HONORS PROJECT

YIXUAN ZHOU

ABSTRACT. Short description

1. Introduction

Polynomial optimization are ubiquitous in applied mathematics and engineering. For instance, Ahmadi and Majumdar [AM15] suggested to use polynomial optimization in the field of *artificial intelligence* for solving *real-time decision problems*. In the field of electric power system design, the *optimal power flow problem* can be reduced to polynomial optimization problem as well, see [Jos16].

In general, an *optimization* problem takes for the form

$$\min \ f(\mathbf{x}) \quad s.t. \ g_i(\mathbf{x}) \geq 0, h_j(\mathbf{x}) = 0,$$

where f, g, h are arbitrary functions of the variables $\mathbf{x}$. The function f is called the *objective function*, and the functions g, h are called the *constraints*. That is 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 includes *gradient decent* and *Newton's method*. Those optimization algorithms seeks to exploit the properties of the objective functions to find a local minimizer $\mathbf{x}^*$ and hopes (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, by its name, is the optimization problem where f, g, h are all polynomials. In this thesis, we focus on the polynomial optimization in unconstrained case, that is the feasible domain of the problem is $\mathbb{R}^n$. Then, instead of finding a local minimizer $\mathbf{x}^*$, polynomial optimization tries to minimize the polynomial f globally by finding its max lower bound. That is we can equivalently formulate the problem as

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

where the objective function f is a polynomial of $\mathbf{x}$, and the lower bound r is a real number.

To solve this problem, we would need to decide whether a given polynomial, $f(\mathbf{x}) - r$ is *nonnegative* (see Definition 2.5). It turns out that this problem has been well studied in real algebraic geometry since the beginning of the 20th century. However, it turns out that deciding whether an arbitrary multivariate polynomial is nonnegative is, in the language of computational complexity, *NP-hard* (computationally infeasible).

Key words and phrases. some keywords go here.

Therefore, instead of directly deciding the *nonnegativity* of a polynomial, one wants to find sufficient conditions that certify nonnegativity of a polynomial which are easier to check. Such conditions are called *certificate of nonnegativity.* The most famous and canonical certificate is *sums of squares (SOS)* (see Definition 2.6). Deciding if a polynomial is a sums of squares can be formulated as a *semidefinite program (SDP)* (see Definition 2.11). SDPs have known algorithms that, under mild conditions, can be solved in polynomial time (efficiently) with respect to the input size [BPT13, Chapter 2].

When solving the *SDP*, the solver is actually dealing with a set of linear constraints, 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 $\mathbb{R}[\mathbf{x}]_{n,d}$. Therefore, the process of *COC* is just formulating the equality constraints by identifying the constants $c_0, ..., c_k$. For more about vector space and basis, one could refer to the book Linear Algebra and Its Applications by Lay, Lay and McDonald[Str06].

Because the linear system that formed by *COC* depends on the basis $\mathcal{B}$, the stability of the linear system, measured by the *condition number* (see Definition 2.13), is drastically fluctuated depends on the $\mathcal{B}$ that we chose. The stability of the system is very important in practice because this measure how robust the method is under noises in the data. Yet, it is unclear what basis generates the best result. In this paper, we perform numerical experiments, with the computer's aid, of different choices of the basis $\mathcal{B}$ to determine, in different scenarios, what is the best choice of basis to carry out the process of *COC* so that the resulted linear system is most stabled.

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

Acknowledgements

I would like to express my deepest gratefulness to my program advisor 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 paper. 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.

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 base that are considered in this thesis.

2.1. Real Polynomials.

First, we rigorously define the decision problem that is considered in this paper. Thus, in this subsection we define the terms that are used 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 $\mathbb{R}^{m\times n}$, and the ring (set) of real-valued n-variate polynomials is denoted as $\mathbb{R}[\mathbf{x}]$, 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.

A polynomial is an expression consisting of variables and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponentiation of variables. Therefore, a univariate polynomial takes the form, $p(x) = \sum_{i=1}^{l} c_i x^{\alpha(i)}$, with c_i be the leading coefficient and $\alpha(i)$ be the degree of the term. And we can introduce more variables to it.

Example 2.1. $p(x, y, z) = x^4 + 2xyz - 6y + 7$ with x, y, z being variables. This polynomial has 3 variables and has degree 4, thus it belongs to $\mathbb{R}[\mathbf{x}]_{3,4}$.

