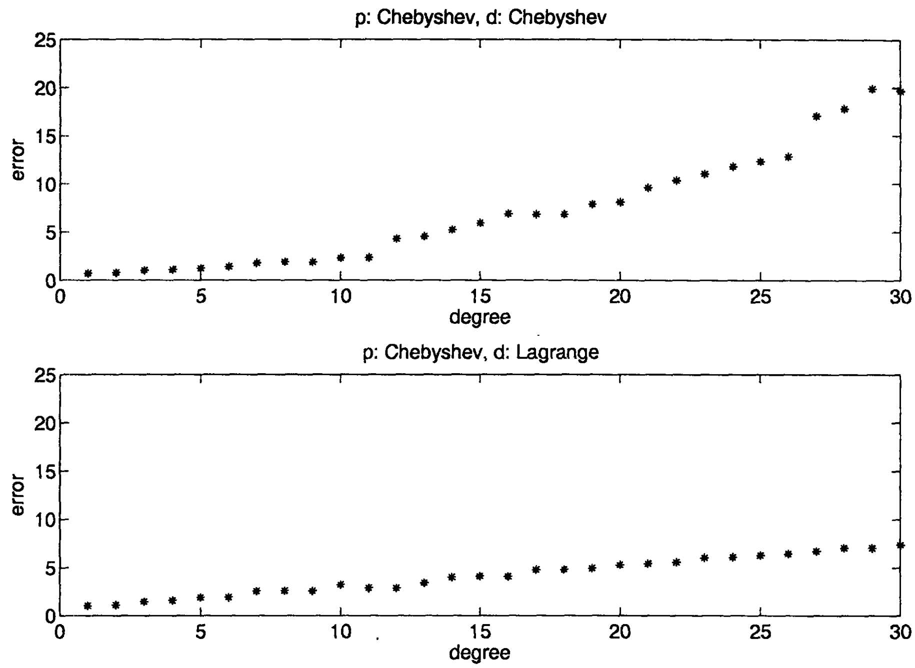

Figure 5-7: Comparison of running time of 1). Chebyshev basis on the primal and dual, and 2). Chebyshev basis on the primal and the Lagrange basis on the dual for 6th degree polynomial problem.

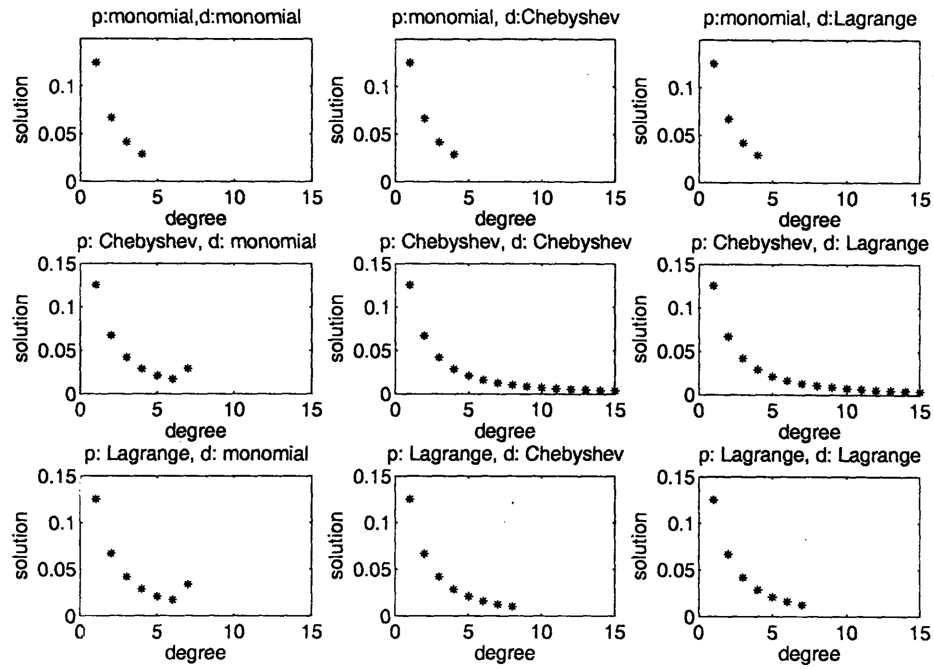

Figure 5-8: Computed solution against degree for 6th degree polynomial problem.

ing. When the monomial basis is used as primal, numerical problems occur at small degree (degree four). With the Lagrange basis as primal, the performances are generally poor. Numerical problems are also encountered at small degree (degree eight) of the polynomial. While the Chebyshev basis is used as primal and the monomial basis is used as dual, numerical problems occur at degree nine, which is slightly later as compared to using other bases as primal. It is encouraging to discover that no numerical problems occur when the Chebyshev basis is chosen as primal with either the Chebyshev basis or the Lagrange basis as dual. The result shows that numerically it is advantageous to use the Chebyshev basis as primal in this example.

