2018

Polynomial optimization using semidefinite programming

Michael D. Smith

ABSTRACT

POLYNOMIAL OPTIMIZATION USING SEMIDEFINITE PROGRAMMING

Michael D. Smith, M.S.
Department of Mathematical Sciences
Northern Illinois University, 2018
Nathan Krislock, Director

In this thesis, our goal is to study the problem of minimizing a polynomial $p(x)$ using semidefinite matrices. Our discussion will cover Lagrangian duality and conic programming, followed by a discussion on how nonnegative polynomials can be approximated by sums of squares. We will use this approximation to create our semidefinite programming problems. This will lead us to being able to solve the problem of wireless coverage using minimum transmission.

NORTHERN ILLINOIS UNIVERSITY
DE KALB, ILLINOIS

MAY 2018

POLYNOMIAL OPTIMIZATION USING SEMIDEFINITE PROGRAMMING

BY

MICHAEL D. SMITH

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICAL SCIENCES

Thesis Director:
Nathan Krislock

ACKNOWLEDGEMENTS

I would like to thank my friends, family, and wife, Jennifer, for their constant love and moral support throughout all of my studies. These are the people who have helped to keep me sane.

I would like to thank Dr. Sien Deng and Dr. Jose Yunier Bello Cruz for agreeing to be on my defense committee. Your time and help with my studies are greatly appreciated.

Most of all, I would like to thank Dr. Nathan Krislock for advising me throughout this entire process. You have been my teacher, proofreader, career coach, technical support, and so much more. Thank you for all of the hard work and long hours you have put in helping me prepare this thesis.

DEDICATION

To Jennifer, for helping to keep my hopes and spirits up through this long process

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Optimization is a topic in mathematics that has many practical uses. It is always helpful to know what the best possible outcome for a given problem may be. However, optimization is a topic that is barely discussed in the undergraduate curriculum. There is a brief discussion of the simplex method for linear programming, but overall the topic of optimization is left off the table. This can be attributed in part to the lack of ability to perform most optimization algorithms by hand. It can also be attributed to the amount of mathematical insight required to comprehend more complicated optimization problems.

In this thesis, we will explore semidefinite programming, a branch of conic programming that will allow us to tackle some polynomial optimization problems. Our goal is to minimize a polynomial $p(x)$. We will show how we can approximate the answer to this problem with the use of semidefinite matrices.

In Chapter 2, we will introduce the concept of a dual problem and discuss how it is used to create conditions for optimality. Chapter 2 will also discuss linear, conic, and semidefinite programming. In Chapter 3, we will discuss nonnegative polynomials and their relation to sum-of-squares polynomials. We will then show how sum-of-squares polynomials can be rewritten using a semidefinite matrix and use that to create polynomial optimization problems using sum-of-squares polynomials and semidefinite programming. We will then wrap up Chapter 3 with some numerical tests of our solvers, as well as show how to solve the problem of obtaining wireless coverage using minimum transmission.

CHAPTER 2

MATHEMATICAL BACKGROUND

This chapter will give us a theoretical framework for understanding semidefinite programming. We will start with an introduction to the Lagrangian dual problem. Many optimization problems utilize a dual problem, and our hope is to give some explanation as to why they do.

This chapter also introduces us to three types of mathematical optimization: linear programming, conic programming, and semidefinite programming. The purpose is to provide motivation for studying semidefinite programming.

2.1 Lagrangian Dual Problem

Sometimes we are faced with optimization problems that are particularly difficult to solve. As such, we may wish to look at a related problem that is easier to solve. We also would like to have some defined conditions at which we can obtain our optimal solution. The purpose of this section is to provide a framework for which we can formulate a related problem and determine some conditions with which we can obtain an optimal solution. The information in this section will be primarily adapted from Boyd and Vandenberghe (2004).

When discussing this topic, we will often reference whether or not our functions are convex. While it is not necessary for our optimization problems to contain only convex functions, we will find that they aid in our ability to obtain optimal values for our optimization

problems. As such, we will define them here. A *convex function* $f : \mathbf{R}^n \to \mathbf{R}$ is a function that satisfies

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$$

for all $x, y \in \mathbf{R}^n$ and all $\alpha \in [0,1]$.

2.1.1 Lagrangian

Consider an optimization problem in standard form:

$$\begin{aligned} &\text{Minimize } f_0(x) \\ &\text{subject to } f_i(x) \leq 0, \quad i = 1 \ldots m, \\ &\qquad\qquad h_j(x) = 0, \quad j = 1 \ldots p, \end{aligned} \tag{2.1}$$

with $x \in \mathbf{R}^n$ and assuming that the domain $D = \bigcap_{i=0}^{m} \mathbf{dom} f_i \cap \bigcap_{j=1}^{p} \mathbf{dom} h_j$, where $\mathbf{dom} f$ is the domain of f, is nonempty. An optimization problem will be considered a *convex optimization problem* if it is attempting to minimize a convex function and all of its constraints are of the form $f_i(x) \leq 0$ and $h_j(x) = 0$, where f_i are convex functions and h_j are affine functions. A function h is *affine* if h and $-h$ are both convex. We will denote the optimal value of (2.1) as p^*. The basic idea behind Lagrangian duality is to transform the objective function by adding a weighted sum of the constraints to it. We define the *Lagrangian* $L : \mathbf{R}^n \times \mathbf{R}^m_+ \times \mathbf{R}^p \to \mathbf{R}$ of this problem as