Some quick observations that we can get about the set $\mathbb{R}[\mathbf{x}]_{n,d}$ are:

Proposition 2.2. *$\mathbb{R}[\mathbf{x}]_{n,d}$ forms a finite real vector space, with dimension $\binom{n}{d}$.*

Proof. Suppose $p(\mathbf{x}), q(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]_{n,d}$, $c \in \mathbb{R}$, then we have, Then we have $cp \in \mathbb{R}[\mathbf{x}]_{n,d}$ because multiplication by a scalar does not increase the degree nor introduce new variables. And $p + q \in \mathbb{R}[\mathbf{x}]_{n,d}$ 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 d degree 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 $\mathbb{R}[\mathbf{x}]_{n,d}$. Also, it is a linear independent set. Thus, it forms a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$. By counting, the cardinality of B, $|B| = \binom{n+d}{d}$, which is thus the dimension of $\mathbb{R}[\mathbf{x}]_{n,d}$.

□

Let $\mathcal{B}_{n,d}$ be a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$. A canonical basis to $\mathbb{R}[\mathbf{x}]_{n,d}$ is the one used in the proof above, 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 2.3. *Given $\mathcal{B}_{n,d}$ be a basis of $\mathbb{R}[\mathbf{x}]_{n,d}$. If we list the elements of the basis $\mathcal{B}_{n,d}$ 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 2.4. 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 elements of the upper triangle of the resulted 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 2.5. Let $P_{n,2d}$ denote the set of *nonnegative polynomials* with n variables and degree at most $2d$, that is

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

The reason that we chose to write $2d$ 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 2.6. Let $\Sigma_{n,2d}$ denote the set of polynomials with n variables and degree at most $2d$ that are *sum of squares*, that is

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

Notice that $\Sigma_{n,2d} \subseteq P_{n,2d}$ because sum of squares of real numbers are always nonnegative. In fact, these two sets are *convex cones.* For more about the convex cone formulations, see [BPT13, Chapter 3]

In 1888, Hilbert in [Hil88] showed that 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 $d = 4$. It is interesting that though Hilbert proved the $\Sigma_{n,2d}$ is a strict subset of $P_{n,2d}$ other than the 3 special cases, a concrete example of a nonnegative polynomial that is not SOS is very hard to find. In fact, it takes almost one hundred years until Motzkin in [Mot67] discovered *Motzkin' polynomial*

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

which is in $\mathbb{R}[x]_{2,6}$ a nonnegative polynomial but cannot be written in SOS.

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. As a result, although using the latter certificate we fail to identify some nonnegative polynomials, but that is the trade off that we make in order to solve the problem in a computationally feasible way.

Just to reiterate, the *decision problem* that is considered in this paper 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, the tools from linear algebra play an important role in the analysis. Besides, both *SDP* and the measurement of stability of system require some tools from linear algebra. Thus, we devote this subsection introducing all the required tools.

Definition 2.7. 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 matries* is:

Theorem 2.8 (Spectral Theorem [Gol96]). *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 $\mathbb{R}^n$.

Then we introduce the key idea that related to *SDP*, the *positive semidefinite matrix.*

Definition 2.9. A matrix $A \in \mathbb{R}^{n\times n}$, it is *positive semidefinite (psd)* if A is symmetric and

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

We denote it as $A \succcurlyeq 0$.

Proposition 2.10. *A matrix is psd if and only if all its eigenvalues are greater than or equal to 0.*

Proof. Towards a contradiction, suppose matrix 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 all positive eigenvalues. 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 that $x^T Ax \geq 0$. □

Definition 2.11. A *semidefinite program (SDP)* in standard primal form is

$$\begin{aligned} &\textit{minimize } \langle C, X\rangle \quad \textit{subject to } \langle A_i, X\rangle = b_i, \quad i = 1, ..., k \\ &X \succcurlyeq 0 \end{aligned} \tag*{.(2.2)}$$

One can compactly write the constraint $\langle A_i, X\rangle = b_i$ compactly in a matrix form, we can collect all the constraints and write it is $\langle A, X\rangle = \mathbf{b}$ The stability of this linear system is then of the interest of this paper. The *condition number* of the matrix A is used to measure the stability of the above linear system. The *condition number* can be nicely calculated using the inverse (when the matrix is square) and the *pesudo-inverse* (when the matrix is rectangle) of the matrix A.

Definition 2.12. Given a matrix $A \in \mathbb{R}^{m\times n}$, the *pesudo-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 pesudo-inverse, and when $A \in \mathbb{R}^{m\times n}$ is *full rank*, that is $rank(A) = min\{n, m\}$, A can be expressed in simple algebraic form.

