University of California, San Diego

Department of Mathematics

Numerics for Different Bases in Certifying Nonnegativity of Polynomials

Author: Yixuan Zhou

Advisors: Mareike Dressler

May 22, 2022

Abstract

Polynomial optimization is widely applicatible in solving both practical and theoretical computation problem. One particularly successful way to solve the polynomial optimization certifying the nonnegativity of a polynomial using the sum of squares (SOS) certificate. In this thesis, we study how different choices of the polynomial basis can affect the numerical stability of the linear system generated when using Semidefinite programming (SDP) to test SOS. We run numerical experiments comparing the performance of monomial basis and different types of orthogonal basis, and we discuss the different trade-offs between the choice two types of polynomial basis.

Acknowledgement

I would like to express my deepest gratefulness to my program advisor Professor Mareike Dressler for offering this opportunity to do this honor project. During the past three quarters, she not only has been providing insightful ideas about the direction of this project should be going, but also has been guiding me through the process of doing research. As an expert in this field, she patiently explained in details of the concepts behind the problem. As an advisor, she showed me how to find sources and how to write a thesis. As a mentor, she shared insightful thoughts with me about academia, industry, and life. Especially with these special period of time, when we are all blocked by the pandemic, she still offered incredible patience and shining positive attitudes whenever we talk. This project could not have done without her support and help. Thank you.

Contents

1 Introduction

Polynomial optimization problems are ubiquitous in applied mathematics and engineering. For instance, Ahmadi and Majumdar [AM16] suggested to use polynomial optimization in the field of *artificial intelligence* for solving *real-time decision problems.* In electric power system design, the *optimal power flow problem* can be reduced to a polynomial optimization problem as well, see [Jos16].

In general, an *optimization problem* [BBV04, Chapter 1] takes the form

$$\begin{aligned} \min_{\mathbf{x}} \; f(\mathbf{x}) \quad \textit{subject to } g_i(\mathbf{x}) &\geq 0 \\ h_j(\mathbf{x}) &= 0, \end{aligned} \tag{1.1}$$

where f, g_i, h_j are arbitrary functions of the variables $\mathbf{x} = (x_1, ..., x_n)$. The function f is called the *objective function*, and the functions g_i, h_j are called the *constraints.* Hence, when solving an optimization problem, one tries to minimize the objective function subject to certain constraints. There are several ways to address this problem, which are collected as *nonlinear optimization methods.* Some famous optimization approaches are the *Gradient decent method*, *Newton's method* and *Interior-point method* [BBV04, Chapter 9-11]. Those optimization algorithms seek to exploit the properties of the objective function to find a local minimizer $\mathbf{x}^*$ and hope (with justifications in special cases) that the function value at the minimizer is close to the global minimum. Whereas polynomial optimization takes a different approach.

Polynomial optimization, as the name suggests, is the specific case when the functions f, g_i, h_j in (1.1) are polynomials. In this thesis, we focus on unconstrained polynomial optimization, that is, the feasible domain is $\mathbb{R}^n$. (In general, constrained optimization problem can be reduced to their unconstrained counterpart.) Then, instead of minimizing a polynomial, one can search for its maximal lower bound, see [Dre18, Chapter 1]. Hence, we can equivalently formulate the unconstrained problem as

$$\max \; r \quad \textit{s.t. } f(\mathbf{x}) - r \geq 0 \quad \textit{for all } \mathbf{x} \in \mathbb{R}^n,$$

where the objective function f is a polynomial, and the lower bound r is a real number [BPT12, Chapter 3].

To solve this problem, we need to decide whether a given polynomial is *nonnegative* (see Definition 2.1.5). It turns out that this problem has been well studied in real algebraic geometry since the beginning of the 19th century. However, it turns out that deciding whether an arbitrary multivariate polynomial is nonnegative is, in the language of computational complexity, *co-NP-hard* (computationally infeasible), see [BPT12, Chapter 3].

Therefore, one wants to find sufficient conditions that certify nonnegativity of a polynomial which are easier to check, rather than directly deciding the nonnegativity of a polynomial. Such conditions are called *certificates of nonnegativity.* The most famous and canonical nonnegativity certificate is *sums of squares (SOS)* (see Definition 2.1.6). Deciding if a polynomial is a sum of squares can be formulated as a *semidefinite program (SDP)* (see Definition 2.2.6). Under mild conditions, SDPs have known algorithms that can solve the problem in polynomial time (efficiently) with respect to the input size, see [BPT12, Chapter 2].

When solving the SDP, the solver is actually dealing with a set of linear equations, which can be compactly written in the form $Ax = b$, obtained by the process of *comparison of coefficients*

(COC). Though this process is formally introduced in Section 2.3, the intuition behind it is quite simple. It can be understood in terms of vectors after realizing that the set of n-variate real polynomials with degree less than or equals to d, denoted by $\mathbb{R}[\mathbf{x}]_{n,d}$ forms a vector space. Then any polynomial $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]_{n,d}$ can be uniquely identified as $p(\mathbf{x}) = \sum_{j=0}^{k} c_j b_j$ given a basis $\mathcal{B} = \{b_0, ..., b_k\}$ of $\mathbb{R}[\mathbf{x}]_{n,d}$. Hence, to determine whether 2 polynomials are equal, one can express the 2 polynomials in the same basis, and comparing whether the coefficients are the same. This is the intuition behind COC. The generated equalities will then be used as the linear equations in SDP. For more about vector space and basis, one could refer to the book Linear Algebra and Its Applications [Lay16].

The linear system that is formed by COC depends on the two bases: The basis $\mathcal{B}$ we use to compare the coefficients and the basis $\mathcal{B}'$ that the original polynomial is written in. We measure the stability of the linear system, as suggested by [Rec14], via the *(generalized) condition number* (see Definition 2.2.10), which heavily depends on the choice of both $\mathcal{B}$ and $\mathcal{B}'$. The stability of the system is very important in practice because this measures how robust the method is under noisy data, see [Rec14]. Yet, it is unclear what bases combinations generate the best result.

In this thesis, we generalize the work of [Rec14] to the multivariate setting and perform computer aided numerical experiments of different combinations of the bases, to determine what choice of basis to carry out COC so that the resulted linear system is most stable. With our result, practitioners can find the best basis choice to plug into an SDP solver to certify a polynomial given by the basis determined by the corresponding real life problem.

1.1 Thesis Outline

The thesis is organized as follows: In Section 2, we introduce the preliminary material, including the tools that we need throughout this thesis, and the algorithm that will be employed to carry out the COC. A brief survey of polynomial bases that are considered in this thesis is also included in this section. Section 3 presents the main numerical results of the condition number. Finally, in Section 4, we discuss the result that is obtained in Section 3 and some thoughts on what are the further efforts that can be made to this problem.

2 Preliminaries

In this section, we introduce the notation and the necessary background material to understand the problem. Moreover, we describe the algorithm that is used to carry out the *comparison of coefficients (COC)* process and provide a brief survey of the polynomial bases that are considered in this thesis.

2.1 Real Polynomials

To rigorously define the considered decision problem, we first recall the terms related to *polynomials.*

Throughout, we use bold letters for vectors, e.g. $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$. The set of all m by n real matrices is denoted as $\mathbb{R}^{m\times n}$. Further, to ease the notations, we define the following function:

$$\mathcal{N} \colon \mathbb{N} \to \mathbb{N}$$
$$\mathcal{N}(d) \mapsto \binom{n+d}{d},$$

where $\binom{n+d}{d}$ stands for the binomial coefficient $n + d$ choose d.

The primary interest of this thesis is on polynomials, so we first spend a paragraph define all the terms of polynomials that will be used.

Let $\mathbb{R}[\mathbf{x}] = \mathbb{R}[x_1, ..., x_n]$ be the ring of n-variate polynomials. We only consider polynomials $p \in \mathbb{R}[\mathbf{x}]$ that are supported on a finite set $A \in \mathbb{N}^n$. Hence, we can write p as the form $p(\mathbf{x}) = \sum_{\mathbf{a}\in A} c_{\mathbf{a}}\mathbf{x}^{\mathbf{a}}$. We call a polynomial of the form $\mathbf{x}^{\mathbf{a}}$ a *monomial*, and we define the *degree* of the monomials to be $|\mathbf{a}| = \sum_{i=1}^{n} a_i$. Then the *degree* of the polynomial p is given by the maximum degree over all monomials of p that has a non-zero coefficient. That is, *the degree of* $p = \max\{|\mathbf{a}| : c_{\mathbf{a}} \neq 0\}$ [Dre18].

We use $\mathbb{R}[\mathbf{x}]_{n,d}$ for the set of all n-variate real polynomials with degree less than or equal to d.

Example 2.1.1. *Consider $p(x, y, z) = x^4 + 2xyz - 6y + 7$ with x, y, z being variables. This polynomial has 3 variables and has degree 4, thus it belongs to $\mathbb{R}[\mathbf{x}]_{3,4}$.*

Immediate observations about $\mathbb{R}[\mathbf{x}]_{n,d}$ are:

Proposition 2.1.2. *$\mathbb{R}[\mathbf{x}]_{n,d}$ forms a finite real vector space of dimension $\mathcal{N}(d)$.*

Proof. Suppose $p(\mathbf{x}), q(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]_{n,d}$, $c \in \mathbb{R}$, then we have, $cp \in \mathbb{R}[\mathbf{x}]_{n,d}$ because multiplication by a scalar neither increases the degree nor introduces new variables. We have $p + q \in \mathbb{R}[\mathbf{x}]_{n,d}$ for the same reason. This proves the first part of the claim.

For the dimension of $\mathbb{R}[\mathbf{x}]_{n,d}$, we observe that any n-variate degree d polynomial can be written in the form

$$c_0 + c_1x_1 + c_2x_2 + ... + c_{n+1}x_1^2 + c_{n+2}x_1x_2 + ... + c_l x_n^d.$$

So the set $B = \{1, x_1, x_2, ..., x_1^2, x_1x_2, ...x_n^d\}$ spans $\mathbb{R}[\mathbf{x}]_{n,d}$. Moreover, it is a linear independent set. Thus, it forms a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$. By counting the involved monomials, the cardinality of B is $|B| = \binom{n+d}{d} = \mathcal{N}(d)$, which yields the dimension of $\mathbb{R}[\mathbf{x}]_{n,d}$.

□

Let $\mathcal{B}_{n,d}$ be a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$. In the proof above, we used a canonical basis to $\mathbb{R}[\mathbf{x}]_{n,d}$, namely the monomial basis,

$$\mathcal{B}_{n,d} = \{1, x_1, x_2, ..., x_n, x_1^2, x_1x_2, ..., x_n^d\}.$$

Remark 2.1.3. *If we list the elements of $\mathcal{B}_{n,d}$ in a column vector $\mathbf{b}$, then $\mathbf{b}\mathbf{b}^T$ forms a matrix whose upper triangular entries can be collected to form a basis of $\mathbb{R}[\mathbf{x}]_{n,2d}$.*

Example 2.1.4. *Let $\mathcal{B}_{2,1} = \{1, x, y\}$ be a basis of $\mathbb{R}[\mathbf{x}]_{2,1}$, then*

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \begin{bmatrix} 1 & x & y \end{bmatrix} = \begin{bmatrix} 1 & x & y \\ x & x^2 & xy \\ y & xy & y^2 \end{bmatrix}$$

The entries of the upper triangle of the matrix, $\{1, x, y, x^2, xy, y^2\}$ form a basis of $\mathbb{R}[\mathbf{x}]_{2,2}$.

Next, we define the terminologies related to the decision problem.

Definition 2.1.5. *Let $P_{n,2d}$ denote the set of* nonnegative polynomials *with n variables and degree at most $2d$, that is*

$$P_{n,2d} = \{p \in \mathbb{R}[\mathbf{x}]_{n,2d} : p(\mathbf{x}) \geq 0, \text{ for all } \mathbf{x} \in \mathbb{R}^n\}.$$

The reason that we chose to consider the degree $2d$ in the definition is that nonnegative polynomials always have even degrees. A polynomial with odd degree would be negative when we fix all the other variables and move one variable to positive or negative infinity.

Definition 2.1.6. *Let $\Sigma_{n,2d}$ denote the set of polynomials with n variables and degree at most $2d$ that are* sum of squares (SOS)*, that is*

$$\Sigma_{n,2d} = \{p \in \mathbb{R}[\mathbf{x}]_{n,2d} : \text{ there exists } q_1(\mathbf{x}), ..., q_k(\mathbf{x}) \in \mathbb{R}[x]_{n,d} \text{ s.t. } p(\mathbf{x}) = \sum_{i=1}^{k} q_i^2(\mathbf{x})\}.$$

Notice that $\Sigma_{n,2d} \subseteq P_{n,2d}$ because sum of squares of real numbers are always nonnegative. In fact, geometrically, these two sets are *convex cones*, see [BPT12, Chapter 3].