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x), \tag{2.2}$$

with $\mathbf{dom} L = D \times \mathbf{R}^m_+ \times \mathbf{R}^p$, where $\mathbf{R}^m_+ = \{x \in \mathbf{R}^m : x_i \geq 0, i = 1, \ldots, m\}$. We call λ and ν the *dual variables* or the *Lagrangian multiplier vectors* associated with (2.1).

2.1.2 The Lagrange Dual Function

We define the *Lagrange dual function* (or just *dual function*) $g : \mathbf{R}^m_+ \times \mathbf{R}^p \to \mathbf{R}$ as the minimum value of the Lagrangian over x. So for $\lambda \in \mathbf{R}^m_+, \nu \in \mathbf{R}^p$,

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) = \inf_{x \in D} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \right). \tag{2.3}$$

The dual function takes on the value of $-\infty$ when the Lagrangian is unbounded below in x. With this restriction, we can show that the dual function provides a lower bound for our optimal value p^* of (2.1). Suppose $\tilde{x}$ is a feasible point for problem (2.1). Then $f_i(\tilde{x}) \leq 0, h_j(\tilde{x}) = 0$, and $\lambda \geq 0$, so we have

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{j=1}^{p} \nu_j h_j(\tilde{x}) \leq 0,$$

since the first summation is nonpositive and the second summation is zero. Thus,

$$L(\tilde{x}, \lambda, \nu) = f_0(\tilde{x}) + \sum_{i=1}^{m} \lambda_i f_i(\tilde{x}) + \sum_{j=1}^{p} \nu_j h_j(\tilde{x}) \leq f_0(\tilde{x}).$$

This gives us

$$g(\lambda, \nu) = \inf_{x \in D} L(x, \lambda, \nu) \leq L(\tilde{x}, \lambda, \nu) \leq f_0(\tilde{x}),$$

and as such,

$$g(\lambda, \nu) \leq p^*. \tag{2.4}$$

2.1.3 The Lagrange Dual Problem

Now that we have found an equation related to our original problem (2.1) and know that this problem will always provide a lower bound to our desired solution, we can attempt to maximize (2.3) to provide the closest lower bound possible to the optimal value p^*. This results in us formulating the *Lagrange dual problem* (or just *dual problem*) for problem (2.1):

$$\begin{aligned} &\text{Maximize } g(\lambda, \nu) \\ &\text{subject to } \lambda \geq 0. \end{aligned} \tag{2.5}$$

Now that we have this context, we can say that (2.1) is typically referred to as the *primal problem*. We can also now define the term *dual feasible*, which is used to describe a pair (λ, ν) that satisfies $\lambda \geq 0$ and $g(\lambda, \nu) > -\infty$. That means that the pair (λ, ν) is feasible for the dual problem. We say that if (λ^*, ν^*) provides the optimal value for (2.5), then they are the *dual optimal* or *optimal Lagrange multipliers* for that problem.

It is important to note that (2.5) is a convex problem because its objective function is *concave* (meaning that $-g(\lambda, \nu)$ is convex) and its constraint is convex, and maximizing a concave function is equivalent to minimizing a convex function. This does not make any assumptions on the convexity of the primal problem (2.1).

2.1.4 Weak Duality and the Duality Gap

We refer to the optimal value of (2.5) as d^*. By definition, d^* is the best possible lower bound that we can obtain on the primal problem (2.1) from the Lagrangian dual function (2.3). Applying our earlier observation (2.4) to (2.5), we see that

$$d^* \leq p^*. \tag{2.6}$$

This is an important property known as *weak duality*. We will always have this property, as this does not make any assumptions as to whether or not the primal problem (2.1) is convex.

The difference $p^* - d^*$ is referred to as the *optimal duality gap*, since it is the difference between the optimal solutions for the primal and dual problems. Because we always have weak duality, the duality gap is always nonnegative.

2.1.5 Strong Duality and Constraint Qualifications

While weak duality is a useful feature, our goal is to find an exact optimal value for the primal problem if possible. For this, we will need more than just a lower bound on p^*. Suppose that equality

$$d^* = p^* \tag{2.7}$$

holds. Then the duality gap is zero, and we say that *strong duality* holds.

Strong duality is an incredibly useful feature, as it allows us to solve either the primal or dual problem in order to find our desired optimal value. However, we are typically not

guaranteed to have strong duality. If primal problem is a convex optimization problem, i.e., of the form

$$\begin{aligned} &\text{Minimize } f_0(x) \\ &\text{subject to } f_i(x) \leq 0, \quad i = 1 \dots m, \\ &\qquad\qquad h_j(x) = 0, \quad j = 1 \dots p, \end{aligned}$$

with affine h_j's, we often (but not always) find that strong duality holds. There are many results which impose other conditions on the problem that establish strong duality. We call such conditions *constraint qualifications*.