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

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

In this case, the pesudo-inverse is called the *left inverse* since $A^{\dagger}A = I$.

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

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

In this case, the pesudo-inverse is called the *right inverse* since $AA^{\dagger} = I$.

Definition 2.13. Given a matrix $A \in \mathbb{R}^{m\times n}$, the 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, for instance, *Frobenius norm.*

Here, I give a brief description of how the *condition number* is related to the stability of the system by introducing another way to define it.

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

where the σ denotes the singular values of A.

Thus, it can be understood as how stale our system is. Intuitively, when the condition number is large, some error in the input along the max direction of the singular value, our result would largely fluctuate because the error, magnified by the singular value, would dominate the input that is along the direction of the minimum singular value. Therefore, the smaller the condition number is, the more stable our system is under fluctuations caused by noises. The rigorous explanation of the condition number can be found in [CK07].

2.3. **Comparing Coefficient Algorithm.** With all the tools in hand, we are now ready to introduce the *COC* algorithm that solves the decision problem described in (2.1).

The algorithm is build upon the following theorem, one can find a detailed explanation of it in the third chapter of [BPT13].

Theorem 2.14. *Given $p(x) \in P_{n,2d}$, if $p(x) \in \Sigma_{n,2d}$, then for any basis $\mathcal{B}_{n,d}$ of $\mathbb{R}_{n,d}$, there exists a matrix such that*

$$\mathcal{B}_{n,d}^T \mathcal{Q} \mathcal{B}_{n,d} = p(x) \textit{ and } \mathcal{Q} \succcurlyeq 0. \tag{2.3}$$

Proof. For any $p(x) \in \mathbb{R}_{n,2d}$, if $p(x) \in \Sigma_{n,2d}$, then we can write

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

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

Now given $\mathcal{B}_{n,d} = \{b_1, ..., b_{\binom{n+d}{d}}\}$ be a basis of $P_{n,d}$, we have

$$q_j(x) = \sum_{i=1}^{\binom{n+d}{d}} c_j b_j = \begin{bmatrix} c_1, ..., c_{\binom{n+d}{d}} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_{\binom{n+d}{d}} \end{bmatrix}$$

By substituting the section equation into the first, we have

$$p(x) = \begin{bmatrix} b_1 & \cdots & b_{\binom{n+d}{d}} \end{bmatrix} \begin{bmatrix} c_{1,1} & \cdots & c_{1,k} \\ \vdots & & \\ c_{\binom{n+d}{d},1} & \cdots & c_{\binom{n+d}{d},k} \end{bmatrix} \begin{bmatrix} c_{1,1} & \cdots & c_{1,\binom{n+d}{d}} \\ \vdots & & \\ c_{k,1} & \cdots & c_{k,\binom{n+d}{d}} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_{\binom{n+d}{d}} \end{bmatrix}$$

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

On the other hand, if we know $p(x) = \mathcal{B}_{n,d}^T \mathcal{Q} \mathcal{B}_{n,d}$ 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 $p(x)$ as $\mathcal{B}_{n,d}^T L^T L \mathcal{B}_{n,d}$. □

Therefore, we have reduced our problem decision problem to finding this *psd* matrix. The above theorem provides a hint that the above problem can be solved via *SDP*. Actually, it would be solving a sub-part of *SDP*. When examine the formulation of *SDP* in (2.2), we would minimize a target function subject to a set of linear constraints and the variable being a *psd* matrix. By examining (2.3), we can see that we have a set of constraints (later we translate this set of constraints exactly into the constraints in *SDP*) and the requirement of $\mathcal{Q}$ being a *psd*. Therefore, we found that the existence condition that is provided in the Theorem 2.14 is a subpart of the *SDP*, which is determining whether there is a feasible point that satisfies the constraints that are imposed.