One natural question to ask is whether these two cones are the same. In 1888, David Hilbert showed that they are usually not the same.

Theorem 2.1.7 ([Hil88]). *There are only three special cases where the two cones coincide, i.e. $\Sigma_{n,2d} = P_{n,2d}$: univariate polynomials, quadratic polynomials, and bivariate polynomials of degree four. Namely, when $n = 1$, or $d = 2$, or $n = 2$ and $d = 4$.*

However, Hilbert's proof is highly non-constructive. Therefore, it took almost 70 years until Motzkin, accidentally, found the first nonnegative polynomial that is not a SOS. In [Mot67], he presented the *Motzkin' polynomial*,

$$p(x, y) = x^4y^2 + x^2y^4 - 3x^2y^2 + 1.$$

Proof. In this proof, we show Motzkin's polynomial is nonnegative, and it can't be written in a sum of squares.

The nonnegativity is immediate from the AM-GM inequality. Notice that,

$$x^2y^2 = \sqrt[3]{(x^4y^2)(x^2y^4)\cdot 1} \leq \frac{x^4y^2 + x^2y^4 + 1}{3}.$$

Multiply both sides by 3 and move the left term to the right, which shows Motzkin's polynomial is nonnegative.

To show it cannot be written in a sum of squares, we need to make the following observation. Suppose $p(x,y) = x^4y^2 + x^2y^4 - 3x^2y^2 + 1 = \sum_{i=1}^m h_i(x,y)^2$ can be written as a sum of squares, then $h_i(x,y) = a_ix^2y + b_ixy^2 + c_ixy + d$ with $a_i, b_i, c_i \in \mathbb{R}$.

To make this observation, we notice that for any two terms $a = x^{i_1}y^{j_1}$ and $b = x^{i_2}y^{j_2}$, $ab = x^{i_1+i_2}y^{j_1+j_2}$. Hence, we know h_i would be expressive enough if $p(x,y)$ can be written in a sum of squares, as we can see every term x^iy^j in $p(x,y)$, there exists a term $x^{i/2}y^{j/2}$ in $h_i(x,y)$. Note, this argument can also be made using *Newton polytope*, see [BPT12, Chapter 4].

Now, we can see that the coefficients of the term x^2y^2 in $\sum_{i=1}^m h_i(x,y)^2$ equals to $\sum_{i=1}^m c_i^2 \geq 0$, yet in Motzkin's polynomial the coefficient of the term is -3, a contradiction. Hence, we conclude that Motzkin's polynomial is not a sum of squares. $\square$

Since the discovery of Motzkin's polynomials, there are many efforts devoted to find more nonnegative polynomials that cannot be written in the form of sum of squares, including *Choi-Lam polynomial* and *Reznick polynomial* [CLR95]. Choi and Lam offered two examples that we list in the following equations, and Reznick in [Rez00] suggested a constructive way of generating nonnegative polynomials that can't be written in the form of sum of squares following what Hilbert's original proof.

$$\textit{Choi-Lam polynomial:} \quad \begin{aligned} p(x,y,z) &= x^2y^2 + y^2z^2 + z^2x^2 + 1 - 4xyz \\ p(x,y,z,w) &= x^2y^2 + y^2z^2 + z^2x^2 + w^4 - 4wxyz. \end{aligned}$$

Coming back to our original problem, as discussed in the introduction, the decision problem of whether an arbitrary polynomial $p(\mathbf{x}) \in P_{n,2d}$ is computationally hard. Whereas we can efficiently decide whether $p(\mathbf{x}) \in \Sigma_{n,2d}$ using *semidefinite program*, which we introduce in next subsection. Hence, we make the following trade off. We use the certificates to check the nonnegativity in a computationally feasible way, accepting fact that we fail to identify some nonnegative polynomials.

Remark 2.1.8. *We note that SOS certificate, with minor modification, can also be used to for constrained optimization in which we only require the polynomial p to be nonnegative on an algebraic set $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n | g_i(\mathbf{x}) \geq 0, i = 1, \ldots, n\}$, where g_i are also polynomials. For example, if we can decompose p as $p(\mathbf{x}) = s_0(\mathbf{x}) + \sum_{i=1}^n s_i(\mathbf{x})g_i(\mathbf{x})$, with $s_0, s_1, \ldots, s_n$ being SOS polynomials, then we can certify p is nonnegative over $\mathcal{S}$.*

Lastly, we conclude that the *decision problem* that is considered in this thesis is the following:

$$\textit{Given } p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]_{n,2d}, \textit{ decide whether } p(\mathbf{x}) \in \Sigma_{n,2d}. \tag{2.1}$$

2.2 Linear Algebra And Semidefinite Programming

Since $\mathbb{R}[\mathbf{x}]_{n,2d}$ forms a vector space, we use tools from linear algebra to analyze its structure and properties. Thus, we devote this subsection introducing all required tools.

Definition 2.2.1. *Given a matrix $A \in \mathbb{R}^{n\times n}$, we say it is* symmetric *if $A^T = A$. We denote the set of* symmetric matrix *as $\mathcal{S}^n$.*

A famous result of *symmetric matrices* is:

Theorem 2.2.2 (Spectral Theorem [Dem90])**.** *Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, it can be diagonalized as*

$$A = P^{-1}DP,$$

where D is a diagonal matrix with real entries, and P is an orthonormal matrix.

In other words, all eigenvalues of A are real, and their corresponding eigenvectors form an orthonormal basis of $\mathbb{R}^n$.

Now, we introduce the key idea relating to semidefinite programming — *positive semidefinite matrices.*

Definition 2.2.3. *A matrix $A \in \mathbb{R}^{n\times n}$ is* positive semidefinite (psd) *if A is symmetric and*

$$x^T Ax \geq 0 \quad \text{for all } x \in \mathbb{R}^n.$$

We denote it as $A \succcurlyeq 0$.

Proposition 2.2.4. *A matrix is psd if and only if all its eigenvalues are nonnegative.*

Proof. Towards a contradiction, suppose A has eigenvalue $\lambda < 0$. For x being its corresponding eigenvector, we have $x^T Ax = \lambda x^T x < 0$, a contradiction.

On the other hand, assume A has only positive eigenvalues λ_i. By A being a symmetric matrix, its eigenvectors form a basis. Thus, for any $x \in \mathbb{R}^n$, we have $x = \sum_{i=1}^n c_i v_i$ where v_i are the eigenvectors of A, that are also orthonormal to each other. Hence, $x^T Ax = \sum_{i=1}^n \lambda_i$. Since all $\lambda_i \geq 0$, we have $x^T Ax \geq 0$. □

Definition 2.2.5. *Given $A, B \in \mathbb{R}^{m\times n}$, then the* Frobenius inner product
$\langle \cdot, \cdot \rangle : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}$ *is defined as*

$$\langle A, B \rangle = \mathrm{Tr}(A^T B),$$

where Tr *stands for the trace of the matrix.*

We finally introduce the program that we want use to solve the optimization problem.

Definition 2.2.6. *A* semidefinite program (SDP) *in standard primal form is the following optimization problem. Given $A, C \in \mathbb{R}^{n\times n}$,*

$$\begin{aligned} \min_{X\in\mathcal{S}^n} \langle C, X\rangle \quad &\text{subject to } \langle A_i, X\rangle = b_i, \quad i = 1, ..., k \\ &X \succcurlyeq 0. \end{aligned} \tag{2.2}$$

To compactly write the constraints, it is natural to generalize the above definition by the following:

Definition 2.2.7. *Let k be a positive integer, $A \in \mathbb{R}^{n\times kn}, B \in \mathbb{R}^{n\times n}$, then the* inner product *between A and B, $\langle \cdot, \cdot \rangle : \mathbb{R}^{n\times kn} \times \mathbb{R}^{n\times n} \to \mathbb{R}^k$ is defined as*

$$\langle A, B\rangle = \begin{bmatrix} \langle A_1, B\rangle \\ \vdots \\ \langle A_k, B\rangle \end{bmatrix},$$

where $A = \begin{bmatrix} A_1 \\ \vdots \\ A_k \end{bmatrix}$.

Note that semidefinite programming can be viewed as a generalization of linear programming. As we can see from the following definition, semidefinite programming is linear programming that optimize over the set of symmetric matrices instead of vectors.

Definition 2.2.8. *A linear programming (LP) in standard primal form can be written as the following. Given $A \in \mathbb{R}^{m\times n}, \mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m$,*

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \ \mathbf{c}^T\mathbf{x} \quad \textit{subject to } A\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq 0. \end{aligned} \tag{2.3}$$

.

Using Definition 2.2.7, we can rewrite the constraints $\langle A_i, X\rangle = b_i$ in (2.2) as

$$\langle A, X\rangle = \mathbf{b}. \tag{2.4}$$

The aim of this thesis is to study the stability of the System (2.4), where we measure stability using *(generalized) condition numbers* of a matrix.

If A is a square matrix, then the condition number can easily be calculated using its inverse. However, if A is rectangular, then we need to use its *pseudo-inverse*, see the following definition.

Definition 2.2.9. *Given a matrix $A \in \mathbb{R}^{m\times n}$, the* pseudo-inverse*, which is also knows as the* Moore-Penrose *inverse of A, is the matrix $A^\dagger$ satisfying:*

- $AA^\dagger A = A$,
- $A^\dagger AA^\dagger = A^\dagger$,
- $(AA^\dagger)^T = AA^\dagger$,
- $(A^\dagger A)^T = AA^\dagger$.

Every matrix has its pseudo-inverse, and when $A \in \mathbb{R}^{m\times n}$ is *full rank*, that is $\text{rank}(A) = \min\{n, m\}$, A can be expressed in simple algebraic form.

In particular, when A has linearly independent columns, $A^\dagger$ can be computed as

$$A^\dagger = (A^T A)^{-1} A^T.$$

In this case, the pseudo-inverse is called the *left inverse*, since $A^\dagger A = I$, with I being the identify matrix.

And when A has linearly independent rows, $A^\dagger$ can be computed as

$$A^\dagger = A^T (A A^T)^{-1},$$

and the pseudo-inverse is called the *right inverse*, since $AA^\dagger = I$.

Definition 2.2.10. *Given a matrix $A \in \mathbb{R}^{m\times n}$, the* (generalized) condition number *of A, $\kappa(A)$ is defined as*

$$\kappa(A) = \begin{cases} ||A|| \cdot ||A^\dagger|| & \textit{if } A \textit{ is full rank} \\ \infty & \textit{otherwise} \end{cases}$$

for any norm $||\cdot||$ imposed on A.

To understand how the condition number is related to the stability of (2.4), we assume A is a square matrix, then $\kappa(A)$ can be equivalently written as:

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, \tag{2.5}$$

where the $\sigma_{\max}$ and $\sigma_{\min}$ denote the maximal and minimal singular values of A respectively.

Intuitively, when the condition number is large, an error in the input along the max direction of the singular value would dominate "correct data" that is along the direction of the minimum singular value. Therefore, if the condition number of (2.4) is small, this system is more stable under fluctuations caused by noises.

Again, we note that (2.5) does in general not apply to our problem, and we present it simply for the purpose of providing an intuition of the concept. A more rigorous explanation of the condition number can be found in [CK12].

2.3 Comparing of Coefficients Algorithm

In this section, we introduce the algorithm that solves the decision problem described in (2.1).

To start, we introduce the following theorem which shows how to convert our decision problem into an SDP.

Theorem 2.3.1 (Chapter 3 [BPT12]). *Let $p(\mathbf{x}) \in P_{n,2d}$. If $p(\mathbf{x}) \in \Sigma_{n,2d}$, then for any basis $\mathcal{B}_{n,d}$ of $\mathbb{R}[\mathbf{x}]_{n,d}$, there exists a matrix $Q \in \mathbb{R}^{\mathcal{N}(d)\times\mathcal{N}(d)}$ such that*

$$\mathcal{B}_{n,d}^T \mathcal{Q} \mathcal{B}_{n,d} = p(\mathbf{x}) \textit{ and } \mathcal{Q} \succcurlyeq 0. \tag{2.6}$$

Note, by abuse of notation, we also use $\mathcal{B}_{n,d}$ to denote the vector formed by all elements in the basis $\mathcal{B}_{n,d}$.

