POLYNOMIAL AND MOMENT CONIC OPTIMIZATIONS: THEORY AND APPLICATIONS

BY

MOHAMMAD MEHDI RANJBAR

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Farid Alizadeh

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2018

ABSTRACT OF THE DISSERTATION

Polynomial and Moment Conic Optimizations:

Theory and Applications

by:

Mohammad Mehdi Ranjbar

Dissertation Director:

Farid Alizadeh

We investigate the non-negative univariate polynomial conic optimization (uPCO) problem from the perspective of applications and the algorithms.

We start by considering the applications of uPCO, specifically in: 1- Time-variant network flow problems, and 2- Non-parametric estimation under shape constraints with splines . Regarding algorithms, we use non-symmetric interior point methods (IPMs) to solve this conic optimization problem.

It is well known that uPCO can be formulated as a semidefinite programming (SDP) problem, and therefore, it can be solved by available software for SDP. However, doing so will result in squaring the number of decision variables, and thus it is impractical even for moderate size problems. In addition, straightforward SDP formulation involves numerically unstable processes.

Regarding the latter issue, we propose an orthogonal change of basis. Using the Chebyshev polynomials (which form an orthogonal basis), uPCO problems with significantly higher dimensions can be solved. As for the former issue, we propose two direct non-symmetric interior-point algorithms, by specializing the non-symmetric homogeneous self-dual predictor-corrector (HSD P-C) IPM (proposed by Skajaa-Ye 2015) and a Mehrotra version (i.e. HSD M-P-C IPM) of this algorithm (proposed by Akle-Ye 2015).

We consider implementing these algorithms in two approaches. In the first approach, we develop these IPMs for the dual of the uPCO problem, i.e., the univariate

moment conic optimization (uMCO) problem, where the algorithms can be utilized by the efficient barrier function of the moment cone. In the second approach, we consider developing the previous algorithms directly for the uPCO problem, by utilizing the algorithms by the Faybusovich universal barrier function of the non-negative univariate polynomial cone. We present numerical results of our implementations of these algorithms for each approach and a comparison among them.

Next, we consider a general conic optimization problem which contains the non-negative polynomial conic constraints, as well as second order and linear conic constraints. We propose a unified non-symmetric HSD IPM for this problem. Finally, we present that the numerical results of our implementations are comparable to the results of the symmetric HSD IPM for the symmetric formulation of the general problem, without the need to square the non-negative polynomial variables.

Acknowledgements

Ph.D. is a work that cannot be accomplished without the help and support of others. First and foremost, I would like to thank my adviser, professor Farid Alizadeh, for his support, help, and ideas throughout this project. I sincerely appreciate his contribution of time, ideas, and funding to make my Ph.D. happen. It was an honor to be one of his students.

I would like to thank professor Tamas Terlaky for his time and ideas regarding the numerical implementations part of my research. I thank my dissertation committee members, professor Farid Alizadeh, Jonathan Eckstein, Andrzej Ruszczynski and Tamas Terlaky for their valuable guidance. Also, I would like to thank professor Yurii Nestrov, Dr. Andres Skajaa, Dr. Santiago Akle Serrano and professor Yinyu Ye for their outstanding research on non-symmetric interior point methods, which helped to initiate and accelerate this project.

I am especially grateful to Rutgers University for being a wonderful place to learn and grow, and for the opportunities it provided from facilities to a vast variety of offered courses.

I would like to express my gratitude to Shaya Famenini for her support, help and guidance, especially for introducing me to the beautiful world of economy and quantitative finance.

I would like to thank professor Farid Alizadeh and professor Alexander Amati for their time and valuable advice in the process of seeking an employment opportunity.

I would like to thank all of my teachers who I have learned from them. Specially, I would like to thank professor Alexander Amati, professor Daniel Ocone and Dr. Gordan Ritter, for all I have learned from their quantitative finance courses, and also professor Vladimir Pavlovic for his wonderful machine learning courses.

Lastly, I am deeply grateful to my family (mom, dad, brothers and sisters) for their love, support, encouragement and sacrifice. Without them, this thesis would have never materialized. They have taught me honesty, perseverance, faithfulness and optimism. I learned from them not to give up hope and aspire to be a good human being, even during the times of adversity. Thank you, mom and dad.

Dedication

I would like to dedicate this thesis to my parents, Maryam and Heidar Ali, my brothers and sisters who are my true love.

Table of Contents

Chapter 1

Introduction

Optimization has become one of the most active fields in science for more than a half century. A general class of optimization problems is the class of convex optimization. Perhaps among all types of algorithms for solving this class of optimization problems, Interior Point Methods (IPMs) are one of the most stable and efficient classes of algorithms. Specifically, IPMs have been widely used for solving conic optimization problems which are a subset of the convex optimization problems.

One particular class of conic optimization problems which has been getting more attention recently, is the non-negative univariate polynomial conic optimization problems. This is conic optimization over the cone of non-negative univariate polynomials, which can represent many real world problems from a wide variety of areas, e.g., engineering, statistics, mathematics, economy, finance, etc., that can be mathematically formulated as this type.

In this thesis, we investigate the non-negative univariate polynomial conic optimization problem from different perspectives, i.e. applications, formulation, and algorithms for solving this type of optimization problems.

In terms of applications, we formulate several interesting problems in approximation, network flow problems, statistics and statistical learning, as a conic optimization problem, which contains the non-negative univariate polynomial conic constraints, as well as second order and linear conic constraints, to motivate the reader for the rest of the thesis. We refer to the latter problem as general conic optimization problem.

In terms of formulation, conic optimization problems involving the non-negative univariate polynomial cones can be cast as different conic optimization problems, i.e. as the 1- Semidefinite programming problem, 2- Non-negative univariate polynomial conic

optimization problem (i.e. the original problem), and 3- Moment conic optimization problem. Formulating the original problem as a semidefinite programming problem requires embedding the non-negative univariate polynomial cone into a much larger cone, i.e. the set of semidefinite matrices, by introducing many new decision variables. This increases the size of the problem quadratically, and therefore, even solving a medium size problem of this type becomes impractical. Solving the original problem directly needs a non-symmetric interior point method, which needs an efficient barrier function available for the non-negative univariate polynomial cone. However, the known barrier function for the original cone is not efficient to the best of our knowledge. Alternatively, using a well-known duality between the non-negative polynomial cone and the moment cone, we can formulate the original problem as a moment conic optimization problem. This keeps the size of the problem unchanged, and in addition, allows us to benefit from an efficient barrier function for the moment cone. We intend to investigate these issues in more details in this work.

Finally, in terms of algorithms, we develop two fast and numerically stable algorithms based on two non-symmetric interior point methods which have been recently proposed: 1- Homogeneous self-dual predictor-corrector interior point method (proposed by Skajaa-Ye [48]), 2- Homogeneous self-dual Mehrotra predictor-corrector interior point method (proposed by Akle-Ye [2]). Our algorithms can directly be applied to the original and moment conic optimization problems. Furthermore, we extend these algorithms such that they can handle the general conic optimization problems, which contain non-symmetric cones as well as symmetric cones.

1.1 Outline of Thesis

This thesis is organized as follows: Chapter 2 provides the necessary background on general convex cones. The definition of three well-known symmetric cones, along with a number of non-symmetric cones are given, specifically, the definition of non-negative univariate polynomial cone and moment cone. In fact, the former is a subset of a more general family of cones, i.e., the cone of sum of squared functions, and the latter is a subset of a more general family of cones which are induced by the Chebyshev system.

Chapter 3 provides a brief overview of interior point methods for the conic optimization problems. We give a general overview of interior point methods for both symmetric and non-symmetric conic optimization problems, which are conic optimization over symmetric and non-symmetric cones, respectively. Then a homogeneous self-dual embedding model will be reviewed. Finally, we give a general overview of a homogeneous self-dual predictor-corrector path-following algorithm, and a Mehrotra version of this algorithm.

In Chapter 4, we formulate several interesting real world problems as the non-negative univariate polynomial conic optimizations to highlight its importance. Specifically, we formulate problems in approximation, network flow problems with time varying parameters and flow, and non-parametric estimation under shape constraints.

In Chapter 5, we investigate solving the non-negative univariate polynomial conic optimization through solving the moment conic optimization. First, we formulate this conic optimization problem as a semidefinite programming problem (SDP) to point out the major drawbacks of this formulation: 1- The SDP formulation requires working with semidefinite matrices which involve numerically unstable matrix computations, 2- Lifting the polynomial optimization problems to semidefinite programs increases the number of decision variables by a quadratic factor. Working directly on the non-negative univariate polynomial conic optimization has its own challenges, which will be investigated in Section 6. Alternatively, using the existing duality between the non-negative univariate polynomial cone and the moment cone, we can work directly on the moment conic optimization problem instead of the original problem. By doing so, first we avoid increasing the dimension of the problem quadratically. Second, we benefit from an efficient logarithmic barrier function for the moment cone. Considering this exchange of problems, we develop a fast and numerically stable interior point algorithm directly for the moment conic optimization. Our approach is based on the general non-symmetric method of Skajaa and Ye [48]. We improve our algorithm performance by adopting a variant of Mehrotras predictor-corrector method, which is based on the non-symmetric method of Akle and Ye [2]. Finally, we use the Chebyshev polynomial basis to overcome the numerical instability of the standard basis, and in addition, to

be able to use the Fast Fourier Transform (FFT) algorithm for intermediate computations. FFT algorithm can be used in calculating the gradient and Hessian of the barrier function of the moment cone. We demonstrate the effectiveness of our algorithms for several classes of polynomial optimization problems, including non-negative polynomial approximation and time-varying network flow problems. Specifically, we develop numerical experiments to compare the performance of these two algorithms, and to compare the performance of the non-symmetric algorithm with the symmetric algorithm (i.e., the interior point method for the semidefinite programming).

In Chapter 6, we investigate solving the non-negative univariate polynomial conic optimization directly by using a barrier function for the non-negative univariate polynomial cone. We consider Faybusovich's universal barrier function. He proposed this class of barrier functions for the cones induced by the Chebyshev systems. Based on this barrier function in the standard basis, the non-symmetric homogeneous self-dual model is constructed, where this time, the primal and dual problems are the non-negative univariate polynomial and the moment problems, respectively. Then, we adapt the previous non-symmetric algorithm to this model. We compare the performance of this algorithm with the one developed in Chapter 5. Finally, we investigate numerical issues of this algorithm and suggest a few remediations. Specifically, we suggest the change of basis and discuss a closed-form solution for the integrations that arise in the computation of this barrier function.

In Chapter 7, we consider a more general conic optimization, i.e., the conic optimization problems containing the non-negative univariate polynomial/moment constraints, as well as second order and linear constraints. This is a conic optimization problem which contains non-symmetric and symmetric cones. We develop a unified interior point method for this general conic problem. We investigate the Newton systems and the Schur complement, and compare them with the symmetric counterpart. Finally, we present the numerical results of the implementations.

Chapter 2

Convex Cones: Symmetric and Non-Symmetric

The purpose of this chapter and the next is to give an overview and introduce the fundamental concepts of conic optimization which are essential for this thesis. We first start by overviewing the concept of convex cones and the properties of these sets, which are the foundation of conic optimization. Generally speaking, conic optimization is minimization of a linear objective function over linear constraints, where the decision variables belong to a convex cone. The set of convex cones, in general, can be divided into two classes: 1- Symmetric, and 2- Non-Symmetric.

Symmetric cones were the main subject of research for years, since several real world applications can be formulated as a conic optimization problem using these cones. Therefore, we begin with an overview of these cones.

Recently, non-symmetric cones have attracted more attention due to the variety and applications of these cones. In particular, we consider the univariate non-negative polynomials and the moment cones.

We continue this chapter by introducing a number of notations which will be used throughout the thesis. At the end, we review the SDP representation of these cones.

2.1 Convex Cones

First, we present the definition of cone duality. Let $\mathcal{K}$ be a proper cone, and let $\langle ., . \rangle$ show the inner product.

Definition 1. *Suppose $\mathcal{K}$ is a proper cone. The dual cone of $\mathcal{K}$ is defined as:*

$$\mathcal{K}^* = \{s | \langle s, x \rangle \geq 0 \quad \forall x \in \mathcal{K}\}. \tag{2.1}$$

The following properties of the dual cone are often useful:

Proposition 1.

The followings holds true.

1. *If $\mathcal{K}$ is a cone, then the dual cone is a closed cone.*

2. *If $\bar{\mathcal{K}}$ denotes set closure, then $\bar{\mathcal{K}} = (\mathcal{K}^*)^*$. When $\mathcal{K} = (\mathcal{K}^*)^*$, we call $\mathcal{K}$ self-dual.*

3. *If $\mathcal{K}$ is a proper cone, then so is $\mathcal{K}^*$.*

Proof. Refer to any convex optimization book. □

The definition of a dual cone depends on the choice of the inner product. For a given cone, two different inner products yield different dual cones. For most of this work, the selected inner product is the Euclidean dot product.

2.2 Symmetric Cones

Symmetric cones were the subject of research for many years, where the conic optimization over these cones led to the so-called symmetric interior point methods. To define symmetric cones, we need the following.

Definition 2. *If the set of all linear maps $\mathcal{L}$ such that $\mathcal{L}\mathcal{K} = \mathcal{K}$ acts transitively on $\mathcal{K}$, then the cone $\mathcal{K}$ is called homogeneous. This means that for a homogeneous cone $\mathcal{K}$, we have:*

$$\forall x, y \in Int(\mathcal{K}): \quad \exists \mathcal{L} \;\; s.t. \;\; \mathcal{L}\mathcal{K} = \mathcal{K} \quad for\ which \quad \mathcal{L}x = y. \tag{2.2}$$

Definition 3. *A convex proper cone that is both homogeneous and self-dual is called symmetric.*

It has been shown that any symmetric cone consists of a (unique) Cartesian product of irreducible symmetric cones, of which only five exist (see e.g. [21]). These irreducible cones can therefore be thought of as a basis of all symmetric cones. Three well-known symmetric cones which are of practical interest in optimization are:

1. **Positive Orthant:**

$$\mathcal{L}^n = \{x \in \mathbb{R}^n : \quad x_i \geq 0 \quad \forall i = 1, ..., n\} \tag{2.3}$$

where the conic optimization over this cone leads to the linear programming (LP). This cone is reducible to a semidefinite cone. In fact, this is a special case of semidefinite cones.

2. **Lorentz Cone or Second Order Cone:**

$$\mathcal{S}^n = \{(x_0, \bar{x}) \in \mathbb{R}^{n+1} : \quad ||\bar{x}||_2 \leq x_0\}. \tag{2.4}$$

where the conic optimization over the direct sum of these cones leads to the second order conic programming (SOCP).

3. **Semidefinite Cone:**

$$\mathcal{SD}^n = \{X \in \mathbb{R}^{n\times n} : \quad X^T = X \text{ and } s^T X s \geq 0, \ \forall s \in \mathbb{R}^n\} \tag{2.5}$$

where the conic optimization over this cone leads to the semidefinite programming (SDP).

Although the conic optimizations over these cones, model a large variety of real world problems, many cannot be directly formulated into one of these cones.

2.3 Non-Symmetric Cones

We categorize the proper cones which are not symmetric as non-symmetric cones. Of special interest in this work are two non-symmetric cones: 1- The non-negative univariate polynomial cone and 2- The moment cone. Next, we will define these cones.

2.3.1 The Non-Negative Univariate Polynomial Cone

The cone of non-negative univariate polynomial is defined as:

Definition 4. *The cone of non-negative univariate polynomial of degree n in the standard basis over interval $[a, b]$ is defined as:*

$$\mathcal{P}^{n+}_{[a,b]} = \{x \in \mathbb{R}^{n+1} | \ \sum_{j=0}^{n} x_j t^j \geq 0, \ \forall t \in [a, b]\}. \tag{2.6}$$

From now on throughout this work, we use $\mathcal{P}^{n+}_{[a,b]}$ to show the non-negative univariate polynomial cone. We use $\mathcal{P}^{+}_{[a,b]}$ when the degree is clear from the context, and $\mathcal{P}^{n+}$ is used when the interval is understood from the context. Finally, we simply write $\mathcal{P}^{+}$ if both the degree and interval are clear.

Like any cone, the boundary and interior of the cone can be defined. In particular, the boundary of $\mathcal{P}^{n+}_{[a,b]}$, which is shown by $\partial\mathcal{P}^{n+}_{[a,b]}$, can be defined as the set of all non-negative polynomials that have at least one real root in $[a, b]$. Also, the interior of the cone is defined as the set of all non-negative polynomials that do not have any real roots in $[a, b]$, or in other words are strictly positive.

It is well-known that $\mathcal{P}^{2n+}_{\mathbb{R}} = \sum_{2n,1}$ (the cone of sum of squares of degree $2n$ and univariate). Specifically, we have:

Proposition 2.

Suppose $p \in \mathcal{P}^{2n+}_{\mathbb{R}}$. Then the following are equivalent:

1. $p \in \mathcal{P}^{2n+}_{\mathbb{R}}$.
2. $p = q^2 + r^2$ *for some polynomials* q *and* r *of degree at most* n.
3. $p \in \sum_{2n,1}$.
4. *There exists a* $X \in \mathcal{SD}^{m+1}$ *satisfying* $p_k = \sum_{i+j=k} x_{i,j}$ *for* $k = 0, ..., 2n$ *and* $i,j=0,...,m$.

Proof. See for example [44]. □

More generally, this theorem can be extended to the case of a non-negative polynomial over an interval or half line, which is covered by the next theorem.

Proposition 3.

For every polynomial of degree n,

1. $p \in \mathcal{P}^{2m+}_{[a,\infty)}$ *if and only if* $p(t) = r^2(t) + (t - a)s^2(t)$ *for some polynomials* r *and* s *of degree at most* $\lfloor n/2 \rfloor$.

2. $p \in \mathcal{P}^{n+}_{[a,b]}$ *if and only if*

$$p(t) = \begin{cases} r^2(t) + (t-a)(b-t)q^2(t), & \text{if } n = 2m, \\ (t-a)r^2(t) + (b-t)s^2(t), & \text{if } n = 2m+1, \end{cases} \tag{2.7}$$

for some polynomials r *and* s *of degree* k *and* q *of degree* $k-1$.

Proof. See for example [42]. □

It can be seen that part 2 of proposition 3 can be written in terms of $\mathcal{P}^{2n}_{\mathbb{R}}$ as:

$$\mathcal{P}^{2n}_{[a,b]} = \mathcal{P}^{2n}_{\mathbb{R}} + A\mathcal{P}^{2n-2}_{\mathbb{R}},$$
$$\mathcal{P}^{2n+1}_{[a,b]} = B\mathcal{P}^{2n}_{\mathbb{R}} + C\mathcal{P}^{2n}_{\mathbb{R}},$$

where A, B, C are linear transformations representing multiplication by polynomials $(b-t)(t-a), (b-t)$ and $(t-a)$, respectively.

2.3.2 The Moment Cone

To define the moment cone, we need the definition of the Chebyshev system:

Definition 5. *Let* $u_0(t), u_1(t), ..., u_n(t)$ *denote continuous real-valued functions for* $t \in [a,b]$. *Then we say* $\{u_0(t), u_1(t), ..., u_n(t)\}$ *is a Chebyshev system of order* n *if*

$$det \begin{bmatrix} u_0(t_0) & u_0(t_1) & \dots & u_0(t_n) \\ \vdots & \vdots & \vdots & \vdots \\ u_n(t_0) & u_n(t_1) & \dots & u_n(t_n) \end{bmatrix} > 0, \tag{2.8}$$

where $a \leq t_0 < t_1 < ... < t_n \leq b$.

Let us consider several examples:

1. The functions

$$1, t, ..., t^n \tag{2.9}$$

form a Chebyshev system on any closed interval.

2. The functions

$$T_0(t), T_1(t), ..., T_n(t) \tag{2.10}$$

where $T_i(t)$ is the i-th Chebyshev polynomial form a Chebyshev system on any closed interval.

3. The functions

$$\frac{1}{t+\alpha_0}, \frac{1}{t+\alpha_1}, ..., \frac{1}{t+\alpha_n}, \tag{2.11}$$

for $0 < \alpha_0 < \alpha_1 < ... < \alpha_n$ form a Chebyshev system for any interval such that $a + \alpha_0 > 0$.

4. The functions

$$e^{\alpha_0 t}, e^{\alpha_1 t}, ..., e^{\alpha_n t} \tag{2.12}$$

for different α_i's form a Chebyshev system on any closed interval.

5. The functions

$$1, sin(t), ..., sin(nt), cos(t), ..., cos(nt) \tag{2.13}$$

form a periodic Chebyshev system on $[0, 2\pi]$.

It has been shown in [41] that the cones induced by these Chebyshev systems are isomorphic to the cone of non-negative univariate polynomials.

Now the moment cone can be defined with respect to Chebyshev system $\{u_0(t), u_1(t), ..., u_n(t)\}$ as following:

Definition 6. *Suppose $u_0(t), u_1(t), ..., u_n(t)$ form a Chebyshev system. Then, the moment cone over interval $[a, b]$ is defined as:*

$$\mathcal{M}^n_{[a,b]} = \{c \in \mathbb{R}^{n+1} : \ \ c_i = \int_a^b u_i(t) dF, \ \ \forall i = 0, .., n\}, \tag{2.14}$$

where F traverses the set of all non-decreasing right continuous functions of bounded variation.

It is worth mentioning that F can be considered as a non-negative multiple of a cumulative distribution function (CDF) of a random variable with support set $[a, b]$.

For example, if we consider the standard basis, $\{1, t, ..., t^n\}$, as Chebyshev system, then the moment cone in the standard basis over $[a, b]$ is defined as:

$$\mathcal{M}^n_{[a,b]} = \{c \in \mathbb{R}^{n+1} : \ \ c_i = \int_a^b t^i dF, \ \ \forall i = 0, .., n\}. \tag{2.15}$$

Equivalently, if we define $u^n_t = \big(u_0(t), ..., u_n(t)\big)^T$, then the moment cone is defined as:

$$\mathcal{M}^n_{[a,b]} = cone\{u^n_t : \ \ \forall t \in [a, b]\}, \tag{2.16}$$

where *cone* mean the convex cone of the set.

In fact, the u^n_t's can be considered as the extreme rays of $\mathcal{M}^n_{[a,b]}$. Figure 2.1 shows the moment cone of dimension 3.

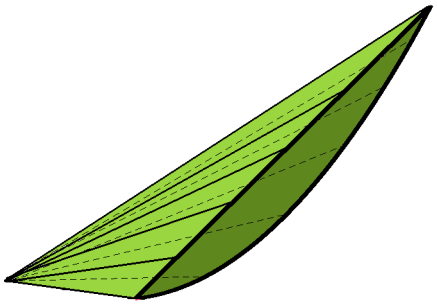

Figure 2.1: Moment cone of dimension 3

Notice that the actual dimension of $\mathcal{M}^n_{[a,b]}$ and $\mathcal{P}^{n+}_{[a,b]}$ is $n + 1$. We refer to n as the degree of the cone.

Now let us consider two properties of the non-negative polynomial cone and the moment cone which make them well-suited for optimization purposes.

Proposition 4.

$\mathcal{P}^n_{[a,b]}$ and $\mathcal{M}^n_{[a,b]}$ are proper cones.

Proof. See for example Lemma 1 in [34]. □

The second property which is the foundation of primal-dual algorithms, is about the duality of these two cones.

Proposition 5.

$$(\mathcal{M}^n_{[a,b]})^* = \mathcal{P}^{n^+}_{[a,b]}.$$

Proof. See for example [56]. □

From the previous duality, it can be seen that:

$$(A\mathcal{P}^{n^+}_{[a,b]})^* = A^{-T}(\mathcal{P}^{n^+}_{[a,b]})^* = A^{-T}\mathcal{M}^n_{[a,b]},$$

where A is an invertible matrix.

For more about the properties of $\mathcal{P}^n_{[a,b]}$ and $\mathcal{M}^n_{[a,b]}$, refer to [56].

2.3.3 Other Non-Symmetric Cones

Recently, non-symmetric cones are getting more attention. We mention a few other examples of non-symmetric cones and their dual, and refer the reader to the references for more properties and applications of these cones.

Definition 7. *The p-cone $\mathcal{K}_p \subset \mathbb{R}^3$ is defined as:*

$$\mathcal{K}_p = \{(x_0, \bar{x}) : \quad ||\bar{x}||_p \leq x_0\}, \tag{2.17}$$

where $||.||_p$ is the p-norm.

The dual cone of the p-cone is given by:

$$(\mathcal{K}_p)^* = \{(x_0, \bar{x}) : \quad ||\bar{x}||_q \leq x_0\}, \tag{2.18}$$

where $\frac{1}{p} + \frac{1}{q} = 1$. See more about these cones in [15].

Definition 8. *The exponential cone $\mathcal{K}_e \subset \mathbb{R}^3$ is defined as:*

$$\mathcal{K}_e = cl\{(x, y, z) : \quad z > 0, \ \ exp(\frac{x}{z}) \leq \frac{y}{z}\}. \tag{2.19}$$

The dual cone of the exponential cone is given by:

$$\mathcal{K}^*_e = cl\{(u, v, w) : \quad u < 0, \ \ exp(\frac{w}{u}) \leq -\frac{ev}{u}\}, \tag{2.20}$$

where $e = exp(1)$. See more about these cones in [18] and [15].

Definition 9. *For given* $\alpha = (\alpha_1, ..., \alpha_m)$ *with* $\sum_{i=1}^{m} \alpha_i = 1$ *and* $\alpha_i > 0$ *for* $i = 1, .., m$ *the power cone is defined as:*

$$\mathcal{K}_\alpha = \{(x, y) \in \mathbb{R}_+^m \times \mathbb{R} : \quad |y| \leq x_1^{\alpha_1} ... x_m^{\alpha_m}\}. \tag{2.21}$$

The dual cone of the power cone is given by:

$$(\mathcal{K}_\alpha)^* = \{(u, v) \in \mathbb{R}_+^m \times \mathbb{R} : \quad |v| \leq \left(\frac{u_1}{\alpha_1}\right)^{\alpha_1} ... \left(\frac{u_m}{\alpha_m}\right)^{\alpha_m}\} \tag{2.22}$$

See more about these cones in [15].

Definition 10. *The geometric cone* $\mathcal{G}^n$ *is defined as:*

$$\mathcal{G}^n = \{(x, \theta) \in \mathbb{R}_+^2 \times \mathbb{R}_+ : \quad \sum_{i=1}^{n} exp(-\frac{x_i}{\theta}) \leq 1\}. \tag{2.23}$$

The dual cone of the geometric cone is given by:

$$(\mathcal{G}^n)^* = \{(x^*, \theta^*) \in \mathbb{R}_+^2 \times \mathbb{R}_+ : \quad \theta^* \geq \sum_{i|x_i^*>0} x_i^* \log \frac{x_i^*}{\sum_{i=1}^n x_i^*}\}. \tag{2.24}$$

See more about these cones in [18] and [15].

Definition 11. *The extended geometric cone* $\mathcal{G}_2^n$ *is defined as:*

$$\mathcal{G}_2^n = \{(x, \theta, \kappa) \in \mathbb{R}_+^2 \times \mathbb{R}_+ \times \mathbb{R}_+ : \quad \theta \sum_{i=1}^{n} exp(-\frac{x_i}{\theta}) \leq \kappa\}. \tag{2.25}$$

The dual cone of the extended geometric cone is given by:

$$(\mathcal{G}_2^n)^* = \{(x^*, \theta^*, \kappa^*) \in \mathbb{R}_+^2 \times \mathbb{R}_+ \times \mathbb{R}_+ : \quad \theta^* \geq (\sum_{0<x_i^*<\kappa^*} x_i^* log\frac{x_i^*}{\kappa^*} - x_i^*) - \sum_{i=1}^{n} \kappa^*\}. \tag{2.26}$$

For more about this cone and its properties see Section 6 in [18].

As we have mentioned, non-symmetric cones often can be embedded into a symmetric cone, which usually results in a much larger dimension space. Next, we are going to show the semidefinite representation of $\mathcal{P}_{[a,b]}^{n^+}$ and $\mathcal{M}_{[a,b]}^n$. First, we need to introduce notations that can be used to show the results in a matrix format and simplify the formulations throughout this work.

2.4 Notation

Definition 12. *The i-th elementary Hankel and Toeplitz matrices in $\mathbb{R}^{(n+1)\times(n+1)}$ are defined as the following matrices:*

$$([H_i]_{(k,l)})_{i=0}^{2n} = \begin{cases} 1, & \text{if } k+l=i, \\ 0, & \text{otherwise}, \end{cases}$$

$$([T_i]_{(k,l)})_{i=0}^{n} = \begin{cases} 1, & \text{if } |k-l|=i, \\ 0, & \text{otherwise}, \end{cases}$$

where $k, l = 0, ..., n$.

Definition 13. *The Hankel operator is defined as:*

$$H(.) : \mathbb{R}^{2n+1} \to \mathbb{R}^{(n+1)\times(n+1)}$$

$$H(x) = \begin{pmatrix} x_0 & x_1 & \dots & x_n \\ x_1 & x_2 & \iddots & x_{n+1} \\ \vdots & \iddots & \iddots & \vdots \\ x_n & x_{n+1} & \dots & x_{2n} \end{pmatrix}$$

and the symmetric Toeplitz operator as:

$$T(.) : \mathbb{R}^{n+1} \to \mathbb{R}^{(n+1)\times(n+1)}$$

$$T(x) = \begin{pmatrix} x_0 & x_1 & \dots & x_n \\ x_1 & x_0 & \dots & x_{n-1} \\ \vdots & \ddots & \ddots & \vdots \\ x_n & \dots & x_1 & x_0 \end{pmatrix}.$$

In fact, for $x \in \mathbb{R}^{2n+1}$, we have $H(x) = \sum_{i=0}^{2n} x_i H_i$ and $T(x) = \sum_{i=0}^{n} x_i T_i$. Also, since both operators are linear, therefore, $H(x) + T(x) = (H+T)(x)$.

Notice the differences between H_i and $H(.)$, where the latter means the Hankel operator, and the former means the i-th elementary Hankel matrix. The same holds for the Toeplitz.