In addition to numerical stability, running time of the various methods is compared in Figure 5-6. We can observe that the computation time required by various methods is approximately the same when the degree of polynomials is small. Two numerically better approaches, the Chebyshev basis as primal with the Chebyshev basis as dual and the Chebyshev basis as primal and the Lagrange basis as dual, are examined

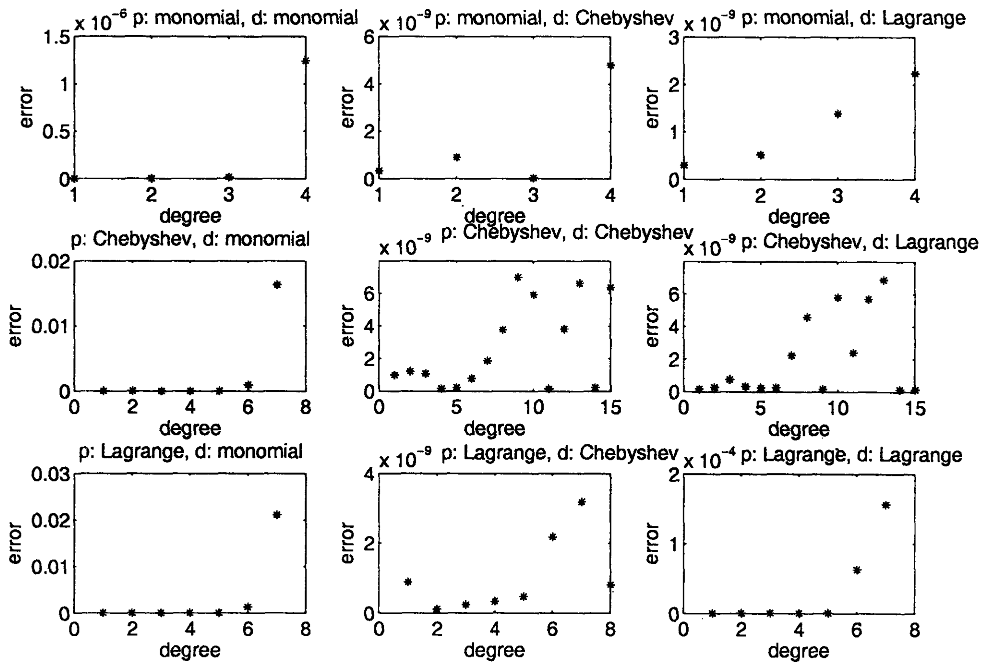

Figure 5-9: Error against degree for 6th degree polynomial problem.

further. Their running time profiles are plotted to higher degree of polynomials in Figure 5-7. We observe from the figure that the running time is shorter for the method of using the Chebyshev basis as primal and the Lagrange basis as dual. In fact, execution time of the two methods differs by more than 60% when the degree of polynomials is 30. The faster computation speed of Chebyshev-Lagrange method is probably due to the rank one property of constraint matrices when the Lagrange basis is used as dual.

In terms of accuracy of the solutions, it is shown in Figure 5-8 that all the methods yield comparable accuracy at small degree of polynomials. As degree of polynomials grows larger, only two of the methods are able to produce accurate solutions. The two better methods coincide with the two numerically better methods. Their solutions and errors are plotted out in Figure 5-10 and Figure 5-11 for more thorough comparison. As can be seen from Figure 5-10, the accuracy of the solutions obtained from the two methods is comparable. In Figure 5-11, it is noticed that errors are negligible for

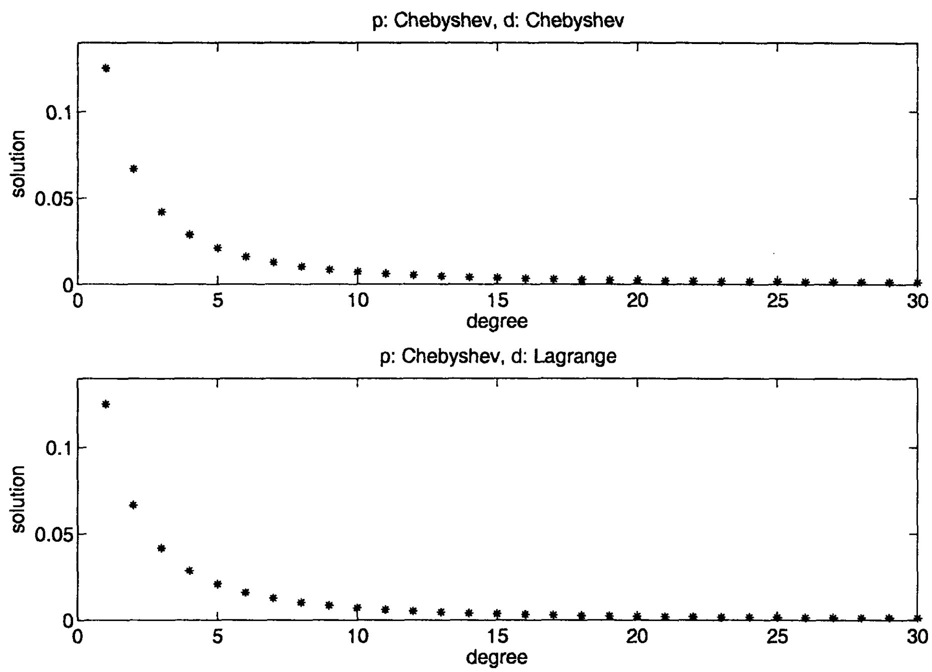

Figure 5-10: Comparison of computed solution of 1). Chebyshev basis on the primal and dual, and 2). Chebyshev basis on the primal and the Lagrange basis on the dual for 6th degree polynomial problem.