Proof. If $p(\mathbf{x}) \in \Sigma_{n,2d}$, we can write

$$p(\mathbf{x}) = \sum_{i=1}^{k} q^2(\mathbf{x}) = \big[q_1(\mathbf{x}), ..., q_k(\mathbf{x})]\big] \begin{bmatrix} q_1 \\ \vdots \\ q_k \end{bmatrix}$$

Notice that $q_j(x) \in P_{n,d}$.

Now given $\mathcal{B}_{n,d} = \{b_1, ..., b_{\mathcal{N}(d)}\}$ be a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$, we have

$$q_j(\mathbf{x}) = \sum_{i=1}^{\mathcal{N}(d)} c_j b_j = \big[c_1, ..., c_{\mathcal{N}(d)}]\big] \begin{bmatrix} b_1 \\ \vdots \\ b_{\mathcal{N}(d)} \end{bmatrix}$$

By substituting the section equation into the first, we have

$$p(\mathbf{x}) = \begin{bmatrix} b_1 & ... & b_{\mathcal{N}(d)} \end{bmatrix} \begin{bmatrix} c_{1,1} & \cdots & c_{1,k} \\ \vdots & & \\ c_{\mathcal{N}(d),1} & \cdots & c_{\mathcal{N}(d)} \end{bmatrix} \begin{bmatrix} c_{1,1} & \cdots & c_{1,\mathcal{N}(d)} \\ \vdots & & \\ c_{k,1} & \cdots & c_{k,\mathcal{N}(d)} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_{\mathcal{N}(d)} \end{bmatrix}$$

Now the matrices in the middle is $CC^T = \mathcal{Q}$ a psd matrix, which proofs the forward direction of this theorem.

On the other hand, if we know $p(\mathbf{x}) = \mathcal{B}_{n,d}^T \mathcal{Q} \mathcal{B}_{n,d}$ where $\mathcal{Q}$ is a psd matrix, we can just apply the Cholesky decomposition to get $\mathcal{Q} = L^T L$, and recover the SOS form of $p(\mathbf{x})$ as $\mathcal{B}_{n,d}^T L^T L \mathcal{B}_{n,d}$. □

Therefore, we have reduced our decision problem to finding a specific psd matrix. Actually, it would be deciding the feasibility of the SDP problem stated above. When we examine the formulation of the SDP in (2.2), we minimize a target function subject to a set of linear constraints and the variable being a psd matrix. By examining (2.6), we can see that we have a set of constraints (later we translate this set of constraints exactly into the constraints in the SDP) and the requirement of $\mathcal{Q}$ being a psd. Therefore, the existence condition that is provided in the Theorem 2.3.1 is the subpart of the SDP which determines whether there is a feasible point that satisfies the constraints that are imposed.

To address how the constraints $\mathcal{B}_{n,d}^T \mathcal{Q} \mathcal{B}_{n,d} = p(\mathbf{x})$ are translated into the constraints $\langle A, X \rangle = B$ in the SDP, we have the following proposition.

Proposition 2.3.2. *Let $\mathcal{B}_{n,d} = \{b_1, \cdots, b_{\mathcal{N}(d)}\}$ be a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$, and let $\mathcal{B}_{n,2d} = \{b'_1, ..., b'_{\mathcal{N}(2d)}\}$ be the basis of $\mathbb{R}[\mathbf{x}]_{n,2d}$ induced by $\mathcal{B}_{n,d}$ using Remark 2.1.3. Given $p(\mathbf{x}) = \sum_{i=1}^{\mathcal{N}(2d)} c_i b'_i \in \Sigma_{n,2d}$, then the task of finding $\mathcal{Q}$ in* 2.6 *can be reformulated as solving the feasibility part of an SDP stated in 2.2.6.*

Proof. By the above theorem, the constraints are,

$$\begin{aligned} p(\mathbf{x}) = \mathcal{B}_{n,d}^T \mathcal{Q} \mathcal{B}_{n,d} = \langle \mathcal{B}_{n,d} \mathcal{B}_{n,d}^T, \mathcal{Q} \rangle \\ \mathcal{Q} \succcurlyeq 0, \end{aligned}$$

as stated in 2.2.6. □

Note the constraints $p(\mathbf{x}) = \langle \mathcal{B}_{n,d}\mathcal{B}_{n,d}^T, \mathcal{Q}\rangle$ require us to compare the equality of two polynomials. By Theorem 2.1.2, polynomials form a vector space. Thus, when fixing a basis, the polynomials are uniquely identified via its coefficients. As a result, two polynomials are equal if they have the same coefficients when regarded under the same basis.

Notice that $\langle \mathcal{B}_{n,d}\mathcal{B}_{n,d}^T, \mathcal{Q}\rangle$ and $p(\mathbf{x})$ are polynomials expressed in the basis $\mathcal{B}_{n,2d}$. Thus, the equality can be established by comparing the coefficient vectors of the two polynomials.

Goal 2.3.3. *We define the above process as comparison of coefficients (COC) and this thesis is designated to evaluate the stability of the constraints* $p(\mathbf{x}) = \langle \mathcal{B}_{n,d}\mathcal{B}_{n,d}^T, \mathcal{Q}\rangle$*.*

As we have mentioned, the stability is measured by the condition number of a matrix. Thus, we need to re-write the constraints into the form $Ax = c$. Therefore, the problem needs to be further reformulated. To distinguish the choices of the involved basis in this process, we introduce two specific matrices following the ideas of [Rec14].

Definition 2.3.4 ([Rec14])**.** *We call the matrix* $\mathcal{B}_{n,d}\mathcal{B}_{n,d}^T$ *in Proposition 2.3.2 the* moment matrix*, we denote this matrix as* $\mathcal{M}$*. By definition this matrix is symmetric and of rank* 1*.*

Example 2.3.5. *Consider the basis* $\mathcal{B}_{1,2} = \{1, 2x, 4x^2 - 4\}$ *of* $\mathbf{R}[x]_{1,2}$*, one can easily verify this is a basis.*

If we write $\mathcal{B}_{1,2} = \begin{bmatrix} 1 \\ 2x \\ 4x^2 - 4 \end{bmatrix}$*, the moment matrix* $\mathcal{M}$ *is*

$$\mathcal{M} = \mathcal{B}_{1,2}\mathcal{B}_{1,2}^T = \begin{bmatrix} 1 & 2x & 4x^2 - 4 \\ 2x & 4x^2 & 8x^3 - 8x \\ 4x^2 - 4 & 8x^3 - 8x & 16x^4 - 32x^2 + 4 \end{bmatrix}.$$

The moment matrix provides the first basis choice involved in the COC process. Suppose the given polynomial $p(\mathbf{x}) = \sum_{j=0}^{k} c_j b_j$ is in the same basis as the basis of the moment matrix, that is the upper triangle of $\mathcal{B}_{n,d}\mathcal{B}_{n,d}^T$ consists of $b_0, ..., b_k$. Since $\mathcal{Q} \succcurlyeq 0$, $\mathcal{Q}$ is symmetric and hence it is completely determined by its upper triangle entries. Let $\mathbf{q} = \left[q_{0,0}, ..., q_{0,m}, q_{1,1}, q_{1,2}, ..., q_{m,m}\right]^T$ be the vector consisting of all the elements of the upper triangle of $\mathcal{Q}$. The constraints $p(\mathbf{x}) = \langle \mathcal{B}_{n,d}\mathcal{B}_{n,d}^T, \mathcal{Q}\rangle$ can then be reformulated as a set of linear equations $A\mathbf{q} = \mathbf{c}$ where $\mathbf{c} = \left[c_0, ..., c_k\right]^T$ is the coefficients vector of $p(\mathbf{x})$, and A is a matrix that is used to establish the equality of polynomials. Then, we can measure the stability of the constraints $p(\mathbf{x}) = \langle \mathcal{B}_{n,d}\mathcal{B}_{n,d}^T, \mathcal{Q}\rangle$ by the condition number of A.

Since, this matrix A is not the final matrix that is used in the analysis, the procedure of obtaining A is omitted.

Now, what if $p(\mathbf{x})$ is written in a basis that is different from the basis constructed by the moment matrix? One might argue that we can simply multiply a change of basis matrix to convert $p(\mathbf{x})$ into the basis that is used in moment matrix. However, this process would be inefficient and inaccurate. The reason is that a change of basis matrix involves writing the basis of one polynomial in terms of the basis of the other. This itself is a process of COC and results in an increase of fluctuations to (2.4) because the condition number is never smaller than 1 [Dem90]. Therefore, we introduce the *coefficient moment matrix* in what follows.

Suppose the moment matrix is constructed with the basis $\mathcal{B}_{n,d}$ and the polynomial $p(\mathbf{x})$ is written in the basis $\mathcal{B}'_{n,2d}$. That is, we have $p(\mathbf{x}) = c_0 b'_0 + ... + c_k b'_k$, with $\mathcal{B}'_{n,2d} = \{b'_0, ..., b'_k\}$ where $k = \binom{n+2d}{2d} - 1$.

Definition 2.3.6 ([Rec14]). *We define the coefficient extraction map as the following map,*

$$\begin{aligned} \mathcal{C} : \mathbb{R}[x]_{n,2d} \times \mathcal{B}'_{n,2d} &\to \mathbb{R} \\ \mathcal{C}(p, b'_j) &\mapsto c_j. \end{aligned}$$

When fixing j, we have the coefficient extraction map being a linear map with respect to the polynomial $p(\mathbf{x})$. Indeed, we have

$$\begin{aligned} \mathcal{C}(\lambda p, b'_j) &= \lambda \mathcal{C}(p, b'_j) \quad \text{for } \lambda \in \mathbb{R} \text{ and} \\ \mathcal{C}(p_1 + p_2, b'_j) &= \mathcal{C}(p_1, b'_j) + \mathcal{C}(p_2, b'_j). \end{aligned}$$

Remark 2.3.7. *When the $\mathcal{B}'{}_{n,2d} = \{b'_1, ..., b'_k\}$ is an orthonormal basis, i.e. $\langle b_i, b_j \rangle = \delta_{i,j}$ (the Dirac delta function), where the inner product is defined as*

$$\langle p, q \rangle = \int_a^b p(\mathbf{x}) q(\mathbf{x}) d\alpha(\mathbf{x}),$$

there is a natural concrete definition for the coefficients extraction map*, namely*

$$\mathcal{C}(p(\mathbf{x}), b'_j) = \langle p(\mathbf{x}), b'_j \rangle.$$

When the basis is only orthogonal, we can still define the coefficients extraction map *concretely as,*

$$\mathcal{C}(p(\mathbf{x}), b'_j) = \frac{1}{||b'_j||} \langle p(\mathbf{x}), b'_j \rangle,$$

where the norm $|| \cdot ||$ is induced by the corresponding inner product.

Since the coefficient extraction map depends on the basis itself, it is more accurate to write the coefficient extraction map as $\mathcal{C}_{\mathcal{B}'{}_{n,2d}}$, However, when the basis is clear, we omit the basis and just write it as $\mathcal{C}$.