A simple example of a constraint qualification is *Slater's constraint qualification*. It states that if the primal problem is convex and $\exists x \in \textbf{relint } D$ (defined as $\text{relint}(D) := \{x \in D \mid \exists \epsilon > 0, B(x, \epsilon) \cap \text{aff}(D) \subseteq D\}$, where $B(x, \epsilon) := \{y \in \mathbf{R}^n \mid \|x - y\|_2 \leq \epsilon\}$ and $\text{aff}(D)$ is the smallest affine set containing D) such that

$$f_i(x) < 0, \quad i = 1, \dots, m, \quad h_j(x) = 0, \quad j = 1 \dots p, \tag{2.8}$$

then strong duality holds. Such a point x is referred to as *strictly feasible* since the inequalities hold strictly. A proof of Slater's constraint qualification can be found in Boyd and Vandenberghe (2004, p. 234). Slater's condition also implies that the dual optimal value is attained when $d^* > -\infty$.

2.1.6 KKT Optimality Conditions

Now that we understand strong duality, it is worthwhile to observe some properties that we will have if it holds. The *Karush-Kuhn-Tucker* (KKT) conditions provide us with these properties. As we will see, they are more useful to us if our primal problem is convex. This is

because the KKT conditions provide necessary and sufficient conditions for optimality if we know that our primal problem is convex. However, we will not assume that (2.1) is convex unless otherwise stated.

Certificate of suboptimality: If we can find a single dual feasible pair (λ, ν), we have a lower bound on the optimal value of the primal problem p^* because of (2.6). We say then that (λ, ν) provides a *certificate* that $p^* \geq g(\lambda, \nu)$. If we also have a primal feasible x, then we define the *duality gap* between x and (λ, ν) as

$$f_0(x) - g(\lambda, \nu).$$

These feasible points provide us an interval for the optimal values p^* and d^*:

$$p^* \in [g(\lambda, \nu), f_0(x)], \quad d^* \in [g(\lambda, \nu), f_0(x)].$$

If the duality gap is zero, then we have strong duality and as such the primal optimal is x^* and the dual optimal pair is (λ^*, ν^*).

Complementary slackness: Suppose that the primal and dual optimal values are attained and strong duality holds. Let x^* be the primal optimal and (λ^*, ν^*) be the dual optimal pair. Then we have

$$\begin{aligned} f_0(x^*) &= g(\lambda^*, \nu^*) \\ &= \inf_x \left(f_0(x) + \sum_{i=1}^m \lambda_i^* f_i(x) + \sum_{j=1}^p \nu_j^* h_j(x) \right) \\ &\leq f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{j=1}^p \nu_j^* h_j(x^*) \\ &\leq f_0(x^*). \end{aligned}$$

The first line comes from strong duality (2.7). The second line is the definition of the dual function (2.3). The third line comes from the definition of infimum. The last line is because $f_i(x^*) \leq 0, h_j(x^*) = 0$, and $\lambda \geq 0$. Because of this, we can see that $f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{j=1}^p \nu_j^* h_j(x^*) = f_0(x^*)$, and as a result,

$$\sum_{i=1}^m \lambda_i^* f_i(x^*) = 0.$$

Since each term in the summation is nonpositive, we have

$$\lambda_i^* f_i(x^*) = 0, \quad i = 1, \dots m. \tag{2.9}$$

This is a result known as *complementary slackness*. It says that for a primal optimal x^* and a dual optimal pair (λ^*, ν^*), if strong duality holds, then

$$\lambda_i^* > 0 \implies f_i(x^*) = 0 \qquad \textbf{and} \qquad f_i(x^*) < 0 \implies \lambda_i^* = 0.$$

This means that the ith optimal Lagrange multiplier is zero unless the ith constraint is active at the optimum.

KKT optimality conditions: One of the things that we need to assume for the KKT optimality conditions is for $f_0, f_1, \dots, f_m, h_1, h_2, \dots, h_p$ to be differentiable.

When the primal problem is nonconvex, let x^* and (λ^*, ν^*) be any primal and dual optimal points with zero duality gap. Note here we are assuming that strong duality holds. Because x^* minimizes $L(x, \lambda^*, \nu^*)$ over x, it follows that its gradient must equal zero at x^*. As such, we have that

$$\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(x^*) = 0.$$

Therefore, we have

$$\begin{aligned} f_i(x^*) &\leq 0, \qquad i = 1, \ldots, m, \\ h_j(x^*) &= 0, \qquad j = 1, \ldots, p, \\ \lambda_i^* &\geq 0, \qquad i = 1, \ldots, m, \\ \lambda_i^* f_i(x^*) &= 0, \qquad i = 1, \ldots m, \\ \nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{j=1}^{p} \nu_j^* \nabla h_j(x^*) &= 0. \end{aligned} \tag{2.10}$$

These are known as the KKT conditions.