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

Proposition 2.15. *We pick a basis of $\mathcal{B}_{n,d} = \{b_1, ..., b_{\binom{n+d}{d}}\}$ of $\mathbb{R}_{n,d}$, and list it in a vector form $\mathbf{b} = \begin{bmatrix} b_1 & \cdots & b_{\binom{n+d}{d}} \end{bmatrix}^T$. Then by Remark 2.3, we can form a basis $\mathcal{B}_{n,2d} = \{b'_1, ..., b'_{\binom{n+2d}{2d}}\}$ from the vector b. Suppose the $p(x)$ that we are interested in is written in the form $\sum_{i=1}^{\binom{n+2d}{2d}} c_i b'_i$. Then, we have the reformulation of the constraints as $p(x) = \mathbf{b}^T \mathcal{Q} \mathbf{b} = \langle Q, \mathbf{b}\mathbf{b}^T \rangle$, which when written separately in different rows is exactly the formulation in Definition 2.11.*

Notice that the constraints $p(x) = \langle Q, \mathbf{b}\mathbf{b}^T \rangle$ involved in comparing two polynomials. By Theorem 2.2, $\mathbb{R}[\mathbf{x}]_{n,2d}$ form a vector space. The equality of two vectors is established by comparing the coordinates of the two vectors when under the same basis. Thus, when the basis of $p(x)$ is the same as the basis formed by the upper triangle of $\mathbf{b}\mathbf{b}^T$, we can compare the coordinates of the vectors to establish the equality. And the coordinates for polynomials are called their coefficients. Thus, we name this process as *Comparing of Coefficients (COC)* and the paper is designated to evaluate the stability of the constrants $p(x) = \langle Q, \mathbf{b}\mathbf{b}^T \rangle$.

As we have mentioned, the stability is measured by the *condition number* of a matrix. Thus, we would need to re-write the constraints in to the form $Ax = c$. Therefore, the problem need to be further reformulated. And how the choices of the basis are involved in this process would also be introduced.

Definition 2.16 ([Rec14])**.** We call the matrix bb^T in Proposition 2.15 the *Moment Matrix*, we denote this matrix as $\mathcal{M}$, and it is a symmetric matrix by definition.

Example 2.17. Here is another place to do an example to illustrate Moment Matrix.

The *Moment Matrix* provides the first choice of basis that is involved in the process of *COC*. Suppose the given polynomial $p(x) = \sum_{j=0}^{k} c_j b_j$ is in the same basis as the resulted basis of the *Moment Matrix*, that is the upper triangle of $\mathbf{b}\mathbf{b}^\mathbf{T}$ consists $b_0, ..., b_k$. Because $\mathcal{Q} \succcurlyeq 0$, $\mathcal{Q}$ is symmetric, it is completely determined by its upper triangle. Let $\mathbf{q} = \left[q_{0,0}, ..., q_{0,m}, q_{1,1}, q_{1,2}, ..., q_{m,m}\right]^T$ be the vector consists of all the elements of the upper triangle of $\mathcal{Q}$. The constrants $p(x) = \langle Q, \mathbf{b}\mathbf{b}^T \rangle$ 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 vector of the coefficients of the $p(x)$, and A is a matrix that is used to establish the equality of polynomials. Then, we can measure the stability of the constraints $p(x) = \langle Q, \mathbf{b}\mathbf{b}^T \rangle$ by the *condition number* of A.

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

Now, what if the $p(x)$ is written in a basis that is different from the basis constructed by the *Moment Matrix*? One might argue that we can simply apply a change of basis matrix to convert $p(x)$ into the basis that is used in *Moment Matrix*. However, that is no efficient and accurate way to obtain a change of basis matrix. The reason is that a change of basis matrix would involve writing the basis of one polynomial in terms of the basis of the other. This itself is a huge process of *Comparing of Coefficients* and would result in an increase of perturbation to the system because the *condition number* is never smaller than 1 [Gol96]. Therefore, we shall introduce the *Coefficient Moment Matrix*. In the remaining part of this section, we shall build our way to it.

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

Definition 2.18 ([Rec14])**.** We define the *coefficient extraction map* as the following map,

$$\mathcal{C} : \mathbb{R}[x]_{\leq,2d} \times \mathbb{R}[x]_{\leq,2d} \to \mathbb{R}$$
$$\mathcal{C}(p, b'_j) \mapsto c_j$$

When fixing s_j, we have the *coefficient extraction map* being a linear map with respect to the polynomial $p(x)$. Indeed, we have

$$\mathcal{C}(\lambda p, b'_j) = \lambda\mathcal{C}(p, b'_j) \quad \lambda \in \mathbb{R}$$
$$\mathcal{C}(p_1 + p_2, b'_j) = \mathcal{C}(p_1, b'_j) + \mathcal{C}(p_2, b_j)$$

Remark 2.19. *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*

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

There is a natural concretely definition for the coefficients extraction map*. That is*