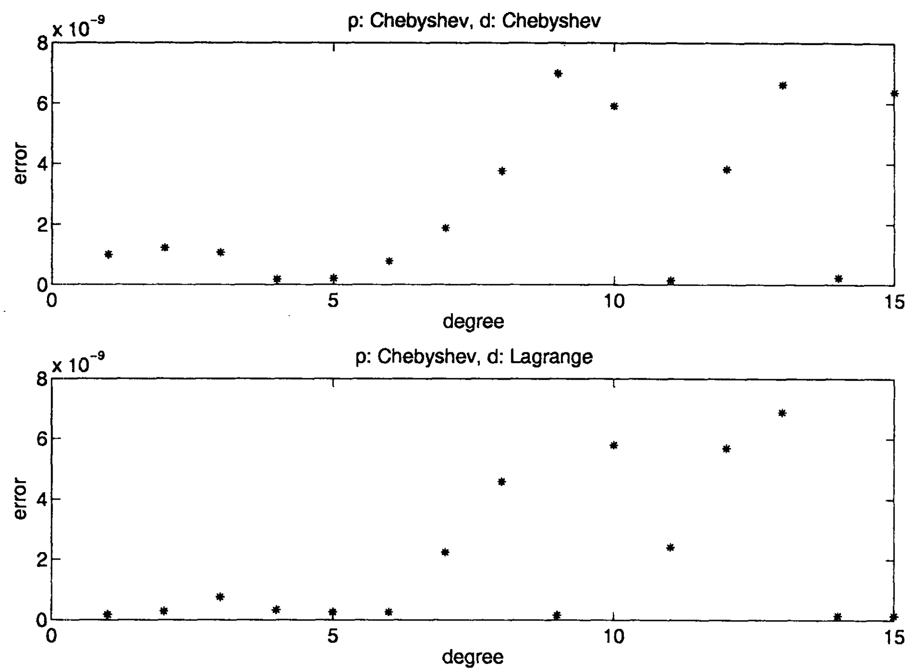

Figure 5-11: Comparison of error of 1). Chebyshev basis on the primal and dual, and 2). Chebyshev basis on the primal and the Lagrange basis on the dual for 6th degree polynomial problem.

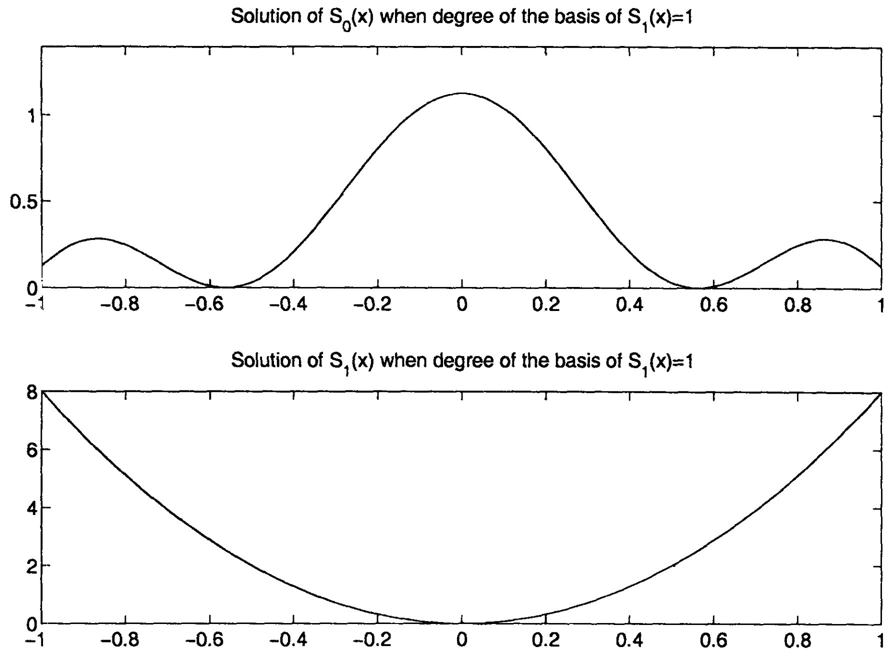

Figure 5-12: Solution of $S_0(x)$ and $S_1(x)$ when degree of basis of $S_1(x) = 1$.

both approaches. We have also plotted some sample solutions of $S_o(x)$ and $S_1(x)$ in Figure 5-12 and Figure 5-13.

5.5 The lower bound of a bivariate polynomial

Example 5.4 *Given*

$$p(x,y) \geq 0, \textit{ for } x,y \in [-1,1],$$

find a lower bound of $p(x,y)$.