We can generalize the coefficient extraction map to take in a matrix as the first argument, and just apply the map entry-wise. With an abuse of notation, we have:

Definition 2.3.8. *Given a matrix of polynomials $(p_{i,j}(\mathbf{x}))_{i,j}$ all in the basis $\mathcal{B}'_{n,2d}$, the coefficient extraction map is defined as*

$$\begin{aligned} &\mathcal{C} : \mathbb{R}[x]^{m \times n}_{n,2d} \times \mathcal{B}'_{n,2d} \to \mathbb{R}^{m \times n} \\ &\mathcal{C}(\begin{bmatrix} p_{1,1} & ... & p_{1,n} \\ & \vdots & \\ p_{m,1} & ... & p_{m,n} \end{bmatrix}, b'_j) = \begin{bmatrix} \mathcal{C}(p_{1,1}, b'_j) & ... & \mathcal{C}(p_{1,n}, b'_j) \\ & \vdots & \\ \mathcal{C}(p_{m,1}, b'_j) & ... & \mathcal{C}(p_{m,n}, b'_j) \end{bmatrix}. \end{aligned}$$

Remark 2.3.9. *An immediate result from the above definition is that, given $Q \in \mathbb{R}^{m\times m}$, $\mathcal{M} \in \mathbb{R}[x]^{m\times m}_{n,2d}$, and a basis $\mathcal{B}'_{n,2d} = \{b'_1, ..., b'_k\}$, the matrix inner product provides the following relation,*

$$\mathcal{C}(\langle Q, \mathcal{M}\rangle, b'_j) = \langle Q, \mathcal{C}(\mathcal{M}, b'_j)\rangle = \langle \mathcal{C}(\mathcal{M}, b'_j), Q\rangle$$

Let $\mathcal{B}_{n,d} = \{b_0, ..., b_l\}$, where $l = \mathcal{N}(d) - 1$. By abuse of notation, we again let $\mathcal{B}_{n,d} = [b_1, ..., b_l]^T$ and $\mathcal{M} = \mathcal{B}_{n,d}\mathcal{B}^T_{n,d} \in \mathbb{R}^{l\times l}$. Then given $p(\mathbf{x})$ in $\mathcal{B}'_{n,2d} = \{b'_0, ..., b'_k\}$, $p = c_0 b'_0 + ... + c_k b'_k$ and $Q \in \mathbb{R}^{l\times l}$ being the change of basis matrix from $\mathcal{B}_{n,2d}$ to $\mathcal{B}'_{n,2d}$, where $\mathcal{B}_{n,2d}$ is generated by $\mathcal{B}_{n,d}$ using Remark 2.1.3, we have:

$$c_j = \mathcal{C}(p, b'_j) = \mathcal{C}(\langle Q, \mathcal{M}\rangle, b'_j) = \langle \mathcal{C}(\mathcal{M}, b'_j), Q\rangle. \tag{2.7}$$

By the above relationship, we find the tool we are looking for. Given any moment matrix and a basis that is different from the one used to construct the moment matrix, we can extract the coefficient of any element of the basis when being used to represent the moment matrix. This object is crucial, so we have the following definition:

Definition 2.3.10. *We define the matrix $\mathcal{A}_j = \mathcal{C}(\mathcal{M}, b'_j)$ to be the coefficient moment matrix of b'_j.*

An immediate observation we can make is the following:

Remark 2.3.11. *$\mathcal{A}_j$ is symmetric, because $\mathcal{M}$ is symmetric.*

Recall, the SOS problem is to decide, given a polynomial $p(\mathbf{x})$, whether there exists a $Q \succcurlyeq 0$ such that $p = \mathcal{B}^T_{n,d} Q \mathcal{B}_{n,d}$. Suppose $p(\mathbf{x})$ is given in $\mathcal{B}'_{n,2d}$, that is $p(\mathbf{x}) = \sum_{i=0}^{\mathcal{N}(2d)-1} c_j b'_j$, we can then reformulate the constraints $\mathcal{B}^T_{n,d} Q \mathcal{B}_{n,d}$ using the coefficient moment matrix as

$$\langle \mathcal{A}_j, Q\rangle = c_j \quad \forall j = 0, ..., \mathcal{N}(2d) - 1. \tag{2.8}$$

Let

$$\mathcal{A}_j = \begin{bmatrix} a^{(j)}_{0,0} & a^{(j)}_{0,1} & \cdots & a^{(j)}_{0,l} \\ a^{(j)}_{0,1} & a^{(j)}_{1,1} & \cdots & a^{(j)}_{1,l} \\ & \vdots & & \\ a^{(j)}_{0,l} & a^{(j)}_{1,l} & \cdots & a^{(j)}_{l,l} \end{bmatrix}, \quad Q = \begin{bmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,l} \\ q_{0,1} & q_{1,1} & \cdots & q_{1,l} \\ & \vdots & & \\ q_{0,l} & q_{1,l} & \cdots & q_{l,l} \end{bmatrix},$$

and set

$$\mathbf{a}_j = \begin{bmatrix} a^{(j)}_{0,0} \\ 2a^{(j)}_{0,1} \\ \vdots \\ 2a^{(j)}_{0,l} \\ a^{(j)}_{1,1} \\ 2a^{(j)}_{1,2} \\ \vdots \\ a^{(j)}_{l,l} \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} q_{0,0} \\ q_{0,1} \\ \vdots \\ q_{0,l}, \\ q_{1,1}, \\ q_{1,2} \\ \vdots \\ q_{l,l} \end{bmatrix} \in \mathbb{R}^{l(l+1)/2}.$$

We can reformulate the inner product $\langle \mathcal{A}_j, Q\rangle$ using the fact that both $\mathcal{A}_j$ and Q are symmetric:

$$\langle \mathcal{A}_j, Q\rangle = \mathbf{a}_j^T \cdot \mathbf{q}.$$

Then, finally, we can re-write the constraint $p(\mathbf{x}) = \mathcal{B}_{n,d}^T Q \mathcal{B}_{n,d}$, as the following system of linear equations:

$$\mathcal{A}\mathbf{q} = \begin{bmatrix} a_0^T \\ \vdots \\ a_l^T \end{bmatrix} \mathbf{q} = \begin{bmatrix} c_0 \\ \vdots \\ c_l \end{bmatrix} = \mathbf{c}.$$

Thus, the numerical property of the constraints for the SDP problem is completely captured by the condition number of $\mathcal{A}$, and we can measure it given $\mathcal{A}$ is full-rank. Therefore, we write computer programs to examine different combinations of bases under different degrees and number of variate. In particular, in this thesis we write our code in *python*. In the rest of this section, we discuss the polynomial bases that we are will use in this comparison, and briefly touch upon SDP before we present our results in the next section.

2.4 Polynomial Bases

In this section, we briefly survey the polynomial bases that is considered in this thesis, and their associated coefficient extraction map (defined in Definition 2.3.6). We also analyze the runtime of the coefficient extraction map based on the algorithms that we employed.

2.4.1 Monomial Basis

The most canonical basis for $\mathbb{R}[\mathbf{x}]_{n,d}$ is the monomial basis:

$$\mathcal{B}_{n,d} = \{1, x_1, x_2, ..., x_1^2, x_1x_2, ..., x_n^d\}.$$

The coefficient extraction map associated to this basis is straightforward: one would expand a given polynomial and group them by terms. Formally, we can capture this process with the following algorithm.

Algorithm 1: Coefficient Extraction Map for Monomial

```
**Result:** Coefficient of $b$ in $p$
**Input:** Polynomial $p$, Monomial basis vector $b$, Monomial basis $B$
expand $p$ so that it is a sum of terms in $B$;
store coefficient of each term in a hash table $H$;
return $H[b]$;
```

Suppose the input polynomial $p \in \mathbb{R}[\mathbf{x}]_{n,d}$ has k parentheses, has at most m terms in each parenthesis, and has l characters (exclude all the operators) in it. The run time of the above algorithm would be $O(mk + nd + l)$. The $O(mk)$ is the cost of operations required to expand p; $O(l)$ is the cost of operations to iterate through the polynomial to group terms; $O(nd)$ is the cost of operations to get the coefficient for each basis b in B.

Example 2.4.1. *The polynomial of the form* $p = (3x_1 + x_2)(x_3 + x_4) + x_2$ *has* $k = 2$, $m = 2$, $l = 6$, *and is in* $\mathbb{R}[\mathbf{x}]_{2,2}$.

2.4.2 Orthogonal Polynomial Bases

Other than the monomial basis, our focus in this thesis is on orthogonal polynomials. The reason is that due to their orthogonality, the coefficient extraction map is relatively easy to construct.

Chebyshev Polynomials. There are two kinds of polynomials that are widely used in numerical analysis, the *Chebyshev Polynomial of the First Kind* and *Chebyshev Polynomial of the Second Kind* [MH02]. The roots of these polynomials, called *Chebyshev nodes* are used in polynomial interpolation because the resulting interpolation polynomial minimizes the effect of *Runge's phenomenon.*

There are many ways to generate the two sequence of polynomials in univariate settings. Here, we decide to include the recursive definition since this is the most straightforward one to implement.

The univariate *Chebyshev Polynomial of the First Kind* can be constructed as

$$\begin{aligned}
&T_0(x) = 1\\
&T_1(x) = x\\
&T_{d+1}(x) = 2xT_d(x) - T_{d-1}(x),
\end{aligned}$$

where $T_d(x)$ denotes the d degree univariate Chebyshev Polynomial of the First Kind [MH02].

Due to its orthogonality, the associated coefficient extraction map would be

Algorithm 2: Coefficient Extraction Map for Chebyshev First Kind

```
Result: Coefficient of $b$ in $p$
Input: $p$ be any polynomial, $b$ be an element of a multivariate Chebyshev First Kind basis
coeff $= \int_{-1}^{1} \frac{pb}{\sqrt{1-x^2}} dx$;
norm $= \int_{-1}^{1} \frac{bb}{\sqrt{1-x^2}} dx$;
return $\frac{\text{coeff}}{\text{norm}}$;
```

When handling bivariate cases, we can use a similar treatment as in Remark 2.1.3 to obtain bases in higher dimensions.

The coefficient extraction map seems to be tedious to define in higher dimension cases. However, building upon the idea of Mádi-Nagy [MN12] we have the following result which enables us to easily construct the coefficient extraction map for orthogonal polynomials in higher dimensions.

Proposition 2.4.2. *Given an orthogonal univariate polynomial basis $\mathcal{B} = \{b_1, ..., b_n\}$, and assume the orthogonality is under the inner product*

$$\langle b_i, b_j \rangle = \int_l^u b_i(x) b_j(x) w(x) dx,$$

where $w(x)$ is a weight function, then the multivariate polynomial basis induced by this univariate polynomial basis is also orthogonal, and the corresponding inner product is

$$\langle b_i, b_j \rangle = \int_l^u ... \int_l^u b_i(x) b_j(x) w(x_1) ... w(x_n) dx_1 ... dx_n.$$

Proof. Consider the case where we start with an orthogonal univariate polynomial basis $\mathcal{B} = \{b_1(x), ..., b_n(x)\}$. Then, using modified apporach as in Remark 2.1.3 to construct a basis for bivariate polynomials, we would get

$$\mathcal{B}' = \{b'_1(x,y), ..., b'_{n^2}(x,y)\} = \{b_1(x)b_1(y), b_1(x)b_2(y), ..., b_2(x)b_1(y), ..., b_n(x)b_n(y)\}.$$

Now it is immediate from Fubini's theorem that

$$\begin{aligned}\int_l^u \int_l^u b'_i(x,y)b'_j(x,y)w(x)w(y)dxdy &= \int_l^u \int_l^u b_{i_1}(x)b_{j_1}(y)b_{i_2}(x)b_{j_2}(y)w(x)w(y)dxdy \\ &= \int_l^u b_{i_1}(x)b_{i_2}(x)w(x)dx \int_l^u b_{j_1}(y)b_{j_2}(y)w(y)dy \\ &= \begin{cases} C & \textit{if } i_1 = i_2 \textit{ and } j_1 = j_2 \\ 0 & \textit{otherwise} \end{cases},\end{aligned}$$

where $C \neq 0$ is the norm of a basis $b' \in \mathcal{B}'$. Inductively, one can generalize this to higher dimensions as well. □

Now using Proposition 2.4.2, we have the coefficient extraction map for multivariate *Chebyshev Polynomials of the First Kind* be

$$\mathcal{C}(p(\mathbf{x}), b(\mathbf{x})) = \int_{-1}^{1} \frac{p(x_1, \ldots, x_n)b(x_1, \ldots, x_n)}{\sqrt{(1-x_1^2)(1-x_2^2)\cdots(1-x_n^2)}} dx_1 dx_2 \cdots dx_n.$$

The univariate *Chebyshev Polynomials of the Second Kind*, as suggested in [MH02], can be constructed with the following recurrence relation:

$$\begin{aligned}&U_0(x) = 1 \\ &U_1(x) = 2x \\ &U_{d+1}(x) = 2xU_d(x) - U_{d-1}(x),\end{aligned}$$

where $U_d(x)$ denotes the d degree univariate Chebyshev polynomials of the Second Kind.

The associated coefficient extraction map is

Algorithm 3: Coefficient Extraction Map for Chebyshev Second Kind

```
Result: Coefficient of $b$ in $p$
Input: $p$ be any polynomial, $b$ be an element of a multivariate Chebyshev Second Kind
       basis
coeff = $\int_{-1}^{1} pb\sqrt{1-x^2}dx$;
norm = $\int_{-1}^{1} bb\sqrt{1-x^2}dx$;
return $\frac{\text{coeff}}{\text{norm}}$;
```

By the same process described above, we can construct multivariate Chebyshev polynomials of the Second Kind again using a similar treatment Remark 2.1.3, and we can construct the associated coefficient extraction map as in Proposition 2.4.2.

Legendre Polynomials *Legendre polynomials* provide special orthogonal polynomial basis that has a vast number of mathematical properties, and numerous applications. One of the most modern applications that Legendre polynomials are used in is machine learning. According to Yang, Hou and Luo, who developed a neural network based on Legendre polynomials to solve ordinary differential equations [YHL18]. Also, according to Dash, implementing Legendre polynomials in *recurrent neural networks* based predicter can significantly increase the performance of the prediction [Das20].