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

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

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

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

Remark 2.20. *We should actually write the* coefficient extraction map *as $\mathcal{C}_{\mathcal{B}'_{n,2d}}$, since it depends on the base itself. However, when the base is clear, we just write it as $\mathcal{C}$.*

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

Definition 2.21. Given a matrix of polynomials $(p_{i,j}(x))_{i,j}$ all in the bases Let the *coefficient extraction map* be defined as

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

Remark 2.22. *An immediate result from the above definition is that, given $Q \in \mathbb{R}^{m\times m}$, $M \in \mathbb{R}[x]^{m\times m}_{n,2d}$, and a basis $\mathcal{B}'_{n,2d} = \{b'_1, ..., b'_k\}$, the matrix inner product provides the following relation,*

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

Notice that, let $\mathcal{B}_{n,d} = \{b_0, ..., b_l\}$, where $l = \binom{n+d}{d} - 1$, let $\mathbf{b} = [b_1, ..., b_l]^T$, $M = \mathbf{b} \cdot \mathbf{b}^T \in \mathbb{R}^{l,l}$. Then given $p(x)$ in $\mathcal{B}'_{n,2d} = \{b'_0, ..., b'_k\}$, $p = c_0b'_0 + ... + c_kb'_k$, given $Q \in \mathbb{R}^{l,l}$ be 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 **??**, we have

$$c_j = \mathcal{C}(p, b'_j) = \mathcal{C}(\langle Q, M\rangle, b'_j) = \langle Q, \mathcal{C}(M, b'_j)\rangle \tag{2.4}$$

Definition 2.23. Define the matrix $\mathcal{A}_j = \mathcal{C}(M, b'_j)$ be the *coefficient moment matrix* of b'_j.

Proposition 2.24. *$\mathcal{A}_j$ is symmetric, because M is symmetric.*

Recall, the *SOS* problem is to decide, given a polynomial $p(x)$, whether there exists a $Q \succcurlyeq 0$ such that $p = \mathbf{b}^T Q\mathbf{b}$, where $\mathbf{b}$ be the vector generated by $\mathcal{B}_{n,d}$. Suppose $p(x)$ is given in $\mathcal{B}'_{n,2d}$, we can then reformulate the constraints $\mathbf{b}^T Q\mathbf{b}$ using the *coefficient moment matrix* as

$$\langle Q, \mathcal{A}_j\rangle = c_j \quad \forall j = 0, ..., \binom{n+2d}{2d} - 1 \tag{2.5}$$

Let $\mathcal{A}_j = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,l} \\ a_{0,1} & a_{1,1} & \cdots & a_{1,l} \\ & \vdots & & \\ a_{0,l} & a_{1,l} & \dots & a_{l,l} \end{bmatrix}, Q = \begin{bmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,l} \\ q_{0,1} & q_{1,1} & \cdots & q_{1,l} \\ & \vdots & & \\ q_{0,l} & q_{1,l} & \dots & q_{l,l} \end{bmatrix}$

Set $\mathbf{a_j} = [a_{0,0}, 2a_{0,1}..., 2a_{0,l}, a_{1,1}, 2a_{1,2}, ..., a_{l,l}]^T \in \mathbb{R}^{l(l+1)/2}$, and $\mathbf{q} = [q_{0,0}, ..., q_{1,1}, q_{1,2}..., q_{l,l}]^T \in \mathbb{R}^{l(l+1)/2}$. We can re-write the inner product $\langle Q, \mathcal{A}_j\rangle$ using the fact that both $\mathcal{A}_j$ and Q are symmetric.

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

Then, finally, we can re-write the constraint $p(x) = \mathbf{b}^T Q\mathbf{b}$, as the system of linear equations that

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

Thus, the numerical property of the *SDP* problem is completely captured by the *condition number* of A. Therefore, we write code in *python* to examine different combinations of bases under different degrees and number of variate in the Preliminaries sections. We list the polynomial bases that we are interested in the following sections, and briefly touch upon *semidifinite program* before we present our results.

2.4. Polynomial Basis.

In this section, we briefly survey the polynomial bases that is considered in this thesis, and their associated *coefficient extraction map* (defined in Definition 2.18). We also analyze the runtime of the *coefficient extraction map.*

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 b, Monomial Base 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 2.25. For example, 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 Polynomials.* Other than the monomial basis, the bulk of the focus of this thesis is on the orthogonal polynomial. The reason is that due to its orthogonality, the coefficient extraction map is relatively easy to construct.