This problem can be formulated as a SOS problem as shown below:

$$\begin{gathered} \max \gamma \\ st.\ p(x) - \gamma = s(x,y) + t(x,y)(1-x^2) + v(x,y)(1-y^2) \\ s(x,y),\ t(x,y) \textit{ and } v(x,y) \textit{ are SOS} \end{gathered}$$

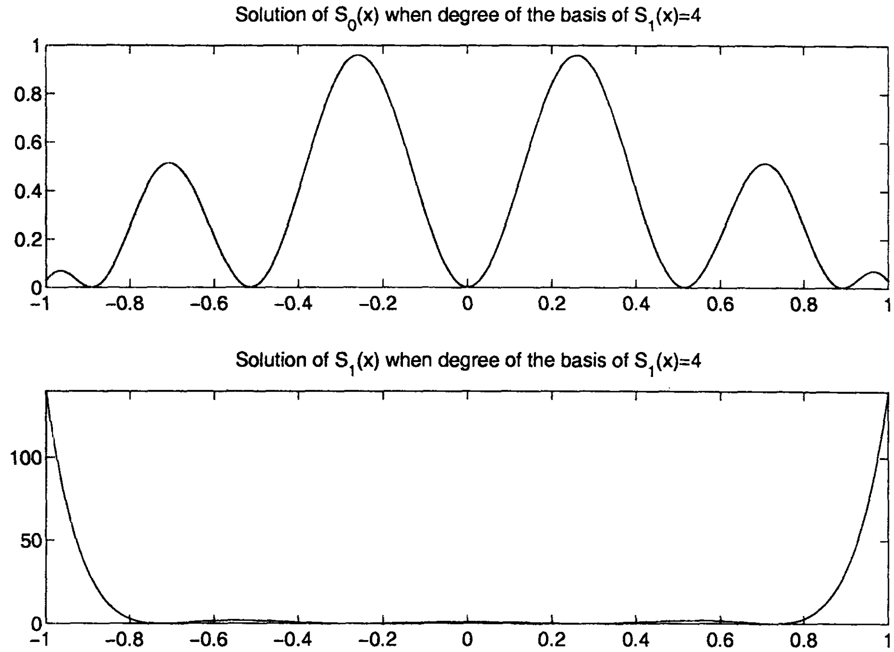

Figure 5-13: Solution of $S_0(x)$ and $S_1(x)$ when degree of basis of $S_1(x) = 4$.

This is an example for multivariate polynomials. From the univariate examples, we have found the approaches that generally perform better. These approaches are selected for simulation in this example. The three approaches chosen are: the Chebyshev basis on the primal and the Lagrange basis on the dual; the monomial basis on the primal and dual; the Chebyshev basis on the primal and dual.

There are many ways to choose the location of the sampling points for multivariate polynomials. When the Lagrange basis is used on the dual, the locations of sampling points are critical. We tried several point schemes, namely, the Chebyshev nodes of first kind, the Chebyshev nodes of second kind, equidistant points, and the Gaussian-Legendre points. These points are used on both axis. The performances of the above named four point schemes are compared in Figure 5-14 and Figure 5-15. We notice that there is no significant differences in terms of running time among the four selected point schemes. However, the Chebyshev nodes appear to result in much greater errors. It can be seen that the equidistant points perform the best among the

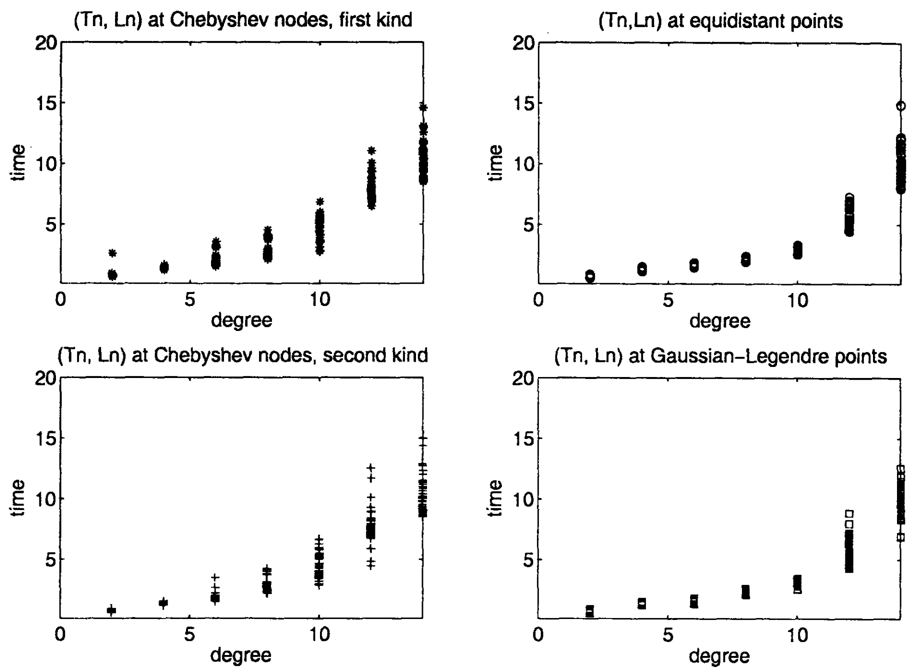

Figure 5-14: Comparison of running time against degree for lower bound for bivariate polynomial problem at different sets of sampling points using primal: Chebyshev basis and dual: Lagrange basis.