There are many ways to construct the Legendre polynomials. In order to implement Legendre polynomials, we present the *Rodrigues' formula* that generates univariate Legendre polynomials [Bli13]:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2 - x)^n,$$

where $P_n(x)$ denotes the univariate Legendre polynomials of degree n, and $\frac{d^n}{dx^n}$ denotes the nth derivative with respect to x.

Another way to construct the Legendre polynomials is to define this basis as the solution to an orthogonal system with respect to the weight function $w(x) = 1$ over the interval $[-1, 1]$. That is, $P_n(x)$ is a polynomial such that

$$\int_{-1}^{1} P_m(x)P_n(x)dx = 0 \quad \textit{if } n \neq m.$$

Though this formulation is bad for solving the exact expression of Legendre polynomials, it shows that Legendre polynomials are orthogonal with respect to the inner product $\int_{-1}^{1} dx$. Thus, this definition directly provides us the coefficient extraction map associated to *Legendre polynomials* in the univariate case.

Then, via the same treatment introduced in Remark 2.1.3 and Proposition 2.4.2, we have the coefficient extraction map associated to Legendre polynomials be the following algorithm.

Algorithm 4: Coefficient Extraction Map for Legendre Polynomial

```
**Result:** Coefficient of $b$ in $p$
**Input:** $p$ be any polynomial, $b$ be an element of a multivariate Legendre basis
coeff = $\int_{-1}^{1} ... \int_{-1}^{1} p(\mathbf{x})b(\mathbf{x})dx_1...dx_n$;
norm = $\int_{-1}^{1} ... \int_{-1}^{1} b(\mathbf{x})b(\mathbf{x})dx_1...dx_n$;
return $\frac{\text{coeff}}{\text{norm}}$;
```

Jacobi Polynomials The last kind of orthogonal polynomials that we consider are *Jacobi polynomials.* As for the other polynomials, there are many ways to construct the Jacobi polynomial. The easiest way to implement them is by the *Rodrigues's formula* [Sue01]. In the univariate case, we have

$$P_n^{(\alpha,\beta)}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{-\alpha}(1+x)^{-\beta}\frac{d^n}{dx^n}[(1-x)^{\alpha}(1+x)^{\beta}(1-x^2)^n],$$

where $P_n^{(\alpha,\beta)}(x)$ is a Jacobi polynomial with degree n of parameter α, β.

The Jacobi polynomials are orthogonal under the inner product

$$\langle P_i^{(\alpha,\beta)}(x), P_j^{(\alpha,\beta)}(x)\rangle = \int_{-1}^{1}(1-x)^{\alpha}(1+x)^{\beta}P_i^{(\alpha,\beta)}(x)P_j^{(\alpha,\beta)}(x)dx.$$

In fact, Jacobi polynomials generalize the orthogonal polynomials discussed above. When setting $\alpha = 0, \beta = 0$, we obtain the Legendre polynomials. (So it is not surprising that they both can be generated using Rodrigues' formula).

Applying Remark 2.1.3 and Proposition 2.4.2, yields the following algorithm for the coefficient extraction map associated to multivariate Jacobi polynomials:

Algorithm 5: Coefficient Extraction Map for Jacobi Polynomial

```
Result: Coefficient of $b$ in $p$
Input: $p$ be any polynomial, $b$ be an element of a multivariate Jacobi basis, parameters $\alpha, \beta$
coeff = $\int_{-1}^{1} \dots \int_{-1}^{1} (1-x_1)^{\alpha}(1+x_n)^{\beta} \dots (1-x_n)^{\alpha}(1+x_n)^{\beta} p(\mathbf{x}) b(\mathbf{x}) dx_1 \dots dx_n$;
norm = $\int_{-1}^{1} \dots \int_{-1}^{1} (1-x_1)^{\alpha}(1+x_n)^{\beta} \dots (1-x_n)^{\alpha}(1+x_n)^{\beta} b(\mathbf{x}) b(\mathbf{x}) dx_1 \dots dx_n$;
return $\frac{\text{coeff}}{\text{norm}}$;
```

2.5 Solving Semidefinite Programs

Lastly, we provide a brief overview on how to solve SDPs, see [BBV04]. The most common algorithms utilize the duality theory. Recall the formulation of an SDP in (2.2). This is usually referred to as the *primal* SDP. Utilizing the duality theory, we can also formulate a *dual* version of the SDP. The advantage of the dual version is that it is usually easier to solve and has a better problem structure that one can exploit.

The *dual* SDP can be formulated as,

$$\max_{\mathbf{y} \in \mathbb{R}^k} \langle \mathbf{b}, \mathbf{y} \rangle \quad \text{subject to } \sum_{i=1}^{k} y_i A_i \preccurlyeq C, \tag{2.9}$$

where $\mathbf{b} = \begin{bmatrix} b_1 & \cdots & b_k \end{bmatrix}^T$, and the inner product is the regular vector inner product. As we can see from this formulation, we can without loss of generalization assume $k \leq n$ since otherwise we can fully determine the unknown matrix X from (2.2), then the dimension of the matrix in our problem become smaller, which leads to a faster running time. What is more, it can be proven that the dual problem will always be convex no matter whether the primal is convex or not. This ensures that we can use convex optimization to solve the problem.

Hence, we can use interior point methods to solve the dual SDP. In what follows, we first provide a gentle introduction of duality theory. Then, we present the interior point method in a general setting. Lastly, we conclude this subsection with how this method is used to solve SDPs.

2.5.1 Duality Theory

For the ease of notation and understanding, we work with real valued functions that take in vectors as their parameters. However, the input can be easily generalized to matrices by replacing regular inequalities with *generalized inequalities*, denoted as $x \preccurlyeq_K y$ with K being a proper cone, meaning $x - y \in K$ [BBV04].

Let $f\colon \mathbb{R}^n \to \mathbb{R}, f_i\colon \mathbb{R}^n \to \mathbb{R}, h_i\colon \mathbb{R}^n \to \mathbb{R}$, then any constrained optimization problem, in its primal form, can be written as:

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \text{subject to} \quad & f_i(\mathbf{x}) \leq 0 \quad \text{i} = 1,\dots,m \\ & h_i(\mathbf{x}) = 0 \quad \text{i} = 1,\dots,p.\end{aligned} \tag{2.10}$$

Then, we define the *Lagrangian* of the primal problem to be a function $\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}$ that

$$\mathcal{L}(\mathbf{x}, \lambda, \nu) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}).$$

We call the vectors $\lambda = \begin{bmatrix}\lambda_1 & \cdots & \lambda_m\end{bmatrix}^T, \nu = \begin{bmatrix}\nu_1 & \cdots & \nu_p\end{bmatrix}^T$ the *Lagrange multiplier vectors* associated with the primal problem.

Then we define the *Lagrange dual function* or just *dual function* $g\colon \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as,

$$g(\lambda, \nu) = \inf_{\mathbf{x}} \{f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x})\}.$$

Observe that $g(\lambda, \nu) \leq p^*$ for all λ, ν, where p^* is the optimal value of the primal problem (2.10). Hence, the dual function provides a lower bound of the optimal value of the primal problem. Also, note that the lower bound is trivial if there exists $\lambda_i < 0$, since that gives $g = -\infty$. So we say the points (λ, ν) are *dual feasible* if $\lambda_i \geq 0$ for all $i = 1, \dots, m$.

Because the dual function is a lower bound of the optimal value of (2.10), a natural question to ask is: what is the largest lower bound we can obtain? This leads us to define the *dual optimization problem* (or *dual problem*):

$$\max_{\lambda,\nu} g(\lambda, \nu) \quad \text{subject to} \quad \lambda_i \geq 0 \quad \text{i} = 1,\dots,m. \tag{2.11}$$

Observe that if we set d^* as the optimal value of the dual problem, then $d^* \leq p^*$. We call this inequality *weak duality.* If we have $d^* = p^*$, then we say *strong duality* holds. Strong duality is very powerful as it offers a tight lower bound, and it can be asserted that it holds if *Slater's condition* holds [BBV04]. Note, there are also other conditions that leads to strong duality, see [BBV04] as well.

Now, suppose *strong duality* holds, let $\mathbf{x}^*$ denote the primal optimal point and (λ^*, ν^*) denote the dual optimal point, then we have:

$$\begin{aligned} f(\mathbf{x}^*) &= g(\lambda^*, \nu^*) \\ &= \inf_{\mathbf{x}} \{f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^* f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i^* h_i(\mathbf{x})\} \\ &\leq f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* h_i(\mathbf{x}^*) \\ &\leq f(\mathbf{x}^*). \end{aligned} \tag{2.12}$$

The last inequality is due to the fact that $\mathbf{x}^*$ must be feasible, which leads to $f_i(\mathbf{x}^*) \leq 0$ and $h_i(\mathbf{x}^*) = 0$.

By the chain of inequalities in (2.12), we can conclude that the two inequalities must be equalities. This result is significant, as we can observe two key points. First, from the second line to the third line with equality being placed as inequality, we see that $\mathbf{x}^*$ minimize $\mathcal{L}(\mathbf{x}, \lambda^*, \nu^*)$. Secondly, from line three to line four again with equality, we must have $\sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}^*) = 0$. Notice that for both primal and dual feasibility, $\lambda_i^* \geq 0$, $f_i(\mathbf{x}^*) \leq 0$, which implies that each term in the sum is nonnegative. Hence, for it to sum to 0, we must have $\lambda_i^* f_i(\mathbf{x}^*) = 0$ for all $i = 1, \ldots, m$. We call this property the *complementary slackness*.

With the above two observations, we are ready to formulate the *KKT* (*Karush-Kuhn-Tucker*) optimality conditions.

Theorem 2.5.1. *Given the primal problem stated in* (2.10)*, suppose that f, f_i are convex and h_i are affine, and we assume they are all differentiable, then the following holds: If $\mathbf{x}^*, \lambda^*, \nu^*$ satisfy the KKT conditions*

$$\begin{aligned} f_i(\mathbf{x}^*) &\leq 0, \quad i = 1, \ldots, m \\ h_i(\mathbf{x}^*) &= 0, \quad i = 1, \ldots, p \\ \lambda_i^* &\geq 0, \quad i = 1, \ldots, m \\ \lambda_i^* f_i(\mathbf{x}^*) &= 0, \quad i = 1, \ldots, m \\ \nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(\mathbf{x}^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(\mathbf{x}^*) &= 0, \end{aligned} \tag{2.13}$$

then $\mathbf{x}^*$ and (λ^*, ν^*) are primal and dual optimal.

Proof. Given $\mathbf{x}^*, \lambda^*, \nu^*$ satisfying the KKT conditions, we have $\mathbf{x}^*$ is primal feasible by the first two conditions. The third condition grants dual feasibility. Further, by the third condition, we can observe that the Lagrangian $\mathcal{L}(\mathbf{x}, \lambda^*, \nu^*)$ is convex in $\mathbf{x}$. Thus, it is minimized when the gradient with respect to $\mathbf{x}$ is 0, which is precisely what the last KKT condition states. From this, we conclude that

$$\begin{aligned} g(\lambda^*, \nu^*) &= \mathcal{L}(\mathbf{x}^*, \lambda^*, \nu^*) \\ &= f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}^*) + \sum_{i=1}^p \nu_i^* h_i(\mathbf{x}^*) \\ &= f(\mathbf{x}^*), \end{aligned}$$

where the last inequality follows from the fourth KKT condition, the complementary slackness. By the fact that the dual value is always a lower bound of the primal optimal value, we have $\mathbf{x}^*$ and (λ^*, ν^*) are primal and dual optimal. □

The KKT condition gives us a certificate of how to use duality to find an optimal point of both the primal and dual problem. Now, we are ready to introduce the interior point method that utilizes the duality theory.

2.5.2 Interior Point Method

When the objective function f and the inequality constraints f_i are all convex and twice continuously differentiable, and the equalities h_i are affine, then we can rewrite the primal problem as:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \text{subject to} \quad f_i(\mathbf{x}) \leq 0 \quad \text{i} = 1\dots m \\ A\mathbf{x} = \mathbf{b}, \tag{2.14}$$

where $A \in \mathbb{R}^{p\times n}, b \in \mathbb{R}^p$ representing the h_i.

The interior point method is devised to handle this form of optimization problem. In particular, it handles the inequality constraints in (2.14) and utilizes Newton's method to satisfies the equality constraints. We first outline the procedure of Newton's method, starting with the unconstrained version and move to the version with equality constraints. Then, we present how the interior point method successfully reduces to the problem having only equality constrained form. In the rest of this section, we always assume f and f_i are convex and twice continuously differentiable.