Definition 14. *The dehankel operator is defined as:*

$$H^*(.) : \mathbb{R}^{(n+1)\times(n+1)} \to \mathbb{R}^{2n+1}$$

$$[H^*(Y)]_{i=0}^{2n} = [\langle H_i, Y\rangle]_{i=0}^{2n} = \begin{pmatrix} y_{0,0} \\ y_{0,1} + y_{1,0} \\ \vdots \\ y_{n,n} \end{pmatrix},$$

and the detoeplitz operator is defined as:

$$T^*(.) : \mathbb{R}^{(n+1)\times(n+1)} \to \mathbb{R}^{n+1}$$

$$[T^*(Y)]_{i=0}^{n} = [\langle T_i, Y\rangle]_{i=0}^{n} = \begin{pmatrix} y_{0,0} + ... + y_{n,n} \\ 2(y_{0,1} + ... + y_{n-1,n}) \\ \vdots \\ 2y_{0,n} \end{pmatrix},$$

where $\langle ., . \rangle$ *means the matrix inner product. It is convenient to consider* T^* *as the concatenation of the* T^* *with a zero vector of dimension* n*, to make it of the same dimension of* H^**.*

In fact, H^* and T^* are the adjoint linear operators of H and T, (see [35]). Since, both operators are linear, therefore, $H^*(Y) + T^*(Y) = (H^* + T^*)(Y)$.

Also, in this work we consider the gradient of a function to be in a vector form which makes the notation easier.

2.5 Semidefinite Representation of $\mathcal{P}^n_{[a,b]}$ and $\mathcal{M}^n_{[a,b]}$

It is well-known that $\mathcal{P}^n_{[a,b]}$ and $\mathcal{M}^n_{[a,b]}$ are semidefinite representable, which is shown in the following propositions. (See [56] and [35] for a more general setting.)

The semidefinite representation of $\mathcal{P}^n_{[a,b]}$ can be shown as:

Proposition 6.

a. $\mathcal{P}^{2n^+}_{\mathbb{R}} = \{p \in \mathbb{R}^{2n+1} |\ \exists Y \succcurlyeq 0,\ s.t.\ p = H^*(Y)\}$*, where* $Y \in \mathbb{R}^{(n+1)\times(n+1)}$

b. $\mathcal{P}^{2n^+}_{[a,b]} = \{p \in \mathbb{R}^{2n+1} |\ \exists Y_1, Y_2 \succcurlyeq 0,\ s.t.\ p = H^*(Y_1) + \mathcal{A}H^*(Y_2)\}$*, where* $Y_1 \in \mathbb{R}^{(n+1)\times(n+1)}$ *and* $Y_2 \in \mathbb{R}^{n\times n}$*.*

c. $\mathcal{P}_{[a,b]}^{(2n+1)^+} = \{p \in \mathbb{R}^{2n+2} |\ \exists Y_1, Y_2 \succcurlyeq 0,\ s.t.\ p = \mathcal{B}H^*(Y_1) + \mathcal{C}H^*(Y_2)\}$, *where* $Y_1, Y_2 \in \mathbb{R}^{(n+1)\times(n+1)}$ *where* $\mathcal{A}, \mathcal{B}, \mathcal{C}$ *are:*

$$\mathcal{A} := \begin{pmatrix} -ab & a+b & -1 & & \\ & -ab & a+b & -1 & \\ & & -ab & a+b & -1 \\ & & & \ddots & \ddots & \ddots \end{pmatrix}, \tag{2.27}$$

$$\mathcal{B} := \begin{pmatrix} -a & 1 & & & \\ & -a & 1 & & \\ & & -a & 1 & \\ & & & \ddots & \ddots \end{pmatrix}, \tag{2.28}$$

$$\mathcal{C} := \begin{pmatrix} b & -1 & & & \\ & b & -1 & & \\ & & b & -1 & \\ & & & \ddots & \ddots \end{pmatrix}. \tag{2.29}$$

Proof. See for example, Theorem 17.1 part 1, Theorem 17.12 and Theorem 17.13 in [35]. □

In fact, the linear transformations $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ represent polynomial multiplications by $(b-t)(t-a)$, $(t-a)$ and $(b-t)$, respectively.

Alternately, the semidefinite representation of $\mathcal{M}_{[a,b]}^n$ can be shown as:

Proposition 7.

a. $(x_i)_{i=0}^{2n} \in \mathcal{M}_{\mathbb{R}}^{2n} \Leftrightarrow H(x) \succcurlyeq 0$.

b. $(x_i)_{i=0}^{2n} \in \mathcal{M}_{[a,b]}^{2n} \Leftrightarrow \begin{cases} H(x) \succcurlyeq 0, \\ H(\mathcal{A}x) \succcurlyeq 0. \end{cases}$

c. $(x_i)_{i=0}^{2n+1} \in \mathcal{M}_{[a,b]}^{2n+1} \Leftrightarrow \begin{cases} H(\mathcal{B}x) \succcurlyeq 0, \\ H(\mathcal{C}x) \succcurlyeq 0. \end{cases}$

Proof. See for example, Corollary 1.1 in [56], or more generally see Theorem 17.1 part 2, Theorem 17.12 and Theorem 17.13 in [35]. □

Chapter 3

Interior Point Methods

3.1 A Brief History of Interior Point Methods

Interior point methods (IPMs), primarily in the form of logarithmic barrier methods, can be traced back to Frisch [54], and the seminal work of Fiaco and MaCormick [55], where they were considering problems with nonlinear constraints. Research lost interest in these techniques in the late 1960s and 1970s, when it became apparent that the subproblems of the original nonlinear problem that needed to be solved, became increasingly ill-conditioned as the solution was being approached. During the 1970s, new techniques, like augmented Lagrangian and sequential quadratic programming methods, superseded barrier methods to the point that this technique was almost forgotten by the early 1980s. Karmarkar's paper [24] began a new chapter in the history of optimization, where he proposed a polynomial-time interior point algorithm for linear programming (LP). The importance of his algorithm was not just for the polynomiality of the algorithm, but rather because this was a bridge from LP to nonlinear optimization, where many techniques and algorithms for LP can be generalized to nonlinear optimization. Soon after Karmarkar's algorithm, rapid widespread efforts were dedicated to making IPMs as efficient as possible (see [60], [58], [59]), and generalizing IPMs to other classes of optimization problems (see [38], [13], [45]). The most efficient interior point algorithms in practice, use path-following techniques for symmetric conic optimizations, where they can benefit from the symmetry between primal and dual problems. Renegar [46] and Gonzaga [20] introduced path-following methods with an improved iteration complexity. Megiddo [29] connected the logarithmic barrier to the complementarity gap between primal and dual problems of LP. Kojima, Mizuno and

Yoshise [25] and later Monteiro and Adler [31], realized a feasible primal-dual path-following algorithm which was the most successful algorithm in practice. Mehrotra [30] introduced a remarkable higher order primal-dual logarithmic barrier for LP. Lusting [27] and Lusting et al. [28] made important contributions to the development of the implementation of an infeasible primal-dual Mehrotra predictor-corrector. The drawback to their algorithm and its implementation was that it could not detect the infeasibility and unbound status of LP. Ye et al. [53] presented a homogeneous self-dual primal-dual IPM, where the model embeds the original LP into a slightly larger LP problem that always has a solution and the algorithm is able to start from any feasible or infeasible point near the central path. The optimal solution of the embedded model can detect the infeasibility or unbounded status or can easily be converted to an optimal solution of the original model. Xu et al. [52] simplified and generalized their algorithm, and the numerical implementation of his algorithm showed that this algorithm was one of the most efficient and stable implementations of an infeasible homogeneous self-dual primal-dual predictor-corrector IPM (for more details see [7]). These algorithms have been implemented in software packages like Mosek [32] (see the details about the implementation in [8]), or the solvers in CVX [19] (see the details about the implementations in [49], [50]).

Nesterov and Nemirovski [38] investigated a more fundamental idea: "Which properties of IPMs made it so powerful for LP, and how to employ IPMs to a more general class of optimizations?" It turned out that the key property is that the barrier function should be self-concordant. This made it clear which class of optimization problems can be solved in an efficient way using IPMs, at least in theory, and that was convex optimization, since they proved that, at least in principle, any convex optimization problem could be provided with a self-concordant barrier. Soon after their work another wave of research occurred, which developed IPMs for other well-known convex optimizations. Nesterov and Todd [39] [40], suggested primal-dual IPM for convex optimization problems which admit a self-scaled barrier function and have a closed-form and computationally tractable logarithmically homogeneous self-concordant barrier function (LHSCB). But only a few convex optimization problems admit a well-defined LHSCB

function. A special subclass of these problems are the problems which contains only self-scaled cones. Three important self-scaled cones which are also symmetric, are: 1. Positive orthant, 2. Lorentz cone, and 3. Semidefinite cone. Although conic optimizations over these cones, model a large variety of real world problems, many cannot be directly formulated as one of these, for example, entropy constraints ($x \log x \leq t$), p-cone constraints ($||x||_p \leq t$), constraints in geometry ($cx_1^{\alpha_1} x_2^{\alpha_2} ... x_p^{\alpha_p} \leq 1$), exponential conic constraints $exp(\frac{x}{z}) \leq \frac{y}{z}$, power conic constraints $x_1^{\alpha} x_2^{1-\alpha} \geq -z$, single variable non-negative polynomial constraints $x \succcurlyeq_{\mathcal{P}^{n+}_{[a\ b]}} 0$, moment conic constraints $x \succcurlyeq_{\mathcal{M}^{n+1}_{[a\ b]}} 0$, etc.

Embedding these non-symmetric constraints into a symmetric conic constraint, usually introduces many new decision variables and many new constraints. In particular, since the IPMs need to compute the Hessian and to store it, increasing the dimension of the problem, increases both the running time and the memory usage. For example, as we will see later in Chapter 5, embedding a non-negative single variable polynomial constraint or a moment constraint into a semidefinite constraint requires squaring the number of decision variables.

In this chapter, we give a brief overview of IPMs for conic optimization. We start with some definitions and basic foundation. Then a general comparison of symmetric vs non-symmetric IPM will be given. Finally, we focus on an important model, i.e. homogeneous self-dual model, in a primal-dual setting. We elaborate on two important primal-dual path-following algorithms for this model, i.e., homogeneous self-dual predictor-corrector IPM and a Mehrotra version of this algorithm. We use these algorithms for our numerical implementations. The main references for this chapter are [15], [47], [18], [2], [33], [38].

3.2 Conic Optimization

An important subclass of optimization is the conic optimization. In fact, any convex optimization problem can be converted to a conic optimization problem by introducing at most one variable. Throughout this work, we consider the multi-blocks conic

optimization problem, which can be formulated as following:

$$\begin{aligned} \min \;\; & c_1^T x_1 + \cdots + c_k^T x_k \\ s.t. \;\; & A_1 x_1 + \cdots + A_k x_k = b, \\ & x_i \succcurlyeq_{\mathcal{K}_i^{n_i}} 0, \;\; i = 1, ..., k, \end{aligned} \tag{COP}$$

where $A_i \in \mathbb{R}^{m \times n_i}$, $b \in \mathbb{R}^m$, $c_i \in \mathbb{R}^{n_i}$ and $\mathcal{K}_i^{n_i}$ are proper cones of dimension n_i for $i = 1, ..., k$. To simplify the notation in the rest of the work, we use Matlab notations to concatenate matrices A_is and vectors c_is as:

$$A = [A_1, ..., A_k]$$

$$c = [c_1; ...; c_k]$$

Also, we denote $\mathcal{K} = \mathcal{K}_1^{n_1} \otimes ... \otimes \mathcal{K}_k^{n_k}$, where $\otimes$ means the sum product of the cones. Therefore, the conic optimization in a compact format is formulated as:

$$\begin{aligned} \min \;\; & c^T x \\ s.t. \;\; & Ax = b, \\ & x \succcurlyeq_{\mathcal{K}} 0, \end{aligned} \tag{3.1}$$

Proper cones possess special properties that make them well-suited for optimization. Having a well-defined dual cone is one of those properties which was briefly covered in Chapter 2. Next, we will explain another important property, i.e. possessing a well-characterized barrier function.

3.2.1 LHSCB Function

One important aspect of IPMs is the concept of barrier function, which is defined as:

Definition 15. *Let $\mathcal{C}$ be a closed convex set with non-empty interior. A convex function F is said to be a barrier function for $\mathcal{C}$ if*

$$dom\ F = Int(\mathcal{C}) \tag{3.2}$$

and

$$F(x) \to \infty \quad for \quad x \to \partial\mathcal{C} \tag{3.3}$$

where $\partial\mathcal{C}$ means the boundary of $\mathcal{C}$.

To define a well-suited barrier function in the case of conic optimization, we need the following definition:

Definition 16. *A closed convex function $F \in \mathcal{C}^3$ (three times continuously differentiable) with open domain is called self-concordant if*

$$|D^3F(x)[h,h,h]| \leq 2D^2F(x)[h,h]^{3/2}, \tag{3.4}$$

for all $x \in dom\ F$, and for all $h \in \mathbb{R}^n$.

In order to have a well-defined convergence analysis we need to impose one more assumption on the self-concordant barrier function F:

Definition 17. *A self-concordant barrier function F is called ν-self-concordant barrier for convex set $\mathcal{C} \subseteq \mathbb{R}^n$ if*

$$\nabla F(x)^T \nabla^2 F(x)^{-1} \nabla F(x) \leq \nu, \quad \forall x \in Int(\mathcal{C}). \tag{3.5}$$

Finally, IPMs for the conic optimization problems are associated with specific self-concordant barrier functions for cones, in particular, those that satisfy the so-called logarithmic homogeneity condition.

Definition 18. *A barrier function F is called a logarithmic homogeneous with barrier parameter ν if F satisfies:*

$$F(\tau x) = F(x) - \nu \log \tau, \quad \forall x \in Int(dom\ F),\ \ \forall \tau > 0. \tag{3.6}$$

If in addition, F is self-concordant, we call it a Logarithmically Homogeneous Self-Concordant Barrier function, or in a short notation ν-LHSCB function.

The followings are a few examples of ν-LHSCB functions for well-known cones:

- **Positive Orthant:**

$$F(x) = -\sum_{j=1}^{n} \log x_j, \quad \forall x \in \mathrm{Int}(\mathcal{L}^n) \tag{3.7}$$

  is a n-LHSCB function for $\mathcal{C} = \mathcal{L}^n$.

- **Second Order Cone:**

$$F(x) = -\log(x_0^2 - ||\bar{x}||_2^2), \quad \forall x \in \text{Int}(\mathcal{S}^n) \tag{3.8}$$

is a 2-LHSCB function for $\mathcal{C} = \mathcal{S}^n$.

- **Semidefinite Cone:**

$$F(x) = -\log\det(X), \quad \forall X \in \text{Int}(\mathcal{SD}_+^n) \tag{3.9}$$

is a n-LHSCB function for $\mathcal{C} = \mathcal{SD}_+^n$.

- **Moment Cone:** If n is an even integer number,

$$F(x) = -\log\det(H(x)), \quad \forall x \in \text{Int}(\mathcal{M}_{\mathbb{R}}^n) \tag{3.10}$$

is a n-LHSCB function for $\mathcal{C} = \mathcal{M}_{\mathbb{R}}^n$. We will see more on this in Chapter 5.

- **Positive Polynomial Cone:** If n is an odd integer number,

$$F(x) = -\frac{1}{2}\ln\det\mathcal{D}(x), \quad \forall x \in \text{Int}(\mathcal{P}_{[a,b]}^{n^+}) \tag{3.11}$$

is a n-LHSCB function for $\mathcal{C} = \mathcal{P}_{[a,b]}^{n^+}$, where $\mathcal{D}(x)$ is a skew symmetric matrix of size $(n+1)\times(n+1)$, and its (i,j)-th entry, $i,j = 0,...,n$, is defined as:

$$\mathcal{D}_{i,j}(x) = \int_a^b \frac{(j-i)t^{i+j-1}}{(x_0 + x_1 t + ... + x_n t^n)^2} dt.$$

We will see more on this in Chapter 6.

For more examples refer to [15] for p-cone and power cone, [18], [15], [2] for exponential cone, and [18] for geometric cone as well as the extended geometric cone.

A direct consequence of Definition 18 is the following properties:

$$\nabla^2 F(x)x = -\nabla F(x) \tag{3.12}$$

$$\nabla F(\tau x)x = \tau^{-1} - \nabla F(x) \tag{3.13}$$

$$\nabla^2 F(\tau x) = \tau^{-2} - \nabla^2 F(x) \tag{3.14}$$

$$x^T \nabla F(x) = -\nu \tag{3.15}$$

$$||x||_x^2 = \nu \tag{3.16}$$

where $||.||_x$ is the Hessian norm. See [38] for the proofs and further details.

Next, we will describe an important property of convex cones, specifically in the case of symmetric cones.

3.2.2 Conjugate Barrier Function

In Section (2.1) we have seen that for any given proper cone $\mathcal{K}$, there is an associated proper cone which is the dual cone. In the case of symmetric cones, the dual cone is the cone itself, but this is not true anymore in the case of non-symmetric cones. The same analogy exists for the ν-LHSCB function. This can be expressed in terms of the conjugate barrier function, which is defined below.

Definition 19. *Given a barrier function F for convex set $\mathcal{C}$, the function*

$$F^*(s) = \sup_{x \in \mathcal{C}} \{-s^T x - F(x)\} \tag{3.17}$$

is called conjugate barrier function of F.

Since the set of cones is a subset of convex sets, then the same definition holds for the conjugate barrier of cones. But the advantage of cones over a convex set, is that, if F is a barrier function for $\mathcal{K}$, then F^* is also a barrier function for $\mathcal{K}^*$. Furthermore, if F is a ν-LHSCB function for $\mathcal{K}$, then F^* is also a ν-LHSCB function for $\mathcal{K}^*$.

The following equations connect the two barrier functions (see [38] for proofs):

$$-\nabla F(x) \in \text{Int}(\mathcal{K}^*) \tag{3.18}$$

$$-\nabla F^*(s) \in \text{Int}(\mathcal{K}) \tag{3.19}$$

$$\nabla F(-\nabla F^*(s)) = -s \tag{3.20}$$

$$\nabla F^*(-\nabla F(x)) = -x \tag{3.21}$$

$$\nabla^2 F(-\nabla F^*(s)) = (\nabla^2 F^*(x))^{-1} \tag{3.22}$$

$$\nabla^2 F^*(-\nabla F(x)) = (\nabla^2 F^*(s))^{-1} \tag{3.23}$$

$$\nabla^2 F(x)x = -\nabla F(x) \tag{3.24}$$

$$\nabla^2 F^*(s)s = -\nabla F^*(s) \tag{3.25}$$

$$x^T \nabla F(x) = -\nu \tag{3.26}$$

$$s^T \nabla F(s) = -\nu \tag{3.27}$$

$$\langle x, \nabla^2 F(x) x \rangle = \nu \tag{3.28}$$

$$\langle s, \nabla^2 F^*(s) s \rangle = \nu \tag{3.29}$$

$$\langle \nabla F(x), \nabla^2 F(x)^{-1} \nabla F(x) \rangle = \nu \tag{3.30}$$

$$\langle \nabla F^*(s), \nabla^2 F^*(s)^{-1} \nabla F^*(s) \rangle = \nu \tag{3.31}$$

It can be shown that if $\mathcal{K} = \mathcal{K}_1 \otimes ... \otimes \mathcal{K}_k$, each with ν_i-LHSCB function F_i, then $\sum_{i=1}^k F_i$ is a ν-LHSCB function for $\mathcal{K}$ with $\nu = \sum_{i=1}^k \nu_i$.

Also, the local Hessian-norms on $\mathcal{K}$ and $\mathcal{K}^*$ can be defined as:

$$||u||_x = ||\nabla^2 F(x)^{1/2} u||_2 \quad \text{for } x \in \text{Int}(\mathcal{K}) \tag{3.32}$$

$$||u||_s^* = ||\nabla^2 F^*(s)^{1/2} u||_2 \quad \text{for } s \in \text{Int}(\mathcal{K}^*) \tag{3.33}$$

$$||u||_x^* = ||\nabla^2 F(x)^{-1/2} u||_2 \quad \text{for } x \in \text{Int}(\mathcal{K}) \tag{3.34}$$

The connection between F and F^* is very important, particularly in the case of symmetric cones, and therefore, the conic optimization containing these cones. In fact, because of this symmetry between $\mathcal{K}$ and $\mathcal{K}^*$, and between F and F^*, all information about one cone can be transferred to information about the other cone without extra computation. This property of symmetric cones makes the primal-dual interior point methods computationally very cheap and efficient in practice. But this is not the case for non-symmetric cones. To see the connection between F and F^*, and the differences between the symmetric cones and non-symmetric cones, we consider the following two examples.

- **Symmetric Cone:** Positive Orthant ($\mathcal{K} = \mathbb{R}^n_+$)

  As we have seen previously, the LHSCB function for this cone is defined as:

$$F(x) = -\sum_{j=1}^n \log(x_j), \quad \forall x \in \text{Int}(\mathbb{R}^n_+) \tag{3.35}$$

  Therefore, using Definition 19, the conjugate barrier function is defined as:

$$F^*(s) = \sup_{x \in \mathbb{R}^n_+} \{-s^T x + \sum_{j=1}^n \log(x_j)\} \tag{3.36}$$

for any given $s \in \text{Int}(\mathcal{K}^*) = \text{Int}(\mathbb{R}^n_+)$. After some calculation we have:

$$F^*(s) - \sum_{j=1}^{n} \log(s_j) - n. \tag{3.37}$$

Therefore, F^* is the same as F except for a constant parameter n.

- **Non-Symmetric Cone:** Moment Cone ($\mathcal{K} = \mathcal{M}^{2n}_{\mathbb{R}}$)

  As we have seen, the LHSCB function for the moment cone in the standard basis over $\mathbb{R}$ is defined as:

$$F(x) = -\log\det(H(x)), \quad \forall x \in \text{Int}(\mathcal{M}^{2n}_{\mathbb{R}}). \tag{3.38}$$

  Therefore, using Definition 19, the conjugate barrier function is defined as:

$$\begin{aligned} F^*(s) &= \sup_{x \in \text{Int}(\mathcal{M}^{2n}_{\mathbb{R}})} \{-\langle x, s\rangle - F(x)\} \\ &= -\inf_{x \in \text{Int}(\mathcal{M}^{2n}_{\mathbb{R}})} \{\langle x, s\rangle + \ln\det(H(x))\} \end{aligned} \tag{3.39}$$

  for any given $s \in \text{Int}(\mathcal{K}^*) = \text{Int}(\mathcal{P}^{2n^+}_{\mathbb{R}})$. Since the function in (3.39) is convex, from the first order condition, the minimizer will be:

$$\nabla_x(x^T s - \ln\det(H(x))|_{x^*} = 0 \implies H^*(H^{-1}(x^*)) = s. \tag{3.40}$$

  Since (3.40) is a system of nonlinear equations, there does not exist a closed-form formulation for F^*. It is an interesting problem to see if there are fast and numerically stable algorithms that can compute H^*, its gradient and Hessian efficiently for the moment cone.

As we can see, computing the value of the conjugate barrier for non-symmetric cones is not straightforward in comparison with symmetric cones anymore. Therefore, when dealing with non-symmetric cones, it is preferable to work with algorithms that do not require the dual barrier function. This will be the main topic of Chapter 5.

Before explaining a more sophisticated model for the conic optimization in (3.1) and the interior point methods for that model, we can have a general comparison between the interior point method for the conic optimizations over a symmetric cone and a non-symmetric cone. We refer to them as symmetric and non-symmetric IPMs, respectively.

3.3 Symmetric vs Non-Symmetric IPM

The central path for the problem in (3.1), can be defined by the unique solution of the following parametric problem:

$$\begin{aligned} \min \quad & c^T x + \mu F(x) \\ s.t. \quad & Ax = b. \end{aligned} \tag{3.41}$$

The Lagrangian function for the parametrized problem in (3.41) is defined as:

$$\mathcal{L}(x, \lambda) = c^T x + \mu F(x) + \lambda^T (b - Ax). \tag{3.42}$$

Applying the first order optimality condition to (3.42) results in:

$$\begin{aligned} \nabla_x \mathcal{L}(x, \lambda) &= c + \mu \nabla F(x) - A\lambda = 0 \\ \nabla_\lambda \mathcal{L}(x, \lambda) &= b - Ax = 0. \end{aligned} \tag{3.43}$$

Consider the change of variable $s = -\mu \nabla_x F(x)$. Then, the KKT optimality conditions states that, x_μ is an optimal solution for (3.41) if and only if there exists (y_μ, s_μ) such that:

$$\begin{aligned} c - s_\mu - A^T y_\mu &= 0 \\ b - Ax_\mu &= 0 \\ s_\mu + \mu \nabla F(x_\mu) &= 0. \end{aligned} \tag{3.44}$$

On the other hand, using the duality theory, it is well-known that for any conic problem of the form (3.1), there exists an associated problem known as the dual problem, which in a standard form is defined as:

$$\begin{aligned} \max \quad & b^T y \\ s.t. \quad & A^T y + s = c \\ & s \succcurlyeq_{\mathcal{K}^*} 0. \end{aligned} \tag{3.45}$$

If $F^*(s)$ is the barrier function for the dual cone $\mathcal{K}^*$, then the central path for the problem in (3.45), can be defined by the unique solution of the following parametric problem:

$$\begin{aligned} \max \quad & b^T y + \mu F^*(s) \\ s.t. \quad & A^T y + s = c. \end{aligned} \tag{3.46}$$

With the same analogy as we have previously observed, the KKT optimality conditions states that (y_μ, s_μ) is an optimal solution for (3.46) if and only if there exists x_μ such that:

$$\begin{aligned} c - s_\mu - A^T y_\mu &= 0 \\ b - Ax_\mu &= 0 \\ x_\mu + \mu \nabla F^*(s_\mu) &= 0. \end{aligned} \tag{3.47}$$

Although (3.44) and (3.47) are very similar in formulation, in principle they are different since they result in different Newton directions. To understand this better consider a strictly feasible point x and (y, s) for the primal and dual problems, respectively. Then the first order approximation results in the following Newton directions for (3.44) and (3.47), respectively:

$$\begin{aligned} \Delta s - A^T \Delta y &= 0 \\ A\Delta x &= 0 \\ \Delta s + \mu \nabla^2 F(x) \Delta x &= -s - \mu \nabla F(x) \end{aligned} \tag{3.48}$$

and

$$\begin{aligned} \Delta s - A^T \Delta y &= 0 \\ A\Delta x &= 0 \\ \Delta x + \mu \nabla^2 F^*(s) &= -x - \mu \nabla F^*(s), \end{aligned} \tag{3.49}$$

where F^* is the convex conjugate of F.

When $\mathcal{K}$ is symmetric, it can be shown that the Newton direction in (3.48) and (3.49) can be defined in a symmetric form with respect to the primal and dual problems. In order to demonstrate this, we need the concept of the scaling-point which needs the definition of self-scaled barrier.

Definition 20. *A ν-self-concordant barrier F for $\mathcal{K}$ is called a ν-self-scaled barrier if for all $v \in Int(\mathcal{K})$, $\nabla^2 F(v)$ maps $Int(\mathcal{K})$ to $Int(\mathcal{K}^*)$ and*

$$F^*(\nabla^2 F(v)x) = F(x) - 2F(v) - \nu. \tag{3.50}$$

If a cone admits such self-scaled barrier, we call it a self-scaled cone.

It can be shown that if F is a self-scaled barrier for cone $\mathcal{K}$, then the conjugate barrier F^* is also a self-scaled barrier for the dual cone.

Nesterov and Todd [39] proposed the concept of scaling-point as follows:

Proposition 8.

If F is a ν-self-scaled barrier for $\mathcal{K}$, then for every $x \in Int(\mathcal{K})$ and $s \in Int(\mathcal{K}^*)$, there is a unique $w \in Int(\mathcal{K})$ such that:

$$\nabla^2 F(w)x = s. \tag{3.51}$$

Moreover,

$$\nabla^2 F(w)\nabla F^*(s) = \nabla F(x) \tag{3.52}$$

$$\nabla^2 F(w)\nabla^2 F^*(s)\nabla^2 F(w) = \nabla^2 F(x). \tag{3.53}$$

For example, the scaling-point for $\mathcal{L}^n$ and $\mathcal{SD}^n_+$, can be computed explicitly as:

- If $\mathcal{K} = \mathcal{L}^n$, then $w = (\sqrt{x_1/s_1}, ..., \sqrt{x_n/s_n})$.
- If $\mathcal{K} = \mathcal{SD}^n_+$, then $w = X^{1/2}(X^{1/2}SX^{1/2})X^{1/2}$.

To see how using the concept of scaling-point defines a symmetric Newton direction, consider the last equation of (3.48). Replacing $\mu\nabla^2 F(x)$ by $\nabla^2 F(w)$, results in:

$$\Delta s + \nabla^2 F(w)\Delta x = -s - \mu\nabla F(x). \tag{3.54}$$

Multiplying (3.54) by $\nabla^2 F(w)^{-1}$ results in:

$$\nabla^2 F(w)^{-1}\Delta s + \Delta x = \nabla^2 F(w)^{-1}(-s - \mu\nabla F(x)). \tag{3.55}$$

Let $v = -\nabla F(w) \in \text{Int}(\mathcal{K}^*)$, then taking ∇F^* from both sides of (3.55) and using (3.21) we have:

$$\nabla F^*(v) = \nabla F^*(-\nabla F(w)) = -w. \tag{3.56}$$

Now, we have

$$\begin{aligned} \nabla^2 F(-\nabla F^*(v)) = \nabla^2 F(w) \quad &\Rightarrow \\ \nabla^2 F^*(v)^{-1} = \nabla^2 F(w) \quad &\Rightarrow \\ \nabla^2 F^*(v) = \nabla^2 F(w)^{-1} \quad & \end{aligned} \tag{3.57}$$

which are applying $\nabla^2 F$ to both sides of (3.56), using (3.22) and taking the inverse.