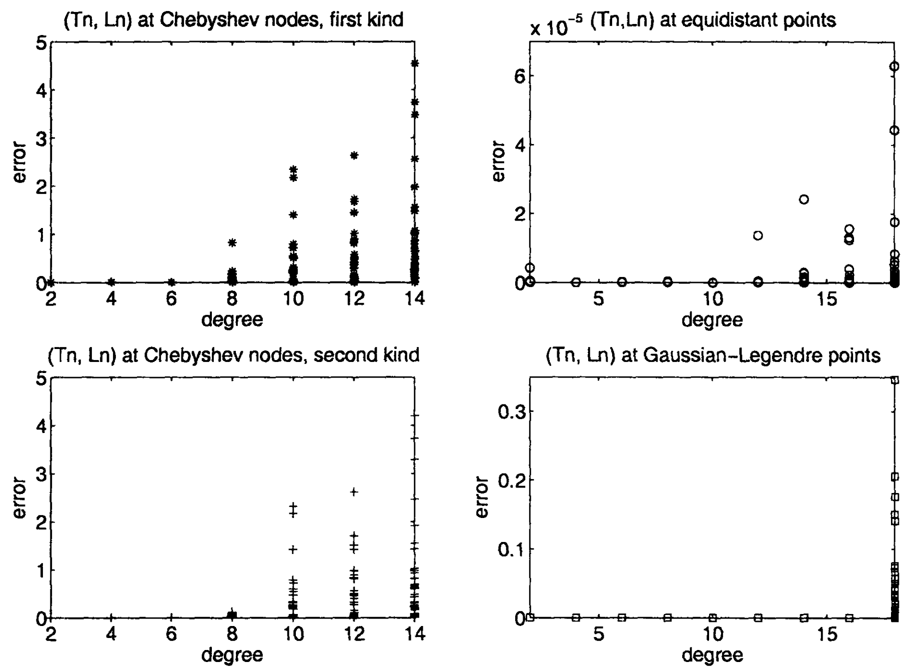

Figure 5-15: Comparison of error against degree for lower bound for bivariate polynomial problem at different sets of sampling points using primal: Chebyshev basis, and dual: Lagrange basis (unnormalized).

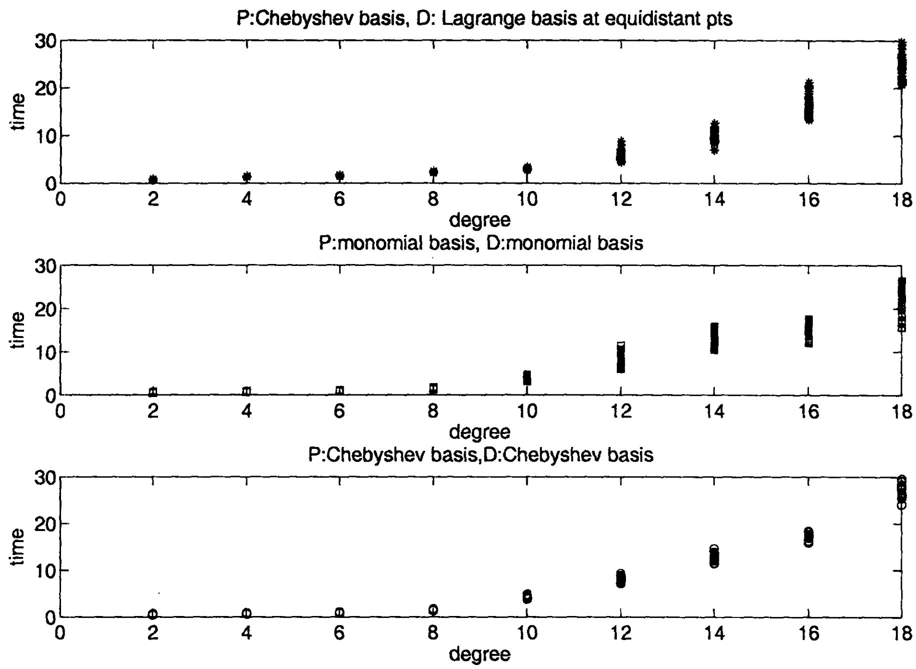

Figure 5-16: Running time against degree for lower bound for bivariate polynomial problem.