Newton's Method. Consider the following unconstrained optimization problem of the form:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

Then for any point $\mathbf{x} \in \mathbb{R}^n$, we define the *Newton's step* be $\Delta_{nt}\mathbf{x} := -\nabla^2 f(\mathbf{x})^{-1}\nabla f(\mathbf{x})$. The convexity of f ensure that $\nabla^2 f(\mathbf{x})$ is a positive definite matrix, which implies

$$\nabla f(\mathbf{x})^T \Delta_{nt}\mathbf{x} = -\nabla f(\mathbf{x})^T \nabla^2 f(\mathbf{x})^{-1}\nabla f(\mathbf{x}) < 0,$$

unless $\nabla f(\mathbf{x}) = 0$. Note the convexity of f grants that given $\mathbf{x}, \mathbf{y}^*$ with $\mathbf{y}^*$ be the optimal solution, if $\nabla f(\mathbf{x})(\mathbf{y}^* - \mathbf{x}) \geq 0$, then $f(\mathbf{y}^*) \geq f(\mathbf{x} + c\Delta_{nt}\mathbf{x})$ for some constant c. This means the Newton's step is a decent step.

Also, consider $\hat{f}$ to be the second-order Taylor's approximation of f at $\mathbf{x}$, that is

$$\hat{f}(\mathbf{x}+\mathbf{v}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T\mathbf{v} + \frac{1}{2}\mathbf{v}^T\nabla^2 f(\mathbf{x})\mathbf{v},$$

then $\hat{f}$ is minimized exactly at $\mathbf{x} + \Delta_{nt}\mathbf{x}$. This suggests if the function f is nearly quadratic, after taking the Newton's step, we get a good estimate of the minimizer.

Further, we define the *Newton's decrement* as $\gamma(\mathbf{x}) := \nabla f(\mathbf{x})^T\nabla^2 f(\mathbf{x})^{-1}\nabla f(\mathbf{x})$. Then we have

$$f(\mathbf{x}) - \inf_{\mathbf{y}} \hat{f}(\mathbf{y}) = f(\mathbf{x}) - f(\mathbf{x} + \Delta_{nt}\mathbf{x}) = \frac{1}{2}\gamma(\mathbf{x})^2.$$

That is $\frac{1}{2}\gamma(\mathbf{x})^2$ gives an estimate of the gap of the current function value and its approximated minimal value. Thus, it can be used as a stopping condition of the Newton's method algorithm.

Algorithm 6: Newton's Method (unconstrained)

```
Input: Starting point $\mathbf{x}$, tolerance $\epsilon > 0$
while True do
    Calculate Newton's step $\Delta^2\mathbf{x}$ and Newton's decrement $\gamma(\mathbf{x})$
    if $\gamma(\mathbf{x}) \leq \epsilon$ then
        return $\mathbf{x}$
    end
    Choose step $s$ by line search [BBV04, Chapter 9.2]
    Update $\mathbf{x} = \mathbf{x} + s\Delta^2\mathbf{x}$.
end
```

We then investigate how Newton's method can be applied for optimization problem with only equality constraints:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \text{subject to} \;\; A\mathbf{x} = \mathbf{b},$$

where $A \in \mathbb{R}^{p\times n}, b \in \mathbb{R}^p$. In the constrained case, we need another approach to calculate the Newton's step to ensure it is not only a decrement direction but also a feasible direction.

Newton's method, as we saw in the unconstrained case, utilizes the second order Taylor's approximation of the objective function f to find the Newton's step. Let $\hat{f}$ be the second order Taylor's approximation in variable $\mathbf{v}$ of the objective function $f(\mathbf{x})$, so, $\hat{f}(\mathbf{x}+\mathbf{v}) = f(\mathbf{x}+\mathbf{v}) + \nabla f(\mathbf{x})^T\mathbf{v} + \frac{1}{2}\mathbf{v}^T\nabla^2 f(\mathbf{x})\mathbf{v}$. Then we want to solve the following optimization problem:

$$\min_{\mathbf{v}\in\mathbb{R}^n} f(\mathbf{x}+\mathbf{v}) + \nabla f(\mathbf{x})^T\mathbf{v} + \frac{1}{2}\mathbf{v}^T\nabla^2 f(\mathbf{x})\mathbf{v} \quad \text{subject to} \;\; A(\mathbf{x}+\mathbf{v}) = \mathbf{b}.$$

As explained in the unconstrained case, the Newton's step at $\mathbf{x}$, $\Delta_{nt}\mathbf{x}$, is the optimal $\mathbf{v}$ that minimizes $\hat{f}$, so solving the above optimization problem is finding the feasible Newton's step. The above optimization problem can be solved analytically using Theorem 2.5.1. It states that $\Delta_{nt}\mathbf{x}$ is optimal if and only if the KKT conditions are satisfied. That is, there exists $\nu^* \in \mathbb{R}^p$ such that $A\nu^* = b - A\mathbf{x} = 0$ (if $\mathbf{x}$ is feasible) and $\nabla^2 f(\mathbf{x})\nu^* + \nabla f(\mathbf{x}) + A^T\nu^* = 0$. Hence, we can characterize this primal dual optimal pair as:

$$\begin{bmatrix} \nabla^2 f(\mathbf{x}) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta_{nt}\mathbf{x} \\ \nu^* \end{bmatrix} = \begin{bmatrix} -\nabla f(\mathbf{x}) \\ 0 \end{bmatrix}.$$

The Newton's decrement is the same as in the unconstrained case and is still an estimate of the difference between $f(\mathbf{x})$ and the optimal value p^*. Therefore, the same algorithm holds for the constrained case, with the only modification that the Newton's step is calculated differently to satisfy the equality constraints.

Interior Point Method: Logarithmic Barrier Function and Central Path. In order to reduce the problem such that Newton's method can be applied, the most natural thing to do is to absorb the inequality constraints into the objective function as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + \sum_{i=1}^{m} I(-f_i(\mathbf{x})) \quad \text{subject to} \;\; A(\mathbf{x}) = \mathbf{b}, \tag{2.15}$$

where,

$$I(x) = \begin{cases} 0 & x \leq 0 \\ \infty & x > 0. \end{cases}$$

However, a problem of this idea is that the objective function is no longer differentiable, which again forbids us from applying the Newton's method. Hence, we approximate the step function by a twice differentiable function: $\hat{I}(x) = -\frac{1}{t}\log(x)$, where t is a parameter indicating the accuracy of the approximation, see Figure 1.

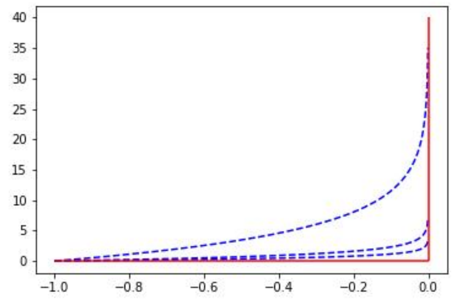


Figure 1: Logarithmic barrier approximation to the indicator function. For larger values of t, the approximation becomes more accurate.

Further, we call the function $\phi(\mathbf{x}) := -\sum_{i=1}^{m} \log(-f_i(\mathbf{x}))$ the *logarithmic barrier* function, whose domain, $\{\mathbf{x}|f_i(\mathbf{x}) < 0, i = 1, ..., m\}$, is the set of points that satisfies the inequality constraints.

Replacing I by $\hat{I}$ and writing it in terms of the logarithmic barrier function, we can approximate (2.15) as the following optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + \frac{1}{t}\phi(\mathbf{x}) \quad \text{subject to} \quad A(\mathbf{x}) = \mathbf{b}, \tag{2.16}$$

where we have a convex and twice differentiable objective function. Hence, the assumptions of Newton's Method are satisfied.

Now, we assume that for every $t > 0$, the Newton's method can produce a unique solution $\mathbf{x}^*(t)$, and we define the *central path* associated to the problem (2.16) the set of *central points*, $\{\mathbf{x}^*(t)|t > 0\}$.

Then, by Theorem 2.5.1, a central point can be characterized as

$$A\mathbf{x}^*(t) = 0, \quad f_i(\mathbf{x}^*(t)) < 0 \text{ for } i = 1 \dots m, \tag{2.17}$$

and there exists $\nu \in \mathbb{R}^p$ such that

$$\nabla f(\mathbf{x}^*(t)) + \frac{1}{t}\nabla\phi(\mathbf{x}^*(t)) + \frac{1}{t}A^T\nu = \nabla f(\mathbf{x}^*(t)) + \sum_{i=1}^{m} \frac{1}{-tf_i(\mathbf{x}^*(t))}\nabla f_i(\mathbf{x}^*(t)) + \frac{1}{t}A^T\nu = 0. \tag{2.18}$$

An important property for the central path is that each central point yields a dual feasible point:

$$\lambda_i^*(t) = -\frac{1}{tf_i}, i = 1, \dots, m, \quad \nu^*(t) = \nu/t.$$

Substituting it into (2.18), we have

$$\nabla f(\mathbf{x}^*(t)) + \sum_{i=1}^{m} \lambda_i^*(t)\nabla f_i(\mathbf{x}^*(t)) + \frac{1}{t}A^T\nu^*(t) = 0 \tag{2.19}$$

Notice that (2.19) is the derivative of the Lagrangian $\mathcal{L}(\mathbf{x}, \lambda, \nu)$ of the original inequality constrained optimization problem (2.14) at $\mathbf{x}^*(t)$ equals 0. Since the Lagrangian is convex, we moreover have $\mathbf{x}^*(t)$ minimizes the Lagrangian given $\lambda = \lambda^*(t)$ and $\nu = \nu^*(t)$. Hence, the dual function

$$\begin{aligned} g(\lambda^*(t), \nu^*(t)) &= f(\mathbf{x}(t)) + \sum_{i=1}^{m} \lambda_i^*(t) f_i(\mathbf{x}^*(t)) + \nu^*(t)^T (A\mathbf{x}^*(t) - b) \\ &= f(\mathbf{x}^*(t)) - m/t, \end{aligned}$$

has finite value at $\lambda^*(t), \nu^*(t)$.

Recall that the dual function forms a lower bound of the optimal value of the primal problem. Hence, if the optimal value of the primal problem is p^*, then we have,

$$f(\mathbf{x}^*(t)) - p^* \leq m/t,$$

and as $t \to \infty$, we have $\mathbf{x}^*(t)$ converge to the minimizer of the primal objective function.

It is very tempting to device an algorithm that directly chooses $t = m/\epsilon$, where ϵ is a tolerance term, and directly invokes Newton's method. However, it only works well for small problems with moderate accuracy.

Hence, Fiacco and McCormick in the 1960s proposed the *barrier method*, which can be proven to have good convergence. There are also many generalizations of the method to handle different edge cases including starting with infeasible $\mathbf{x}$ etc. However, those are beyond the scope of this thesis and we refer the readers to [BBV04]. Here, we only present a simple version of the barrier method:

Algorithm 7: Barrier method

```
Input: Strictly feasible starting point $\mathbf{x}$, $t > 0$, $\mu > 1$ tolerance $\epsilon > 0$
while True do
    Centering step: computer $\mathbf{x}^*(t)$ starting at $\mathbf{x}$
    Update $\mathbf{x} = \mathbf{x}^*(t)$
    if $m/t < \epsilon$ then
        return $\mathbf{x}$
    end
    $t = \mu t$
end
```

2.5.3 Interior Point Method for Semidefinite Program

At the end of this subsection, we examine how interior point method can be applied to semidefinite programming. We mainly follow the lecture notes by Professor Pena [Pen15].

Recall that we have derived the standard and dual form of the SDP in (2.2) and (2.9). Introducing the slack variable S to the dual SDP, we can re-write the problem as

$$\begin{aligned} \max_{\mathbf{y} \in \mathbb{R}^k, S \in \mathcal{S}^n} b^T y \quad \text{subject to } &\sum_{i=1}^{k} y_i A_i + S = C \\ &S \succcurlyeq 0. \end{aligned} \tag{2.20}$$

Assume that both (2.2) and (2.20) are strictly feasible, that is we have strong duality, then $b^T y^* = \langle C, X^* \rangle$ with X^*, y^* being the primal and dual optimal variables respectively. As what we have seen above, we want to utilize Newton's method to solve the problem. To do so, we have to circumvent the inequality constraints. The KKT conditions for the SDP are as follows:

$$\begin{aligned} \langle A_i, X\rangle &= b_i \quad i = 1,\dots,k \\ X &\succcurlyeq 0 \\ S &\succcurlyeq 0 \\ XS &= 0 \\ \sum_{i=1}^{k} y_i A_i + S &= C. \end{aligned} \tag{2.21}$$

As suggested by the interior point method, we just employ a logarithmic barrier function for X and S. The barrier function in this case is defined as $\phi\colon \mathcal{S}^n \to \mathbb{R}$, $\phi(X) \coloneqq -\log\det X$. Notice that the conditions $X \succcurlyeq 0$ and $S \succcurlyeq 0$ imply X and S have nonnegative determinants. Hence, we want to bound the determinant above from 0 by the negative of the logarithmic function of the determinant. Then, we can just apply what we have said above to calculate the optimal conditions. Note that the Newton's method requires slight modifications. Yet, this is out of the scope of this thesis, so we refer the reader to [Pen15] for more details.