When the primal problem is convex, we will claim that the KKT conditions are also sufficient for the points to be primal and dual optimal; i.e., if f_i are convex and h_j are affine, and $\tilde{x}, \tilde{\lambda}, \tilde{\nu}$ are any points that satisfy the KKT conditions

$$\begin{aligned} f_i(\tilde{x}) &\leq 0, \qquad i = 1, \ldots, m, \\ h_j(\tilde{x}) &= 0, \qquad j = 1, \ldots, p, \\ \tilde{\lambda}_i &\geq 0, \qquad i = 1, \ldots, m, \\ \tilde{\lambda}_i f_i(\tilde{x}) &= 0, \qquad i = 1, \ldots m, \\ \nabla f_0(\tilde{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i \nabla f_i(\tilde{x}) + \sum_{j=1}^{p} \tilde{\nu}_j \nabla h_j(\tilde{x}) &= 0, \end{aligned} \tag{2.11}$$

then $\tilde{x}, \tilde{\lambda}$, and $\tilde{\nu}$ are primal and dual optimal, respectively, and there is zero duality gap.

We will now justify this claim. The first two lines of (2.11) let us know that $\tilde{x}$ is primal feasible. The third line gives us that $L(x, \tilde{\lambda}, \tilde{\nu})$ is convex in x. The final line lets us see

that the gradient of L vanishes with respect to x when $x = \tilde{x}$, so it follows that $L(x, \tilde{\lambda}, \tilde{\nu})$ is minimized with respect to x by $\tilde{x}$. We can then see that

$$\begin{aligned} g(\tilde{\lambda}, \tilde{\nu}) &= L(\tilde{x}, \tilde{\lambda}, \tilde{\nu}) \\ &= f_0(\tilde{x}) + \sum_{i=1}^{m} \tilde{\lambda}_i f_i(\tilde{x}) + \sum_{j=1}^{p} \tilde{\nu} h_j(\tilde{x}) \\ &= f_0(\tilde{x}), \end{aligned}$$

where the last line comes from the facts that $h_j(\tilde{x}) = 0$ and $\tilde{\lambda}_i f_i(\tilde{x}) = 0$. Therefore, $\tilde{x}$ and $(\tilde{\lambda}, \tilde{\nu})$ have zero duality gap and are primal and dual optimal. What we have shown is that for any convex optimization problem that satisfies Slater's condition (2.8) and has differentiable objective and constraint functions, the KKT conditions provide necessary and sufficient conditions for optimality. That is, since Slater's condition implies that the optimal duality gap is zero and that the dual optimum is obtained, x is optimal if and only if there exists (λ, ν) that satisfy the KKT conditions with x.

The KKT conditions are an important part of optimization. This is because they provide a powerful result for convex problems. As such, many convex optimization algorithms are based on methods for solving for the KKT conditions.

2.2 Linear Programming

The most commonly used tool for mathematical optimization is linear programming. This can be attributed to its simplicity and efficiency, as well as its ability to handle a large number of real-world applications. The majority of the remaining material in this chapter is adapted from the second chapter of Blekherman, Parrilo, and Thomas (2013), unless otherwise noted.

2.2.1 Problem Formulation

Linear programming is the problem of finding the minimal value of a linear function subject to linear constraints. It typically has the following standard form:

$$\begin{aligned} &\text{Minimize } c^T x \\ &\text{subject to } Ax = b, \\ &x \geq 0. \end{aligned} \tag{LP-P}$$

This is called the **primal problem** of the linear program, denoted henceforth as (LP-P). To solve these problems, we can use the Lagrangian to form a **dual problem** for the linear problem, denoted henceforth as (LP-D). In order to formulate (LP-D), we first observe that the Lagrangian of (LP-P) is

$$L(x, \lambda, \nu) = c^T x - \sum_{i=1}^{n} \lambda_i x_i + \nu^T (b - Ax) = b^T \nu + (c - A^T \nu - \lambda)^T x,$$

which results in a dual function of

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = b^T \nu + \inf_x (c - A^T \nu - \lambda)^T x.$$

Since the infimum term is linear, it equals $-\infty$ unless $c - A^T \nu - \lambda = 0$. Thus, we can rewrite our dual function as

$$g(\lambda, \nu) = \begin{cases} b^T \nu, & \text{if } c - A^T \nu - \lambda = 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

Therefore, the standard form of (LP-D) is as follows:

$$\begin{aligned} &\text{Maximize } b^T\nu \\ &\text{subject to } c - A^T\nu - \lambda = 0, \\ &\lambda \geq 0, \end{aligned} \tag{LP-D}$$

or rewritten as

$$\begin{aligned} &\text{Maximize } b^T\nu \\ &\text{subject to } c - A^T\nu \geq 0. \end{aligned} \tag{2.12}$$

There is also an inequality form for linear programming problems. This takes the form

$$\begin{aligned} &\text{Minimize } c^T x \\ &\text{subject to } Ax \leq b. \end{aligned} \tag{2.13}$$