Substituting $\nabla^2 F^*(v)$ for $\nabla^2 F(w)^{-1}$ in (3.55), then using (3.51) and (3.52), results in the following equation:

$$\nabla^2 F^*(v)\Delta s + \Delta x = -x - \mu \nabla F^*(s). \tag{3.58}$$

Comparing (3.54) and (3.58) shows that they are symmetric with respect to duality. To see this symmetry better, we consider the affine search direction of (3.54), i.e.

$$\Delta s + \nabla^2 F(w)\Delta x = -s. \tag{3.59}$$

Let $B^2 = \nabla^2 F(w)$. Then (3.59) can be written as:

$$B^{-1}\Delta s + B\Delta x = -B^{-1}s. \tag{3.60}$$

Now, if we switch the role of Δx and Δs, then B will be changed to B^{-1} according to (3.57). Therefore, the symmetry of Newton direction with respect to the primal and dual problems is clearly holding.

Nesterov and Todd [39], [40], used the concept of the scaling-point and developed an efficient symmetric primal-dual interior point method for a general symmetric conic optimization problem which contains linear, second order, and semidefinite conic constraints. These symmetric primal-dual interior point methods have been successfully implemented in SDPT3 [50], SeDuMi [49] and Mosek [32].

In general, there are two main differences between the interior point methods for symmetric cones and non-symmetric cones:

1. The conjugate barrier function in the case of symmetric cones is almost identical to the primal barrier function except for a constant, while the conjugate barrier function in the case of non-symmetric cones usually does not have a closed-form and is not easy to compute.

2. The symmetric cones are provided with a self-scaled barrier function, which results in a scaling-point. This makes the Newton direction symmetric with respect to the primal and dual cones. But this is not true anymore in case of the non-symmetric

cones. Although, Nesterov [37] generalized the concept of scaling-point for the non-symmetric cones (primal-dual lifting), the resulted Newton direction is not symmetric anymore with respect to the primal and dual cones.

Next, we will present an overview of a general model for the conic optimization problem in the form of (3.2), and the interior point algorithms for this model. We will use these algorithms in the coming sections.

3.4 Homogeneous Self-Dual Model

Generic IPMs assume that the starting point (x, y, s) is strictly feasible with respect to the primal and dual problems. This means, there exist a $x \in \text{Int}(\mathcal{K})$ such that $Ax = b$ and a $s \in \text{Int}(\mathcal{K}^*)$ such that $A^T y + s = c$. Also they assumed that both problems have a bounded optimal solution.

In practice, these assumptions are too restrictive. For example, usually a feasible point with respect to the primal and dual constraints is not directly available, and finding one might be as difficult as solving the original problem.

There are different strategies that can handle these restrictions of which two most efficient ones are:

1. Modify the algorithm to make it work with infeasible iterations, which leads to infeasible primal-dual path-following interior point methods. These algorithms have difficulty in detecting the source of infeasibility or unboundedness.

   But more efficient approach is:

2. Embed the problem into the homogeneous self-dual model, which leads to infeasible homogeneous self-dual path-following interior point methods. This approach can overcome all those restrictions.

In this work, we only consider the infeasible homogeneous self-dual path-following interior point methods.

To move forward, we need to assume that the strong duality theorem is satisfied, which guarantees the existence of an optimal solution. The strong duality says that,

if there exists $x \in \text{Int}(\mathcal{K})$ such that $Ax = b$ and $s \in \text{Int}(\mathcal{K}^*)$, $y \in \mathbb{R}^m$ such that $A^T y + s = c$, then, the primal and dual problems have an optimal solution that satisfies

$$\begin{aligned} Ax - b &= 0 \\ A^T y + s - c &= 0 \\ x \circ s &= 0 \\ x \in \mathcal{K},\ y \in \mathbb{R}^m,\ s \in \mathcal{K}^* & \end{aligned}$$

where $\circ$ is the inner product associated to the duality definition.

One can construct the homogeneous self-dual model from the primal and dual problems, which have been formulated in (3.1) and (3.45), respectively, by introducing two extra non-negative variables τ and κ, as the following:

$$\begin{array}{lrcl} \min & 0 & & \\ s.t. & Ax - b\tau & = & 0 \\ & A^T y + c\tau - s & = & 0 \\ & b^T y - c^T x - \kappa & = & 0 \end{array} \qquad \text{(HSD)}$$

$$(x, \tau) \in \mathcal{K} \times \mathbb{R}^+,\ y \in \mathbb{R}^m,\ (s, \kappa) \in \mathcal{K}^* \times \mathbb{R}^+.$$

The following theorem shows that the solution of (HSD) provides either an optimal solution for (3.1) and (3.45), or it will detect infeasibility or unboundedness of them.

Proposition 9.
Assume $(x^*, \tau^*, y^*, s^*, \kappa^*)$ solves (HSD). Then

a) $(x^*, \tau^*, s^*, \kappa^*)$ is complementary, that is, $x^* \circ s^* + \tau^* \kappa^* = 0$.

b) If $\tau^* > 0$ then x^*/τ^* and $(y^*, s^*)/\tau^*$ are optimal solution for (3.1) and (3.45), respectively.

c) If $\kappa^* > 0$, then either primal or dual is infeasible.

Proof. See [53], [48]. □

It is clear that (HSD) is self-dual (see [48] Appendix 2) and thus a path-following primal-dual interior point algorithm is suitable for finding the solution of (HSD). This follows next.

3.4.1 Path-Following IPM for HSD Model

The idea of path-following interior point algorithms is to follow the central path of the model closely from an initial point towards the optimal solution.

The central path of (HSD) is defined by the unique solution of following parametrized problem by $\gamma \in [0, 1]$:

$$\begin{aligned} Ax - b\tau &= \gamma r_p, \quad x \in \mathcal{K},\ \tau > 0 \\ A^T y + s - c\tau &= \gamma r_d, \quad s \in \mathcal{K}^*,\ \kappa > 0 \\ c^T x + b^T y - \kappa &= \gamma r_c, \end{aligned} \tag{3.61}$$

coupled with

$$\begin{cases} x \circ s = \gamma \mu e \\ \tau \kappa = \gamma \mu \end{cases} \tag{3.62}$$

if $\mathcal{K}$ is symmetric and coupled with

$$\begin{cases} s + \gamma \mu \nabla F(x) = 0 \\ \tau \kappa = \gamma \mu \end{cases} \tag{3.63}$$

if $\mathcal{K}$ is non-symmetric where

$$r_p = b\tau - Ax \tag{3.64}$$

$$r_d = c\tau - A^T y - s \tag{3.65}$$

$$r_c = \kappa - b^T y - c^T x. \tag{3.66}$$

Now to follow the central path of (HSD) closely one should approximately find the solution of the system of nonlinear equations in (3.61) coupled with (3.62) in the case

of symmetric cones, or (3.61) coupled with (3.63) in the case of non-symmetric cones, for a fixed parameter μ at each iteration.

In order to do so, suppose a given initial interior point $z = (x, \tau, y, s, \kappa)$ (which might be infeasible with respect to the primal and/or the dual constraints) in some neighborhood (we will define it later) of the central path. Using the first order approximation, the Newton search direction is defined as:

$$\begin{aligned} Adx - bd\tau &= \eta r_p \\ A^T dy + ds - cd\tau &= \eta r_d \\ c^T dx + b^T y - d\kappa &= \eta r_c \end{aligned} \tag{3.67}$$

coupled with

$$\begin{cases} Bdx + B^{-1}ds = -\gamma\mu B^{-1}\nabla F(x) - B^{-1}s \\ \kappa d\tau + \tau d\kappa = \gamma\mu - \tau\kappa \end{cases} \tag{3.68}$$

if the cone is symmetric and coupled with

$$\begin{cases} \mu\nabla^2 F(x)\Delta x + \Delta s = -\gamma\mu\nabla F(x) - s \\ \kappa d\tau + \tau d\kappa = \gamma\mu - \tau\kappa \end{cases} \tag{3.69}$$

if the cone is non-symmetric.

The parameters η and γ are in $[0, 1]$, where η is called infeasibility parameter and γ is called centering parameter. In the case of $\eta = \gamma = 1$, the search direction defined by (3.67) accompanied by (3.68) or (3.69), is equivalent to one iteration of Newton's method applied to (3.61) accompanied by (3.62) or (3.63), respectively.

After the search direction has been computed, the current point will be updated as:

$$z^+ := (x^+, \tau^+, y^+, s^+, \kappa^+) = (x, \tau, y, s, \kappa) + \alpha(dx, d\tau, dy, ds, d\kappa) \tag{3.70}$$

where $\alpha \in [0, 1]$ is a suitable step-length.

This claim has been proven that by taking this step, the infeasibility will be decreased as:

$$(r_p^+, r_d^+, r_g^+) = (1 - \alpha\eta)(r_p, r_d, r_g) \tag{3.71}$$

and the complementary gap (duality gap) will be decreased by:

$$\mu(z^+) \approx (1 - (1 - \gamma)\alpha)\mu(z) \tag{3.72}$$

where the duality gap is defined by $\mu(z) = (x^T s + \tau\kappa)/(\nu + 1)$.

Also, if $\eta = 1 - \gamma$ then (3.71) and (3.72) imply that the infeasibility and the complementary gap are both reduced at the same rate η. For proofs and more details, see [53] in the case of symmetric cones, and [48] in the case of non-symmetric cones.

The method of choosing the centering parameter, γ, distinguishes different path-following algorithms. For example

- **Short-step path-following method:** In the short-step algorithm, the centering parameter will be chosen near its largest value, i.e. 1. This usually allows to take a full step on the Newton direction, but this step is usually very short, and therefore, does not make too much progress towards the optimal solution. As a result of this step, the next point stays near the central path.

- **Long-step path-following method:** In the long-step algorithm, the centering parameter will be chosen near its smallest value, i.e. 0. This produces a larger Newton direction and makes more progress towards the optimal solution, but this step is generally infeasible, and needs to be damped such that the next point is close to the central path.

- **Predictor-corrector path-following method:** In each iteration of this algorithm one alternates between two phases, i.e. predictor and corrector. In the predictor phase a large step can be taken without being concerned about the centering, and in the corrector phase the point will move back to the close vicinity of the central path. Computational results showed that this method is more efficient compared to other methods. We explain this with more details in next section.

First, we need a definition for the neighborhood of the central path which is based on the Hessian-norm of the complementarity gap, as shown below:

$$\mathcal{N}(\beta) = \{z \in \mathcal{K} \times \mathbb{R}_+ \times \mathbb{R}^m \times \mathcal{K}^* \times \mathbb{R}_+ : \ ||\psi(x, \tau, s, \kappa, \mu(z))||^*_{x,\tau} \leq \beta\mu(z)\},$$

where for a symmetric cone

$$\psi(x,\tau,s,\kappa,\mu(z)) = \begin{pmatrix} x \circ s - \mu(z)e \\ \tau\kappa - \mu(z) \end{pmatrix} \tag{3.73}$$

and for a non-symmetric cone

$$\psi(x,\tau,s,\kappa,\mu(z)) = \begin{pmatrix} s + \mu(z)\nabla F(x) \\ \tau\kappa - \mu(z) \end{pmatrix}. \tag{3.74}$$

Next, we give an overview of the predictor-corrector algorithm.

3.4.2 Predictor-Corrector Path-Following Method

In the predictor-corrector algorithm, the centering parameter alternates between its extreme values, i.e. 0 and 1. In fact, this algorithm tries to make a balance between moving towards the optimal solution and following the central path by alternating the value of the centering parameter. The algorithm consists of the following two phases:

Predictor phase: When the centering parameter is set at its smallest value, the resulting Newton direction, which will be referred as prediction direction dz_p, is large and points towards the boundary of the feasible area. Therefore, moving the current point along the prediction direction results in a large reduction in the infeasibility and the duality gap. But the new point will be far from the central path. Figure 3.1 shows this phase.

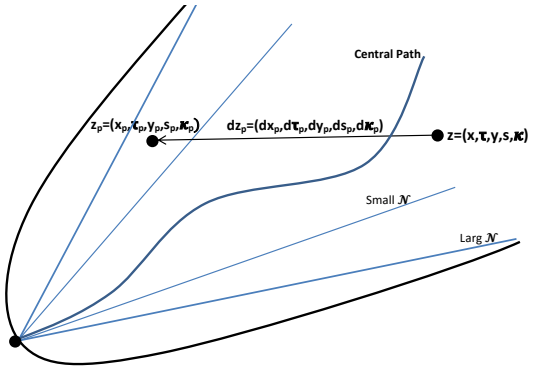

Figure 3.1: Prediction phase: In this phase the prediction direction, dz_p, is computed and the current point, z, will move along dz_p which results in $z_p = z + \alpha_p dz_p$. The updated point has smaller infeasibility and duality gap than z.

Corrector phase: Alternatively, when the centering parameter is set at its largest value, the resulting Newton direction, which will be referred as the corrector direction dz_c, points towards the central path and approximately perpendicular to it. Therefore, moving the current point along the corrector direction results in a reduction in distance from the central path but at the same time keeps the infeasibility and the duality gap unchanged. This means that by taking this direction, the next point will be once again close to the central path. Figure 3.2 shows this phase.

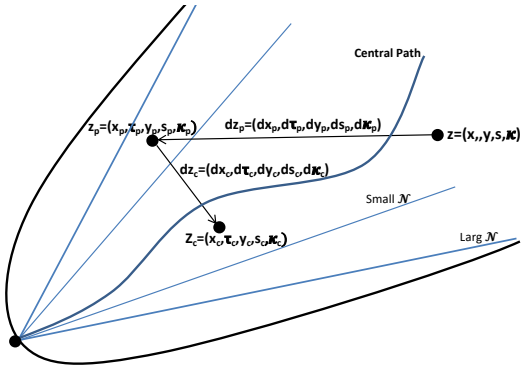

Figure 3.2: Correction phase: In this phase the corrector direction, dz_c is computed and the current point, z_p, will move along dz_c which results in $z_c = z_p + \alpha_c dz_c$. The updated point is once again near the central path and has the same infeasibility and duality gap compared to z_p.

This algorithm is given in Algorithm 1.

Algorithm 1: Predictor-corrector path-following IPM

Initialize: Initialize $0 < \beta_s < \beta_l < 1$, initial point $z = (x^0, \tau^0, y^0, s^0, \kappa^0) \in \mathcal{N}(\beta_s)$

While not converged **do**

Predictor Phase

Find the prediction direction dz_p starting from the current z

Choose the largest $\alpha_p \in [0, 1]$ such that $z + \alpha_p dz_p \in \mathcal{N}(\beta_l)$

Set $z_p = z + \alpha_p dz_p$

Corrector Phase

Repeat

Find the corrector direction dz_c starting from current z_p

Choose $\alpha_c \in [0, 1]$ such that minimizes $||\psi(z_p + \alpha_c dz_c)||^*_{x,\tau}$

Set $z_p = z_p + \alpha_c dz_c$

Until $z_p \in \mathcal{N}(\beta_s)$

Set $z = z_p$

End of while

There are different versions of the predictor-corrector path-following method that are more efficient and common in practice. For example, Mehrotra [30] designed an efficient way to update the centering parameter. The algorithm is called Mehrotra predictor-corrector algorithm which comes next.

3.4.3 Mehrotra Predictor-Corrector Path-Following Method

A practical and efficient way of choosing the centering parameter is to choose its value according to the progress which can be made in the current iteration. In other words, the centering parameter will be set to a small value if we can take a large step along the Newton direction obtained from a pre-iteration phase, and will be set to a large value if a large step cannot be taken along this direction. The pre-iteration phase is called the affine phase. The algorithm consists of the following two main phases:

Affine phase: In this phase, we are not concerned on the centering. This means that the centering parameter is set equal to zero. Then, the Newton direction will be computed, which will be referred to as affine search direction, and is shown by dz_a. Then a suitable step-length α_a will be computed, where $z + \alpha_a dz_a$ is in a large neighborhood, usually the cone itself.

Combined phase: In this phase, first we measure how much progress has been made in the affine phase, and then based on this measure, the centering parameter will be set. Then, the Newton direction will be computed, which will be referred to as combined search direction, and is shown by dz_c. Finally, the current point

will move along this direction such that the new point will be in the neighborhood of the central path.

The intuition for this approach is that if a large progress has been made in the affine phase, this indicates that the current point is close to the central path, or at least far from the boundary. Therefore, a small centering is enough. Alternatively, if a small progress has been made in the affine phase, this indicates that the current point is close to the boundary, and there is no room to move. Therefore, a large centering is needed.

The measure of progress can be computed based on different factors for different applications. Two common factors are:

1. **Duality measure:** Mehrotra [30] suggested the following heuristic for linear programming:
$$\gamma = \left(\frac{\mu_a^{k+1}}{\mu^k} \right)^3 \tag{3.75}$$
where k shows the number of iterations and
$$\mu_a^{k+1} = \frac{(x^{k+1} + \alpha_a dx_a^{k+1})^T (s^{k+1} + \alpha_a ds_a^{k+1})}{\nu + 1}. \tag{3.76}$$
In fact, this is the ratio of the duality measure obtained from the affine phase at the current iteration, and the duality measure from the previous iteration.

2. **Affine step-length:** Andersen et al. [9], suggested the following heuristic centering parameter in the case of symmetric interior point algorithms:
$$\gamma = (1 - \alpha_a)^3. \tag{3.77}$$
Recently, Akle [2] used this for non-symmetric interior point algorithms.

This algorithm is given in Algorithm 2.

Algorithm 2: Mehrotra predictor-corrector path-following IPM

Initialize: Initialize $\beta_0 \approx .99$, $0 < \beta < 1$, initial point $z = (x^0, \tau^0, y^0, s^0, \kappa^0) \in \mathcal{N}(\beta)$

While not converged **do**

Affine Phase

Find the affine search direction dz_a starting from current z

Choose the largest $\alpha_a \in [0,1]$ such that $z + \alpha_a dz_a \in \mathcal{N}(\beta_0)$

Combined Phase

Set γ based on (3.75) or (3.77)
Find the combined direction dz_c starting from current z
Choose $\alpha_c \in [0,1]$ such that minimizes $z + \alpha_c dz_c \in \mathcal{N}(\beta)$

Set $z = z + \beta_0 \alpha_c dz_c$

End of while

This algorithm is one of the most efficient and stable interior point algorithms for the conic optimization, which has been implemented in professional and commercial solver packages in the case of symmetric cones (e.g. Mosek, SeDuMi, SDPT3, etc.), and is becoming a popular algorithm in the case of non-symmetric cones.

Chapter 4

Applications

Conic optimization containing non-negative polynomial conic constraints ($\succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}}$) and moment conic constraints ($\succcurlyeq_{\mathcal{M}^{n}_{[a,b]}}$), has many applications, particularly in statistics, finance, business, economics, and engineering.

In this chapter, we consider a number of the applications of non-negative univariate polynomial conic optimization (referred to as polynomial conic optimization below, for simplicity), or of the moment conic optimization in combination with other cones, i.e. second order cones and positive orthant.

4.1 Approximation

One field where the polynomial optimization is intensively used is engineering, especially with the purpose of approximation. A key idea in approximation is given by the Weierstrass approximation theorem, which states that every continuous function defined on a closed interval $[a, b]$, can be uniformly approximated as closely as desired by a polynomial when the degree of polynomial is large enough. Now, let us consider the situation where we are given a set of real value functions, $f_i : \mathbb{R} \to \mathbb{R}$ for $i = 1, ..., k$, and the goal is to find the best envelope curve of these functions. Here, the best envelope curve means a curve which contains all other functions from below (or above) and is the closest one to these functions. If we focus on the set of polynomials, a broadly-studied family of curves with very well-behaved properties, the problem is that of finding the best envelope polynomial $x(t)$ of a fixed degree n for a given set of functions. We will define what we mean by "the best" later. In addition to the best envelope property, there might be cases where the envelope polynomial needs to have other desired properties. For example, we may want the polynomial, or even its

derivatives, to be non-negative (or negative) or/and increasing (or decreasing) or/and convex (or concave). We can formulate this with the following optimization problem, involving polynomial conic constraints:

$$\begin{aligned}
&\text{Best } x \\
&s.t.\ x(t) \le f_i(t), \quad \forall t \in [a,b], \quad i = 1, ..., k, \\
&\quad x \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0 \quad \longleftarrow \text{positivity} \\
&\quad x' \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0 \quad \longleftarrow \text{monotonicity} \\
&\quad x'' \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0 \quad \longleftarrow \text{convexity}
\end{aligned}$$

This formulation can be cast as a polynomial conic optimization problem if the constraints of type $x(t) \le f_i(t)$ can be cast as a polynomial conic constraint. There are different approaches to handle $x(t) \le f_i(t), \forall t \in [a,b], i = 1, ..., k$. One is to approximate functions f_i by a polynomial, i.e. $f_i \approx p_i$ as closely as desired. We can then consider the following substitution:

$$x(t) \le f_i(t) \quad \text{is replaced by} \quad x \preccurlyeq_{\mathcal{P}^{n+}_{[a,b]}} p_i, \tag{4.1}$$

where the latter is a polynomial conic constraint. An alternative approach is discretization, where we consider a sufficiently refined grid of points:

$$S_{[a\ b]} = \{a \le t_1 < t_2 < ... < t_p \le b\}. \tag{4.2}$$

and then consider the following substitution:

$$x(t) \le f_i(t) \quad \text{is replaced by} \quad x(t_j) \le f_i(t_j), \quad \forall t_j \in S_{a,b}, \tag{4.3}$$

where the latter is a linear constraint. Although we consider the first approach here, our methodology can be applied to the second approach as well.

Let us for now, consider the positivity constraints. We will see how other conic constraints can be easily added into the formulation.

"The best" quantification for an envelope curve, can be defined in terms of integration $\int_a^b x(t)dt$, which can be thought as the average area under polynomial x over

the interval $[a, b]$. It is clear that the integration of a polynomial can be defined as a function of the inner product of the polynomial coefficients and a constant vector, that is, $\int_a^b x(t)dt = \langle e_{[a,b]}, x\rangle$, where the constant vector $e_{[a,b]}$ depends on the basis of the polynomials and the interval. For example, in the Chebyshev polynomial basis, the i-th element of vector $e_{[a,b]}$ is $e^i_{[a,b]} = \int_a^b T_i(t)dt$, where $T_i(t)$ is the i-th Chebyshev polynomial basis. Therefore, finding the best non-negative envelope polynomial can mathematically be formulated as:

$$\begin{aligned} \max \quad & \langle e_{[a,b]}, x\rangle \\ s.t. \quad & x \preccurlyeq_{\mathcal{P}^{n+}_{[a,b]}} p_i(t), \quad i = 1, ..., k, \\ & x \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0. \end{aligned} \tag{Envelope}$$

Casting (Envelope) as a standard polynomial conic optimization problem results in the following problem:

$$\begin{aligned} \min \quad & -\langle e_{[a,b]}, x\rangle \\ s.t. \quad & \begin{bmatrix} I & I & 0 & \dots & 0 \\ I & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ I & 0 & 0 & \dots & I \end{bmatrix} \begin{bmatrix} x \\ z_1 \\ \vdots \\ z_k \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix} \\ & x, z_i \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0, \quad i = 1, ..., k, \end{aligned}$$

where I is an identity matrix of appropriate dimension.

It will be shown later (see Chapter 6) that it is usually better to find the optimal solution of the polynomial conic optimization from solving its dual problem, which we will now derive. Using the conic duality between the non-negative polynomial cone and the moment cone, the dual problem of (Envelope) can be written as:

$$\begin{aligned} \min \quad & \sum_{i=1}^{k} \langle p_i, y_i\rangle \\ s.t. \quad & \sum_{i=1}^{k} y_i \succcurlyeq_{\mathcal{M}^n_{[a,b]}} e_{[a,b]}, \\ & y_i \succcurlyeq_{\mathcal{M}^n_{[a,b]}} 0. \end{aligned} \tag{Dual-Envelope}$$

The standard conic formulation of (Dual-Envelope) is given by:

$$\begin{aligned} \min \quad & p_1^T y_1 + ... + p_k^T y_k \\ s.t. \quad & \begin{bmatrix} I & \dots & I & -I \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ s \end{bmatrix} = e_{[a,b]}, \\ & y_i, s \succcurlyeq_{\mathcal{M}^n_{[a,b]}} 0, \quad i = 1, ..., k. \end{aligned} \tag{Dual-Approx}$$

4.2 Time-Varying Network Flow Problems

Network flow problems are well-studied optimization problems with a variety of applications. Generic network flow problems are induced by a graph $G(V, E)$, with given constant input data V and E, corresponding to the sets of vertices and edges, respectively. In this section, we focus on a particular network flow problem known as the "maximum flow" problem, but the results we discuss can also be applied to other types of network flow problems, like minimum cost flow, shortest path flow, etc.

The standard maximum flow problem is given by $G(V, E, s_0, s_1, c)$, where s_0 and s_1 are source and sink nodes, and c is a constant edge capacity mapping, i.e.,

$$c : E \to \mathbb{R}^+.$$

In other words, in the maximum flow problem one looks for the maximum flow that can be pushed from the source node to the sink node, given the constant edge capacities.

Now, let us consider the case where the capacities are univariate continuous functions over $[a, b]$. Usually, the capacities are considered to be non-negative functions, thus the capacity mapping can be defined as:

$$c : E \to \mathcal{F}_{[a,b]} := \{f : \quad f : \mathbb{R} \to \mathbb{R}_+ \quad \text{and continuous}\}.$$

Then, the goal in time-varying maximum flow problem, is to find the maximum flows, i.e. $x_{i,j}(t)$, that can be pushed from source to sink with respect to the capacities,

i.e. $x_{i,j}(t) \leq f_{i,j}(t)$ for all $t \in [a,b]$. In this case, by "maximum flow" we actually mean the coefficients of the polynomials representing the maximum flow, i.e. $x_{i,j}$.

There are different approaches to convert these capacity constraints to the same type of polynomial conic constraints we illustrated in the previous section. The first approach consists of approximating each capacity function by a non-negative polynomial $p_{i,j} \approx f_{i,j}$ for all $(i,j) \in E$ with desired degree. We can then consider the following substitution:

$$x(t) \leq f_{i,j}(t) \quad \text{is replaced by} \quad x_{i,j} \preccurlyeq_{\mathcal{P}^{n+}_{[a,b]}} p_{i,j}, \quad (i,j) \in E. \tag{4.4}$$

Finally, one measure for maximality can be expressed in term of integration as we have seen in Section 4.1. Therefore, the time-varying maximum flows problem can be formulated as:

$$\begin{aligned}
\max \quad & \sum_{i|(s_0,i)\in E} \langle e_{[a,b]}, x_{s_0,i} \rangle \\
s.t. \quad & \sum_{j|(j,i)\in E} x_{j,i} - \sum_{j|(i,j)\in E} x_{i,j} =_{\mathcal{P}^{n+}_{[a,b]}} 0, \qquad \forall i \in V, \\
& x_{i,j} \preccurlyeq_{\mathcal{P}^{n+}_{[a,b]}} p_{i,j}, \quad (i,j) \in E, \\
& x_{i,j} \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0, \qquad (i,j) \in E.
\end{aligned} \tag{Max-Flow}$$

It is clear that (Max-Flow) is a polynomial conic optimization problem. The dual of (Max-Flow) is given by:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} \langle p_{i,j} y_{i,j} \rangle \\
s.t. \quad & u_i - u_j + y_{i,j} \succcurlyeq_{\mathcal{M}^n_{[a,b]}} e_{[a,b]}, \quad (i,j) \in E \\
& u_i, u_j, y_{i,j} \succcurlyeq_{\mathcal{M}^n_{[a,b]}} 0, \qquad (i,j) \in E,
\end{aligned} \tag{Dual Max-Flow}$$

which is a moment conic optimization problem.

4.3 Non-parametric Estimation Under Shape Constraints

One important and well-known concept in statistical modeling is curve fitting, where one tries to construct the best fit to a data set in two dimensions, that is, $(t_i, y_i) \in \mathbb{R}^2$ for

$i = 1, .., p$. In this context, we may be interested in finding the best fitting polynomial of a given degree such that the fitting polynomial has some special characteristics, such as being positive/negative, increasing/decreasing, convex/concave etc. In the realm of statistics, this problem is called non-parametric estimation under shape constraints. Applications of this problem can be drawn, for example, from economics and finance. For instance:

- A cost function must be increasing and convex.
- A production function must be increasing and concave (see [5]).
- A utility function must be increasing and concave.
- A call option pricing function must be decreasing and convex (see [1]).
- An arrival-rate function of a non-homogeneous Poisson process should be non-negative (see [4]).

If we consider the set of polynomials as fitting curve, then these shape-constraint problems can be in general formulated as:

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{p} (x(t_i) - y_i)^2 \\ s.t \quad & x(t) \geq 0, \ \ \forall \ t \in [a, b], \ \ \longleftarrow \text{positivity} \\ & x'(t) \geq 0, \ \ \forall \, t \in [a, b], \ \ \longleftarrow \text{monotonicity} \\ & x''(t) \geq 0, \ \ \forall t \in [a, b], \ \ \longleftarrow \text{convexity or concavity} \end{aligned}$$

where x' and x'' are the first and the second derivative of x, respectively. Clearly this can be cast as a conic optimization problem, containing non-negative polynomial and

second order conic constraints, as shown below:

$$\begin{aligned} \min \quad & r \\ s.t \quad & \begin{pmatrix} r \\ Vx - y \end{pmatrix} \succcurlyeq_{\mathcal{S}} 0, \\ & x \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0, \\ & x' \succcurlyeq_{\mathcal{P}^{(n-1)+}_{[a,b]}} 0, \\ & x'' \succcurlyeq_{\mathcal{P}^{(n-2)+}_{[a,b]}} 0, \end{aligned} \tag{Shaped-Estim}$$

where V is the Vandermonde matrix ($V_{i,j} = t_i^j$), y is the vector of observations, and $\succcurlyeq_{\mathcal{S}} 0$ represents a second order conic constraint.