At the end, we note that there are many well-developed SDP solvers, like *SDPT3* and *SeDuMi*. One can refer to [BPT12, Chapter 2.3] for a list of available SDP solvers.

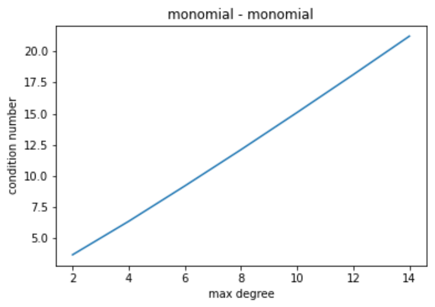
3 Numerical Results

In this section, we discuss the results obtained from our numerical experiments. The experiments are run on a 2015 MacBook Pro with a 3.1 GHz Quad-Core Intel Core i7 Processor and 16 GB of RAM. The code for the experiment is available at `https://github.com/YixuanSeanZhou/math_honor_project.git`.

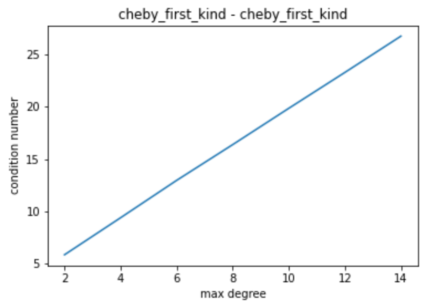
We divide the discussion into two subsections, one devoted for the univariate polynomials, and the other for the bivariate case. The title of each figure appearing in this section uses the notation $\mathcal{B}$ - $\mathcal{B}'$ to describe the basis for the input polynomial as $\mathcal{B}$ and the basis for the moment matrix as $\mathcal{B}'$.

3.1 Univariate Polynomials

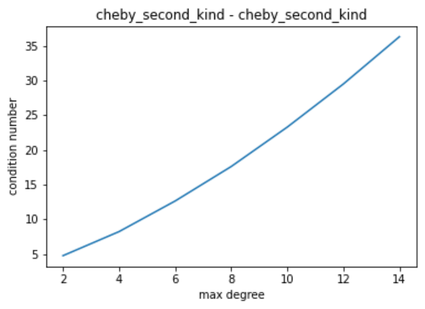
We carry out the algorithms described above for univariate polynomials and plot the results in different figures depending on the combination of bases. First, we study the condition numbers when the two bases are the same, see Figure 2. We observe that the condition number of the resulting coefficient moment matrix tends to be small. In particular, the monomial polynomial basis has the smallest condition number among the other polynomials. And among all the orthogonal polynomial bases, the Chebyshev first kind has the best behavior.



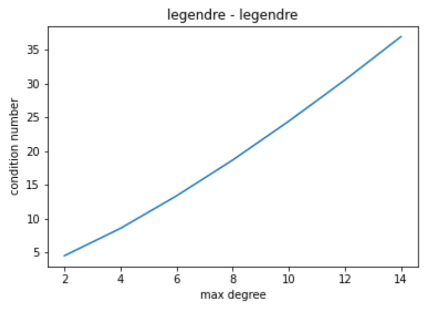
(a) Monomial polynomial with monomial moment matrix



(b) Chebyshev 1st kind polynomial with Chebyshev 1st kind moment matrix



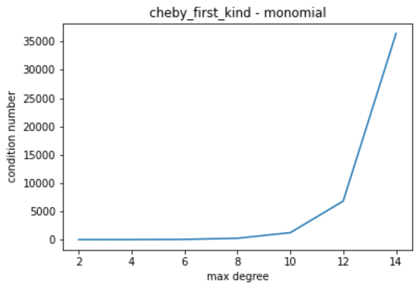
(c) Chebyshev second 2nd polynomial with Chebyshev 2nd kind moment matrix



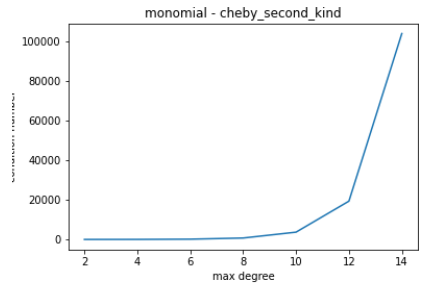
(d) Legendre polynomial with legendre moment matrix

Figure 2: Condition number of the coefficient moment matrix for same basis (univ)

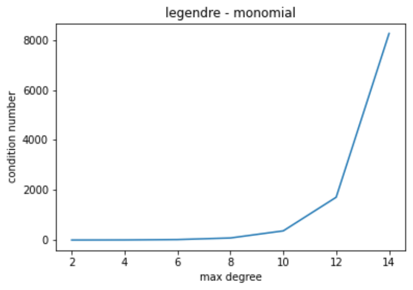
After that, we discuss the performance when we pair orthogonal bases with monomial bases, see Figure 3. We observe that the monomial polynomial basis does not work well with orthogonal polynomial bases and the condition number of the coefficient moment matrix tends to blow up as the degree of the polynomials increases.



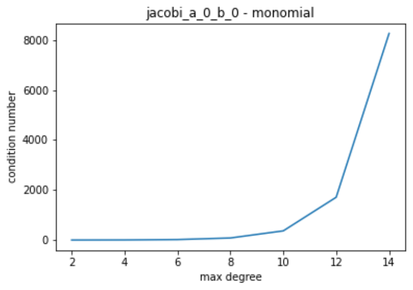
(a) Chebyshev 1st kind polynomial with monomial moment matrix



(b) monomial polynomial with Chebyshev 2nd Kind moment matrix



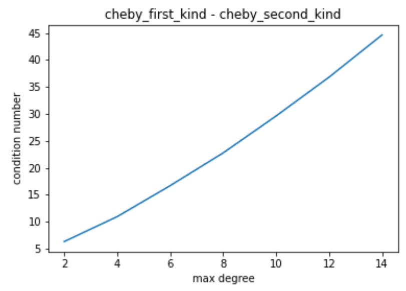
(c) Legendre polynomial with monomial moment matrix



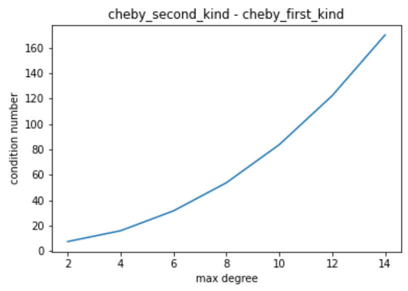
(d) Jacobi polynomial with monomial moment matrix

Figure 3: Monomial with orthogonal polynomials (univ)

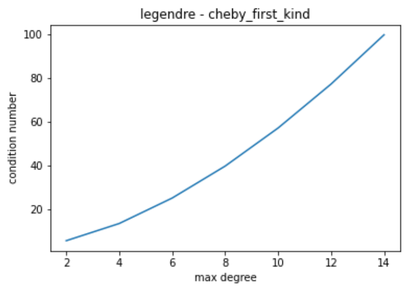
Lastly, in Figure 4 we capture the condition number of the pairing of different orthognal bases. We see the scale of the condition number is dramatically smaller than the condition numbers when monomial basis is involved.



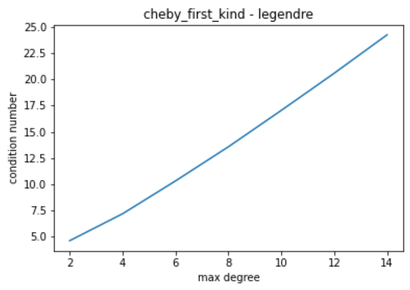
(a) Chebyshev first kind polynomial with Chebyshev second kind moment matrix



(b) Chebyshev second kind polynomial with Chebyshev first kind moment matrix



(c) Legendre polynomial with Chebyshev first kind moment matrix



(d) Chebyshev first kind polynomial with Legendre moment matrix

Figure 4: Monomial with orthogonal polynomials (univ)

Concluding, we list the condition number of the coefficient moment matries of the four degree 14 polynomials we experimented with in Table 1. Note, we omit the Jacobi polynomial, because Chebyshev and Legendre polynomials are special cases of it. We highlight the smallest condition number in blue, and the smallest condition number among orthogonal bases in orange.

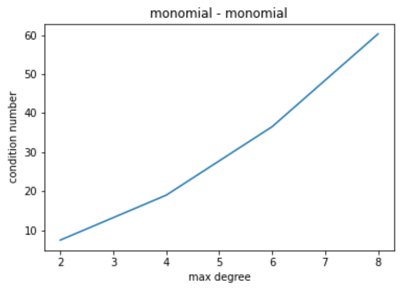
Moment Matrix Basis / Polynomial Basis	Monomial	Chebyshev 1st	Chebyshev 2nd	Legendre
Monomial	2.122801e+01	8.866559e+04	1.040619e+05	4.401169e+04
Chebyshev 1st	3.643853e+04	2.673064e+01	4.463266e+01	2.428417e+01
Chebyshev 2nd	5.248902e+04	1.701201e+02	3.629213e+01	7.221286e+01
Legendre	8.260979e+03	9.981569e+01	6.871414e+01	3.691980e+01

Table 1: Condition number of coefficient moment matrix with degree 14 (univ).

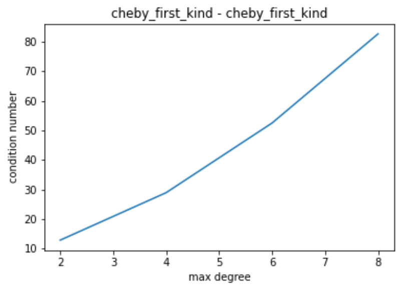
3.2 Bivariate Polynomials

In this subsection, we perform the same experiments for the bivariate cases, and plot the results similarly as in univariate case. Since the computational cost is much higher in this case, we only experiment with polynomials up to degree 8. Nonetheless, we do observe analogous phenomena as in the univariate case.

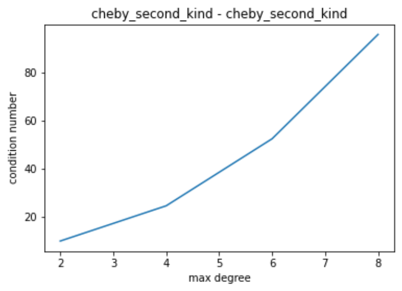
Like in the univariate case, the condition number is smaller when the basis of the polynomials and the basis of the moment matrix are the same, see Figure 5. The monomial still has the smallest condition number, and Chebyshev first kind outperforms other orthogonal bases by a small margin.



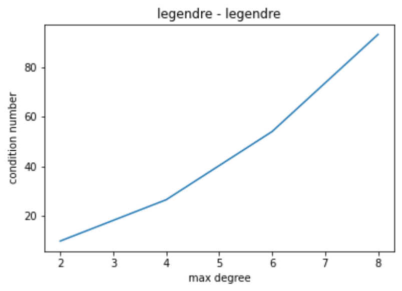
(a) monomial polynomial with monomial moment matrix



(b) Chebyshev first kind polynomial with Chebyshev first kind moment matrix



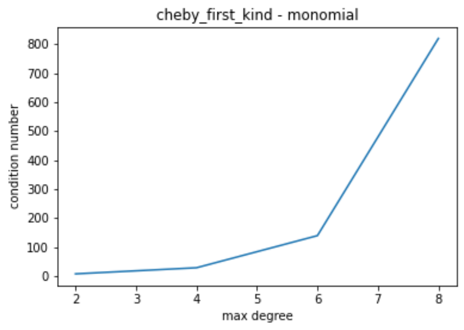
(c) Chebyshev second kind polynomial with Chebyshev second kind moment matrix



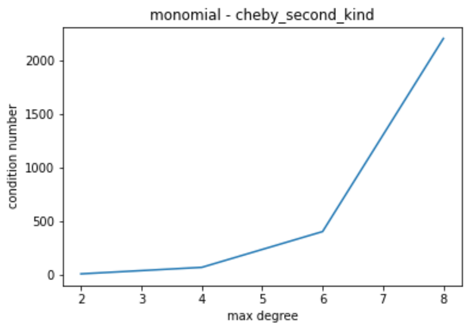
(d) Legendre polynomial with legendre moment matrix

Figure 5: Condition number of the coefficient moment matrix for same basis (biv)

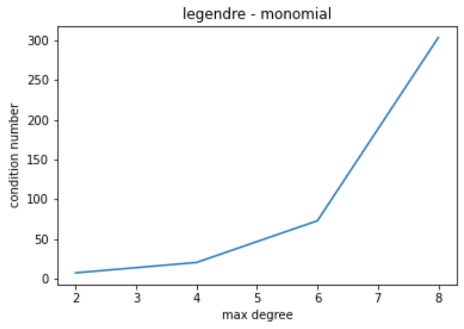
When we have a combination of monomials and orthogonal polynomials, the condition numbers again explode with growing polynomial degree, see Figure 6.