In this case, the Lagrangian is

$$L(x, \lambda) = c^T x + \lambda^T (b - Ax) = b^T \lambda + (c - A^T \lambda)^T x,$$

which results in a dual function of

$$g(\lambda) = \inf_x L(x, \lambda) = b^T \lambda + \inf_x (c - A^T \lambda)^T x.$$

Once again, since the infimum term is linear, it equals $-\infty$ unless $c - A^T\lambda = 0$. Thus, we can rewrite our dual function as

$$g(\lambda, \nu) = \begin{cases} b^T\lambda, & \text{if } c - A^T\lambda = 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

Therefore, the dual problem is as follows:

$$\begin{aligned} &\text{Maximize } b^T\lambda \\ &\text{subject to } c - A^T\lambda = 0, \\ &\lambda \geq 0. \end{aligned} \tag{2.14}$$

For linear programming, we do not need Slater's constraint qualification (2.8) to hold to obtain strong duality. In other words, strong duality always holds for linear programming.

2.2.2 Simple Examples

Example 1: The following example of standard form linear programming comes from Blekherman et al. (2013, p. 4). While we will not go into the steps for solving here, our numerical solvers and their methods will be discussed by the end of this chapter.

Consider the following linear programming problem:

$$\begin{aligned} &\text{Minimize } x_1 - 8x_2 \\ &\text{subject to} \\ &-x_1 + 3x_2 + x_3 \quad = 4, \\ &4x_1 - x_2 + x_4 \quad = 6, \\ &x_1, x_2, x_3, x_4 \quad \geq 0. \end{aligned} \tag{2.15}$$

The optimal solution for this problem is that $x^* = (2, 2, 0, 0)$ with an optimal value $p^* = -14$.

The corresponding dual problem is

$$\begin{aligned} &\text{Maximize } 4y_1 + 6y_2 \\ &\text{subject to} \\ &-y_1 + 4y_2 \quad \leq 1, \\ &3y_1 - y_2 \quad \leq -8, \\ &y_1, y_2 \quad \leq 0. \end{aligned} \tag{2.16}$$

Here, the optimal solution is $y^* = (-\frac{31}{11}, -\frac{5}{11})$ with an optimal value $d^* = -14$. Notably, strong duality does indeed hold since $p^* = d^* = -14$.

Example 2: The following example of the inequality form for linear programming is adapted from Anderson et al. (2011, p. 30).

Springson, Inc., is a small textbook printing company whose management is trying to determine how many paperback and hardcover versions of a new textbook to print. Springson's distributor is excited about this new book, as it claims to have a proof of one of the millennium problems, and has decided to buy all of the books Springson manages to print. Knowing this, Springson would like to maximize the amount of revenue that it brings in.

In order to print a textbook, each textbook is required to go through three steps:

- Printing.
- Binding.
- Inspection and packaging.

The director of printing analyzed the steps for each type of book. A paperback book will require $\frac{1}{2}$ hour in the printing department, 2 hours in the binding department, and 1 hour for inspection and packaging. The paperback books will be sold for \$48 each. A hardcover book will require $\frac{3}{4}$ hour in the printing department, 3 hours in the binding department, and 1 hour for inspection and packaging. The hardcover books will be sold for \$76 each. There are 450 hours available for printing, 1200 hours available for binding, and 500 hours available for inspection and packaging.

In order to maximize the revenue made by Springson, Inc., we will create a linear program. Since we want to maximize revenue, we also want to minimize the negative of our revenue. Our primal problem is

$$\begin{aligned} &\text{Minimize } -48x_1 - 76x_2 \\ &\text{subject to} \\ &0.5x_1 + 0.75x_2 \leq 450, \\ &2x_1 + 3x_2 \leq 1200, \\ &x_1 + x_2 \leq 500, \\ &x \geq 0, \end{aligned} \tag{2.17}$$

where x_1 and x_2 are the numbers of paperback and hardcover textbooks, respectively. The optimal solution is $x^* = (0, 400)$ with an optimal primal value $p^* = -30400$. The dual problem for this problem is

$$\begin{aligned} &\text{Maximize } 450y_1 + 1200y_2 + 500y_3 \\ &\text{subject to} \\ &0.5y_1 + 2y_2 + y_3 \quad \leq -48, \\ &0.75y_1 + 3y_2 + y_3 \quad \leq -76, \\ &y \leq 0. \end{aligned} \tag{2.18}$$

The optimal solution is $y^* = (0, -\frac{76}{3}, 0)$ with an optimal dual value $d^* = -30400$. As we can see, strong duality holds since $p^* = d^* = -30400$. This means that in order to maximize our revenue, we should produce 400 hardcover books and no paperback books, for a revenue of $\$34,000$.

2.2.3 Motivation to Study Past Linear Programming

Needless to say, linear programming restricts us to linear equations. This means that anything that we would wish to be more complicated is outside of the realm of possibilities for this type of problem. In the next two sections, we will take a look at conic programming and semidefinite programming. Conic programming is the larger theoretical umbrella that we will look at to help us with providing an abstract foundation. Semidefinite programming is a subset of conic programming that we will focus on in this paper.