For the sake of simplicity, let us for the time being consider only the positivity constraints, and rewrite (Shaped-Estim) as:

$$\begin{aligned} \min \quad & r \\ s.t \quad & \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} \succcurlyeq_{\mathcal{S}} \begin{pmatrix} 0 \\ y \end{pmatrix} \\ & x \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0, \end{aligned} \tag{Pos-Estim}$$

which is a conic optimization problem with second order and non-negative polynomial conic constraints. The dual of (Pos-Estim) can be written as:

$$\begin{aligned} \max \quad & \begin{pmatrix} 0 \\ y \end{pmatrix}^T s_{\mathcal{S}} \\ s.t. \quad & \begin{pmatrix} 1 & 0 & 0 \\ 0 & V^T & I \end{pmatrix} \begin{pmatrix} s_{\mathcal{S}} \\ s_{\mathcal{M}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ & s_{\mathcal{S}} \succcurlyeq_{\mathcal{S}} 0,\ s_{\mathcal{M}} \succcurlyeq_{\mathcal{M}^n_{[a,b]}} 0. \end{aligned} \tag{Dual-Pos-Estim}$$

To include constraints involving derivatives of different order to (Pos-Estim) or

(Dual-Pos-Estim), we may consider the first order differential operator:

$$D = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & n \end{pmatrix} \tag{4.5}$$

and the second order differential operator:

$$DD = \begin{pmatrix} 0 & 0 & 2 & 0 & \dots & 0 \\ 0 & 0 & 0 & 6 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & 0 & \dots & n(n-1) \end{pmatrix}, \tag{4.6}$$

which give us the following equalities:

$$x' \succcurlyeq_{\mathcal{P}^{(n-1)^+}_{[a,b]}} 0 \quad \Leftrightarrow \quad Dx \succcurlyeq_{\mathcal{P}^{(n-1)^+}_{[a,b]}} 0 \tag{4.7}$$

$$x'' \succcurlyeq_{\mathcal{P}^{(n-1)^+}_{[a,b]}} 0 \quad \Leftrightarrow \quad DDx \succcurlyeq_{\mathcal{P}^{(n-1)^+}_{[a,b]}} 0 \tag{4.8}$$

4.4 Non-parametric Estimation Under Shape Constraints with Splines

In practice, it is common to use splines instead of a polynomial for estimation. With splines, the interval on which the estimation is computed is divided into k sub-intervals, and non-parametric estimation under shape constraints is then carried out in each of these sub-intervals, possibly under smoothing constraints.

Let us consider a two-dimensional data set $\{(t_j, y_j)\} \in \mathbb{R}^2$ for $j = 1, .., l$, with $t_j \in [a, b]$. Let us consider a k-partition of $[a, b]$

$$[a, b] = \cup_{i=1}^{k} [c_i, d_i] \tag{4.9}$$

where the number of data points in each $[c_i, d_i]$ is l_i. For the sake of simplicity, let us for the time being only consider the positivity constraints, and formulate non-parametric

estimation under shape constraints with splines as:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{k}\sum_{j=1}^{l_i}(x_i(t_{i,j}) - y_{i,j})^2 \\
s.t. \quad & x_i(d_i) - x_{i+1}(d_i) = 0, && i = 1, ..., k-1 \\
& x_i'(d_i) - x_{i+1}'(d_i) = 0, && i = 1, ..., k-1 \\
& x_i''(d_i) - x_{i+1}''(d_i) = 0, && i = 1, ..., k-1 \\
& x_i \succcurlyeq_{\mathcal{P}^{n+}_{[c_i,d_i]}} 0, && i = 1, ..., k.
\end{aligned} \tag{4.10}$$

The only thing we should take care of, to convert this problem into a standard polynomial optimization, is to shift each sub-interval $[c_i, d_i]$ to $[a, b]$, and consequently shift each piece of polynomial $x_i(t)$ from $[c_i, d_i]$ to $[a, b]$. This implies finding the matrix operator $\mathcal{A}^{[a,b]}_{[c_i,d_i]} \in \mathbb{R}^{(n+1)\times(n+1)}$ that shifts $x_i \in \mathcal{P}^{n+}_{[c_i,d_i]}$ to $x_{s_i} \in \mathcal{P}^{n+}_{[a,b]}$, that is:

$$\mathcal{P}^{n+}_{[a,b]} = \mathcal{A}^{[a,b]}_{[c_i,d_i]}\mathcal{P}^{n+}_{[c_i,d_i]}. \tag{4.11}$$

To find $\mathcal{A}^{[a,b]}_{[c_i,d_i]}$, we first let $t \in [c_i, d_i]$, and define t_{s_i} as:

$$t_{s_i} : [c_i, d_i] \longrightarrow [a, b] \tag{4.12}$$

$$t_{s_i} = \frac{a-b}{c_i - d_i}t + \frac{c_i b - d_i a}{c_i - d_i}. \tag{4.13}$$

Then we define t in terms of t_{s_i} as:

$$t = \frac{d_i - c_i}{b-a}t_{s_i} + \frac{c_i b - d_i a}{b-a}. \tag{4.14}$$

If $x_i(t) = \sum_{i=0}^{n} p_i t^i$, then

$$x_{s_i}(t_s) = x_i(t) = \sum_{i=0}^{n} p_i(\frac{d_i - c_i}{b-a}t_{s_i} + \frac{c_i b - d_i a}{b-a})^i \tag{4.15}$$

$$= \sum_{i=0}^{n} p_i \sum_{j=0}^{i}\binom{i}{j}(\frac{c_i b - d_i a}{b-a}))^j(\frac{d_i - c_i}{b-a})^{i-j}t_{s_i}^{i-j}. \tag{4.16}$$

Therefore, the $(i-j, i)$-th entry of $\mathcal{A}^{[a,b]}_{[c_i,d_i]}$ is defined as:

$$[\mathcal{A}^{[a,b]}_{[c_i,d_i]}]_{i-j,i} = \binom{i}{j}(\frac{c_i b - d_i a}{b-a}))^j(\frac{d_i - c_i}{b-a})^{i-j} \quad \text{for } i = 0, ..., n, j = 0, ..., i. \tag{4.17}$$

Back to the problem in (4.10), we can use (4.11) to obtain the following change of variable:

$$x_i = \left(\mathcal{A}^{[a,b]}_{[c_i,d_i]}\right)^{-1} x_{s_i}. \tag{4.18}$$

Now, using (4.7) and (4.8), and the change of variable in (4.18), the problem in (4.10) can be written as:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{k} r_i \\
s.t. \quad & \begin{pmatrix} 1 & 0 \\ 0 & V_i \left(\mathcal{A}^{[a,b]}_{[c_i,d_i]}\right)^{-1} \end{pmatrix} \begin{pmatrix} r_i \\ x_{s_i} \end{pmatrix} \succcurlyeq_{\mathcal{S}_{l_i+1}} \begin{pmatrix} 0 \\ y_i \end{pmatrix}, && \text{for } i = 1, ..., k, \\
& \left(\mathcal{A}^{[a,b]}_{[c_i,d_i]}\right)^{-1} x_{s_i}(d_i) - \left(\mathcal{A}^{[a,b]}_{[c_{i+1},d_{i+1}]}\right)^{-1} x_{s_{i+1}}(d_i) = 0, && \text{for } i = 1, ..., k, \\
& D\left(\mathcal{A}^{[a,b]}_{[c_i,d_i]}\right)^{-1} x_{s_i}(d_i) - D\left(\mathcal{A}^{[a,b]}_{[c_{i+1},d_{i+1}]}\right)^{-1} x_{s_{i+1}}(d_i) = 0, && \text{for } i = 1, ..., k, \\
& DD\left(\mathcal{A}^{[a,b]}_{[c_i,d_i]}\right)^{-1} x_{s_i}(d_i) - DD\left(\mathcal{A}^{[a,b]}_{[c_{i+1},d_{i+1}]}\right)^{-1} x_{s_{i+1}}(d_i) = 0, && \text{for } i = 1, ..., k, \\
& x_{s_i} \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0, && \text{for } i = 1, ..., k.
\end{aligned}$$

(N.P.E. Spline)

which is a conic optimization problem with linear, second order, and non-negative polynomial conic constraints.

Chapter 5

Moment Conic Optimization

In this chapter we consider solving the non-negative univariate polynomial conic optimization (uPCO) via the univariate moment conic optimization (uMCO) in a primal-dual setting which is suitable for primal-dual interior point method algorithms. The reason for this is that solving a non-negative univariate polynomial conic optimization directly by IPM using its barrier function (see Section 5.4.1), is not computationally efficient at least not with the available methodologies. We investigate this in more details in Chapter 6. Also, it is well-known that the non-negative univariate polynomial cone is semidefinite representable. Therefore, a uPCO can be cast as a SDP. But this is not desirable because the SDP is extremely ill-conditioned and increases the size of the problem quadratically. These drawbacks make solving a uPCO of a small size impossible.

Alternatively, instead of solving a uPCO directly, we can use the existing duality between the non-negative univariate polynomial cone and the univariate moment cone and solve a univariate moment conic optimization problem which can be solved more efficiently. Then, the optimal solution of the uPCO can be obtained from the optimal solution of the uMCO through duality. To solve a uMCO, we can use the well-known semidefinite reprehensibility of the moment cone and cast the uMCO as a SDP. But this is not desirable because again the SDP is (i) extremely ill-conditioned and (ii) quadratically increases the size of the problem.

One remedy for the first issue is to use an orthogonal change of basis for which we use the Chebyshev polynomial basis. Through this change of basis, problems of a much larger size can be solved. Also, one remedy for the second issue is to use a non-symmetric IPM that can be applied directly to the uMCO which keeps the size of

the original problem unchanged.

Therefore, to find the optimal solution of the uPCO, we consider solving a primal-dual model, where the primal problem is a uMCO and the dual problem is a uPCO where the computation is carried out on the moment cone. By doing so, first we avoid increasing the dimension of the problem quadratically. Second, we benefit from an efficient logarithmic barrier function for the moment cone in both bases (i.e. standard and Chebyshev).

Based on the rationale explained above, we develop a fast and numerically stable non-symmetric primal-dual interior point algorithm for the pair of uMCO and uPCO problems. Our non-symmetric algorithm is based on the general non-symmetric algorithm of Skajaa and Ye [48]. We improve the performance of our algorithm by adopting a variant of the Mehrotras predictor-corrector method which is based on the non-symmetric algorithm of Akle and Ye [2].

From now on, we drop "univariate" whenever possible, for brevity.

5.1 Non-Negative Polynomial and Moment Conic Optimizations

We consider the conic optimization problem in the standard form as follows:

$$\begin{aligned} \min \;\; & c_1^T x_1 + \cdots + c_k^T x_k \\ s.t. \;\; & A_1 x_1 + \cdots + A_k x_k = b, \\ & x_i \succcurlyeq_{\mathcal{K}_i^{n_i}} 0, \;\; i = 1, ..., k, \end{aligned} \tag{COP}$$

where $A_i \in \mathbb{R}^{m \times n_i}$, $b \in \mathbb{R}^m$, $c_i \in \mathbb{R}^{n_i}$, and $\mathcal{K}_i^{n_i}$ is either of the following two non-symmetric cones:

1. Non-negative polynomial cone of dimension $n_i + 1$ over interval $[a, b]$ which is shown by $\mathcal{P}_{[a,b]}^{n_i^+}$.

2. Moment cone of dimension $n_i + 1$ over interval $[a, b]$ which is shown by $\mathcal{M}_{[a,b]}^{n_i}$.

When $\mathcal{K}_i^{n_i} = \mathcal{P}_{[a,b]}^{n_i^+}$ for $i = 1, ..., k$, we refer to (COP) as the *non-negative univariate polynomial conic optimization problem* (uPCO). This is the original problem which we

want to solve. Alternatively, when $\mathcal{K}_i^{n_i} = \mathcal{M}_{[a,b]}^n$ for $i = 1, ..., k$, we refer to (COP) as the *moment univariate conic optimization problem* (uMCO).

One approach to solve uPCO or uMCO is to cast these problems as SDP, as explained in the following section.

5.2 SDP Formulation of uPCO and uMCO in the Standard Basis

It was mentioned earlier that the non-negative polynomial and the moment cones are semidefinite representable. Therefore, using proposition 6, one can cast the uPCO as a SDP, depending on a particular case of interest. For example, in the case where $s \in \mathcal{P}_{\mathbb{R}}^{2n^+}$, we have:

$$\begin{aligned} (uPCO)\min\ & c^T s && \Rightarrow\ (SDP)\min\ && H(c)\bullet Y \\ s.t.\ & As = b, && s.t.\ && H(a_i)\bullet Y = b_i,\ i = 1, .., m, \\ & s \succcurlyeq_{\mathcal{P}_{\mathbb{R}}^{2n^+}} 0, && && Y \succcurlyeq 0, \end{aligned}$$

where $\bullet$ means the inner product and a_i is the i-th row of matrix A. It is clear that the optimal solution of the uPCO is $s^* = H^*(Y^*)$, where Y^* is the optimal solution of the SDP.

Alternatively, using proposition 7, one can cast the uMCO as a SDP, depending on a particular case of interest. For example, for the case where $x \in \mathcal{M}_{\mathbb{R}}^{2n}$, we have:

$$\begin{aligned} (uMCO)\min\ & c^T x && \Rightarrow\ (SDP)\min\ && c^T x \\ s.t.\ & Ax = b, && s.t.\ && Ax = b, \\ & x \succcurlyeq_{\mathcal{M}_{\mathbb{R}}^{2n}} 0, && && H(x) \succcurlyeq 0. \end{aligned}$$

5.3 Drawbacks of SDP Formulation and Remedies

Formulating the non-negative polynomial and moment conic optimizations as a SDP is problematic at least for two main reasons. These will be investigated in more details along with the proposed remedy for each issue in Sections 5.3.1 and 5.3.2.

5.3.1 Ill-Conditioned Issue and Chebyshev Change of Basis

The SDP formulation of the uMCO or uPCO is extremely ill-conditioned. This is due to working with Hankel matrices in the SDP formulation. It is well-known that the condition number of Hankel matrices is exponential in the matrix dimension. We have experimented that with this approach, even solving a problem of dimension $n \geq 50$ with the standard SDP solvers such as Mosek, SeDuMi or SDPT3, failed to converge since the automatic SDP solvers attempt to solve SDP problems with standard methods such as Cholesky factorization.

To avoid the ill-conditioned computations, we use a more suitable basis rather than the standard $\{1, t, t^2, ..., t^n\}$. Orthogonal polynomials of various kinds are an obvious alternative. We choose Chebyshev polynomials, since in addition to numerical stability, it is possible to carry out certain parts of the computations using the Fast Fourier Transform (FFT) and the FFT inverse in a straightforward manner. Chebyshev polynomials of the first kind are defined as:

$$T_k(t) = cos(k\ arccos(t))$$

for $-1 \leq t \leq 1$. These polynomials form a basis for the linear space of polynomials of degree at most n. So we can make a change of basis and represent a polynomial $p(t)$ as:

$$p(t) = p_0 + p_1 t + \ + p_n t^n = q_0 + q_1 T_1(t) + \ + q_n T_n(t),$$

so that the new "coefficients" are $q = (q_0, q_1, ..., q_n)^T$. The vector q can be obtained from p by a triangular linear transformation. Indeed, a well known three term recursive representation of Chebyshev polynomials is:

$$T_0(t) = 1, \ \ T_1(t) = t,$$

$$T_{n+1}(t) = 2tT_n(t) - T_{n-1}(t).$$

The non-negative polynomials and moment cones in the Chebyshev polynomial basis

over $[-1,1]$ are defined as:

$$\mathcal{P}_{Ch}^{n+} = \{x \in \mathbb{R}^{n+1} |\ \sum_{j=0}^{n} x_j T_j(t) \geqslant 0,\ \forall t \in [-1,1]\}, \tag{5.1}$$

$$\mathcal{M}_{Ch}^{n} = Cone\{(T_0(t), T_1(t), ..., T_n(t))^T |\ \forall t \in [-1,1]\}. \tag{5.2}$$

We fixed the interval to $[-1,1]$ to simplify the formulation but this is not restrictive and any interval can be considered. Through our experiments, we observed that the interval $[-1,1]$ is the best in terms of computations.

The semidefinite representation of the non-negative polynomial cone in the Chebyshev polynomial basis over $[-1,1]$ follows from the next lemma.

Lemma 10.

It can be shown that:

a. $\mathcal{P}_{\mathbb{R}_{Ch}}^{2n+} = \{p \in \mathbb{R}^{2n+1} |\ \exists\ Y \succcurlyeq 0,\ s.t.\ p = \frac{1}{2}(H+T)(Y)\}$,

b. $\mathcal{P}_{Ch}^{2n+} = \{p \in \mathbb{R}^{2n+1} |\ \exists\ Y_1, Y_2 \succcurlyeq 0,\ s.t.\ p = \frac{1}{2}(H+T)^*(Y_1) + \frac{1}{2}\mathcal{A}_{Ch}^T(H+T)^*(Y_2)\}$,

c. $\mathcal{P}_{Ch}^{(2n+1)^+} = \{p \in \mathbb{R}^{2n+2} |\ \exists\ Y_3, Y_4 \succcurlyeq 0,\ s.t.\ p = \frac{1}{2}\mathcal{B}_{Ch}^T(H+T)^*(Y_3) + \frac{1}{2}\mathcal{C}_{Ch}^T(H+T)^*(Y_4)\}$,

where $Y, Y_1, Y_3, Y_4 \in \mathbb{R}^{(n+1)\times(n+1)}$, $Y_2 \in \mathbb{R}^{n\times n}$ *and the linear transformations* $\mathcal{A}_{Ch}$, $\mathcal{B}_{Ch}$ *and* $\mathcal{C}_{Ch}$ *represent, respectively, Chebyshev polynomial multiplications by* $(T_0(t) - T_1(t))(T_0(t) + T_1(t))$, $(T_0(t) - T_1(t))$ *and* $(T_0(t) + T_1(t))$ *in the Chebyshev basis.*

Proof. See the Appendix. □

Alternatively, the semidefinite representation of the moment cone in the Chebyshev polynomial basis over $[-1,1]$ follows from the following lemma.

Lemma 11.

It can be shown that:

a. $x \in \mathcal{M}_{Ch}^{2n+1}$ *over* $\mathbb{R}$ *if and only if* $H(x) + T(x) \succcurlyeq 0$.

b. $x \in \mathcal{M}_{Ch}^{2n+1}$ *over* $[-1,1]$ *if and only if*

$$\begin{cases} (H+T)(x) \succcurlyeq 0, \\ (H+T)(\mathcal{A}_{Ch}x) \succcurlyeq 0. \end{cases}$$

c. $x \in \mathcal{M}_{Ch}^{2n+2}$ *over* $[-1,1]$ *if and only if*

$$\begin{cases} (H+T)(\mathcal{B}_{Ch}x) \succcurlyeq 0 \\ (H+T)(\mathcal{C}_{Ch}x) \succcurlyeq 0, \end{cases}$$

Proof. See the Appendix. □

By using the Chebyshev change of basis, a uPCO or uMCO of a much higher size can be solved through the SDP formulation. We observed that problems with blocks of dimension 3000 can be solved. The question that may arise here is, "why a polynomial of such large degree is useful?". A brief answer can be that, in many applications, increasing the degree of solution gives a better optimal solution particularly when the number of constraints is large. Refer to Section 5.9 for further details.

5.3.2 Squared Dimension Issue and Non-Symmetric IPM

Embedding a uPCO or uMCO problem into a SDP requires squaring the number of the decision variables and increases the number of constraints. Table 5.1 shows these for uPCO and uMCO problems with k blocks, and each block of dimension $m \times n_i$.

Formulation	blk	Constraints	Variable
Non-Symmetric	k	m	$\sum_{i=1}^{k} n_i + k$
SDP of uPCO	k	m	$O(k \sum_{i=1}^{k} n_i^2/2)$
SDP of uMCO	k	$m + k \sum_{i=1}^{k} n_i$	$O(k \sum_{i=1}^{k} n_i^2/2)$

Table 5.1: Dimension of SDP formulation of uPCO and uMCO

This makes solving even a medium size uPCO or uMCO problem by SDP impractical, in terms of the running time and memory storage. For example, if uPCO has 2×10^3 blocks and the degree of each block is of size 10^3, then it needs 10^9 floating point memory storage to only store the Hessian. The situation is worse in terms of the running time. As an example, it takes a PC with 32 GB RAM and 3GHz processor about 4 days to solve a problem of 200 blocks and each block of dimension 1200. Therefore, we need to think of an algorithm that maintains the dimension of the original problem unchanged. The non-symmetric interior point method achieves this. In the following sections we describe our approach to circumvent semidefinite programming and work directly with the moment cone, so that we avoid squaring the number of variables.

5.4 Barrier Function of Non-Negative Polynomial and Moment Cones

Interior point methods are accompanied with the barrier function of the primal and dual cones. Since this is a second order method, it also needs the gradient and the Hessian of the barrier function of the primal cone and possibly of the dual cone. In the case of symmetric cones, the information about the dual barrier function can be simply obtained from the primal barrier function without doing any extra computation. But this is not true anymore in the case of non-symmetric cones. We begin with the barrier function of the non-negative polynomial cone to justify the use of the moment cone.

5.4.1 Barrier Function of Non-Negative Polynomial Cone

Faybusovich [16] used the concept of the so-called universal barrier function for a convex set in a finite dimensional vector space [38], and proposed a LHSCB function for cones generated by the Chebyshev systems (see Theorems 4 and 5 in [16]). In fact, the non-negative polynomial cone is a subset of this class of cones. For example, let us consider his LHSCB function for the $\mathcal{P}^{n+}_{[a,b]}$ when n is odd, which is:

$$F(x) = \frac{1}{2} \ln \det \mathcal{D}(x), \quad \text{when } x \in \text{Int}(\mathcal{P}^{n+}_{[a,b]})$$
$$\mathcal{D}_{i,j}(x) = (j-i) \int_a^b \frac{t^{i+j-1}}{x(t)^2} dt, \quad i,j = 0, ..., n.$$

It is evident that the barrier function requires a significant number of one-dimensional integrals. Therefore, any interior point method which needs this barrier is not efficient. We will investigate this more in Chapter 6. Alternatively, there exists an efficient LHSCB function for its dual cone, i.e. moment cone, which will be discussed in the next section.

5.4.2 Barrier Function of Moment Cone, its Gradient and Hessian in Standard Basis

As was observed in proposition 7, the moment cone can be represented by a semidefinite matrix depending on the degree and the interval. Therefore, the barrier function for the moment cone is a variant of the log barrier function for the semidefinite cone, i.e.:

$$\begin{aligned}
&a.\ F(x) = -\ln\det(H(x)), && \text{when } x \in \mathrm{Int}(\mathcal{M}^{2n}_{\mathbb{R}}),\\
&b.\ F(x) = -\ln\det(H(x)) - \ln\det(H(\mathcal{A}x), && \text{when } x \in \mathrm{Int}(\mathcal{M}^{2n}_{[a,b]}),\\
&c.\ F(x) = -\ln\det(H(\mathcal{B}x)) - \ln\det(H(\mathcal{C}x)), && \text{when } x \in \mathrm{Int}(\mathcal{M}^{2n+1}_{[a,b]}).
\end{aligned}$$

Above barrier functions are LHSCB. All self-concordance properties of these barriers are inherited from the semidefinite cone barrier.

Using the above barrier functions, it can be shown that:

Lemma 12.

The gradients of the moment barrier functions in the standard basis are as follows:

$$\begin{aligned}
&a.\ \nabla_x(F(x)) = -H^*(H^{-1}(x)), && \textit{when } x \in \textit{Int}(\mathcal{M}^{2n}_{\mathbb{R}}),\\
&b.\ \nabla_x(F(x)) = -H^*(H^{-1}(x)) - \mathcal{A}^T H^*(H^{-1}(\mathcal{A}x), && \textit{when } x \in \textit{Int}(\mathcal{M}^{2n}_{[a,b]}),\\
&c.\ \nabla_x(F(x)) = -\mathcal{B}^T H^*(H^{-1}(\mathcal{B}x)) - \mathcal{C}^T H^*(H^{-1}(\mathcal{C}x)), && \textit{when } x \in \textit{Int}(\mathcal{M}^{2n+1}_{[a,b]}).
\end{aligned}$$

Proof. See the Appendix. □

Lemma 13.

The Hessian of the moment barrier functions in the standard basis are as follows:

$$\begin{aligned}
&a.\ \nabla_x^2(F(x)) = conv2(H^{-1}(x)), && \text{when } x \in Int(\mathcal{M}_{\mathbb{R}}^{2n}),\\
&b.\ \nabla_x^2(F(x)) = conv2(H^{-1}(x))+ && \\
&\qquad \mathcal{A}^T conv2(H^{-1}(\mathcal{A}x)\mathcal{A}, && \text{when } x \in Int(\mathcal{M}_{[a,b]}^{2n}),\\
&c.\ \nabla_x^2(F(x)) = \mathcal{B}^T conv2(H^{-1}(\mathcal{B}x))\mathcal{B}+ && \\
&\qquad \mathcal{C}^T conv2(H^{-1}(\mathcal{C}x))\mathcal{C}, && \text{when } x \in Int(\mathcal{M}_{[a,b]}^{2n+1}).
\end{aligned}$$

Proof. See the Appendix. □

Here, $conv2(P, Q)$ is the convolution of two bivariate polynomials, P, Q, in the standard basis, and $conv2(P) = conv2(P, P)$. In general, the input arguments of $conv2(.,.)$ are two matrices of dimension $(n+1) \times (n+1)$ and $(m+1) \times (m+1)$, which are the coefficients of the bivariate polynomials. This means that the (i, j) element of the matrix is the coefficient of $x^i y^j$ in a polynomial presentation. The result of $conv2(.)$ is a matrix of dimension $(n+m+1) \times (n+m+1)$, which is the matrix coefficient corresponding to the multiplication.

It is clear from above that, the Hessian is a $(2n+1) \times (2n+1)$ positive definite matrix. This is substantially smaller than the Hessian obtained from the direct SDP formulation where the Hessian would be a $(n+1)^2 \times (n+1)^2$ positive definite matrix.

As we have mentioned, the uMCO in the standard basis is extremely ill-conditioned. Therefore, we use the Chebyshev polynomial basis.

5.4.3 Barrier Function of Moment Cone, its Gradient and Hessian in Chebyshev Polynomial Basis

As it has been discussed earlier, the Chebyshev polynomial basis is preferable computationally. Therefore, with the same methodology mentioned for the moment cone barrier in the case of standard basis and considering Lemma 11, the LHSCB functions for the

moment cone in the Chebyshev basis can be defined as follows:

$$a.\ F(x) = -\ln\det((H+T)(x)), \qquad \text{when } x \in \mathrm{Int}(\mathcal{M}_{Ch_{\mathbb{R}}}^{2n}),$$

$$b.\ F(x) = -\ln\det((H+T)(x)) - \ln\det((H+T)(\mathcal{A}_{Ch}x)), \qquad \text{when } x \in \mathrm{Int}(\mathcal{M}_{Ch}^{2n+1}),$$

$$c.\ F(x) = -\ln\det((H+T)(\mathcal{B}_{Ch}x)) - \ln\det((H+T)(\mathcal{C}_{Ch}x)), \text{when } x \in \mathrm{Int}(\mathcal{M}_{Ch}^{2n+2}).$$

Using the above barrier functions, it can be shown that:

Lemma 14.

The gradients of the moment barrier functions in the Chebyshev basis are as follows:

$$a.\ \nabla_x(F(x)) = -(H^* + T^*)((H+T)^{-1}(x)), \qquad \textit{when } x \in \textit{Int}(\mathcal{M}_{Ch_{\mathbb{R}}}^{2n}),$$

$$b.\ \nabla_x(F(x)) = -(H^* + T^*)((H+T)^{-1}(x)) - \mathcal{A}_{Ch}^T(H^* + T^*)((H+T)^{-1}(\mathcal{A}_{Ch}x)), \qquad \textit{when } x \in \textit{Int}(\mathcal{M}_{Ch}^{2n}),$$

$$c.\ \nabla_x(F(x)) = -\mathcal{B}_{Ch}^T(H^* + T^*)((H+T)^{-1}(\mathcal{B}_{Ch}x)) - \mathcal{C}_{Ch}^T(H^* + T^*)((H+T)^{-1}(\mathcal{C}_{Ch}x)), \qquad \textit{when } x \in \textit{Int}(\mathcal{M}_{Ch}^{2n+1}).$$

Proof. See the Appendix. □

Lemma 15.

The Hessians of the moment barrier functions in the Chebyshev basis are as follows:

$$a.\ \nabla_x^2(F(x)) = \textit{conv2T}((H+T)^{-1}(x)), \qquad \textit{when } x \in \textit{Int}(\mathcal{M}_{Ch_{\mathbb{R}}}^{2n}),$$

$$b.\ \nabla_x^2(F(x)) = \textit{conv2T}((H+T)^{-1}(x)) + \mathcal{A}_{Ch}^T\textit{conv2T}((H+T)^{-1}(\mathcal{A}_{Ch}x))\mathcal{A}_{Ch}, \qquad \textit{when } x \in \textit{Int}(\mathcal{M}_{Ch}^{2n}),$$

$$c.\ \nabla_x^2(F(x)) = \mathcal{B}_{Ch}^T\textit{conv2T}((H+T)^{-1}(\mathcal{B}_{Ch}x))\mathcal{B}_{Ch} + \mathcal{C}_{Ch}^T\textit{conv2T}((H+T)^{-1}(\mathcal{C}_{Ch}x))\mathcal{C}_{Ch}, \qquad \textit{when } x \in \textit{Int}(\mathcal{M}_{Ch}^{2n+1}).$$

where conv2T(.) is the convolution of two bivariate polynomials in the Chebyshev polynomial basis (inspired from the standard basis).

Proof. See the Appendix. □

It is clear from above that, the Hessian of the moment barrier in the Chebyshev polynomial basis is again a $(2n+1)\times(2n+1)$ positive definite matrix. This is in contrast to the SDP representation where the Hessian is the Kronecker product of $(H+T)^{-1}(x)$ by itself, which is a $(n+1)^2 \times (n+1)^2$ positive definite matrix.

We refer to the gradient and Hessian of the moment barrier function, at a given point x, by $\mathbf{g}_x$ and $\mathbf{H}_x$, respectively, for all bases when there is no ambiguity.

In the next section, we discuss the efficient and numerically stable computation of these quantities.