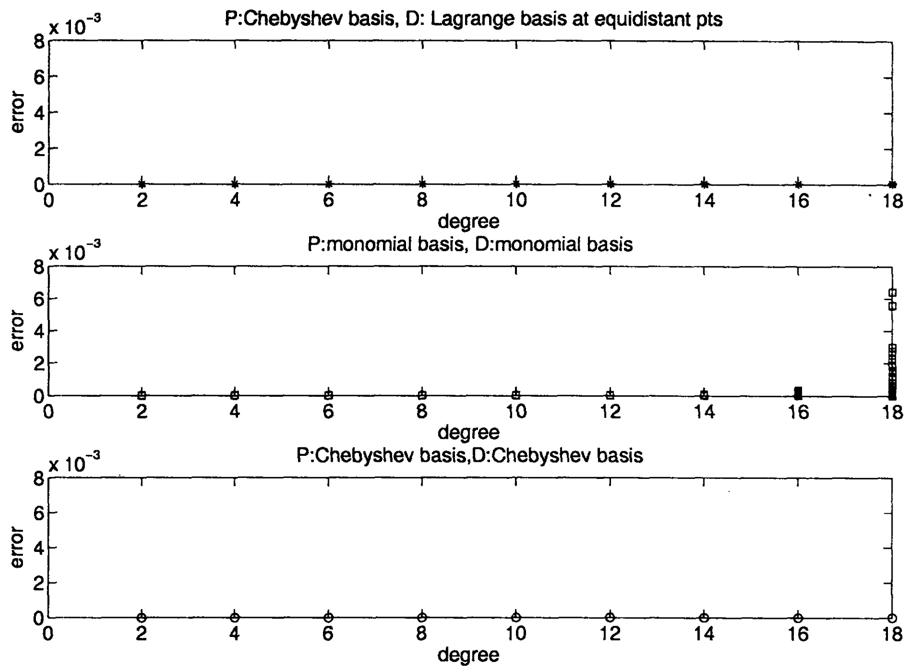

Figure 5-17: Error against degree for lower bound for bivariate polynomial problem (normalized).

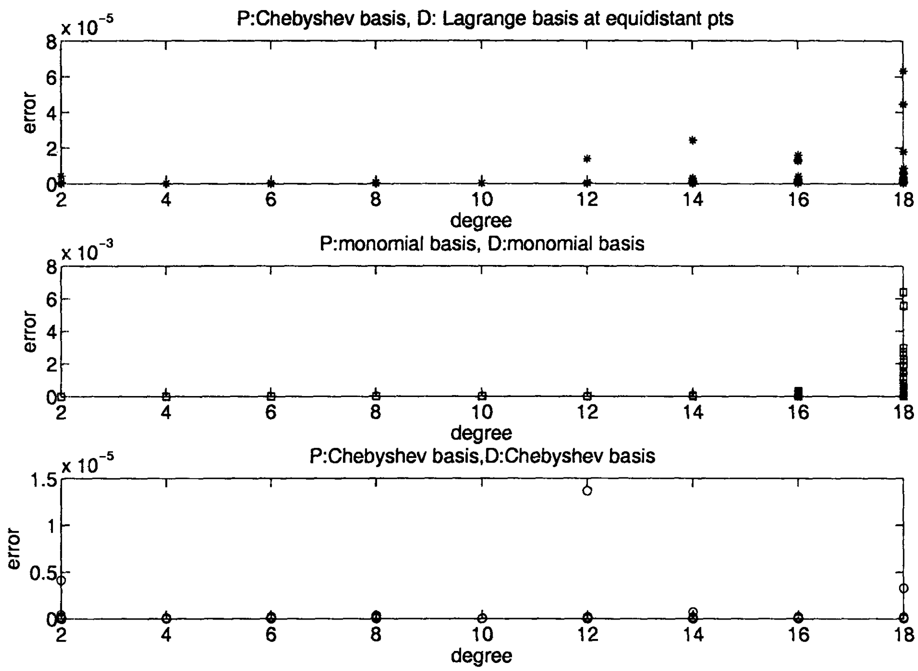

Figure 5-18: Error against degree for lower bound for bivariate polynomial problem (unnormalized).

four point schemes. Therefore, we choose equidistant points in this example.

The running time and errors of the three approaches are plotted in Figure 5-16 and Figure 5-17 respectively. In terms of running time, three methods are approximately the same. The solutions computed from *fmincon* function in MATLAB are treated as the exact solutions for comparison. The errors are the differences of computed solutions and the exact solutions. For the errors, the monomial basis on the primal and dual is much larger than the other two approaches. We compare the unnormalized errors in Figure 5-18. It shows that the Chebyshev basis on both primal and dual has slightly smaller error.

5.6 Summary of results

Based on the examples discussed in this chapter, we are able to conclude the followings:

1. The choice of bases used on the primal and dual leads to differences in terms of numerical conditioning, speed and accuracy.

2. The location of sampling points when the Lagrange basis is used on the dual affects the performance and error significantly.

We observe that the Chebyshev basis on the primal and the Lagrange basis on the dual has particularly good performance. The low rank structure in its constraint matrix is utilized by the SDPT3 solver to improve in speed for calculations in the interior point method. That results in a significant reduction in running time for univariate polynomials. For multivariate polynomials, the Chebyshev basis on both primal and dual approach also demonstrates good performance in terms of numerical conditioning and accuracy. There is no noticeable reduction in time when the Chebyshev basis on the primal and the Lagrange basis on the dual approach is adopted.

Chapter 6

Interface with CVX

6.1 Overview

cvx is a modeling system for disciplined convex programming [19]. Disciplined convex programs [20] [21] imposes a set of rules when we construct convex programs such as linear programs (LPs), quadratic programs (QPs), second-order cone programs (SOCPs), and semidefinite programs (SDPs). cvx specially provides a mode to handle SDP problems. In SDP mode, we can apply inequality operators to matrix representations. In this way, we are able to express linear matrix inequalities conveniently.

cvx is constructed and implemented in Matlab. The functions and operations of cvx can be mixed with common Matlab codes. We can easily do pre-processing and post-processing of our data used in cvx.

The modeling language allows us to define new convex and concave functions, and new convex sets. Thus we can expand the cvx atom library. In this project, I define two new functions to incorporate the sum of squares formulations into cvx.