2.3 Conic Programming

Before we jump into semidefinite programming, we should first discuss conic programming. There are many theoretical concepts under the umbrella of conic programming which will help us to better understand semidefinite programming.

2.3.1 Cones

As we study cones and continue on in this paper, we will often deal with Euclidean spaces. A *Euclidean space* E is a finite-dimensional vector space over the real numbers that is associated with an *inner product* $\langle \cdot, \cdot \rangle : E \times E \to \mathbf{R}$ that satisfies the following conditions $\forall x, y, z \in E$, $\alpha, \beta \in \mathbf{R}$:

- $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$,
- $\langle x, y \rangle = \langle y, x \rangle$,
- $\langle x, x \rangle \geq 0$, and $\langle x, x \rangle = 0$ if and only if $x = 0$.

Associated with the inner product is the concept of the *norm* of an element $x \in E$, which is defined as $\|x\| := \sqrt{\langle x, x \rangle}$.

In order to understand conic programming, we first need to know about cones. Let $C \subseteq E$, a Euclidean space. Then C is a *cone* if $\forall x \in C, \lambda \geq 0, \lambda x \in C$. The set of *positive semidefinite matrices* $\mathcal{S}_+^n$, which is an important set for us later on, is a cone. $\mathbf{R}_+^m$, which we defined earlier, is also a cone.

We will start with some definitions that we will use in our proofs.

- **Cone** : Let $C \subseteq E$, a Euclidean space. Then C is a cone if $\forall x \in C, \lambda \geq 0, \lambda x \in C$.

- **Proper cone** : A proper cone is a cone $C \subset \mathbf{R}^n$ that satisfies the following:
  - C is *convex*, i.e., $\forall x, y \in C, 0 \leq \lambda \leq 1, \lambda x + (1-\lambda)y \in C$.
  - C is *closed*, i.e., $\mathbf{R}^n \backslash C$ is open, or rather the complement of C in $\mathbf{R}^n$ is open.
  - C is *solid*, i.e., C has a non-empty interior.
  - C is *pointed*, i.e., $-C \bigcap C = \{0\}$.
- **Dual cone** : Given a cone $C \subseteq E$, its dual cone is $C^* := \{z \in E^* : \langle z, x \rangle_E \geq 0, \forall x \in C\}$. Clearly, C^* is a cone.
- **Adjoint operator** : Consider two Euclidean spaces U and V and a linear operator $\mathcal{A} : U \to V$. The adjoint operator of $\mathcal{A}$, denoted $\mathcal{A}^*$, is the unique linear map $\mathcal{A}^* : V \to U$ defined by

$$\langle \mathcal{A}^* v, u \rangle_U = \langle v, \mathcal{A}u \rangle_V, \qquad \forall u \in U, v \in V.$$

2.3.2 Problem Formulation

Given a linear map $\mathcal{A} : U \to V$ from Euclidean space U to Euclidean space V and a proper cone $C \subset U$, the primal problem for a conic optimization problem has the form

$$\begin{aligned} &\text{Minimize } \langle c, x \rangle_U \\ &\text{subject to } \mathcal{A}x = b, \\ &x \in C, \end{aligned} \tag{2.19}$$

where $b \in V$, $c \in U^* = U$ since U is finite dimensional. The dual problem for a conic optimization problem has the form

$$\begin{aligned} &\text{Maximize } \langle y, b\rangle_V \\ &\text{subject to } c - \mathcal{A}^* y \in C^*. \end{aligned} \tag{2.20}$$

2.3.3 Theorems

The following two useful theorems result from solving the following exercise (Blekherman et al., 2013, p. 21, Exercise 2.24).

Theorem 1. *$C^{**} = C$ if and only if C is a closed convex cone.*

Proof. Let C be a closed convex cone. We will first show that $C \subseteq C^{**}$. Let $x \in C$. Then $\forall z \in C^*, \langle x, z\rangle \geq 0$. Furthermore, $C^{**} = \{y \in E : \langle z, y\rangle \geq 0, \forall z \in C^*\}$. This shows that $x \in C^{**}$, and thus $C \subseteq C^{**}$.

We will now show that $C^{**} \subseteq C$. Suppose $\exists x \in C^{**}$ such that $x \notin C$. Since C is nonempty and convex, the separating hyperplane theorem (Boyd & Vandenberghe, 2004, Section 2.5) provides us with a nonzero vector v such that

$$\langle v, x\rangle < 0 \leq \langle v, z\rangle, \qquad \forall z \in C.$$

This implies that $v \in C^*$, which means that $\langle v, x\rangle \geq 0$ because $x \in C^{**}$. This creates a contradiction. As such, no such x exists, and $C^{**} \subseteq C$. Thus, $C^{**} = C$. □

Theorem 2. *The dual cone C^* of a proper cone C is also a proper cone.*

Proof. In order to show that C^* is a proper cone, we need to show that it is closed, convex, solid, and pointed.