5.5 Computation of $\mathbf{g}_x$ and $\mathbf{H}_x$

The main computational effort within each iteration of any interior point method for uMCO, is computing the Hessian matrix, $\mathbf{H}_x$, and solving a system of linear equation containing $\mathbf{H}_x$. This, in fact, is the result of applying the Newton method to minimize the objective plus the barrier function. In this section, we describe how these quantities are calculated and how the resulting Newton system is solved efficiently and in a numerically stable manner.

There are two parts in the computation of the Hessian: 1- Inverting a Hankel matrix (or Hankel+Toeplitz in the case of the Chebyshev basis), and 2- Convolution of the Hankel (or Hankel+Toeplitz in the case of the Chebyshev basis) inverse by itself.

For clarity we start with the standard basis, and then move on to the Chebyshev basis.

5.5.1 Computation of $\mathbf{g}_x$ and $\mathbf{H}_x$ in Standard Basis

- Solving a linear system of equations with a $(n+1)\times(n+1)$ Hankel matrix can be accomplished in $O(n\log^2 n)$ time and $O(n)$ space using the FFT (see [10], [51]). As a result, computing $\mathbf{g}_x = H^*(H^{-1}(x))$ requires $O(n\log^2 n)$ time.

- Convolution of two univariate polynomials of degree at most n in the standard basis, can be accomplished by the FFT or type I Discrete Cosine Transform (DCT) [17], in $O(n\log n)$ time and $O(n)$ space.

- More generally, the convolution of two bivariate polynomials of degree $n \times n$ in the standard basis can be accomplished by the convolution of two univariate polynomials of degree n^2 by reordering the power of the variables. Therefore, the convolution of two bivariate polynomials is at most $O(n^2 \log n)$ time and $O(n^2)$ space using FFT. As a results, computing the Schur complement in a non-symmetric formulation (i.e. $A\mathbf{H}_x^{-1}A^T$) can be accomplished in $O(mn^2 \log n)$ whereas this is $O(mn^3)$ in a SDP formulation.

5.5.2 Computation of $\mathbf{g}_x$ and $\mathbf{H}_x$ in Chebyshev Basis

- Solving a linear system of equations with a $(n+1) \times (n+1)$ Hankel+Toeplitz coefficient matrix can be accomplished in $O(rn^2)$ time and $O(rn)$ space by the DCT algorithm (see [10]), where r is the displacement rank of (Hankel+Toeplitz) matrix and is at most 4. As a result, in the Chebyshev basis, $\mathbf{g}_x = (H^* + T^*)(T + H)^{-1}(x)$ can be computed in $O(n^2)$. In our implementation we have used the "drsolve" package developed by Arico and Rodriguez [10] as a subroutine code to compute the inverse of the Hankel+Toeplitz matrix.

- The convolution of two polynomials of degree at most n in the Chebyshev basis can be computed either through FFT [17], or through DCT [12], in $O(n \log n)$ time and $O(n)$ space.

- The convolution of two bivariate polynomials of degree $n \times n$ in the Chebyshev basis can be reduced to the convolution of two univariate polynomials of degree at most n^2.

- We have used a version of the DCT method in our implementation based on the work of Baszenski and Tasche [12]. The algorithm follows the same pattern as the FFT: Compute the values of the two polynomials on the $2n$ equally spaced angels between zero and 2π; multiply these values term by term; and finally, interpolate the product on the $2n$ equally spaced values. In [12], it is showed that this process

can be expressed as:

$$conv2T(A,B) = \frac{4}{n^2} C_n((C_n A C_n^T) \circ (C_n B C_n^T)) C_n^T \tag{5.3}$$

where $\circ$ is the Hadamard product, $A, B, C_n \in \mathbb{R}^{(n+1)\times(n+1)}$, and C_n is:

$$C_n = (\varepsilon_{n,j} \cos \frac{ij\pi}{n})_{i,j=0}^{n},$$

with $\varepsilon_{n,0} = \frac{1}{2}$ and $\varepsilon_{n,j} = 1,\ j = 1, ..., n-1$. It can be shown that, similar to FFT, multiplying a vector of length $n+1$ by C_n, requires $O(n \log n)$ time and $O(n)$ space. Thus, the computation of the Hessian of barrier function in the Chebyshev basis can be achieved in $O(n^2 \log n)$. As a result, the Schur complement in the Chebyshev basis can be accomplished in $O(mn^2 \log n)$ whereas this is $O(mn^3)$ in a SDP formulation. This is similar to the case in the standard basis.

5.6 Non-Symmetric Homogeneous Self-Dual IPM

So far, we have observed that in contrast to the non-negative polynomial cone, the barrier of the moment cone, its gradient and Hessian, can be computed efficiently. Therefore, instead of solving the uPCO, we consider solving its dual which is a uMCO. Furthermore, to benefit from the maximum potential power of an interior point method in the conic optimization, usually a primal-dual setting is needed for which the primal and dual problems are required. Therefore, using the duality between the moment cone and the non-negative polynomial, we have the following primal-dual setting:

$$\begin{array}{lll} \min \quad c^T x & \overset{dual}{\Longleftrightarrow} & \max \quad b^T y \\ s.t. \quad A\,x = b, & & s.t. \quad A^T y + s = c \\ \qquad x \succcurlyeq_{\mathcal{M}_{[a,b]}^n} 0, & & \qquad s \succcurlyeq_{\mathcal{P}_{[a,b]}^{n+}} 0. \end{array} \tag{M-NP CO}$$

For now on, we use $\mathcal{M}$ and $\mathcal{P}^+$ to show $\mathcal{M}_{[a,b]}^n$ and $\mathcal{P}_{[a,b]}^{n+}$, when there is no ambiguity.

The first attempt towards a non-symmetric conic optimization has been made by Nesterov [37]. He proposed a non-symmetric primal-dual predictor-corrector IPM, which requires having a feasible primal and dual initial points and being able to compute the value of the primal and dual barrier function efficiently. But as has been

observed earlier, the barrier function of the non-negative polynomial cone is not efficiently computable. Therefore, his method cannot be applied to our case directly.

Recently, Skajaa and Ye [48] relieved those restrictions. They proposed a non-symmetric homogeneous self-dual primal-dual predictor-corrector IPM (HSD P-C IPM) which has the following properties:

- It does not need feasible initial points and can start from any point near the central path.
- It detects infeasibility and unboundedness if applicable.
- It does not need any information from the dual barrier function.

Even more efficiently, a Mehrotra version of that algorithm has been proposed by Akle and Ye [2]. We have developed these two algorithms for our cases. First, we review the essential parts from Chapter 3 that are needed here. The HSD problem can be constructed from (M-NP CO):

$$
\begin{aligned}
\min \quad & 0 \\
s.t. \quad & A\,x - b\tau = 0 \\
& A^T y + c\tau - s = 0 \\
& b^T y - c^T x - \kappa = 0 \qquad \text{(HSD M-NP)} \\
& \mu g_x + s = 0 \\
& \tau\kappa = \mu \\
(x,\tau) \in \mathcal{M} \times \mathbb{R}^+,\ & y \in \mathbb{R}^m,\ (s,\kappa) \in \mathcal{P}^+ \times \mathbb{R}^+.
\end{aligned}
$$

Theorem 9 guarantees that the solution of (HSD M-NP) provides either an optimal solution for (M-NP CO), or it will detect infeasibility or unboundedness of (M-NP CO).

It has been mentioned that (HSD M-NP) is self-dual (see [48] Appendix 2). Therefore, a path-following primal-dual interior point algorithm is suitable for finding its optimal solution. The idea of the path-following interior point algorithm is to approximately follow the central path of the model and stay close to it towards the optimal solution. Here, "central path" is defined as a parametric problem using the barrier

function, "approximately" uses the first order Newton's method, and to stay close to the central path, prediction and correction in each iteration are used.

Following the notations in [48], we show:

$$\bar{x} = \begin{pmatrix} x \\ \tau \end{pmatrix} \qquad \bar{s} = \begin{pmatrix} s \\ \kappa \end{pmatrix}$$

$$\bar{\mathcal{F}}(\bar{x}) = F(x) - \log \tau, \qquad \bar{\mathcal{F}}^*(\bar{s}) = F^*(s) - \log \kappa$$

$$\bar{\mathcal{K}} = \mathcal{M} \times \mathbb{R}^+, \qquad \bar{\mathcal{K}}^* = \mathcal{P}^+ \times \mathbb{R}^+, \quad \bar{\nu} = \nu + 1$$

where $F(x)$ and $F^*(s)$ are the barrier functions of the moment and non-negative polynomial cones, respectively, which depend on the basis, dimension and interval.

We define G and the residuals of (HSD M-NP) for a given point $z = (\bar{x}, y, \bar{s})$ as:

$$G = \begin{pmatrix} 0 & A & -b \\ -A^T & 0 & c \\ b^T & -c^T & 0 \end{pmatrix}$$

$$\begin{pmatrix} r_p \\ r_d \\ r_c \end{pmatrix} = G \begin{pmatrix} y \\ \bar{x} \end{pmatrix} - \begin{pmatrix} 0 \\ \bar{s} \end{pmatrix}$$

Define the duality gap and function ψ as:

$$\mu(z) = \bar{x}^T \bar{s} / \bar{v},$$

$$\psi(\bar{x}, \bar{s}, \mu(z)) = \bar{s} + \mu(z) \mathbf{g}_{\bar{x}}(\bar{x}).$$

The η-neighborhood of the central path is defined as:

$$\mathcal{N}(\eta) = \{ z \in \bar{\mathcal{K}} \times \mathbb{R}^m \times \bar{\mathcal{K}}^* : \ \ ||\psi(\bar{x}, \bar{s}, \mu(z))||_{\bar{x}}^* \leq \eta \mu(z) \},$$

where the Hessian-norm is defined as $||v||_{\bar{x}}^* = ||\mathbf{H}_{\bar{x}}^{-1/2} v||$. For the general properties of the Hessian-norm, see [48].

5.6.1 Central Path

The central path of (HSD M-NP) is defined as the unique solution of the following parametrized system of equations:

$$G\begin{pmatrix} y \\ \bar{x} \end{pmatrix} - \begin{pmatrix} 0 \\ \bar{s} \end{pmatrix} = \gamma \begin{pmatrix} r_p \\ r_d \\ r_c \end{pmatrix} \qquad \text{(CP)}$$

$$\bar{s} + \gamma\mu \mathbf{g}_{\bar{x}} = 0$$

where $0 \leq \gamma \leq 1$, and is called the centering parameter.

The central path starts from an initial point, z_0, (at $\mu_0 = 1$), and goes towards the optimal solution of the (HSD M-NP) as $\mu \to 0$. In path-following algorithms, it is impossible to precisely follow the central path and step on it in each iteration. Instead, we stay in the neighborhood of the central path, which preserves the centrality and gives us more room to take a large step towards the optimal solution. This is in contrast to what Fiacco and McCormick had done in their seminal work [55]. They knew about the barrier function but they did not consider the central path and that the iterations should stay close to it.

5.6.2 Approximately

To find the solutions of the central path (CP) approximately, one can use the first order Newton's method to find the Newton direction, and then update the current iteration by taking a suitable step size along this direction.

The first order Newton's method is: starting from a given point, $(\bar{x}, y, \bar{s})$, substitute $(\bar{x} + d\bar{x}, y + dy, \bar{s} + d\bar{s})$ into (CP) and then linearize the system of equations. After linearizing the system, the Newton direction will be the unique solution of the following

system of linear equations:

$$G\begin{pmatrix} dy \\ d\bar{x} \end{pmatrix} - \begin{pmatrix} 0 \\ d\bar{s} \end{pmatrix} = (1-\gamma)\begin{pmatrix} r_p \\ r_d \\ r_c \end{pmatrix} \qquad \text{(Direction)}$$

$$ds + \mu \mathbf{H}_x dx = -s - \gamma\mu \mathbf{g}_x$$

$$\tau d\kappa + \kappa d\tau = -\tau\kappa + \gamma\mu.$$

To find a direction which decreases the residuals and duality gap and also stays near the central path, the following algorithms are usually considered:

1. HSD Predictor-Corrector IPM
2. HSD Mehrotra Predictor-Corrector IPM

5.6.3 HSD Predictor-Corrector IPM

To make interior point algorithms efficient in practice, one usually alternates between two phases of (i) Predictor, and (ii) Corrector.

Predictor Phase

In the predictor phase with the goal of taking a large step, one tries to find a direction which is approximately tangent to the central path and points toward the direction that reduces the residuals and duality gap simultaneously, and is not concerned about the centrality. This means to put the centering parameter in (Direction) equal to zero ($\gamma = 0$).

Therefore, given the starting point $z = (\bar{x}, y, \bar{s}) \in \mathcal{N}(\eta)$, the predictor direction is going to be defined as the unique solution of the following system of equations:

$$G\begin{pmatrix} dy_p \\ d\bar{x}_p \end{pmatrix} - \begin{pmatrix} 0 \\ d\bar{s}_p \end{pmatrix} = \begin{pmatrix} r_p \\ r_d \\ r_c \end{pmatrix} \qquad \text{(PD)}$$

$$ds_p + \mu \mathbf{H}_x dx_p = -s$$

$$\tau d\kappa_p + \kappa d\tau_p = -\tau\kappa.$$

Now consider the step-length on this direction be α_p and define:

$$z_p := (\bar{x}_p, y_p, \bar{s}_p) = (\bar{x} + \alpha_p d\bar{x}_p, y + \alpha_p dy_p, \bar{s} + \alpha_p d\bar{s}_p).$$

It is not difficult to see that the residual and the duality gap will be reduced proportionally to $(1 - \alpha_p)$, (see Lemma 3 in [48]). Skajaa and Ye [48] showed if $\eta \leq 1/6$, then by taking a fixed step-length, $\alpha_p = \Omega(1/\sqrt{\nu})$, $z_p \in \mathcal{N}(2\eta)$. But in practical implementation a larger step-length can be taken. To see how to choose the step-length, refer to Section 5.7.2.

Corrector Phase

In the corrector phase, on the other hand, we try to find a search direction which points towards the central path but at the same time keeps the residuals and the duality gap unchanged. By taking a step on this direction, the next point, z_c^+, will be near the central path again, and then in the next predictor phase once again we can take a long step toward the optimal solution. To find the direction pointing toward the central path, we put the centrality parameter in (Direction) at its maximum value ($\gamma = 1$). Therefore, starting from the current iteration, i.e. z_p, the corrector direction, shown by dz_c, will be the unique solution of the following system of equations:

$$G \begin{pmatrix} dy_c \\ d\bar{x}_c \end{pmatrix} - \begin{pmatrix} 0 \\ d\bar{s}_c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{CD}$$

$$ds_c + \mu_p \mathbf{H}_{x_p} dx_c = -s_p - \mu_p \mathbf{g}_{x_p}$$

$$\tau_p d\kappa_c + \kappa_p d\tau_c = -\tau_p \kappa_p + \mu_p.$$

Defining the step-length in the corrector phase by α_c, the next iteration will be $z_c := z_p + \alpha_c dz_c$. Skajaa and Ye (see Lemma 6 in [48]) showed that if z_p is in $\mathcal{N}(2\eta)$, then by applying the corrector phase at most twice, the next point will be close to the central path again, in the sense that $z_c \in \mathcal{N}(\eta)$. But as the implementation results show, even if z_p is not in $\mathcal{N}(2\eta)$, applying the corrector phase twice will still result a point in $\mathcal{N}(\eta)$.

Also, they showed that by using these two phases with the specified step-length, the algorithm terminates with a ϵ-solution of (HSD M-NP) in no more than $O(\sqrt{(}\nu)\log(1/\epsilon))$ iterations. See Algorithm 1 in Chapter 3 for a general overview.

5.6.4 HSD Mehrotra Predictor-Corrector IPM

It is common in practice, to adjust the centering parameter in each iteration rather than keeping it fixed. In addition to updating the centering parameter in each iteration, a second order approximation to the central path can be used. See [30] for Mehrotra second order approximation in the case of linear programming, and [6] for general self-scaled conic problem. Akle [2] proposed a similar idea in the case of the non-symmetric cones. We will not use the second order technique here, since this is still an active area of research which requires more investigation.

There are many ways to update the centering parameter. One general way is to update the centering parameter based on how much improvement can be made in the current iteration. One measure to quantify the "improvement" can be defined based on the step-length which can be taken.

The algorithm composes of two phases: (i) Affine, and (ii) Combined.

Affine Phase

In the affine phase, we are not concerned about the centering. Instead, we want to see how much improvement can be made by taking a step on the affine search direction. Therefore, setting the centering parameter equal to zero in (Direction), the affine search direction, $dz_a = (d\bar{x}_a, dy_a, d\bar{s}_c)$, will be defined as the unique solution of the following system of linear equations:

$$G\begin{pmatrix} dy_a \\ d\bar{x}_a \end{pmatrix} - \begin{pmatrix} 0 \\ d\bar{s}_a \end{pmatrix} = \begin{pmatrix} r_p \\ r_d \\ r_c \end{pmatrix} \qquad \text{(Aff. Dir.)}$$

$$ds_a + \mu \mathbf{H}_x dx_a = -s$$

$$\tau d\kappa_a + \kappa d\tau_a = -\tau\kappa.$$

Then, we define the step-length α_a such that $z + \alpha_a dz_a \in \mathcal{N}(\beta_0)$ for $0 < \beta_0 < 1$.

It is evident that the affine search direction is the same as the predictor search direction in the HSD predictor-corrector method.

Combined Phase

The combined phase is in fact a linear combination of the predictor and corrector phases, i.e. $(1-\gamma)$PD + γCD. The centering parameter is chosen based on how much improvement has been made in the affine phase. We will define "improvement" shortly. This makes sense since if a large improvement can be made in the affine phase, then that means the current point is more and less centered, and therefore, less centering will be needed in the combined phase. On the other hand, if only a small improvement can be made in the affine phase, then this means that the current point is more and less close to the boundary, and therefore, more emphasis will be on centering in the combined phase. This is in-line with the chosen coefficients in the linear combination. Therefore, having set γ, the combined search direction, $dz_c = (d\bar{x}_c, dy_c, d\bar{s}_c)$, will be the unique solution of the following system of linear equations:

$$G \begin{pmatrix} dy_c \\ d\bar{x}_c \end{pmatrix} - \begin{pmatrix} 0 \\ d\bar{s}_c \end{pmatrix} = (1-\gamma) \begin{pmatrix} r_p \\ r_d \\ r_c \end{pmatrix} \qquad \text{(Com. Dir)}$$

$$ds_c + \mu \mathbf{H}_x dx_c = -s - \gamma\mu \mathbf{g}_x$$

$$\tau d\kappa_c + \kappa d\tau_c = -\tau\kappa + \gamma\mu.$$

Regarding the meaning of "improvement", Andersen et al. [9] suggested the following heuristic centering parameter in the case of symmetric interior point algorithms:

$$\gamma = (1-\alpha_a)^{expon} \tag{5.4}$$

where $expon = 1,$ or 2 or 3. Recently, Akle [2] used this centering parameter in non-symmetric interior point algorithms (see algorithm 5 in [2]). Our numerical experiments confirmed that this is a good choice in the case of uMCO.

5.7 Practical Issues

Before showing the numerical results of suggested non-symmetric algorithms for the uMCO, we should mention certain practical issues which are important in our implementation.

5.7.1 Initial Point

In practical IPMs, initial point plays an important role specially in the running time and number of iterations. Our experiments show that

$$x_0 = (\int_a^b t^0 dt, \int_a^b t dt, ..., \int_a^b t^n dt)^T, s_0 = -g_{x_0}, \tau_0 = 1, \kappa_0 = 1 \tag{5.5}$$

is a suitable initial point for the uMCO in the standard basis, which lies exactly on the central path (CP) for $\mu = 1$.

The initial point in the Chebyshev basis, is simply the transformation of this initial point, which is:

$$x_0 = (\int_a^b T_0(t) dt, \int_a^b T_1(t) dt, ..., \int_a^b T_n(t) dt)^T, s_0 = -g_{x_0}, \tau_0 = 1, \kappa_0 = 1. \tag{5.6}$$

5.7.2 Step-Length and Neighborhood

To make the implementation practical, the IPM algorithms usually consider a heuristic approach to find a reasonable large step-length. Although the theory suggests to take the step-length such that z_p^+ is in $\mathcal{N}(2\eta)$ in the predictor, and z_c^+ is in $\mathcal{N}(\eta)$ in the corrector steps, for $0 < \eta < 1$. But in practice we can take a larger step-length to reduce the number of iterations. Generally, in the non-symmetric IPMs there are two stages to compute the step-length. In the first stage, we choose the step-length such that by taking that step-length, the new point will be on the boundary of the cone. Then we take a step backwards such that the new point will be in the strict feasibility area of the cones. This means,

$$\alpha = .995 \min\{\alpha_x, \alpha_s, \alpha_\tau, \alpha_\kappa\},$$

where α_x, α_s, α_τ and α_κ are respectively the step-lengths in the moment cone, the non-negative polynomial cone and positive orthant. In the second stage, we modify the

step-length found in the first stage by using backtracking line search [57], such that the new point using this step-length will be in the right neighborhood. But since computing the Hessian norm in line search is expensive, therefore, in practical implementation, the infinite norm will be used. The η-infinite neighborhood is defined as in the following:

$$\mathcal{N}(\eta) = \{z = (\bar{x}, y, \bar{s}) \in \bar{\mathcal{K}} \times \mathbb{R}^m \times \bar{\mathcal{K}}^* : \left\| \begin{pmatrix} ||\nabla F(x)||_\infty^{-1}(s + \mu \nabla F(x)) \\ \tau\kappa - \mu \end{pmatrix} \right\|_\infty \leq \eta\mu(z)\}.$$

It has been shown that these two neighborhoods are related. See Theorem 3.7.1 in [45].

The step-length in the moment cone is similar to SDP. But how about the step-length in the non-negative polynomial cone? In other words, suppose $s \in \text{Int}(\mathcal{P}^+)$ and ds is the solution of (PD) or (CD). The question is, "How to find α such that $s + \alpha ds \in \partial\mathcal{P}^+$"? One answer is to discretize the cone. This means, consider $\{a \leq t_1 < ... < t_p \leq b\}$. Then α_s will be:

$$\alpha_s = \min_{1 \leq i \leq p} \{-\frac{s(t_i)}{ds(t_i)}\}, \quad \forall t_i \quad s.t. \quad ds(t_i) < 0.$$

We should be cautious about choosing p. This should be large enough to represent a refined grid of points in the interval.

5.8 Numerical Results

To show the numerical results of the implementations, we have considered the following three experiments:

1. The comparison between the performance of the non-symmetric HSD predictor-corrector vs. the performance of the non-symmetric HSD Mehrotra predictor-corrector.

2. The comparison between the performance of the odd degree vs. the performance of the even degree.

3. The comparison between the performance of our implementations which is a non-symmetric HSD Mehrotra predictor-corrector to solve the non-symmetric formulation (i.e. uMCO) of the test problems vs. the performance of a commercial

solver (i.e. Mosek), which is a symmteric HSD Mehrotra predictor-corrector to solve the symmetric formulation (i.e. SDP) of the test problems.

Before explaining the numerical experiments, let us show how to construct a set of test problems which are formulated as uMCO. To do so, we consider the dual problem of the approximation problem, i.e. (4.1) and the dual of time-varying maximum network flow problem, i.e. (Max-Flow) (see Chapter 4).

The dual problem of approximation problem (4.1), after standardization, can be formulated as follows:

$$\begin{aligned} \min \quad & p_1^T x_1 + ... + p_k^T x_k \\ s.t. \quad & I_{n_1} x_1 + ... + I_{n_k} x_k - I_{n_{k+1}} x_{k+1} = e_{[a,b]}, \\ & x_i \succcurlyeq_{\mathcal{M}^{n_i}_{[a,b]}} 0, \quad i = 1, ..., k, \end{aligned} \tag{Dual Approx}$$

where I_{n_i} is the identity matrix of dimension n_i, and p_i's are random generated positive polynomials in the Chebyshev basis, and $e_{[a,b]}$ has been defined in Section (4.1). The numerical experiments have been shown that the best interval is $[-1, 1]$ in the Chebyshev basis. Therefore, (Dual Approx) is in fact a multi-block uMCO with:

$$\begin{aligned} c &= [p_1; ...; p_k], \\ A &= [I_{n_1}, ..., I_{n_k}, -I_{n_{k+1}}], \\ b &= e_{[-1,1]}. \end{aligned}$$

The second set of test problems are constructed from considering the dual problem of (Max-Flow), which can be formulated as the following:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} \langle p_{i,j} y_{i,j} \rangle \\ s.t. \quad & u_i - u_j + y_{i,j} \succcurlyeq_{\mathcal{M}^n_{[a,b]}} e_{[a,b]}, \quad (i,j) \in E \\ & u_i, u_j, y_{i,j} \succcurlyeq_{\mathcal{M}^n_{[a,b]}} 0, \qquad (i,j) \in E. \end{aligned} \tag{Dual Max-Flow}$$

Again, after some standardization, (Dual Max-Flow) can be cast as a multi-block uMCO. Since a library of benchmark test problems does not exist for this class of problems, the test problems were generated randomly. This means, first, we have generated

a network $G(V, E, s0, s_1)$. Then, the capacities, which are non-negative polynomials, have been randomly generated.

5.8.1 Non-Symmetric HSD Predictor-Corrector vs. Non-Symmetric HSD Mehrotra Predictor-Corrector

The first experiment will show the performance of two non-symmetric algorithms that have been mentioned in Section 5.6, i.e. the non-symmetric HSD predictor-corrector IPM, verses the non-symmetric HSD Mehrotra predictor-corrector IPM.

Table 5.2 shows the numerical results of the implementation of these two algorithms. The columns of the table are divided to four blocks:

1. **Net. Inf.:** Shows the general information about each network flow. Specifically, "V" is the number of vertices, "E" is the number of edges and "deg" is the degree of the maximum flows which are the same along all edges.

2. **Prob. Dim.:** Shows the general information about the dimension of the non-symmetric formulation of each test problem. Specifically, "blk" is the number of blocks, "m" is the number of constraints and "n" is the number of decision variables.

3. **Perform. Inf. P-C:** Shows the general information about the numerical performance of HSD predictor-corrector IPM implementation. Specifically, "it." shows the number of iterations and "time" shows the running time (in second) that is needed for the algorithms to reduce the infeasibility and the duality gap by a factor of 10^{-7} as a general rule of convergency.

4. **Perform. Inf. M-P-C:** Shows the general information about the numerical performance of HSD Mehrotra predictor-corrector IPM implementation. Specifically, "it." shows the number of iterations and "time" shows the running time (in second) that is needed for the algorithms to reduce the infeasibility and the duality gap by a factor of 10^{-7} as a general rule of convergency.

Net. Inf.			Prob. Dim.			Perform. Inf. P-C		Perform. Inf. M-P-C	
V	E	deg	blk	m	n	it.	time	it.	time
4	5	99	14	500	1400	12	169	21	170
4	5	199	14	1000	2800	16	883	25	808
6	10	99	28	1000	2800	12	438	28	439
6	10	199	28	2000	5600	26	3540	38	2740
7	12	79	34	960	2720	16	443	26	353
7	12	99	34	1200	3400	12	438	17	345
9	16	69	46	1120	3220	15	417	25	383
10	20	79	56	1600	4480	25	1190	47	1180
11	26	39	70	1040	2800	12	251	15	192
11	26	59	70	1560	4200	12	519	16	366
12	33	39	86	1320	3440	12	352	16	265
17	26	29	82	780	2460	15	254	23	252
21	35	29	108	1050	3240	24	662	36	579
14	42	29	108	1260	3240	12	389	17	345
32	56	29	172	1680	5160	15	879	23	812
45	70	19	226	1400	4520	13	612	19	569
58	91	19	294	1820	5880	22	1730	33	1690

Table 5.2: Predictor-Corrector vs Mehrotra Predictor-Corrector for uMCO

Table 5.2 shows that the Mehrotra predictor-corrector demonstrates better performances in term of running time as a general rule of superiority.

A rough comparison between the two algorithms is given by counting the number of computation of the most expensive machinery in each iteration of the algorithms. This is given in Table 5.3.

Machinery	Number of Computation	
	HSD m-P-C	**HSD P-C**
Computing $\mathbf{H}_x$ and $\mathbf{H}_x^{-1}$	1 time	2 times
Constructing Schur	1 time	2 times
Schur Factorization	1 time	2 times
Solving Newton System	2 times	2 times

Table 5.3: The most expensive machinery and their number of computation

5.8.2 Even Degree vs. Odd Degree

As we have seen in Sections 5.4.2 and 5.4.3, the barrier functions for the moment cone with an even degree and an odd degree are different. Therefore, in the second experiment we have considered a set of test problems from (Dual Approx) and (Dual Max-Flow) to test the performance of these barrier functions.

We have considered the non-symmetric Mehrotra predictor-corrector algorithm for this experiment. To compare the performance of these barriers in a fair manner, the parameters are set to be similar. In addition, we have considered the problems which are exactly the same in terms of input data. The only difference is the degree of optimal solution which differs by one degree.

Table 5.4 shows the numerical results. The first five rows are the results for the approximation problems, and the rest are for the maximum network flow problems. The columns of the table are divided to four blocks:

1. **Net. Inf.:** Shows the general information about the network flow (similar to table 5.2). The (± 1) in the column named "deg" shows that the degree of flows increases/decreases by one.

2. **Prob. Dim.:** Shows the general information about the dimension of each test problem. Since the degree of optimal solution differs by one, therefore the number of linear constraints and the number of decision variables are slightly different. The increases or decreases implied by this difference are shown by $\pm number$ in the

related columns. For example, for the first problem in the first row, the number of linear constraints with the even degree solution (i.e. 10) is 11, whereas with an odd degree solution (i.e. 11) is 12. Similar analogy holds for the number of decision variables.

3. **Perf. Inf. E. D.:** Shows the general performance information of the HSD predictor-corrector IPM implemented for the cones with even degree. Columns "it." and "time" have the same meaning as in Table 5.2.

4. **Perf. Inf. O. D.:** Shows the general performance information of the HSD predictor-corrector IPM implemented for the cones with even degree. Columns "it." and "time" have the same meaning as before.