The two functions work both inside and outside cvx. When the function is invoked inside cvx specification, it will return a function. However, if the function is called outside cvx with some numerical input, it will output a numerical output.

General descriptions are as follows:

- *sos(Q,type)*: the SOS form in terms of entries of Q, where Q is a square matrix constrained by SDP. The 'type' specifies the basis to use for SOS. In this function, two types of bases are supported: the monomial basis and the Chebyshev basis. *sos(Q,type)* returns x^TQx which is a sum of monomials. The output is expressed in a vector form, which only contains the coefficients of monomials.

- *poly_prod(p,q,type)*: the product of two polynomials. We specify two polynomials, p and q, in vector form. The 'type' indicates the type of polynomial. This function also recognizes two types of polynomials: the monomial based polynomial and the Chebyshev polynomial. *poly_prod(p,q,type)* returns the product of two polynomials in vector form.

6.2 Examples

We formulate several examples to illustrate how to implement our functions in cvx to simplify the modeling of SOS problems.

Example 6.1 *Global minimum problem*

$$\begin{gathered} \max \gamma \\ st.\ p(x) - \gamma = s(x) \\ s(x) \text{ is } SOS \end{gathered}$$

We first consider the most basic SOS problem: the global minimum problem. The problem can be expressed in terms of SOS as shown above. Let us create some test problem data for $p(x)$ and *type* in Matlab:

```
type='chebyshev'
len=5;
p=randn(len,1);
d=(len-1)/2+1;
```

We specify the polynomial to be a Chebyshev polynomial with five coefficients (degree four) in the above code. Then we can calculate the dimension of the square matrix Q from the number of coefficients of $p(x)$.

Using cvx, the same problem can be solved as follows:

```
1 cvx_begin sdp
2     variable r
3     variable Q(d,d)
4     maximize r
5     subject to
6     p-[r; zeros(len-1,1)]==sos(Q ,type);
7     Q>=0
8 cvx_end
```

Let us examine this specification line by line:

- Line 1 creates a placeholder for the new cvx specification in SDP mode. At the same time, this line prepares Malab to accept variable declarations, constraints, and objective functions.
- Line 2 declares r to be an optimization variable.
- Line 3 declares Q to be an optimization variable of dimension (d, d).
- Line 4 specifies an objective function to be maximized. Here, we want to maximize the lower bound.
- Line 5 is optional. It is written to make the code more reader friendly. This line does not has any effect.
- Line 6 formulates the equality constraint. Note '==' is used to set the constraint equation. Function $sos(Q, type)$ is implemented here.
- Line 7 restrict Q to be PSD under sdp mode.
- Line 8 marks the end of the cvx specification.

Example 6.2 *Constrained lower bound problem*

$$\max \gamma$$
$$st.\ p(x) - \gamma = s(x) + t(x)(1 - x^2)$$
$$s(x) \text{ and } t(x) \text{ are SOS}$$

Suppose we wish to restrict x to be in the interval $[-1, 1]$, it becomes the constrained lower bound problem as formulated above. Again, we have some test data as follows:

```
type='chebyshev';
len=5;
p=randn(len,1);
d=(len-1)/2+1;
d2=d-1;
```

Using cvx, the formulation is given by

```
cvx_begin sdp
    variable r
    variable Q(d,d)
#   variable Q2(d2,d2)
    maximize r
    subject to
*   p-[r; zeros(len-1,1)]== sos(Q,type)
    + poly_prod( sos(Q2,type), [1; 0;-1], type);
    Q>=0;
&   Q2>=0;
cvx_end
```

Three new lines of cvx code have been added as compared to the global minimum example.

- Line # creates an additional square matrix variable Q_2 with dimension $(d2, d2)$.

- Line * sets the equality constraint. This line uses function $sos(Q, type)$ twice. It also uses the function $poly\_prod(p, q, type)$ to calculate the product of a SOS variable with a numerical polynomial.

- Line & specifies an additional SDP constraint.

Compare the cvx code with the original SOS formulations, it can be seen that cvx code simply specifies SOS condition by an equality constraint with function $sos(Q, type)$ and uses function $poly\_prod(p, q, type)$ to evaluate the product of two polynomials. These functions can be used concurrently with other cvx functions.

Example 6.3 *Chebyshev bound problem*

$$\begin{gathered} \max\ p_n \\ st.\ 1 + p(x) = s_0(x)(1+x) + s_1(x)(1-x) \\ 1 - p(x) = s_2(x)(1+x) + s_3(x)(1-x) \\ s_0(x), s_1(x), s_2(x) \, and\ s_3(x)\ are\ SOS \end{gathered}$$

The Chebyshev bound problem is written in terms of SOS conditions in the above expressions. In cvx, we can easily translate these constraints. The convenience of cvx specifications is demonstrated as follows:

```
cvx_begin sdp
   variable p(len,1)
   variable Q1(d,d)
   variable Q2(d,d)
   variable Q3(d,d)
   variable Q4(d,d)
   maximize p(len)
   subject to
   [1; zeros(len-1,1)]+p== poly_prod( sos(Q1,type), [1;1], type)
   + poly_prod( sos(Q2,type), [1;-1], type);
   [1; zeros(len-1,1)]-p== poly_prod( sos(Q3,type), [1;1], type)
```

```
   + poly_prod( sos(Q4,type), [1;-1], type);
   Q1>=0;
   Q2>=0;
   Q3>=0;
   Q4>=0;
cvx_end
```