Closed : Let $x \in E$. We will first show that $f(y) := \langle y, x\rangle$ is a continuous function. Let $\epsilon > 0$ be given, and let $\delta > 0$ such that $\delta\|x\| < \epsilon$. Then $\forall \epsilon > 0$, if $\|z - y\| < \delta$, we have

$$|\langle z, x\rangle - \langle y, x\rangle| = |\langle z - y, x\rangle| \leq \|z - y\| \ \|x\| < \delta\|x\| < \epsilon$$

by using the Cauchy-Schwarz inequality. Thus, $f(y) = \langle y, x\rangle$ is continuous.

Now let $z \in \bar{C^*}$, the complement of C^*, and let $0 < \epsilon < -\langle z, x\rangle$. Since the inner product is continuous, $\exists \delta > 0$ such that if $\|z - y\| < \delta$, then $|\langle z, x\rangle - \langle y, x\rangle| < \epsilon$. This implies that $\langle y, x\rangle < \langle z, x\rangle + \epsilon < 0$. Thus, if $\|z - y\| < \delta$, then $y \in \bar{C^*}$, which means that $\bar{C^*}$ is open and consequently C^* is closed.

Convex : $\forall x \in C, z_1, z_2 \in C^*, 0 \leq \lambda \leq 1$, we see that

$$\langle \lambda z_1 + (1 - \lambda)z_2, x\rangle = \langle \lambda z_1, x\rangle + \langle (1 - \lambda)z_2, x\rangle \geq 0.$$

Thus, $\lambda z_1 + (1 - \lambda)z_2 \in C^*$, so C^* is convex.

Solid : We will begin by showing that if $C \subseteq \mathbf{R}^n$ is a nonempty convex cone and $\text{int}(C) = \emptyset$, then C^* is not pointed.

Since C is nonempty and convex, the relative interior of C is also nonempty:

$$\text{relint}(C) := \{x \in C \mid \exists \epsilon > 0, B(x, \epsilon) \cap \text{aff}(C) \subseteq C\},$$

where $B(x, \epsilon) := \{y \in \mathbf{R}^n \mid \ \|x - y\|_2 \leq \epsilon\}$ and $\text{aff}(C)$ is the smallest affine set containing C. Since $\text{relint}(C) \neq \emptyset$ but $\text{int}(C) = \emptyset$, it follows that $\text{aff}(C) \neq \mathbf{R}^n$. Since C is a cone, $0 \in C \subseteq \text{aff}(C)$, so $\text{aff}(C) = \text{span}(C)$.

Thus, $\text{span}(C) \neq \mathbf{R}^n$, which means that $\exists \lambda \in \mathbf{R}^n$, $\lambda \neq 0$, such that $\text{span}(C) \subseteq \{x \in \mathbf{R}^n \mid \lambda^T x = 0\}$. This implies that $\lambda^T x = 0, \forall x \in C$, and $\lambda \in C^*$ and $-\lambda \in C^*$. Since $\lambda \neq 0$, we can conclude that C^* is not pointed.

Now recall that we are assuming that C is a proper cone, meaning that it is closed, convex, and pointed. Then $C^{**} = C$. For C^* to be solid, we want to show $\text{int}(C^*) \neq \emptyset$. Suppose $\text{int}(C^*) = \emptyset$. Then because we have shown that C^* is a convex cone, we can conclude that $C^{**} = C$ is not pointed. This contradicts our assumption that C is indeed pointed. As such, C^* is solid.

Pointed : For point of contradiction, assume that C^* is not pointed. Then $\exists z \in C^*$ such that $z \neq 0$ and $-z \in C^*$. This means that $\langle z, x\rangle \geq 0$ and $-\langle z, x\rangle \geq 0$ for all $x \in C$, and thus $\langle z, x\rangle = 0$ for all $x \in C$. This implies that C is not solid since C would be a subset of the hyperplane $\langle z, x\rangle = 0$, which contradicts the assumption that C is a proper cone. Thus, C^* is pointed. □

2.4 Semidefinite Programming

Now that we have some powerful conclusions from conic programming, we can use these to attack semidefinite programming problems. As we will show, the set of semidefinite matrices is a proper cone.

Semidefinite matrices are symmetric, and the set of $n \times n$ real symmetric matrices is denoted $\mathcal{S}^n$. The set of $n \times n$ positive semidefinite matrices is denoted $\mathcal{S}^n_+$, and the set of $n \times n$ positive definite matrices is denoted $\mathcal{S}^n_{++}$. They are defined as

- $\mathcal{S}^n_+ := \{A \in \mathcal{S}^n : x^T Ax \geq 0, \forall x \in \mathbf{R}^n\}$
- $\mathcal{S}^n_{++} := \{A \in \mathcal{S}^n : x^T Ax > 0, \forall x \neq 0\}$

It should be obvious that $\mathcal{S}^n_+$ is a cone since $\forall \lambda \geq 0$, if $A \in \mathcal{S}^n_+$, then $x^T(\lambda A)x = \lambda x^T Ax \geq 0$, so $\lambda A \in \mathcal{S}^n_+$. We denote the concept of $X \in \mathcal{S}^n_+$ as $X \succeq 0$.