Table 5.4 shows that the performance of these two barrier functions are not significantly different. In particular, the number of iterations for both barriers are marginally different, if there are any differences at all. Also, the running time is not significantly different. In fact, on average they are different by less than 2%. These marginal differences can be justified by remembering that the algorithms terminate when the infeasibility and the duality gap are reduced by a factor of 10^{-7}. Therefore, the observed marginal differences can be attributed to the marginal differences in the algorithms' performance.

Net. Inf.			Prob. Dim.			Perf. Inf. E. D.		Perf. Inf. O. D.	
V	E	deg	blk	m	n	it.	time	it.	time
3	2	10+1	3	11+1	34+3	10	3.7	10	3.3
3	2	210+1	3	211+1	634+3	15	101	15	108
3	2	410+1	3	411+1	1234+3	18	662	18	688
3	2	510+1	3	511+1	1534+3	18	1140	18	1180
3	2	610+1	3	612+1	1837+3	18	1860	18	1900
4	4	150+1	12	604+4	1813+12	22	247	22	270
4	5	80+1	14	405+5	1135+14	24	102	24	111
6	8	76-1	24	616-8	1849-24	31	175	33	191
6	10	120+1	28	1210+10	3389+28	36	753	38	762
7	12	66-1	34	804-12	2279-34	22	151	22	150
9	16	60+1	46	976+16	2807+46	29	235	29	249
11	26	150+1	70	3926+26	10571+70	19	1830	19	1980
11	26	56-1	70	1482-26	3991-70	17	230	17	229
12	33	140+1	86	4653+33	12127+86	18	2590	18	2690
12	33	50+1	86	1683+33	4387+86	17	243	17	257
17	26	110+1	82	2886+26	9103+82	42	2690	42	2970
14	42	130+1	108	5502+42	14149+108	18	3630	18	3670
14	42	46-1	108	1974-42	45077-108	18	318	18	323
19	35	120+1	108	4235+35	13069+108	58	7460	57	7710
19	35	40+1	108	1435+35	4429+108	48	580	53	679
17	26	36-1	82	962-26	3035-82	34	267	33	260
32	56	30+1	172	1736+56	5333+172	32	619	32	661
45	70	25-1	226	1890-70	6130-226	72	1600	77	1690
58	91	20+1	294	1911+91	6175+294	77	1950	78	2080

Table 5.4: Mehrotra Predictor-Corrector for Even and Odd degree uMCO

5.8.3 Non-symmetric HSD Mehrotra Predictor-Corrector vs. Symmetric HSD Mehrotra Predictor-Corrector

In the third experiment, we have considered the comparison between the performance of our non-symmetric HSD Mehrotra predictor-corrector and the performance of a symmetric HSD Mehrotra predictor-corrector. We have used Mosek [32], a commercial solver for solving symmetric conic optimizations, as a benchmark. Mosek uses a symmetric Mehrotra predictor-corrector which is the closest algorithm to our non-symmetric algorithm which was used previously. We should mention that Mosek uses a large number of techniques, prepossessing, scaling and subroutines written in C or Java, which makes the solver one of the fastest and most efficient commercial solvers available.

The set of test problems has been constructed once again from (Dual Approx) and (Dual Max-Flow). The test problems have been constructed such that we have two set of problems: (i) Problems with few blocks but large degree polynomials, and (ii) Problems with many blocks but medium degree polynomials. The reason for these choices is first to see how fast the dimension of the problem in the SDP presentation grows when dealing with large degree polynomials (to see why large degree polynomial are considered refer to Section 5.9), and second, to see how Mosek benefits from its embedded enhancements when dealing with problems with many blocks. Therefore, if similar enhancements are applied to the non-symmetric Mehrotra predictor-corrector algorithm, its performance will further improve.

Table 5.5 shows the numerical results. The first seven rows are the results for the approximation problems, and the rest are for the maximum network flow problems. The columns of the table are divided into three blocks:

1. **Net. Inf.:** Shows the general information about the each network, as has been mentioned earlier.

2. **Perf. Inf. NS M-P-C:** Shows the general information about the non-symmetric

formulation (i.e. uMCO), and the performance of the non-symmetric HSD Mehrotra predictor-corrector IPM for uMCO. In particular, "m" and "n" show the number of linear constraints and the number of decision variables, respectively, in the uMCO formulation, and "it." shows the number of iterations that is needed by the non-symmetric Mehrotra predictor-corrector algorithm to converge.

3. **Perf. Inf. SDP/Mosek:** Shows the general information about the symmetric formulation (i.e SDP), and the performance of symmetric algorithm for the same test problems. In particular, "m" and "n" show the number of linear constraints and the number of decision variables, respectively, in the SDP formulation, and "it." shows the number of iterations that are needed by Mosek to converge.

The general rule of convergency here is a reduction of the infeasibility and the duality gap by a factor of 10^{-7}. The running time is not mentioned because of the enhancements in Mosek which speed up its convergence.

Table 5.5 shows that formulating the uMCO as SDP, increases the dimension of the problem significantly. Also, we can see that the number of iterations of these two algorithms are not significantly different for the first set of test problems (problems with few blocks but large degree polynomials). This is because, for these problems, Mosek does not benefit much from the enhancements mentioned earlier. However, the number of iterations are noticeably different for the second set of test problems (problems with many blocks but medium degree polynomials). This is because, for these problems, Mosek benefits significantly from those enhancements.

It should be noted that, in each iteration, Mosek constructs and solves a much larger system of linear equations and computes the Schur in $O(mn^3)$, whereas the non-symmetric algorithm deals with a much smaller system of linear equations and computes the Schur in $O(mn^2 \log n)$.

Net. Inf.				Perf. Inf. NS M-P-C			Perf. Inf. SDP/Mosek		
V	E	deg	blk	m	n	it.	m	n	it.
3	2	10	3	11	34	11	31	117	9
3	2	111	3	112	337	13	335	9687	9
3	2	212	3	213	640	16	637	34558	13
3	2	313	3	314	943	16	941	74731	18
3	2	414	3	415	1246	17	1243	130205	15
3	2	515	3	516	1549	18	1547	200981	16
3	2	616	3	617	1852	17	1849	287058	15
4	4	47	12	192	577	18	572	7388	12
4	4	185	12	744	2233	21	2228	105644	17
6	8	186	24	1488	4465	37	4460	211292	36
4	5	45	14	230	645	20	640	7954	12
6	8	43	24	352	1057	23	1052	12492	13
6	10	41	28	420	1177	34	1172	13352	16
7	12	39	34	480	1361	15	1355	14755	9
9	16	37	46	608	1748	28	1743	18083	12
11	26	35	70	936	2521	16	2514	24870	10
12	33	33	86	1122	2925	16	2914	27428	10
14	42	31	108	1344	3457	17	3381	30645	10
21	37	30	108	1050	3241	43	3231	26961	22
17	26	27	82	728	2297	29	2289	17941	15
32	56	31	172	1456	4472	31	4460	32748	16

Table 5.5: Non-symmetric HSD Mehrotra predictor-corrector vs. SDP for uMCO

5.9 Motivation for Large Degree Polynomials

To give an intuition about the maximum network flow with capacities in the non-negative polynomial cone and how a large degree polynomial might be needed, consider

the network flow in Figure 5.1, where $V = 4$, $E = 4$ and $p_{i,j} \in \mathcal{P}^{5^+}$.

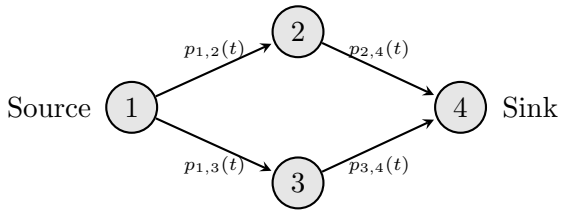

Figure 5.1: Network Flow with $V = 4$, $E = 4$

There are two augmenting paths in the above network flow, i.e. $1 \to 2 \to 4$ and $1 \to 3 \to 4$. Figures 5.2 and 5.3 show the capacities of each augmenting path along with the maximum flows of degree 10, 20 and 50. It is evident from the figures that increasing the degree of the maximum flows increases the flows from source to sink. Therefore, to obtain a desirable solution in a large network with a long augmenting path, increasing the degree of the maximum flows is necessary.

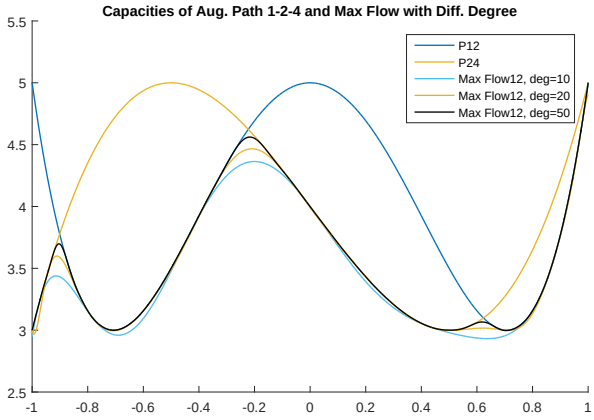

Figure 5.2: Maximum flow of different degrees along with capacities of augmenting path $1 \to 2 \to 4$.

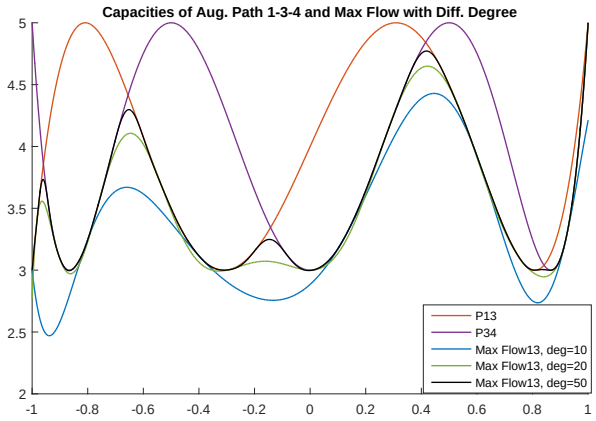

Figure 5.3: Maximum flow of different degrees along with capacities of augmenting path $1 \to 3 \to 4$.

5.10 Conclusion

In this chapter, we have investigated solving the non-negative univariate polynomial conic optimization via the univariate moment conic optimization. We have shown solving uPCO directly using the barrier function for the non-negative polynomial is not efficient. Also, we have observed that the semidefinite presentation of uPCO drastically increases the dimension of the problem and is numerically unstable in the standard basis. In regard to the latter issue, the Chebyshev change of basis resolved the numerical instability of the standard basis without extra computation. In regard to the former issue, we have considered the dual problem. We have observed that in contrast to the non-negative polynomial cone, the moment cone has an efficient barrier function in both bases. Also, we have seen that the Schur in uMCO representation can be computed in $O(mn^2 \log n)$, while this is $O(mn^3)$ in the semidefinite representation.

Next, we have developed a non-symmetric HSD predictor-corrector IPM, and a Mehrotra version of this algorithm to solve uMCO-uPCO in a primal-dual setting, where the moment and the non-negative polynomial problems were the primal and dual problems, respectively.

Finally, we have shown the numerical results for the implementation of these algorithms, which indicate that the non-symmetric algorithms are competitive with the symmetric algorithm in terms of the number of iterations. Note that in the non-symmetric models the problem dimension has been kept unchanged.

Chapter 6

Polynomial Conic Optimization with Universal Barrier Function

In this chapter, we investigate solving the uPCO using a non-symmetric primal-dual IPM utilized by a universal barrier function for the cone of non-negative polynomials. We will use the primal-dual setting, where the primal problem is the uPCO problem and the dual problem is the uMCO problem. This is in contrast to the previous chapter. Particularly, we develop a non-symmetric HSD predictor-corrector IPM for solving the uPCO-uMCO setting. We show the numerical results of this implementation and mention the observations. Finally, we investigate certain difficulties of this approach and point out possible remedies.

6.1 Non-Negative Polynomial and Moment Optimizations in Primal-Dual Setting

In this chapter, we will once again consider the conic optimization involving the non-negative conic constraints, but this time we keep the uPCO as the primal problem, in which we carry out all the computations. In the standard form, this problem is defined as:

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & A\, x = b, \\ & x \succcurlyeq_{\mathcal{P}^{n+}_{[a,b]}} 0. \end{aligned} \tag{uPCO}$$

We have seen in (6.2) that A, b, c and $\mathcal{P}^{n+}_{[a,b]}$ can be considered in a multi-blocks sense. Using the duality between $\mathcal{P}^{n+}_{[a,b]}$ and $\mathcal{M}^{n}_{[a,b]}$ the dual problem of (uPCO) can be

formulated in a standard form as in the following:

$$\begin{aligned} \max \quad & b^T\, y \\ s.t. \quad & A^T y + s = c, \\ & s \succcurlyeq_{\mathcal{M}^n_{[a,b]}} 0. \end{aligned} \tag{uMCO}$$

As explained in previous chapters, the HSD model has attractive practical and theoretical properties for solving conic optimizations. Regarding that, the HSD model for (uPCO) and (uMCO) is as follows:

$$\begin{aligned} \min \quad & 0 \\ s.t. \quad & Ax - b\tau = 0 \\ & A^T y + c\tau - s = 0 \\ & b^T y - c^T x - \kappa = 0 \\ (x,\tau) \in \mathcal{P}^{n^+}_{[a,b]} \times \mathbb{R}^+,\ & y \in \mathbb{R}^m,\ (s,\kappa) \in \mathcal{M}^n_{[a,b]} \times \mathbb{R}^+, \end{aligned} \tag{HSD NP-M}$$

Comparing (HSD NP-M) with (HSD M-NP) from the previous chapter, here the non-negative polynomial cone is considered as the primal cone and the moment cone as the dual cone. This is in contrast to (HSD M-NP) where the moment cone was the primal cone and the non-negative polynomial cone was the dual cone.

We have observed that to consider any path-following IPM for the HSD model, it is required that at least the primal cone has been provided with a suitable barrier function, which has a closed-form gradient and Hessian. Faybusovich has proposed one barrier function [16]. We explain this in the next section.

6.2 Faybusovich LHSCB Function in Standard Basis

Faybusovich [16] proposed a LHSCB function for the class of cones induced by the Chebyshev systems (see definition 5), on intervals of the real line and the circle. This class of cones includes almost all cones of "sum of squared", considered by Nesterov in [35]. The cone of non-negative polynomials is a subset of this class of cones.

To define this class of cones, let $\{u_0, ..., u_n\}$ be a Chebyshev system. Then

$$\mathcal{K} = \{x = \sum_{i=0}^{n} a_i u_i \;\; a_i \geq 0 : \quad x(t) \geq 0 \quad \forall t \in [a, b]\} \tag{6.1}$$

is called the cone induced by the Chebyshev system $\{u_0, ..., u_n\}$.

Based on this definition, he proved that:

Proposition 16.

Let $\{u_0, ..., u_{2\nu-1}\}$ be a Chebyshev system of continuously differentiable functions on $[a, b]$ of odd order. Then

$$F(x) = \ln \epsilon Pf(D(x)) \quad \textit{when } x \in \textit{Int}(\mathcal{K}) \tag{6.2}$$

is a LHSCB function for $\mathcal{K}$ where $\epsilon = \pm 1$, Pf is Pfaffian (see [16]) and the (i, j)-th entry of matrix $D(x)$ is

$$\mathcal{D}_{i,j}(x) = \int_a^b \frac{u_i(t)u_j'(t) - u_i'(t)u_j(t)}{x(t)^2} dt, \quad i, j = 0, ..., 2\nu - 1. \tag{6.3}$$

Proof. See Theorem 4 in [16]. □

The case for the Chebyshev system of even order is slightly different. See Theorem 5 in [16] for this case. In this work, we only consider the case when the order is odd.

The Chebyshev systems of special interest in this work are the standard and Chebyshev bases of polynomials. We present Theorem 16 for the standard basis first, and later on, we will show it for the Chebyshev basis.

Consider the Chebyshev system when $u_i(t) = t^i$ for $i = 0, ..., n$ and $t \in [a, b]$. Then, adjusting Theorem 16 for this system gives:

$$F(x) = -\frac{1}{2} \ln \det \mathcal{D}(x), \quad \text{when } x \in \text{Int}(\mathcal{P}^{n+}_{[a,b]}), \tag{6.4}$$

is a LHSCB function for $\mathcal{P}^{n+}_{[a,b]}$, where $\mathcal{D}(x)$ is a skew symmetric matrix of size $(n+1) \times (n+1)$. The (i, j)-th entry of this matrix has the form

$$\begin{aligned} \mathcal{D}_{i,j}(x) &= \int_a^b \frac{t^i (t^j)' - (t^i)' t^j}{(x_0 + x_1 t + ... + x_n t^n)^2} dt \\ &= (j - i) \int_a^b \frac{t^{i+j-1}}{x(t)^2} dt, \qquad\qquad i, j = 0, ..., n. \end{aligned} \tag{6.5}$$

Considering the components of $\mathcal{D}_{i,j}(x)$, we can write $\mathcal{D}(x)$ as

$$\mathcal{D}(x) = \mathcal{T} \circ H(h(x)), \tag{6.6}$$

where $\mathcal{T} \in \mathbb{R}^{(n+1)\times(n+1)}$ is a constant Toeplitz matrix with (i,j)-th entry equal to $j-i$, $\circ$ is the Hadamard product, H is the Hankel operator and $h(x)$ is following vector:

$$h(x) = \begin{pmatrix} 0 \\ \int_a^b \frac{1}{x(t)^2} dt \\ \vdots \\ \int_a^b \frac{t^{2n-2}}{x(t)^2} dt \\ 0 \end{pmatrix}_{2n+1\times 1}. \tag{6.7}$$

Therefore, (6.4) can be written as:

$$F(x) = -\frac{1}{2} \ln\det \mathcal{T} \circ H(h(x)), \quad \text{when } x \in \mathrm{Int}(\mathcal{P}^{n+}_{[a,b]}). \tag{6.8}$$

To simplify the notations, we use h and $\mathcal{D}$ instead of $h(x)$ and $\mathcal{D}(x)$, respectively.

Considering the definition of the convex conjugate for this barrier function, it is evident that the conjugate barrier does not have a closed-form solution. Therefore, a non-symmetric IPM will be needed, where the gradient and the Hessian of the barrier function of the primal cone are required.

6.3 Gradient and Hessian of Faybusovich Barrier Function in Standard Basis

To perform a non-symmetric IPM, at least the gradient and the Hessian of the primal barrier function should be computable efficiently. In the case of $\mathcal{K} = \mathcal{P}^{n+}_{[a,b]}$ with the barrier function defined in (6.8), the gradient can be computed as:

$$\mathbf{g}_{x_i} := \nabla_{x_i} F = -\frac{1}{2} \langle \mathcal{D}^{-1}, \mathcal{T} \circ H(\frac{\partial h}{\partial x_i}) \rangle, \quad i = 0, ..., n.$$

After some algebraic manipulations, the gradient can be written as:

$$\mathbf{g}_x = H([Dh]_0^n, [Dh]_n^{3n-2})[H^*(\mathcal{D}^{-1} \circ \mathcal{T})]_1^{2n}, \tag{6.9}$$

where

$$Dh = \begin{pmatrix} \int_a^b \frac{1}{x(t)^3} dt \\ \vdots \\ \int_a^b \frac{t^{3n-2}}{x(t)^3} dt \end{pmatrix}_{(3n-1)\times 1} . \tag{6.10}$$

On the other hand, the Hessian can be computed as follows:

$$\begin{aligned} [\mathbf{H}_x]_{i,j} := \nabla^2_{x_i,x_j} F = & -\frac{1}{2}\langle -\mathcal{D}^{-1}(\mathcal{T} \circ H(\frac{\partial h}{\partial x_j}))\mathcal{D}^{-1}, \mathcal{T} \circ H(\frac{\partial h}{\partial x_i})\rangle \\ & -\frac{1}{2}\langle \mathcal{D}^{-1}, \mathcal{T} \circ H(\frac{\partial^2 h}{\partial x_i \partial x_j})\rangle. \end{aligned} \tag{6.11}$$

To compute the Hessian more efficiently by matrix multiplication, we use the Hankel and Toeplitz operators. In order to do that, we need the second derivative of (6.7), which is as follows:

$$DDh = \begin{pmatrix} \int_a^b \frac{1}{x(t)^4} dt \\ \vdots \\ \int_a^b \frac{t^{4n-2}}{x(t)^4} dt \end{pmatrix}_{(4n-1)\times 1} . \tag{6.12}$$

6.4 Numerical Results in Standard Basis

We have developed and implemented a non-symmetric HSD predictor-corrector algorithm (we refer to it as uPCO-uMCO) for the model in (HSD NP-M) using the gradient and Hessian in (6.9) and (6.11), respectively.

To show the numerical result of this implementation and compare it with the one developed in the previous chapter, we consider following two packages:

- **uMCO-uPCO:** This package is developed to solve the HSD model in (HSD M-NP) in standard basis.
- **uPCO-uMCO:** This package is developed to solve the HSD model in (HSD NP-M) in standard basis.

We have considered the set of test problems from the class of maximum network flow problems with univariate input data which was explained in Section 5.8. For each test problem, once it was formulated as a standard non-negative polynomial conic

optimization, i.e. (Max-Flow), and then solved by the uPCO-uMCO package. Next, the same problem was formulated as a standard moment conic optimization, i.e. (Dual Max-Flow), and then solved by the uMCO-uPCO package.

The numerical results are given in Table 6.1, where the columns are divided to six blocks:

1. **Package:** Shows the name of the package that is used to solve the test problems in each formulation.

2. **Net. Inf.:** Shows the general information about the network flow. Specifically, "V", "E" and "deg" show the number of the vertices, the number of the edges and the degree of the flows, respectively.

3. **Formulation Dim.** Shows the dimension of the primal problem of each formulation in the standard form. Specifically, "blk", "m" and "n" show the number of blocks, the number of constraints and the number of variables, respectively.

4. **Status:** Shows the status of solving the problem by the corresponding package. Specifically, "Y" means the package was able to solve the problem with desired accuracy, and "N" means the package failed to solve the problem due to numerical issues.

5. **Perform. Inf.:** Shows the general performance information of each package. Specifically, "it." shows the number of iterations to reduce the primal and dual infeasibility and the duality gap by a factor of 10^{-6}, "A.C" shows the average number of centering phase per each problem, and "time" shows the running time (in second) that is needed by the package to reach the desired accuracy.

6. **SDP:** Shows the number of iterations needed by Mosek to solve the semidefinite representation of each formulation.

Package	Net. Inf.			Formulation Dim.			Status	Perform. Inf.			SDP
	V	E	deg	blk	m	n		it.	A.C.	time	it.
uPCO-uMCO	3	2	5	4	18	24	Y	7	1.1	217	12
uMCO-uPCO	3	2	5	14	40	112	Y	10	1.4	2	7
uPCO-uMCO	4	5	7	10	56	80	Y	8	1.2	128	10
uMCO-uPCO	4	5	7	14	40	112	Y	10	1.1	8	8
uPCO-uMCO	6	8	7	16	96	128	Y	8	1.6	232	13
uMCO-uPCO	6	8	7	24	64	192	Y	9	1.6	13	12
uPCO-uMCO	7	12	9	24	170	240	Y	9	1.8	483	10
uMCO-uPCO	7	12	9	34	120	340	Y	9	1.6	26	9
uPCO-uMCO	9	16	9	32	230	320	Y	11	1.8	802	10
uMCO-uPCO	9	16	9	46	160	460	Y	10	1.5	38	9
uPCO-uMCO	11	26	9	52	350	520	Y	9	1.6	1010	14
uMCO-uPCO	11	26	9	70	260	700	Y	9	1.6	67	9
uPCO-uMCO	14	42	9	84	540	840	Y	9	1.8	1720	14
uMCO-uPCO	14	42	9	108	420	1080	Y	10	1.6	96	9
uPCO-uMCO	14	42	13	84	756	1176	N	–	–	–	11
uMCO-uPCO	14	42	13	108	588	1512	Y	12	1.3	183	10
uPCO-uMCO	17	26	52	7	328	416	Y	9	1.7	907	13
uMCO-uPCO	17	26	82	7	208	656	Y	11	1.5	53	9
uPCO-uMCO	17	26	52	9	410	520	N	–	–	–	10
uMCO-uPCO	17	26	82	9	260	820	Y	12	1.2	71	9
uPCO-uMCO	32	56	112	9	860	1120	N	–	–	–	10
uMCO-uPCO	32	56	172	9	560	1720	Y	12	1.7	201	15
uPCO-uMCO	56	91	182	7	1176	1456	N	–	–	–	24
uMCO-uPCO	56	91	249	7	728	2352	Y	14	2	383	23

Table 6.1: uPCO-uMCO vs uMCO-uPCO in Standard Basis

Two important observations that can be noticed from Table 6.1 are:

- **Running time:** The running time of uMCO-uPCO is much smaller than the one from uPCO-uMCO. This is because of the computation of the integrals in (6.7), (6.10) and (6.12).

- **Numerical issue:** It can be seen that when the degree of the flows or the dimension of the problem gets large, uPCO-uMCO encounters numerical issues. Therefore, the package is not able to converge to an optimal solution with the desired accuracy. For example, consider the rows related to the problem with a network of $V = 14$ and $E = 26$. When the degree of the flows is small ($deg = 9$), uPCO-uMCO is able to solve the problem, but when the degree is increased ($deg = 13$), uPCO-uMCO does not converge. Alternatively, consider the rows related to the last three problems. These show that when the dimension of the network increases, once again, uPCO-uMCO is not able to solve the problem.

Next, we further investigate these challenges.

6.5 Challenges

As discussed earlier, uPCO-uMCO has certain drawbacks compared to uMCO-uPCO. The drawbacks are related to computing the gradient and Hessian of the barrier function of the non-negative polynomial cone. Computing the gradient and Hessian in standard basis has the following major issues:

1. **Numerical issue resulting from the basis:** As we have seen in Chapter 5, polynomials in the standard basis are extremely ill-conditioned. Therefore, computation with large degree polynomials in this basis results in numerical errors.

2. **Numerical issue resulting from computing the integrals:** Computing the integrals in (6.7, 6.10, 6.12), which are the elements in the gradient and Hessian, becomes an ill-posed problem when $x(t)$ gets close to the optimal solution which is on the boundary of $\mathcal{P}^{n^+}_{[a,b]}$. In other words, when $x(t)$ gets close to possess real roots, the integrals become ill-posed problems.

3. **Running time issue:** Computing the integrals by numerical integration are costly which results in long running times. Furthermore, the running time of the integrals in (6.7, 6.10, 6.12) increases as $x(t)$ gets close to the boundary of $\mathcal{P}^{n+}_{[a,b]}$.

As shown in Table 6.1, polynomials of large degree in the standard basis cannot be handled. We explain the other two issues by a numerical example.

For example, consider $x(t) = 2 - 3t + 4t^3$, which is a polynomial of degree 3. If we naively use the functionalities available in Matlab, like "integral()", then as the polynomial approaches the boundary of $\mathcal{P}^{n+}_{[a,b]}$, the magnitude of the integral and the running time increases. These are shown in Figures 6.1 and 6.2.

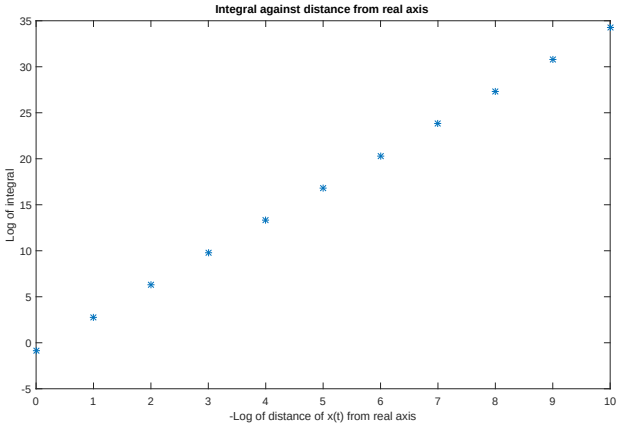

Figure 6.1: log magnitude of $\int_a^b \frac{t}{x(t)^4}$ against -log of distance of x(t) from real axis

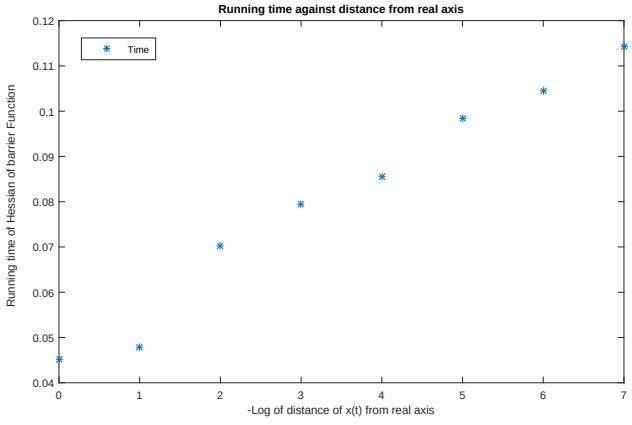

Figure 6.2: Running time of computing the Hessian at $x(t)$ against -log of distance of x(t) from real axis

To overcome these difficulties, we suggest the following remedies:

1. **Change of basis:** As we have seen in the case of moment formulation, the change of basis can alleviate numerical difficulties of working with polynomials of large degree. Among all orthogonal bases, the Chebyshev basis is more favorable. We will examine this in the next section.

2. **Closed-form solution for the integrals:** It was observed previously that computing the integrals in the gradient and the Hessian are ill-posed problems. If these integrals can be computed in a closed-form way, then the numerical issue and running time difficulties of computing the gradient and Hessian will be alleviated. We will explain this in Section 6.9.1.