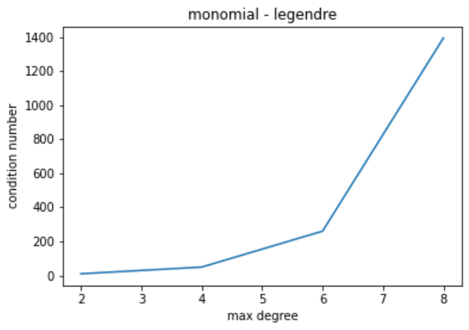
(a) Chebyshev first kind polynomial with monomial moment matrix



(b) monomial polynomial with Chebyshev second Kind moment matrix



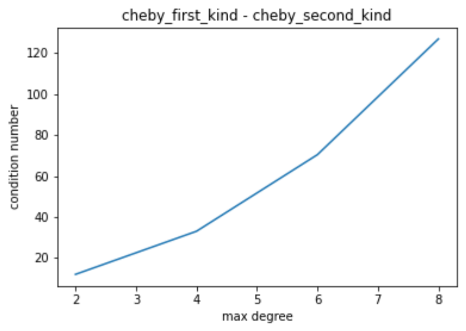
(c) Legendre polynomial with monomial moment matrix



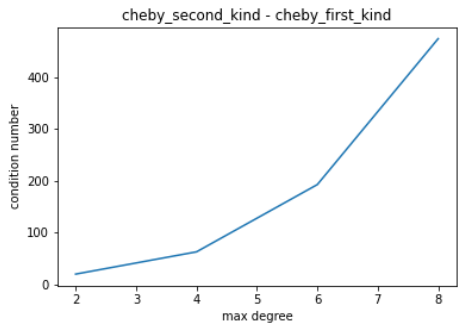
(d) Jacobi polynomial with monomial moment matrix

Figure 6: Monomial with orthogonal polynomials (biv)

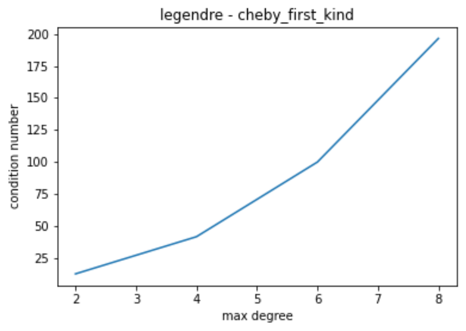
When we pair orthogonal polynomials bases, the behavior of the system is more stable. Notice that even better performance is achieved when combining Chebyshev first kind with Legendre, compared to pairing Chebyshev or Legendre with themselves. The results are shown in Figure 7.



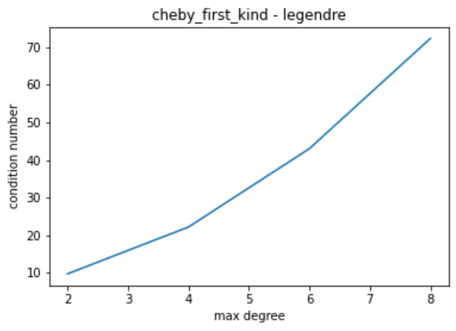
(a) Chebyshev first kind polynomial with Chebyshev second kind moment matrix



(b) Chebyshev second kind polynomial with Chebyshev first kind moment matrix



(c) Legendre polynomial with Chebyshev first kind moment matrix



(d) Chebyshev first kind polynomial with Legendre moment matrix

Figure 7: Monomial with orthogonal polynomials (biv)

Here, we present two tables to summarize the condition number of the combinations of the four degree 8 polynomial bases that we used in both univariate case (Table 2) and bivariate case (Table 3). Comparing the two tables, we observe that the condition numbers of the bivariate case are significantly larger than the ones in the univariate case.

Polynomial Basis \ Moment Matrix Basis	Monomial	Chebyshev 1st	Chebyshev 2nd	Legendre
Monomial	1.2114041e+01	9.2931794e+02	7.3426803e+02	3.4094072e+02
Chebyshev 1st	2.4395198e+02	1.6377406e+01	2.2771550e+01	1.3558274e+01
Chebyshev 2nd	4.4894498e+02	5.3861887e+01	1.7604737e+01	2.9399442e+01
Legendre	8.5482089e+01	3.9905724e+01	3.2969591e+01	1.8699677e+01

Table 2: Condition number of coefficient moment matrix with degree 8 (univ).

Polynomial Basis \ Moment Matrix Basis	Monomial	Chebyshev 1st	Chebyshev 2nd	Legendre
Monomial	6.0303522e+01	3.8821664e+03	2.2060581e+03	1.3954908e+03
Chebyshev 1st	8.2003242e+02	8.2741589e+01	1.2690227e+02	7.2356027e+01
Chebyshev 2nd	1.3583584e+03	4.7372727e+02	9.5693596e+01	2.0181265e+02
Legendre	3.0351806e+02	1.9649165e+02	1.4316038e+02	9.3276473e+01

Table 3: Condition number of coefficient moment matrix with degree 8 (biv).

4 Discussion and Outlook

In this section, we state the observations drawn from Section 3, and we discuss some open problems related to our findings for potential future efforts.

To begin with, we observe that using the monomial basis as both the basis for input polynomials and the basis for the coefficient moment matrix tends to be a good choice, since it constantly has low condition number. Therefore, the linear system generated by COC is more robust when the input data is interpolated by random noises. Moreover, when performing the experiments, the moment matrix with monomial basis has shorter running time, where we measure running time by computer time rather than in complexity theory terms. As a result, deceivingly, one tends to conclude that one should just choose monomial bases for both input polynomial and the coefficient moment matrix. However, this is not true. In fact, the monomial basis has disadvantages in stability and running time:

In terms of the stability, it may happen that the input polynomials are in the orthogonal basis. In that case, from what we can see in the above section, choosing monomial polynomials as the basis for the coefficient moment matrix would horribly destroy the stability of the system. The reason is that the condition number of the linear system blows up very quickly. One might argue that we can simply perform a change of basis operation to convert orthogonal basis to monomials. However, this will also yield the same issues (if not more) of the condition number, because we would need to consider the condition number of the change of a basis matrix as well. Hence, in that case we need to choose the suitable polynomial basis for our coefficient moment matrix that best serves the applications.

In terms of the running time, the reason that monomial basis have faster computing time is because when using orthogonal polynomials as the basis of the coefficients moment matrix, we need to evaluate an integral to extract the coefficients. However, in applications, this is more or less an overhead work that one shall perform before executing optimizations. The reason is that in applications, the number of variables and the degree of the polynomials involved are predetermined. Therefore, one could just calculate the needed integral for each basis beforehand. Then, one can simply apply the linearity of an integral to quickly calculate the coefficients of any linear combination of a basis. Since any polynomial in a corresponding basis is a linear combination of the elements in that basis, each step in the optimization process would only require constant time multiplication to obtain the coefficients, rather than performing $O(mk + nd + l)$ coefficients extraction algorithm as indicated in Section 2.4.1.

As a result, one should try to use orthogonal basis in the coefficient moment matrix when the application is suitable and the stability of the linear system generated by the COC algorithm is relatively well conditioned.

Another observation that we can easily make is that when comingling the monomial basis with the orthogonal basis, the condition number of the linear system explodes very fast. However, we do not have a rigorous explanation for this phenomenon. One reason might be the following: Since the orthogonal polynomials are all special cases of Jacobi polynomials, they share similar structures in their bases. As a result, when performing the integration, we have good numerical stability. However, to fully understand this phenomenon, one has to take a closer look. Maybe there are some structures that polynomial basis has that can be used to deduce the stability of the system, rather than using numerical experiments.

Lastly, we point out that there are other nonnegativity certificates that capture different intuitions or have different usages in applications than the canonical SOS certificate. For example, the *sums of nonnegative circuit polynomials* (SONC) [IDW16], a recently introduced certificate of nonnegativity, is more applicable in constraint polynomial optimizations [Wan18]. This can, for instance, be used to substitute SOS for optimization problems over hypercube, on which SOS has worse performance compared to other algorithms [DKDW21]. Therefore, it would also be interesting to see whether different choices of polynomial basis can impact the stability of the system generated when using other nonnegativity certificates.

References

[AM16] Amir Ali Ahmadi and Anirudha Majumdar, *Some applications of polynomial optimization in operations research and real-time decision making*, Optimization Letters **10** (2016), no. 4, 709–729.

[BBV04] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[Bli13] Sy M Blinder, *Guide to essential math: a review for physics, chemistry and engineering students.*

[BPT12] Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas, *Semidefinite optimization and convex algebraic geometry*, SIAM, 2012.

[CK12] E Ward Cheney and David R Kincaid, *Numerical mathematics and computing*, Cengage Learning, 2012.

[CLR95] Man-Duen Choi, Tsit Yuen Lam, and Bruce Reznick, *Sums of squares of real polynomials*, Proceedings of Symposia in Pure mathematics, vol. 58, American Mathematical Society, 1995, pp. 103–126.

[Das20] Rajashree Dash, *Performance analysis of an evolutionary recurrent legendre polynomial neural network in application to forex prediction*, Journal of King Saud University-Computer and Information Sciences **32** (2020), no. 9, 1000–1011.

[Dem90] James W Demmel, *Matrix computations; (gene golub and charles f. van loan)*, SIAM Review **32** (1990), no. 4, 690.

[DKDW21] Mareike Dressler, Adam Kurpisz, and Timo De Wolff, *Optimization over the boolean hypercube via sums of nonnegative circuit polynomials*, Foundations of Computational Mathematics (2021), 1–23.

[Dre18] Mareike Dressler, *Sums of nonnegative circuit polynomials: Geometry and optimization*, Ph.D. thesis, Universitätsbibliothek Johann Christian Senckenberg, 2018.

[Hil88] David Hilbert, *Über die darstellung definiter formen als summe von formenquadraten*, Mathematische Annalen **32** (1888), no. 3, 342–350.

[IDW16] Sadik Iliman and Timo De Wolff, *Amoebas, nonnegative polynomials and sums of squares supported on circuits*, Research in the Mathematical Sciences **3** (2016), no. 1, 1–35.

[Jos16] Cédric Josz, *Application of polynomial optimization to electricity transmission networks*, arXiv preprint arXiv:1608.03871 (2016).

[Lay16] David C Lay, *Linear algebra and its applications 5th edition*, Pearson (2016).

[MH02] John C Mason and David C Handscomb, *Chebyshev polynomials*, CRC press, 2002.

[MN12] Gergely Mádi-Nagy, *Polynomial bases on the numerical solution of the multivariate discrete moment problem*, Annals of Operations Research **200** (2012), no. 1, 75–92.

[Mot67] Theodore Samuel Motzkin, *The arithmetic-geometric inequality*, Inequalities (Proc. Sympos. Wright-Patterson Air Force Base, Ohio, 1965) (1967), 205–224.

[Pen15] Javier Pena, *Primal-dual interior-point methods part ii*, March 2015.

[Rec14] Marius Recher, *Zur numerischen Auswirkung von Basiswahlen in der polynomiellen Optimierung*, 2014, Master Thesis, Universität zu Köln.

[Rez00] Bruce Reznick, *Some concrete aspects of hilbert's 17th problem*, Contemporary mathematics **253** (2000), 251–272.

[Sue01] PK Suetin, *Jacobi polynomials*, Encyclopaedia of Mathematics, translated from the Russian **5** (2001), 227.

[Wan18] Jie Wang, *Nonnegative polynomials and circuit polynomials*, arXiv preprint arXiv:1804.09455 (2018).

[YHL18] Yunlei Yang, Muzhou Hou, and Jianshu Luo, *A novel improved extreme learning machine algorithm in solving ordinary differential equations by legendre neural network methods*, Advances in Difference Equations **2018** (2018), no. 1, 1–24.