Example 6.4 *6th order bound problem*

$$\begin{aligned} &\max \gamma \\ st.\ &1 - x^2 - \gamma = S_0(x) + S_1(x)(1-x^2)^3 \\ &S_0(x) \text{ and } S_1(x) \text{ are SOS} \end{aligned}$$

This example is similar to previous examples. The **cvx** code further illustrates how to translate SOS conditions to **cvx** specifications.

```
cvx_begin sdp
   variable r
   variable Q(d,d) symmetric
   variable Q2(d2,d2) symmetric
   maximize r
   subject to
   [1; 0; -1; zeros(len-3,1)]-[r; zeros(len-1,1)]==
   sos(Q,type)+ poly_prod( sos(Q2,type), [1; 0;-3; 0;3;0;-1], type);
   Q>=0
   Q2>=0;
cvx_end
```

6.3 Summary

In cvx, we can conveniently expand its atom library by adding more functions. We have successfully incorporated SOS formulations of univariate polynomials in cvx.

Several examples are formulated to illustrate how to utilize these functions. In future, we can expand our scope further to include multivariate polynomials.

Chapter 7

Conclusion

In this project, we have discussed various SDP formulations of SOS optimization. We are particularly interested to investigate the theoretical and numerical advantages of the low rank structure in the interpolation representation of SOS decompositions, which uses the Lagrange basis on the dual. Modeling and simulations have been carried out on MATLAB 7.0. In addition, we have also incorporated SOS functions into cvx, a convex programming modeling toolbox.

The low rank structure is integrated with the SDPT3 solver to utilize its low rank function. Based on experimental results, we can observe the running time of this formulation decreases significantly as compared to the conventional coefficient method of SOS. Low rank property improves in speed for calculations of barrier gradient and Hessian assembling. Particularly, for univariate polynomials, the interpolation method with the Chebyshev basis on the primal demonstrates advantages in terms of numerical stability, accuracy and speed. For multivariate polynomials, this approach also performs well in terms of numerical stability and accuracy. However, there is no observable reduction in time.

Two factors have a big effect in the performance of the interpolation method:

1. The location of sampling points.

2. The choice of basis on the primal.

Appropriate choice of basis and sampling points can improve numerical conditioning greatly.

More investigations can be carried on to explore the numerical properties of multivariate polynomials for improvement of execution speed. Future development may also include interfacing the low rank interpolation method with SOSTOOLS or YALMIP.

Bibliography

[1] Y. Genin, Y. Hachez, Y. Nesterov, and P. V. Dooren. Convex optimization over positive polynomials and filter design. In *Proceedings of the IEE Control Conference*, 2000.

[2] Y. Genin, Y. Hachez, Y. Nesterov, and P. V. Dooren. Optimization problems over positive pseudopolynomial matrices. In *SIAM Journal on Matrix Analysis and Applications*, pages 25(1):57–79, 2003.

[3] T. Roh and L. Vandenberghe. Discrete transforms, semidefinite programming and sum-of-squares representations of nonnegative polynomials. In *SIAM Journal on Optimization*, 2006.

[4] J. Löfberg and P. A. Parrilo. From coefficients to samples: a new approach to SOS optimization. In *IEEE Conference on Decision and Control*, December 2004.

[5] K. C. Toh, R. H. Tütüncü, and M. J. Todd. *On the implementation and usage of* SDPT3, *a* MATLAB *software package for semidefinite-quadratic-linear programming, version 4.0*, 2005. Available at `http://www.math.nus.edu.sg/ mattohkc/sdpt3.html`.

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

[7] B. Reznick. Some concrete aspects of Hilbert's 17th problem. In *Contemporary Mathematics*, volume 253, pages 251–272, 2000.

[8] M. J. Todd. Semidefinite optimization. Technical report, Ithaca, NY 14853, 2001.

[9] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[10] M. G. Krein and A. A. Nudelman. The Markov moment problem and extremal problems. *Translations of Mathematical Monographs, American Mathematical Society*, 50, 1977.

[11] G. Szegö. Orthogonal polynomials. *American Mathematical Society*, 1939.

[12] Wikipedia, the free encyclopedia. Available at `http://en.wikipedia.org/wiki/Monomial_basis`.

[13] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, chapter 22. New York: Dover, 1972.

[14] P. A. Fuhrmann. *A Polynomial Approach to Linear Algebra.* Springer, 1996.

[15] L. Robbiano. On the theory of graded structures. *J. Symbolic Comput.*, 2(2):139–170, 1986.

[16] R. P. Feinerman and D. J. Newman. *Polynomial Approximation.* The Williams and Wilkins Company, 1974.

[17] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11:625–653, 1999.

[18] 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`.

[19] M. Grant, S. Boyd, and Y. Ye. *CVX user's guide*, 2006. Available at `http://www.stanford.edu/ boyd/cvx/`.

[20] M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. *Global Optimization: From Theory to Implementation, Kluwer (Nonconvex Optimization and its Applications series)*, 2004.

[21] M. Grant. *Disciplined convex programming.* PhD thesis, Stanford University, 2004.