6.6 Faybusovich LHSCB Function in Chebyshev Basis

Let us consider the Chebyshev system when $u_i(t) = T_i(t)$ for $i = 0, ..., n$ and $t \in [a, b]$ where $T_i(t)$ is the i-th Chebyshev polynomial in the Chebyshev basis. Then, based on theorem 16,

$$F(x) = -\frac{1}{2} \ln \det \mathcal{D}(x), \quad \text{when } x \in \text{Int}(\mathcal{P}_{Ch}^{n+}), \tag{6.13}$$

is a LHSCB function for $\mathcal{P}_{Ch}^{n+}$, where $\mathcal{D}(x)$ is a skew symmetric matrix of size $(n+1) \times (n+1)$. The (i, j)-th entry of this matrix has the form

$$\mathcal{D}_{i,j}(x) = \int_a^b \frac{T_i(t)T_j^{'}(t) - T_i^{'}(t)T_j(t)}{x(t)^2} dt, \quad i, j = 0, ..., n. \tag{6.14}$$

To drive the gradient and Hessian of F, we need the following facts:

$$(T_i)'(t) = i.U_{i-1} \tag{6.15}$$

$$T_i(t)U_j(t) = \begin{cases} \frac{1}{2}(U_{i+j}(t) + U_{j-i}(t)) & \text{if } j \geq i-1 \\ \frac{1}{2}(U_{i+j}(t) - U_{i-j-2}(t)) & \text{if } j \leq i-2 \end{cases} \tag{6.16}$$

where $U_i(t)$ is the i-th Chebyshev polynomial of the second kind.

Using (6.15) and (6.16), matrix $\mathcal{D}(x)$ in (6.17) can be written as:

$$\mathcal{D}(x) = \mathcal{T} \circ H(h(x)) + \mathcal{H} \circ T([h(x)]_{i=0}^{n-1}), \tag{6.17}$$

where H and T are the Hankel and Toeplitz operators:

$$\mathcal{T} = \begin{pmatrix} 0 & \frac{1}{2} & \dots & \frac{n}{2} \\ -\frac{1}{2} & 0 & \ddots & \frac{n-1}{2} \\ \vdots & \ddots & \ddots & \vdots \\ -\frac{n}{2} & -\frac{n-1}{2} & \dots & 0 \end{pmatrix} \tag{6.18}$$

$$\mathcal{H} = \begin{pmatrix} 0 & \frac{1}{2} & \dots & \frac{n}{2} \\ -\frac{1}{2} & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & \frac{2n-1}{2} \\ -\frac{n}{2} & \dots & -\frac{2n-1}{2} & 0 \end{pmatrix} \tag{6.19}$$

$$h(x) = \begin{pmatrix} 0 \\ \int_a^b \frac{U_0(t)}{x(t)^2} dt \\ \vdots \\ \int_a^b \frac{U_{2n-2}(t)}{x(t)^2} dt \\ 0 \end{pmatrix}_{2n+1\times 1} . \tag{6.20}$$

Again, to simplify the notations, we use h and $\mathcal{D}$ instead of $h(x)$ and $\mathcal{D}(x)$, respectively.

6.7 Gradient and Hessian of Faybusovich Barrier Function in Chebyshev Basis

The gradient of (6.13) with $\mathcal{D}(x)$ in (6.17) can be computed as:

$$\mathbf{g}_{x_i} := \nabla_{x_i} F = -\frac{1}{2}\langle \mathcal{D}^{-1}, \mathcal{T} \circ H(\frac{\partial h}{\partial x_i}) + \mathcal{H} \circ T(\frac{\partial h}{\partial x_i})\rangle. \tag{6.21}$$

After algebraic manipulations, the gradient can be written as:

$$\mathbf{g}_x = \frac{1}{2} HT_1[H^*(\mathcal{D}^{-1} \circ \mathcal{T})]_1^{2n} + \frac{1}{2} HT_2[T^*(\mathcal{D}^{-1} \circ \mathcal{H})]_1^n, \tag{6.22}$$

where

$$HT_1 = H([Dh]_0^n, [Dh]_n^{3n-2}) + T(\begin{bmatrix} Dh_0 \\ 0 \\ -[Dh]_0^{n-2} \end{bmatrix}, [Dh]_0^{2n-2}), \tag{6.23}$$

$$HT_2 = H([Dh]_0^n, [Dh]_n^{2n-1}) + T(\begin{bmatrix} Dh_0 \\ 0 \\ -[Dh]_0^{n-2} \end{bmatrix}, [Dh]_0^{n-1}), \tag{6.24}$$

and

$$Dh = \begin{pmatrix} \int_a^b \frac{U_0(t)}{x(t)^3} dt \\ \vdots \\ \int_a^b \frac{U_{3n-2}(t)}{x(t)^3} dt \end{pmatrix}_{3n-1\times 1}. \tag{6.25}$$

Furthermore, the (i,j)-th entry of Hessian can be computed as:

$$\begin{aligned}[\mathbf{H}_x]_{i,j} =& -\frac{1}{2}\langle -\mathcal{D}^{-1}(\mathcal{T}\circ H(\frac{\partial h}{\partial x_j}) + \mathcal{H}\circ T(\frac{\partial h}{\partial x_j}))\mathcal{D}^{-1}, \mathcal{T}\circ H(\frac{\partial h}{\partial x_i}) + \mathcal{H}\circ T(\frac{\partial h}{\partial x_i})\rangle \\ & -\frac{1}{2}\langle \mathcal{D}^{-1}, \mathcal{T}\circ H(\frac{\partial^2 h}{\partial x_i \partial x_j}) + \mathcal{H}\circ T(\frac{\partial^2 h}{\partial x_i \partial x_j})\rangle. \end{aligned} \tag{6.26}$$

To compute the Hessian more efficiently by matrix multiplication, we use the Hankel and Toeplitz operators. In order to do that, we need the second derivative of (6.20) as:

$$DDh = \begin{pmatrix} \int_a^b \frac{U_0(t)}{x(t)^4} dt \\ \vdots \\ \int_a^b \frac{U_{4n-2}(t)}{x(t)^4} dt \end{pmatrix}_{4n-1\times 1}. \tag{6.27}$$

We will see the numerical results of this change of basis in the next section.

6.8 Numerical Results in Chebyshev Basis

To show the numerical results in Chebyshev basis, we have considered the package uPCO-uMCO which was developed in Section 6.4. But, this time the package is adapted for the Chebyshev basis, i.e. using the barrier function in (6.13), the gradient in (6.22) and the Hessian in (6.26). Furthermore, we compare the performance of this package with the one which was developed in the previous chapter, i.e. uMCO-uPCO for the Chebyshev basis.

The set of test problems are generated similar to the ones in Section 6.4 but this time the capacities are in $\mathcal{P}_{Ch}^{n^+}$. The numerical results are given in Table 6.2, and column definitions are similar to Table 6.1.

Package	Net. Inf.			Formulation Dim.			Status	Perform. Inf.			SDP
	V	E	deg	blk	m	n		it.	A.C	time	it.
uPCO-uMCO	3	2	9	4	30	40	Y	7	1	37	11
uMCO-uPCO	3	2	9	6	20	60	Y	7	1	6	10
uPCO-uMCO	3	2	19	4	60	80	Y	8	1	92	7
uMCO-uPCO	3	2	19	6	40	120	Y	7	1	7	8
uPCO-uMCO	3	2	59	4	180	240	Y	8	1	635	8
uMCO-uPCO	3	2	59	6	120	360	Y	8	1	22	8
uPCO-uMCO	6	8	9	16	120	160	Y	9	1	157	10
uMCO-uPCO	6	8	9	24	80	240	Y	8	1	23	11
uPCO-uMCO	6	8	16	16	240	320	Y	9	1	452	12
uMCO-uPCO	6	8	16	24	160	480	Y	9	1	33	13
uPCO-uMCO	6	8	39	16	480	640	Y	8	1	1130	12
uMCO-uPCO	6	8	39	24	320	960	Y	10	1	76	13

Table 6.2: uPCO-uMCO vs uMCO-uPCO in Chebyshev Basis

By comparing Tables 6.1 and 6.2, we observe that:

- Polynomials of larger degree can be handled in the Chebyshev basis.
- The average number of centering (corrector) phases are less in the Chebyshev basis. This is due to the effect of the change of basis. In fact, this change of basis rescales the neighborhood of the central path and makes the central path more symmetric. Therefore, less centering phases are needed.

6.9 Implicit Solution for Integrals

As we have seen, the computation of the integrals in (6.7, 6.10, 6.12), or in (6.20, 6.25, 6.27) is required for computing the gradient and Hessian of the barrier function for standard or Chebyshev bases. The computation of these integrations, however, is troublesome both numerically and in terms of the running time. Therefore, if these

integrations are formulated as closed-form or at least partially as closed-form, then the computation of the gradient and Hessian can be done in a more stable manner and with less time.

6.9.1 Closed-Form Solution for Integrals

In terms of closed-form integration, Faybusovich [16] considered the following two cases:

1. Chebyshev system $\{1, u_1\}$ for any interval $[a, b]$. Then, the barrier function for the cone induced by this system is:

$$F(x) = -\ln x(a) - \ln x(b) + \ln |U_1(b) - u_1(a)|. \tag{6.28}$$

   In our case, this system is corresponding to $\{1, t\}$ and the induced cone is $\mathcal{P}^{1+}_{[a,b]}$.

2. Chebysev system $\{1, sin\ t, cos\ t\}$ on $[0, 2\pi]$. Then the barrier function for the cone induced by this system is:

$$F(x) = -\ln(x_0^2 - x_1^2 - x_2^2) + \frac{2}{3}\ln(2\pi), \tag{6.29}$$

   where x_0, x_1 and x_2 are the coefficients of x, i.e. $x(t) = x_0 + x_1 sin\ t + x_2 cos\ t$.

It can be seen that, the gradient and Hessian of the barrier based on these bases can be calculated in a closed-form way, (see Section 4 in [16]). But these systems are not of a practical interest. In the best of our knowledge, a Chebyshev system, in which the integrals have an implicit solution, and is practically interesting, is not known.

6.9.2 Partially Closed-From Solution for Integrals

As we have observed, there does not exist an interesting Chebyshev system with a closed-form integration. But there might be ways that the integrals in (6.7, 6.10, 6.12) or similarly (6.20, 6.25, 6.27) can be computed at least partially in a closed-form way.

For example, consider the integrals in (6.7, 6.10, 6.12). One can use a partial fraction of the integrand in the integrals. Therefore, the integrand can be decomposed into fractions with the residues in the numerators and $(t - poles)$ in the denominators.

This means:

$$\frac{t^i}{(x_0 + x_1 t + ... + x_n t^n)^k} = \sum_{j=1}^{l} \frac{r_j}{(t - p_j)^{m_j}} \tag{6.30}$$

where if $x(t)$ has root of multiplicity, say m_j, then the summation in (6.30) contains all powers of that root.

Applying the integral on both sides of (6.30), results in:

$$\int_{-1}^{1} \frac{t^i}{(x_0 + x_1 t + ... + x_n t^n)^k} = \sum_{j=1}^{l_1} \int_{-1}^{1} \frac{r_j}{t - p_j} + \sum_{j=1}^{l_2} \int_{-1}^{1} \frac{r_j}{(t - p_j)^{m_j}} \tag{6.31}$$

where $m_j > 1$.

Now, if we combine the fractions with the conjugate poles and with the same power, we will have:

$$(6.31) = \sum_{j=1}^{l_1'} \int_{-1}^{1} \frac{a_j t + b_j}{(t - u_j)^2 + v_j^2} + \sum_{j=1}^{l_2'} \int_{-1}^{1} \frac{a_{m_j} t^{m_j} + ... + a_{0_j}}{((t - u_j)^2 + v_j^2)^{m_j}} \tag{6.32}$$

From calculus, we know that:

$$\int \frac{1}{t^2 + 1} dt = \arctan(t) + constant \tag{6.33}$$

$$\int \frac{t^j}{(t^2 + 1)^k} dt = \frac{t^{1+j} {}_2F_1(\frac{1+j}{2}, k; 1 + \frac{1+j}{2}; -t^2)}{1 + j} + constant \tag{6.34}$$

where ${}_2F_1(a, b; c; z)$ is the Gauss Hypergeometric function.

Hypergeometric functions are well-known functions, and are defined as:

$${}_2F_1(a, b; c; z) = \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{(c)_k k!} z^k, \tag{6.35}$$

where $(a)_k = \Gamma(a + k)/\Gamma(a)$ (with the same relationship for $(b)_k$ and $(c)_k$). Successful numerical methods exist to compute the hypergeometric functions by Taylor series expansion, efficiently and in a stable way. For example, refer to the R package "hpergeo" by Hankin [22], the work by Pearson and the Matlab code implementation [43].

The right hand side of Equation (6.32), can be analytically computed considering the Equations (6.33) and (6.34).

To some extent, this alleviates the issues mentioned earlier. For example, consider the polynomial mentioned in Section 6.5. Figure 6.3 depicts the log magnitude of

the residues and the log magnitude of $\int_{-1}^{1} \frac{t}{x(t)^4}$. It is clear that the magnitude of the integrals are on the same scale as the magnitude of the residues. This is because, the integrals are a linear combination of ${}_2F_1$ plus a linear combination of arctan, where the coefficients of the linear combination are the residues. As we can see, the magnitude of the residues behave better than the magnitude of the integrals.

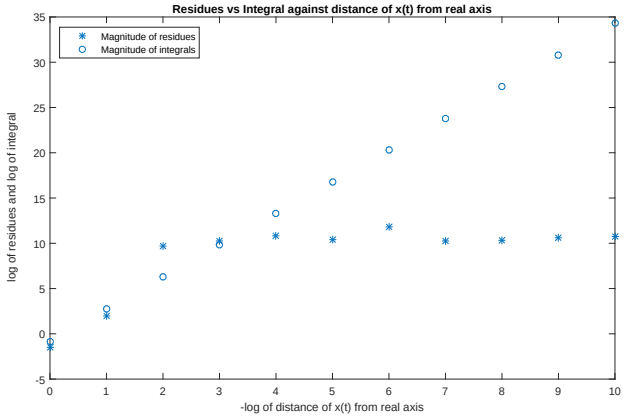

Figure 6.3: Log magnitude of residues and integral of $\frac{t}{x(t)^4}$ against -Log of distance of x(t) from real axis

If we naively use the functionalities available in Matlab, like "residue()", then as the polynomial approaches towards the boundary of the non-negative polynomial cone, the magnitude of the residues increases, as shown in Figure 6.3. This is because, the problem of the partial fraction itself is an ill-posed problem, especially, if the degree of the multiplicity or the degree of the polynomial in the denominator is large. Therefore, even a small change in the data, including round-off errors, can result in arbitrarily large changes in the resulting poles and residues. One way to alleviate this issue is, to make use of the state-space or the zero-pole representations for finding the poles and the residues. The state-space representations are beyond the scope of this work and we leave it as an open area of research.

Furthermore, when we want to compute the following integrations:

$$F(x,i,k) = \int_{-1}^{1} \frac{t^i}{(x_0 + x_1 t + ... + x_n t^n)^k} dt, \tag{6.36}$$

where $x \in \mathrm{Int}(\mathcal{P}^{n^+}_{[-1,1]})$ and i, k are:

$$i = \begin{cases} 0, ..., 2n-1 & k = 2, \\ 0, ..., 3n-1 & k = 3, \\ 0, ..., 4n-1 & k = 4. \end{cases}$$

Then the questions that are open for future research are:

1. Is there any way that the integrals can be computed analytically?

2. Is there any way that the integrals can be computed inductively? Namely, given a fixed $x \in \mathrm{Int}(\mathcal{P}^{n^+}_{[-1,1]})$, is there any way $F(x, i, k)$ can be recursively computed base on knowing $F(x, i-1, k)$, $F(x, i, k-1)$ and $F(x, i-1, k-1)$ or other previous terms?

6.10 Conclusion

We have seen that formulating a non-negative polynomial conic optimization as (HSD M-NP) model and then solving it by the uMCO-uPCO package, which uses the moment cone barrier function, is more stable and more efficient. But formulating the original problem as (HSD NP-M) model and then solving it by uPCO-uMCO package, which uses the Faybusovich barrier function of the non-negative polynomial cone has its own advantages. Particularly, a larger class of optimization problems containing the non-negative polynomial conic constraints can be solved. For example, consider the non-negative polynomial conic optimization problem where the objective function is a nonlinear convex function, e.g. the negative of maximum log likelihood function.

Chapter 7

Conic Optimization Containing Non-Negative Polynomial or Moment Constraints as well as Second Order and Linear Constraints

As we discussed in Chapter 4, many real-world problems can be formulated as a conic optimization problem containing non-negative polynomial or moment, as well as second-order and linear constraints. For example, non-parametric estimation under shape constraints with splines, i.e. (Pos-Estim), which is conic optimization with non-negative polynomial, second-order and linear constraints. The dual of these problems are conic optimizations containing moment, second-order and linear constraints.

In this chapter, a unified interior point method for these conic optimizations will be discussed. We will show the Newton system and Schur complement are similar to the one of symmetric conic optimization, and therefore, can be computed efficiently. Finally, we will show numerical results of our implementations and point out some observations.

7.1 General Conic Optimization

In general, a conic optimization problem with non-negative polynomial, second-order and linear constraints can be formulated as follows:

$$\begin{aligned} \min \quad & c_l^T x_l + \sum_{j=1}^{N_{\mathcal{S}}} c_{s_j}^T x_{s_j} + \sum_{k=1}^{N_{\mathcal{P}^+}} c_{\mathcal{P}_k^+}^T x_{\mathcal{P}_k^+} \\ s.t. \quad & A_l x_l + \sum_{j=1}^{N_{\mathcal{S}}} A_{s_j} x_{s_j} + \sum_{k=1}^{N_{\mathcal{P}^+}} A_{\mathcal{P}_k^+} x_{\mathcal{P}_k^+} = b, \\ & x_l \geq 0,\ x_{s_j} \succcurlyeq_{\mathcal{S}^{n_j}} 0 \ \ \forall j = 1, ..., N_{\mathcal{S}},\ x_{\mathcal{P}_k^+} \succcurlyeq_{\mathcal{P}^{n_k^+}_{[a_k,b_k]}} 0 \ \ \forall k = 1, ..., N_{\mathcal{P}^+}. \end{aligned} \tag{7.1}$$

To simplify the notation, let us use a Matlab-like notation for matrix/vector concatenations:

$$\begin{aligned}
A_s &= [A_{s_1}, ..., A_{s_{N_{\mathcal{S}}}}], \qquad & A_{\mathcal{P}^+} &= [A_{\mathcal{P}_1^+}, ..., A_{\mathcal{P}^+_{N_{\mathcal{P}^+}}}] \\
c_s &= [c_{s_1}; ...; c_{s_{N_{\mathcal{S}}}}], \qquad & c_{\mathcal{P}^+} &= [c_{\mathcal{P}_1^+}; ...; c_{\mathcal{P}^+_{N_{\mathcal{P}^+}}}] \\
x_s &= [x_{s_1}; ...; x_{s_{N_{\mathcal{S}}}}], \qquad & x_{\mathcal{P}^+} &= [x_{\mathcal{P}_1^+}; ...; x_{\mathcal{P}^+_{N_{\mathcal{P}^+}}}] \\
\mathcal{S} &= \mathcal{S}_{n_1} \otimes ... \otimes \mathcal{S}_{n_{N_s}}, \qquad & \mathcal{P}^+ &= \mathcal{P}^{n_1^+}_{[a_1,b_1]} \otimes ... \otimes \mathcal{P}^{n^+_{N_{\mathcal{P}^+}}}_{[a_{N_{\mathcal{P}^+}},b_{N_{\mathcal{P}^+}}]}.
\end{aligned}$$

Using this notation, problem (7.1) can be shown as:

$$\begin{aligned}
\min \quad & c_l^T x_l + c_s^T x_s + c_{\mathcal{P}^+}^T x_{\mathcal{P}^+} \\
s.t. \quad & A_l x_l + A_s x_s + A_{\mathcal{P}^+} x_{\mathcal{P}^+} = b, \\
& x_l \geq 0, x_s \succcurlyeq_{\mathcal{S}} 0, \ x_{\mathcal{P}^+} \succcurlyeq_{\mathcal{P}^{n^+}} 0.
\end{aligned} \tag{NP-S-L}$$

As discussed in Chapter 6, conic optimization over the non-negative polynomial cone, suffers from large running time and numerical instability, unless a stable Chebyshev system with closed-form integration is available. However, in Chapter 5, with numerical results we have also shown that conic optimization over the moment cone is practically very efficient and stable. Therefore, using the duality between the non-negative polynomial cone and the moment cone and the self-duality of symmetric cones, one can attempt to solve the dual problem instead of solving (NP-S-L). To start from standard conic optimization, the dual problem can be considered as:

$$\begin{aligned}
\min \quad & c_l^T x_l + c_s^T x_s + c_{\mathcal{M}}^T x_{\mathcal{M}} \\
s.t. \quad & A_l x_l + A_s x_s + A_{\mathcal{M}} x_{\mathcal{M}} = b, \\
& x_l \geq 0, x_s \succcurlyeq_{\mathcal{S}} 0, x_{\mathcal{M}} \succcurlyeq_{\mathcal{M}} 0.
\end{aligned} \tag{M-S-L}$$

Next, we will try to develop a primal-dual interior point method for this problem, and show that having non-symmetric cones in the conic optimization problem will not ruin the structure of the Newton system in the interior point method computations.

7.2 A Unified HSD IPM

As it was shown in the previous chapters, to develop a HSD primal-dual IPM, the primal as well as the dual problems are needed. Therefore, the dual problem of (M-S-L) in the standard form can be formulated as:

$$\begin{aligned} \max \quad & b^T \ y \\ s.t. \quad & A_l^T \ y + s_l = c_l, \\ & A_s^T \ y + s_s = c_s, \\ & A_{\mathcal{M}}^T y + s_{\mathcal{P}^+} = c_{\mathcal{M}}, \\ & y \ free, s_l \geq 0, s_s \succcurlyeq_{\mathcal{S}} 0, s_{\mathcal{P}^+} \succcurlyeq_{\mathcal{P}^+} 0. \end{aligned} \qquad \text{(Dual M-S-L)}$$

Assuming that strong duality holds, then the first order optimality says that $(x_l, x_s, x_{\mathcal{M}}) \in \mathcal{L}^n \times \mathcal{S}^n \times \mathcal{M}$ and $(s_l, s_s, s_{\mathcal{P}^+}) \in \mathcal{L}^n \times \mathcal{S}^n \times \mathcal{P}^+$ are the optimal solutions for the primal and dual problems, respectively, if the following equations hold:

$$\begin{aligned} A_l x_l + A_s x_s + A_{\mathcal{M}} x_{\mathcal{M}} &= b, \\ A_l^T y + s_l &= c_l, \\ A_s^T y + s_s &= c_s, \\ A_{\mathcal{M}}^T y + s_{\mathcal{P}^{n+}} &= c_{\mathcal{M}}, \\ x_l^T s_l &= 0, \\ x_s \circ s_s &= 0, \\ x_{\mathcal{M}}^T s_{\mathcal{P}^+} &= 0. \end{aligned}$$

With the same analogy to Section 5.6, the homogeneous self-dual problem for (M-S-L) and (Dual M-S-L) can be formulated as:

$$
\begin{aligned}
\min \quad & 0 \\
s.t. \quad & A_l x_l + A_s x_s + A_{\mathcal{M}} x_{\mathcal{M}} - b\tau = 0, \\
& A_l^T y + c_l \tau - s_l = 0, \\
& A_s^T y + c_s \tau - s_s = 0, \\
& A_{\mathcal{M}}^T y + c_{\mathcal{M}} \tau - s_{\mathcal{P}^+} = 0, \qquad \text{(HSD M/NP-S-L)} \\
& b^T y - c_l^T x_l - c_s^T x_s - c_{\mathcal{P}^+}^T x_{\mathcal{P}^+} - \kappa = 0, \\
& (x_l, x_s, x_{\mathcal{M}}, \tau) \in \mathcal{L} \times \mathcal{S} \times \mathcal{M} \times \mathbb{R}^+, \\
& (s_l, s_s, s_{\mathcal{P}^+}, \kappa) \in \mathcal{L} \times \mathcal{S} \times \mathcal{P}^+ \times \mathbb{R}^+, \; y \in \mathbb{R}^m.
\end{aligned}
$$

Similar to (CP) in Chapter 5, the central path of this model can be defined. Then, given a strictly feasible (with respect to the cones) initial point, the Newton search direction is defined by the solution of the following system of linear equations:

$$
\begin{pmatrix}
A_l & A_s & A_{\mathcal{M}} & -b & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -c_l & A_l^T & I & 0 & 0 & 0 \\
0 & 0 & 0 & -c_s & A_s^T & 0 & I & 0 & 0 \\
0 & 0 & 0 & -c_{\mathcal{M}} & A_{\mathcal{M}}^T & 0 & 0 & I & 0 \\
-c_l^T & -c_s^T & -c_{\mathcal{M}}^T & 0 & b^T & 0 & 0 & 0 & -1 \\
S_l & 0 & 0 & 0 & 0 & X_l & 0 & 0 & 0 \\
0 & Arw(s) & 0 & 0 & 0 & 0 & Arw(x) & 0 & 0 \\
0 & 0 & \mu H_{x_{\mathcal{M}}} & 0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & \kappa & 0 & 0 & 0 & 0 & \tau
\end{pmatrix}
\begin{pmatrix}
dx_l \\ dx_s \\ dx_{\mathcal{M}} \\ d\tau \\ dy \\ ds_l \\ ds_s \\ ds_{\mathcal{P}^+} \\ d\kappa
\end{pmatrix}
=
\begin{pmatrix}
\eta . r_p \\ \eta . r_{d_l} \\ \eta . r_{d_s} \\ \eta . r_{d_{\mathcal{P}^+}} \\ \eta . r_g \\ -X_l s_l + \sigma\mu e_l \\ -Arw(x_s) s_s + 2\sigma\mu e_s \\ -s_{\mathcal{P}^+} - \sigma\mu g_{x_{\mathcal{M}}} \\ -\tau\kappa + \sigma\mu
\end{pmatrix}
\tag{7.2}
$$

Now, we multiply the sixth, seventh and ninth rows by X_l^{-1}, $Arw(x_s)^{-1}$ and τ^{-1}, respectively, and subtract them from the second, third and fifth rows, and then subtracts the forth row from the eighth. Next, we delete the last four rows and four columns.

Finally, by reordering rows and columns, we get the following system of linear equations:

$$\begin{pmatrix} -X_l^{-1}S_l & 0 & 0 & A_l^T & -c_l \\ 0 & -Arw(x_s)^{-1}Arw(s_s) & 0 & A_s^T & -c_s \\ 0 & 0 & -\mu H_{x_\mathcal{M}} & A_\mathcal{M}^T & -c_\mathcal{M} \\ A_l & A_s & A_\mathcal{M} & 0 & b \\ -c_l^T & -c_s^T & -c_\mathcal{M}^T & b^T & \kappa/\tau \end{pmatrix} \begin{pmatrix} dx_l \\ dx_s \\ dx_\mathcal{M} \\ dy \\ d\tau \end{pmatrix} = \begin{pmatrix} \eta.r_{d_l} + s_l - \sigma\mu X_l^{-1}e_l \\ \eta.r_{d_s} + s_s - 2\sigma\mu Arw(x_s)^{-1}e_s \\ \eta.r_{d_\mathcal{M}} + s_{\mathcal{P}^+} + \sigma\mu g_{x_\mathcal{M}} \\ \eta.r_p \\ \eta.r_g - \kappa + \sigma\mu/\tau \end{pmatrix} \tag{7.3}$$

To solve (7.3) for $(dx_l, dx_s, dx_\mathcal{M}, dy, d\tau)$, we first solve it for $d\tau$, which results in:

$$d\tau = \frac{\begin{pmatrix} c^T, -b^T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \eta.r_g - \kappa + \sigma\mu/\tau}{\begin{pmatrix} -c^T, b^T \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} + \kappa/\tau} \tag{7.4}$$

where $c = [c_l; c_s; c_\mathcal{M}]$ and $u = [u_l; u_s; u_\mathcal{M}]$ and v are the solutions of the following system of linear equations:

$$\begin{pmatrix} -X_l^{-1}S_l & 0 & 0 & A_l^T \\ 0 & -Arw(x_s)^{-1}Arw(s_s) & 0 & A_s^T \\ 0 & 0 & -\mu H_{x_\mathcal{M}} & A_\mathcal{M}^T \\ A_l & A_s & A_\mathcal{M} & 0 \end{pmatrix} \begin{pmatrix} u_l \\ u_s \\ u_\mathcal{M} \\ v \end{pmatrix} = \begin{pmatrix} \eta.r_{d_l} + s_l - \sigma\mu X_l^{-1}e_l \\ \eta.r_{d_s} + s_s - 2\sigma\mu Arw(x_s)^{-1}e_s \\ \eta.r_{d_\mathcal{M}} + s_{\mathcal{P}^+} + \sigma\mu g_{x_\mathcal{M}} \\ \eta.r_p \end{pmatrix} \tag{7.5}$$

and $p = [p_l; p_s; p_\mathcal{M}]$ and q are the solutions of the following system of linear equations:

$$\begin{pmatrix} -X_l^{-1}S_l & 0 & 0 & A_l^T \\ 0 & -Arw(x_s)^{-1}Arw(s_s) & 0 & A_s^T \\ 0 & 0 & -\mu H_{x_\mathcal{M}} & A_\mathcal{M}^T \\ A_l & A_s & A_\mathcal{M} & 0 \end{pmatrix} \begin{pmatrix} p_l \\ p_s \\ p_\mathcal{M} \\ q \end{pmatrix} = \begin{pmatrix} c_l \\ c_s \\ c_\mathcal{M} \\ b \end{pmatrix}. \tag{7.6}$$

Notice that the left hand side of (7.5) and (7.6) are the same matrices. Therefore, to solve (7.5) and (7.6), factorization of this matrix is needed once.