2.4.2.1. Chebyshev Polynomials. There are two kinds of polynomials are widely used in numerical analysis, the *Chebyshev Polynomial of the First Kind* and *Chebyshev Polynomial of the Second Kind.* 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.*[MF04]

There are many ways to generate the two sequence of polynomials in univariate settings. Here, we decide to include the recursive definition since this would be 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.*

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: Polynomial p, Chebyshev First Kind basis b, Chebyshev First Kind Base B
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 multivariate cases, we can again use Remark 2.3 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 we have the following proposition which enable us to easily construct *coefficient extraction map* for orthogonal polynomials in higher dimensions. [MN12]

Proposition 2.26. *Given an orthogonal univariate polynomial base $\mathcal{B} = \{b_1, ..., b_n\}$, 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 base generated using Remark 2.3 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 base $\mathcal{B} = \{b_1(x), ..., b_n(x)\}$. Then, using Remark 2.3 to construct a base 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

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

where $c \neq 0$ is the norm of a basis $b' \in mathcalB'$. Inductively, one can generalize this to higher dimensions as well. □

Now using this Proposition 2.26, we have the *coefficient extraction map* for multivariate *ChebyShev Polynomials of the First Kind* be

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

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

$$U_0(x) = 1$$
$$U_1(x) = 2x$$
$$U_{d+1}(x) = 2xU_d(x) - U_{d-1}(x)$$

where $U_d(x)$ denotes the d degree univariate *ChebyShev Polynomial of the Second Kind.*

The associated *coefficient extraction map* would be

Algorithm 3: Coefficient Extraction Map for Chebyshev Second Kind

```
Result: Coefficient of b in p
Input: Polynomial p, Chebyshev Second Kind basis b, Chebyshev Second Kind
       Base B
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 Polynomial of the Second Kind* using Remark 2.3, and we can construct the associated *coefficient extraction map* using Proposition 2.26.

2.4.2.2. Legendre Polynomials. *Legendre Polynomials* is 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 Polynomial* in *recurrent neural networks* based predicter can significantly increase the performance of the prediction [Das20].

There are still many ways to construct the *Legendre Polynomials.* In order to implement *Legendre Polynomials*, we present the *Rodrigues' formula* that generates univariate *Legendre Polynomials.*

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

where $P_n(x)$ denotes the univariate *Legendre Polynomials* of degree n, $\frac{d^n}{dx^n}$ denotes the n'th 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

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

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

Then, via the same treatment introduced in Remark 2.3 and Proposition 2.26, 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:** Polynomial p, Legendre basis b, Legendre Base B
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 $\frac{\text{coeff}}{\text{norm}}$;
```

2.4.2.3. Jacobi Polynomials. The last kind of orthogonal polynomial that we will consider is the *Jacobi polynomial*

2.5. **Solving Semidefinite Program.** Toy examples maybe

3. Numerical Results

Proposition 3.1.

Proof. $\square$

maybe a theorem

Theorem 3.2.

or an example...

Example 3.3.

pictures are always a good idea...

4. Maybe Some Proofs

5. Resume, Outlook, or/and Open Problems

what did you do, what questions are still open, natural next steps etc.

References

[AM15] Amir Ali Ahmadi and Anirudha Majumdar, *Some applications of polynomial optimization in operations research and real-time decision making*, 2015.

[BPT13] G. Blekherman, P.A. Parrilo, and R.R. Thomas, *Semidefinite optimization and convex algebraic geometry*, MOS-SIAM Series on Optimization, vol. 13, SIAM and the Mathematical Optimization Society, Philadelphia, 2013.

[CK07] E. Cheney and D. Kincaid, *Numerical mathematics and computing*, International student edition, Cengage Learning, 2007.

[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.

[Gol96] Gene Golub, *Matrix computations*, Johns Hopkins University Press, Baltimore, 1996.

[Hil88] David Hilbert, *Ueber die Darstellung definiter Formen als Summe von Formenquadraten*, September 1888.

[Jos16] Cédric Josz, *Application of polynomial optimization to electricity transmission networks*, Theses, Université Pierre et Marie Curie - Paris VI, July 2016.

[MF04] John H. Mathews and Kurtis K. Fink, *Numerical methods using matlab (4th edition)*, 4 ed., Pearson, January 2004.

[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.

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

[Str06] Gilbert Strang, *Linear algebra and its applications*, Thomson, Brooks/Cole, Belmont, CA, 2006.

[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, 469.

8514 Villa La Jolla Drive # 114, La Jolla, CA, 92037

Email address: `yiz044@ucsd.edu`