The Schur complement of (7.5) and (7.6) can be computed as:

$$\begin{aligned} Schur =& A_l(X_l^{-1}S_l)^{-1}A_l^T + \\ & A_s(Arw(x_s)^{-1}Arw(s_s))^{-1}A_s^T + \\ & A_\mathcal{M}(\mu H_{x_\mathcal{M}})^{-1}A_\mathcal{M}^T. \end{aligned} \tag{7.7}$$

It is not hard to show that:

$$\begin{aligned} Schur^* v = (A_l(X_l^{-1}S_l)^{-1}(\eta.r_{d_l} + s_l - \sigma\mu X_l^{-1}e_l)+ \\ A_s(Arw(x_s)^{-1}Arw(s_s))^{-1}(\eta.r_{d_s} + s_s - 2\sigma\mu Arw(x_s)^{-1}e_s)+ \\ A_{\mathcal{M}}(\mu H_{x_{\mathcal{M}}})^{-1}(\eta.r_{d_{\mathcal{M}}} + s_{\mathcal{P}^+} + \sigma\mu g_{x_{\mathcal{M}}}) + \eta.r_p), \end{aligned} \tag{7.8}$$

and

$$\begin{aligned} Schur^* q = (A_l(X_l^{-1}S_l)^{-1}c_l+ \\ A_s(Arw(x_s)^{-1}Arw(s_s))^{-1}c_s+ \\ A_{\mathcal{M}}(\mu H_{x_{\mathcal{M}}})^{-1}c_{\mathcal{M}} + b). \end{aligned} \tag{7.9}$$

Having u, v, p and q in hand, then

$$\begin{pmatrix} dx_l \\ dx_s \\ dx_{\mathcal{M}} \\ dy \end{pmatrix} = \begin{pmatrix} u_l \\ u_s \\ u_{\mathcal{M}} \\ v \end{pmatrix} + \begin{pmatrix} p_l \\ p_s \\ p_{\mathcal{M}} \\ q \end{pmatrix} d\tau. \tag{7.10}$$

All prior equations are similar to the case of symmetric IPM for symmetric cones (e.g., see [8]), except here, we have a new term for the non-symmetric cone. Also, it can be seen that all prior systems of equations can efficiently be computed. Again, this is similar to the symmetric cases. In addition, the structure of each input data can be exploited in the computation of Schur, v and q. One may wonder if a Nesterov-Todd scaling technique can be used with a non-symmetric cone. This is an open area of research.

Next, we will show the numerical results from the implementation of the algorithm above.

7.3 Numerical Results

We developed two packages of M-S-L and P-S-L. In both packages the HSD Mehrotra predictor-corrector IPM has been used. The algorithm in M-S-L package is based on the homogeneous self-dual model in (HSD M/NP-S-L), where the moment cone is in

the primal cone, and the non-negative polynomial cone is in the dual cone. In fact, all formulations in (7.2)-(7.10) are used in this package. The algorithm in P-S-L package is developed based on the same homogeneous self-dual model, except this time, the roles of the moment cone and the non-negative polynomial cone are exchanged. This means that this time, the non-negative polynomial cone is in the primal cone, and the moment cone is in the dual cone.

To show the numerical results of these implementations, we have considered the non-parametric estimation problem under shape constraints with splines, which was formulated in (4.10). To solve the set of test problems by these packages, once we have formulated (4.10) as a standard conic optimization problem containing the non-negative polynomial, second order and linear constraints, i.e. the problem in (NP-S-L). This formulation is given in (N.P.E. Spline). This is the standard format that can be fed to the P-S-L package. Next, the same problem, i.e. (4.10), has been formulated as a standard conic optimization containing the moment, second order and linear constraints, i.e. the problem in (M-S-L). This is the standard format that can be fed to the M-S-L package. Finally, to compare the performance of these two packages to the benchmark, we have used Mosek. In order to solve the test problems by Mosek, we have used the semidefinite presentations of the non-negative polynomial and the moment cones given in Theorems 6 and 7, respectively. Using those theorems, we have formulated each of the previous formulations as a conic optimization containing semidefinite, second order and linear conic constraints.

The numerical results of the implementations of these packages are given in Table 7.1, where the columns are divided into four blocks:

1. **PKG:** Package name that is used to solve the test problems.

2. **Prob. Dim.:** Problem dimension of each formulation. Particularly, “m” and “n” show the number of constraints and the number of decision variables, respectively. “k” shows the number of splines. “nblk.L”, “nblk.S” and “nblk.M/P” show the dimension of positive orthant, the number of second order cones multiplied by the dimension of each cone, and the number of the moment/non-negative polynomial

cones multiplied by the dimension of each cone, respectively.

3. **Less Accurate ($\epsilon = 10^{-4}$):** General performance of each package to reach 10^{-4} accuracy. Particularly, "it." and "time" show the number of iterations and the time (in second) needed by each package to reduce the primal and dual infeasibility and the duality gap to less than 10^{-4}. Also "it./Mosek" shows the number of iterations needed by Mosek to reach the same level of accuracy.

4. **More Accurate ($\epsilon = 10^{-6}$):** General performance of each package to reach 10^{-6} accuracy. Particularly, "it." and "time" show the number of iterations and time (in second) needed by each package to reduce the primal and dual infeasibility and the duality gap to less than 10^{-6}. Also "it./Mosek" shows the number of iterations needed by Mosek to reach the same level of accuracy.

Several observations can be made from the numerical results in Table 7.1:

1. Comparing the performance of packages (M-S-L and P-S-L) shows that, for a less accurate optimal solution, the number of iterations of both packages are not significantly different.

2. The running time of M-S-L package is much smaller than the P-S-L package.

3. In contrast to M-S-L package, for the more accurate optimal solution, P-S-L package ran into numerical issues.

4. Comparing the number of iterations of M-S-L package and Mosek for both accuracies, shows that they are not significantly different except for the first few instances. This is because Mosek benefits substantially from pre-processing and scaling techniques.

5. On the other hand, comparing the number of iterations of P-S-L package and Mosek for the less accurate solution, shows a significant discrepancy, which may suggest possible improvements for the P-S-L package.

PKG	Prob. Dim.						Less Accurate ($\epsilon = 10^{-4}$)			More Accurate ($\epsilon = 10^{-6}$)		
	m	n	k	nblk.L	nblk.S	nblk.M/P	it.	time	it./Mosek	it.	time	it./Mosek
M-S-L	7	107	1	0	1*101	1*6	9	2	5	11	2.3	7
P-S-L	107	115	1	2	1*101	2*6	12	19.6	6	–	–	8
M-S-L	10	116	2	6	2*61	2*4	9	2.3	5	13	2.9	6
P-S-L	113	122	2	4	2*61	4*4	10	18	5	13	25.3	7
M-S-L	20	238	4	18	4*51	4*4	11	2.8	6	15	3.6	8
P-S-L	229	244	4	8	4*51	8*4	10	36	6	–	–	8
M-S-L	40	282	8	42	8*26	8*4	11	3.6	11	14	4.3	13
P-S-L	261	288	8	16	8*26	16*4	16	92.9	7	–	–	9
M-S-L	40	482	8	42	8*51	8*4	13	4	11	16	4.9	13
P-S-L	461	488	8	16	8*51	16*4	17	119	7	–	–	10
M-S-L	50	504	10	54	10*41	10*4	12	3.7	11	16	5	14
P-S-L	477	510	10	20	10*41	20*4	15	122	7	–	–	9
M-S-L	40	882	8	42	8*101	8*4	13	4.1	10	17	5.2	12
P-S-L	881	888	8	16	8*101	16*4	14	112	7	–	–	10
M-S-L	60	925	12	66	12*66	12*4	13	4.8	13	16	5.8	16
P-S-L	892	931	12	24	12*66	24*4	14	142	7	–	–	10
M-S-L	60	1135	12	66	12*84	12*4	13	4.7	13	16	5.8	17
P-S-L	1092	1131	12	24	12*84	24*4	14	152	7	–	–	10
M-S-L	75	1358	15	84	15*80	15*4	13	5.7	11	17	7.2	18
P-S-L	1316	1364	15	30	15*80	30*4	17	240	8	–	–	10
M-S-L	100	1713	20	114	20*75	20*4	16	8.7	13	19	10.3	20
P-S-L	1656	1719	20	40	20*75	40*4	17	397	9	–	–	11
M-S-L	150	2323	30	174	30*67	30*4	18	14.1	13	22	17.3	22
P-S-L	2236	2329	30	60	30*67	60*4	16	651	10	–	–	12

Table 7.1: M-S-L vs P-S-L

7.4 Conclusion

In this chapter, we have seen a unified interior point method for the conic optimization, which contains the non-symmetric cones as well as the symmetric cones. It has been shown that the Newton systems and Schur complement of this conic optimization can be efficiently computed. Also, we have shown the numerical results from the

implementation of this unified interior point method for the conic optimizations containing non-negative polynomial or moment cones as well as other symmetric cones. The numerical results showed that M-S-L package is superior to P-S-L package, and is competitive to the benchmark.

Appendix A

To show the proofs of Lemma 10 and Lemma 11, we need some facts about polynomial multiplication in Chebyshev basis to use them towards the proofs. Let us consider the Chebyshev polynomial basis for $\mathcal{P}^{n^+}_{[a,b]}$ in a vector form as:

$$u^n_{Ch}(t) := (T_0(t), T_1(t), ..., T_n(t))^T \quad \text{when } t \in [a,b]. \tag{A.1}$$

For simplicity of notation, let us drop (t) from $u^n_{Ch}(t)$. It can be seen that:

$$\begin{aligned} u^n_{Ch}u^{n^T}_{Ch} &= \frac{1}{2}\begin{pmatrix} T_0(t) & T_1(t) & \dots & T_n(t) \\ T_1(t) & T_2(t) & \iddots & T_{n+1}(t) \\ \vdots & \iddots & \iddots & \vdots \\ T_n(t) & T_{n+1}(t) & \dots & T_{2n}(t) \end{pmatrix} + \frac{1}{2}\begin{pmatrix} T_0(t) & T_1(t) & \dots & T_n(t) \\ T_1(t) & T_0(t) & \dots & T_{n-1}(t) \\ \vdots & \ddots & \ddots & \vdots \\ T_n(t) & \dots & T_1(t) & T_0(t) \end{pmatrix} \\ &= \frac{1}{2}H(u^{2n}_{Ch}) + \frac{1}{2}T(u^n_{Ch}) \\ &= \frac{1}{2}(H+T)(u^{2n}_{Ch}). \end{aligned} \tag{A.2}$$

Also, we have the following equations:

$$\begin{aligned} &(1-T_1(t))(1+T_1(t))u^{n-1}_{Ch}u^{n-1^T}_{Ch} \\ &\quad= \left(\frac{1-T_2(t)}{2}\right)u^{n-1}_{Ch}u^{n-1^T}_{Ch} \\ &\quad= \left(\frac{1-T_2(t)}{2}\right)\left(\frac{H+T}{2}\right)(u^{2n-1}_{Ch}) \\ &\quad= \frac{1}{2}H([T_i - \tfrac{1}{2}T_{i+2}]^{2n-2}_{i=0} - \tfrac{1}{2}[T_2;T_1;[T_i]^{2n-4}_{i=0}]) + \\ &\quad\quad \frac{1}{2}T([T_i - \tfrac{1}{2}T_{i+2}]^{n-1}_{i=0}] - \tfrac{1}{2}[T_2;T_1;[T_i]^{n-3}_{i=0}]) \\ &\quad= \frac{1}{2}(H+T)(\mathcal{A}_{Ch}u^{2n}_{Ch}), \end{aligned} \tag{A.3}$$

where $\mathcal{A}_{Ch}$ is:

$$\mathcal{A}_{Ch} := \begin{pmatrix} 1 & 0 & -1 & & & \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} & & \\ -\frac{1}{2} & 0 & 1 & 0 & -\frac{1}{2} & \\ & -\frac{1}{2} & 0 & 1 & 0 & -\frac{1}{2} \\ & & \ddots & & \ddots & & \ddots \end{pmatrix}. \tag{A.4}$$

Similarly, we can show:

$$(1+T_1(t))u_{Ch}^n u_{Ch}^{n^T} = \frac{1}{2}(H+T)(\mathcal{B}_{Ch}u_{Ch}^{2n+1}), \tag{A.5}$$

$$(1-T_1(t))u_{Ch}^n u_{Ch}^{n^T} = \frac{1}{2}(H+T)(\mathcal{C}_{Ch}u_{Ch}^{2n+1}), \tag{A.6}$$

where $\mathcal{B}_{Ch}$ and $\mathcal{C}_{Ch}$ are:

$$\mathcal{B}_{Ch} := \begin{pmatrix} 1 & 1 & & & & & \\ \frac{1}{2} & 1 & \frac{1}{2} & & & & \\ & \frac{1}{2} & 1 & \frac{1}{2} & & & \\ & & \frac{1}{2} & 1 & \frac{1}{2} & & \\ & & & \frac{1}{2} & 1 & \frac{1}{2} & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}, \tag{A.7}$$

$$\mathcal{C}_{Ch} := \begin{pmatrix} 1 & -1 & & & & & \\ -\frac{1}{2} & 1 & -\frac{1}{2} & & & & \\ & -\frac{1}{2} & 1 & -\frac{1}{2} & & & \\ & & -\frac{1}{2} & 1 & -\frac{1}{2} & & \\ & & & -\frac{1}{2} & 1 & -\frac{1}{2} & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{A.8}$$

Proof. **Lemma 10**

a. To show part a, define the linear operator Λ in Equation (17.2) in [35] as (A.2). Then the rest of the proof can be followed from part 1 of Theorem 17.1 in [35].

b. To show part b, in Theorem 17.12 in [35] defines the linear operators Λ_1 as (A.2) and Λ_2 as (A.3). In fact, the adjoint operator Λ_2^* in our case is $\frac{1}{2}\mathcal{A}_{Ch}^T(H+T)^*$.

c. Part c can be worked out in a similar manner, using the rhs of (A.5) and (A.6). □

Proof. **Lemma 11**

Considering the facts in (A.2) and (A.3) and Lemma 10, proving Lemma 11 is straightforward. In fact, we have:

a. Considering Λ as (A.2), part a will be followed from part 2 of Theorem 17.1 in [35].

b. Considering Λ_1 and Λ_2 respectively as (A.2) and (A.3), part b will be followed from Theorem 17.12 in [35].

c. Considering Λ_1 and Λ_2 as in the rhs of (A.5) and (A.6), respectively, part c will be followed from Theorem 17.13 in [35]. □

To show the proofs of Lemma 12 and Lemma 13, we need following facts about matrix differentiations. Considering the first order derivatives, we can show:

$$\begin{aligned}\frac{\partial}{\partial x_i}\ln\det H(Ax) &= H^{-1}(Ax)\bullet\left[\frac{\partial H(Ax)}{\partial Ax}\frac{\partial Ax}{\partial x_i}\right]\\ &= a_i^T H_i^*(H^{-1}(Ax)),\end{aligned}\tag{A.9}$$

where a_i is the i-th column of A, and H_i^* is the dehankel of i-th elementary Hankel matrix defined in Definition 12. Therefore, in a more compact form we have:

$$\mathbf{g}_x := \nabla_x \ln\det H(Ax) = A^T H^*(H^{-1}(Ax)),\tag{A.10}$$

Also, taking the second order derivatives give us:

$$\left[\frac{\partial^2}{\partial x_i \partial x_j}\ln\det H(Ax)\right]_{i,j=0}^{n} = -A^T\left[H^{-1}(Ax)H_iH^{-1}(Ax)\bullet H_j\right]_{i,j=0}^{n}A.\tag{A.11}$$

Finally, let us show the (i,j)-th element of $H^{-1}(Ax)$ by $z_{i,j}$. Then it is easy to show that:

$$H^{-1}(Ax)H_iH^{-1}(Ax)\bullet H_j = \sum_{\substack{a+b=i\\ k+l=j}} z_{a,k}z_{b,l}.\tag{A.12}$$

If we consider the $H^{-1}(Ax)$ as the coefficient of a bivariate polynomial in the standard basis where its (i,j)-th element is the coefficient of $t^i s^j$ in the polynomial, then the rhs of (A.12) can be considered as the coefficient of the $t^i s^j$ in the convolution of $H^{-1}(Ax)$ by itself. Therefore, in more compact form we have:

$$\left[H^{-1}(Ax)H_iH^{-1}(Ax)\bullet H_j\right]_{i,j=0}^{n} = conv2(H^{-1}(Ax), H^{-1}(Ax)).\tag{A.13}$$

Substituting the rhs of (A.13) into (A.11), results in:

$$\mathbf{H}_x := -A^T conv2(H^{-1}(Ax), H^{-1}(Ax))A.\tag{A.14}$$

Proof. **Lemma 12**

a. To show part a, it is enough to substitute I for A in (A.10).

b. To show part b, it is enough that for the first and second part, substitute I and $\mathcal{A}$, respectively, for A in (A.10), where $\mathcal{A}$ was defined in (2.27).

c. To show part c, it is enough that for the first and second part, substitute $\mathcal{B}$ and $\mathcal{C}$, respectively, for A in (A.10), where $\mathcal{B}$ and $\mathcal{C}$ were defined in (2.28) and (2.29). □

Proof. **Lemma 13**

a. To show part a, it is enough to substitute I for A in (A.14).

b. To show part b, it is enough that for the first and second part, substitute I and $\mathcal{A}$, respectively, for A in (A.14), where $\mathcal{A}$ is defined in (2.27).

c. To show part c, it is enough that for the first and second part, substitute $\mathcal{B}$ and $\mathcal{C}$, respectively, for A in (A.14), where $\mathcal{B}$ and $\mathcal{C}$ are defined in (2.28) and (2.29), respectively. □

To show the proofs of Lemma 14 and Lemma 15, we need following facts:

$$\begin{aligned}\frac{\partial}{\partial x_i}\ln\det(H+T)(Ax) &= (H+T)^{-1}(Ax)\bullet\left[\frac{\partial(H+T)(Ax)}{\partial Ax}\frac{\partial Ax}{\partial x_i}\right]\\ &= a_i^T(H_i+T_i)^*((H+T)^{-1}(Ax)),\end{aligned}\tag{A.15}$$

where T_i is the i-th elementary Toeplitz matrix defined in Definition 12. Therefore, in a more compact form we have:

$$\mathbf{g}_x := \nabla_x \ln\det(H+T)(Ax) = A^T(H+T)^*((H+T)^{-1}(Ax)),\tag{A.16}$$

Also, taking the second order derivatives give us:

$$\begin{aligned}&\left[\frac{\partial^2}{\partial x_i\partial x_j}\ln\det(H+T)(Ax)\right]_{i,j=0}^{n} =\\ &-A^T\left[(H+T)^{-1}(Ax)(H_i+T_i)(H+T)^{-1}(Ax)\bullet(H_j+T_j)\right]_{i,j=0}^{n}A.\end{aligned}\tag{A.17}$$

Finally, let us show the (i,j)-th element of $(H+T)^{-1}(Ax)$ by $z_{i,j}$. Then it is easy to show that:

$$\begin{aligned}&(H+T)^{-1}(Ax)(H_i+T_i)(H+T)^{-1}(Ax)\bullet(H_j+T_j) =\\ &\frac{1}{4}\left[\sum_{\substack{a+b=i\\k+l=j}} z_{a,k}z_{b,l} + \sum_{\substack{a+b=i\\|k-l|=j}} z_{a,k}z_{b,l} + \sum_{\substack{|a-b|=i\\k+l=j}} z_{a,k}z_{b,l} + \sum_{\substack{|a-b|=i\\|k-l|=j}} z_{a,k}z_{b,l}\right].\end{aligned}\tag{A.18}$$

If we consider the $(H+T)^{-1}(Ax)$ as the coefficient of a bivariate polynomial where its (i,j)-th element is the coefficient of $T_i(t)T_j(t)$ in the polynomial, then the rhs of (A.18)

can be considered as the coefficient of the $T_i(t)T_j(t)$ in the convolution of $(H+T)^{-1}(Ax)$ by itself. Therefore, in more compact form we have:

$$\Big[(H+T)^{-1}(Ax)(H_i+T_i)(H+T)^{-1}(Ax) \bullet (H_j+T_j)\Big]_{i,j=0}^{n} = conv2T((H+T)^{-1}(Ax),(H+T)^{-1}(Ax)). \quad \text{(A.19)}$$

Substituting the rhs of (A.19) into (A.17), results in:

$$\mathbf{H}_x := A^T conv2T((H+T)^{-1}(Ax),(H+T)^{-1}(Ax))A. \quad \text{(A.20)}$$

Proof. **Lemma 14**

a. To show part a, it is enough to substitute I for A in (A.16).

b. To show part b, it is enough that for the first and second part, substitute I and $\mathcal{A}_{Ch}$, respectively, for A in (A.16), where $\mathcal{A}_{Ch}$ is defined in (A.4).

c. To show part c, it is enough that for the first and second part, substitute $\mathcal{B}_{Ch}$ and $\mathcal{C}_{Ch}$, respectively, for A in (A.16), where $\mathcal{B}_{Ch}$ and $\mathcal{C}_{Ch}$ are defined in (A.7) and (A.8). □

Proof. **Lemma 15**

a. To show part a, it is enough to substitute I for A in (A.20).

b. To show part b, it is enough that for the first and second part, substitute I and $\mathcal{A}_{Ch}$ for A in (A.20), where $\mathcal{A}_{Ch}$ is defined in (A.4).

c. To show part c, it is enough that for the first and second part, substitute $\mathcal{B}_{Ch}$ and $\mathcal{C}_{Ch}$ for A in (A.20), where $\mathcal{B}_{Ch}$ and $\mathcal{C}_{Ch}$ are respectively defined in (A.7) and (A.8). □

References

[1] Aït-Sahalia, Y., Duarte, J., Nonparametric option pricing under shape restrictions, Journal of Econometrics, 116, 9-47 (2003)

[2] Akle Serrano S., Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone, PhD Thesis, Stanford university (2015)

[3] Alizadeh, F., Interior point methods in semidefinite programming with applications to combinatorial optimization. SIAM Journal on Optimization, 5, 13-51 (1995)

[4] Alizadeh, F., Eckstein, J., Noyan, N., Rudolf, G., Arrival Rate Approximation by Nonnegative Cubic Splines, Operations Research, 56(1), 140-156 (2008)

[5] Allon, G., Beenstock, M., Hackman, S., Passy, U., Shapiro, A., Nonparametric estimation of concave production technologies by entropic methods, Journal of Applied Econometrics, 22(4), 795-816 (2007)

[6] Andersen, E., Roos, C., Terlaky, T., On implementing a primal-dual interior-point method for conic quadratic optimization, Math. Program., Ser. B 95: 249 (2003)

[7] Andersen, E. D., Gondzio, J., Meszaros, C., Xu, X., Implementation of interior point methods for large scale linear programming. In Terlaky, T., editor, Interior-point methods of mathematical programming, pp. 189-252. Kluwer Academic Publishers, Germany (1996)

[8] Andersen, E. D., Andersen, K. D., The MOSEK interior point optimization for linear programming: An implementation of the homogeneous algorithm, Frenk, H., and et al, editor, High Performance Optimization, pp. 197-245, Kluwer Academic Publishers, Germany (2000)

[9] Andersen, M. S., Dahl, J., Vandenberghe, L., CVXOPT: A python package for convex optimization, version 1.1.5, (2012)

[10] Arico, A., Rodriguez, G., A fast solver for linear systems with displacement structure, NUMERICAL ALGORITHMS, 55, 529-556 (2010)

[11] Artin E., Über die Zerlegung definiter Funktionen in Quadrate, Abh. math. Sem. Hamburg 5, 110−115 (1927)

[12] Baszenski, G., Tasche, M., Fast Polynomial Multiplication and Convolution Related to the Discrete Cosine Transform, Linear Algebra and Its Applications, 252, 1-25 (1997)

[13] Ben-Tal, A., Nemirovski, A. S.: Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications. SIAM, Philadelphia (2001)

[14] Chandrasekaran, S., Gu, M., Sun, X., Xia, J., Zhu, J., A superfast algorithm for Toeplitz systems of linear equations. SIAM J. Matrix Anal. Appl. 29(4), 1247-1266 (2007)

[15] Chares, P. R., Cones and Interior-Point Algorithms for Structured Convex Optimization involving Powers and Exponentials. PhD thesis, Universite Catholique de Louvain, (2007)

[16] Faybusovich, L., Self-Concordant Barriers for Cones Generated by Chebyshev Systems, SIAM J. Optim., 12(3), 770-781 (2002)

[17] Giorgi, P., On Polynomial Multiplication in Chebyshev Basis, IEEE Transactions on Computers, 61(6), 780-789 (2012)

[18] Glineur, F., Topics in Convex Optimization: Interior-Point methods, Conic Duality and Approximations. Ph.D. thesis, Faculte Polytechnique de Mons, Mons, Belgium, (2001)

[19] Grant, M., Boyd, S., CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`, September 2013

[20] Gonzaga, C. C., An algorithm for solving linear programming problems in $O(n^3L)$ operations. In: Progress in Mathematical Programming. Interior Point and Related Methods (ed. by N. Megiddo), pp. 1-28, Springer-Verlag, New York (1989)

[21] Güler, O., Barrier Functions in Interior Point Methods, Math. Oper. Res. 21, 860-885 (1996)

[22] Hankin, R. K. S., Numerical evaluation of the Gauss hypergeometric function with the hypergeo package. `https://cran.r-project.org/web/packages/hypergeo/` (2016)

[23] Hilbert, D., Über die Darstellung definiter Formen als Summe von Formenquadraten, Math. Ann., 32(1888), 342350

[24] Karmarkar, N., A new polynomial-time algorithm for linear programming, Combinatorica, 4(4), 373-395 (1984)

[25] Kojima, M., Mizuno, S., Yoshise, A., A polynomial-time algorithm for a class of linear complementarity problems. Mathematical Programming, 44, 1-26 (1989)

[26] Lima, J., A Karatsuba-Based Algorithm for Polynomial Multiplication in Chebyshev Form, IEEE TRANSACTIONS ON COMPUTERS, 59(6), 835-841 (2010)

[27] Lustig, I. J., Feasibility issues in primal-dual interior-point methods for linear programming. Math. Programming, 49, 145-162 (1990)

[28] Lustig, I. J., Marsten, R. E., Shanno, D. F., Interior point methods for linear programming: Computational state of the art. ORSAJ. on Comput., 6(1),1-15 (1994)

[29] Megiddo, N., Pathways to the optimal set in linear programming. In N. Megiddo, editor, Progress in Mathematical Programming: Interior Point and Related Methods, Springer Verlag, New York, pp. 131-158 (1989)

[30] Mehrotra, S., On the Implementation of a Primal-Dual Interior Point Method, SIAM J. Optim., 2(4), 575-601 (1992)

[31] Monteiro,R. D. C., Adler, I., Interior path following primal-dual algorithms: Part I: Linear programming. Mathematical Programming, 44, 27-41 (1989)

[32] MOSEK ApS, The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28), url = "http://docs.mosek.com/7.1/toolbox/index.html", (2015)

[33] Nesterov, Y. E., Long-step strategies in interior-point primal-dual methods, Mathematical Programming, 76, 47-94 (1996)

[34] Nesterov, Y. E., Structure of Non-Negative Polynomials and Optimization Problem, (1997)

[35] Nesterov, Y. E., Squared Functional Systems and Optimization Problems, High Performance Optimization, 33, 405-440, Kluwer Academic Press, (2000)

[36] Nesterov, Y. E., Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, (2003)

[37] Nesterov, Y. E., Towards Nonsymmetric Conic Optimization. Optim. Method. Softw., 27, 893-917 (2012)

[38] Nesterov, Y. E., Nemirovskii, A.S., Interior-Point Polynomial Algorithms in Convex Programming, SIAM, Philadelphia (1994)

[39] Nesterov, Y. E., Todd, M. J., Self-scaled barriers and interior-point methods for convex programming. Math. Oper. Res., 22, 1-42 (1997)

[40] Nesterov, Y. E., Todd, M. J., Primal-dual interior-point methods for self-scaled cones. SIAM J. Optim., 8, 324-364 (1998)

[41] Papp, D., Alizadeh, F., Semidefinite Characterization Of Sum-Of-Squares Cones In Algebras, SIAM J. Optim., 23, 1398-1423 (2013)

[42] Papp, D., Optimization models for shape-constrained function estimation problems involving nonnegative polynomials and their restrictions, PhD thesis, Rutgers University, USA (2011)

[43] Pearson, J., Computation of Hypergeometric Functions. Master thesis, University of Oxford, UK (2009)

[44] Pölya, G., Szegö, G., Problems and Theorems in Analysis, vol. 2. Springer, New York (1976)

[45] Renegar, J., A Mathematical View of Interior-Point Methods in Convex Optimization, SIAM, Philadelphia (1987)

[46] Renegar, J., A polynomial-time algorithm based on Newton's method for linear programming, Mathematical Programming, 40 , 59-94 (1988)

[47] Skajaa A., The homogeneous interior-point algorithm: nonsymmetric cones, warmstarting, and applications. PhD thesis, Technical University of Denmark, Denmark (2013)

[48] Skajaa A., Ye Y., A homogeneous interior-point algorithm for nonsymmetric convex conic optimization, Math. Program., Ser. A, 150, 391-422 (2015)

[49] Sturm, J.F., Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. Optimization Methods and Software, 11-12:625653, (1999)

[50] Toh, K. C., R. H. Tütüncü, Todd, M. J., SDPT3 — a Matlab software package for semidefinite programming, Optimization Methods and Software, 11, 545-581 (1999)

[51] Van Barel, M., Heinig, G., Kravanja, P., A stabilized superfast solver for nonsymmetric Toeplitz systems. SIAM J. Matrix Anal. Appl. 23(2), 494-510 (2001, electronic)

[52] Xu, X., Hung, P. -F., Ye, Y., A simplified homogeneous and self-dual linear programming algorithm and its implementation, Annals of Operations Research, 62, 151-171 (1996)

[53] Ye, Y., Todd, M. J., Mizuno, S., An $O(\sqrt{n}L)$ - iteration homogeneous and self-dual linear programming algorithm, Math. Oper. Res., 19, 53-67 (1994)

[54] Frisch, K.R., The logarithmic potential method of convex programming, unpublished manuscript, University Institute of Economics, Oslo, Norway (1955)

[55] Fiacco, A., McCormick, G., Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley and Sons, New York (1968)

[56] Karlin, S., Studden, W. J., Tchebyshev Systems: with Applications in Analysis and Statistics, Interscience Publisher, John Willy & Sons, New York, (1968)

[57] Nocedal, J., Wright, S. J., Numerical Optimization, Springer, 2nd edition (2006)

[58] Roos, C., Vial, J.-P., and Terlaky, T., Theory and Algorithms for Linear Optimization : An Interior Point Approach, John Wiley and Sons, New York (1997)

[59] Ye, Y., Interior Point Algorithms : Theory and Analysis, John Wiley and Sons, New York (1997)

[60] Primal-Dual Interior-Point Methods, SIAM Publications, Philadelphia (1997)