Theorem : $\mathcal{S}^n_+$ is a proper cone.

Proof. In order to show that $\mathcal{S}_+^n$ is a proper cone, we need to show that it is closed, convex, solid, and pointed.

Closed : The complement of $\mathcal{S}_+^n$ in $\mathbf{R}^n$ is the set $C := \{A \in \mathcal{S}^n : x^T Ax < 0, \text{ for some } x \in \mathbf{R}^n\}$. Let $\hat{A} \in C$. Then there exists a $\hat{x} \in \mathbf{R}^n$ such that $\hat{x}^T \hat{A} \hat{x} < 0$, and this statement remains true for all matrices in a sufficiently small neighborhood of $\hat{A}$. Therefore set C is open, and thus $\mathcal{S}_+^n$ is closed.

Convex : $\forall A, B \in \mathcal{S}_+^n$, $0 \leq \lambda \leq 1$, we have that $\forall x \in \mathbf{R}^n$,

$$x^T(\lambda A + (1-\lambda)B)x = \lambda(x^T Ax) + (1-\lambda)(x^T Bx) \geq 0,$$

since $0 \leq \lambda \leq 1$ and $A, B \in \mathcal{S}_+^n$. Thus, $\lambda A + (1-\lambda)B \in \mathcal{S}_+^n$, so $\mathcal{S}_+^n$ is convex.

Solid : We want to show that we have a nonempty interior. We will first show that if X is a symmetric matrix, then $\|X\|_F = \|\lambda(X)\|_2$, where $\lambda(X)$ is the vector of eigenvalues of X. We see that

$$\|X\|_F = \sqrt{\text{tr}(X^2)} = \sqrt{\sum_{i=1}^{n} \lambda_i^2} = \|\lambda(X)\|_2.$$

In order to show we have a nonempty interior, we need at least one element in our interior. We will pick to show that the $n \times n$ identity matrix I is in our interior. Indeed, $I \in \mathcal{S}_+^n$. We want to show that $\exists \epsilon > 0$ such that $\|I - X\|_F = \|\lambda(I - X)\|_2 \leq \epsilon$ implies that $X \in \mathcal{S}_+^n$. We will try $\epsilon = 1$.

Considering the potential eigenvalues λ of X, we would have $Xv = \lambda v$. Furthermore, the potential eigenvalues $\hat{\lambda}$ of $(I - X)$ would be $(I - X)w = w - Xw = (1 - \lambda)w = \hat{\lambda} w$. Thus, we have

$$\|\lambda(I - X)\|_2 = \sqrt{\sum_{i=1}^{n} (1 - \lambda_i)^2} \leq 1,$$

where λ_i is the ith eigenvalue of X. From this, we can see that $\lambda_i \geq 0$ for $i \in \{1, 2, \ldots, n\}$ since otherwise $(1-\lambda)^2 > 1$, which would violate our inequality. Since each eigenvalue of X is nonnegative, $X \in \mathcal{S}_+^n$. Therefore, $I \in \text{int}(\mathcal{S}_+^n)$, and $\mathcal{S}_+^n$ is solid.

Pointed : Suppose that $A \in \mathcal{S}_+^n$ and $A \in \mathcal{S}_-^n$, where $\mathcal{S}_-^n := -\mathcal{S}_+^n := \{A \in \mathcal{S}^n : x^T A x \leq 0, \forall x \in \mathbf{R}^n\}$. Then $\forall x \in \mathbf{R}^n$, we have $x^T A x \geq 0$ and $x^T A x \leq 0$. Therefore, $x^T A x = 0$. We will use this to show that $A = 0$, the matrix of all zeros. Since $x^T A x = 0 \ \forall x \in \mathbf{R}^n$, it holds for e_i, the ith column of the $n \times n$ identity matrix. Thus, we have

$$e_i^T A e_i = a_{ii}^2 = 0 \implies a_{ii} = 0, \forall i \in \{1, 2, \ldots, n\},$$

where a_{ii} is the entry of A in the ith row and ith column. It would also work for a vector $e_{i,j}$ of all zeros except for the ith and jth positions, where the entries are 1. Then we have

$$e_{i,j}^T A e_{i,j} = a_{ii}^2 + 2a_{ij} + a_{jj}^2 = 2a_{ij} = 0 \implies a_{ij} = 0, \forall i, j \in \{1, 2, \ldots, n\}.$$

Thus, $A = 0$, and $\mathcal{S}_+^n$ is pointed. □

2.4.1 Problem Formulation

The primal problem for a semidefinite optimization problem has the form

$$\begin{aligned} &\text{Minimize } \langle C, X \rangle \\ &\text{subject to } \langle A_i, X \rangle = b_i, \qquad i = 1, \ldots, m \\ &\qquad\qquad X \succeq 0, \end{aligned} \tag{SDP-P}$$

where $C, A_i \in \mathcal{S}^n$ and $\langle X, Y \rangle := \text{Tr}(XY) = \sum_{ij} X_{ij} Y_{ij}$. The dual problem for a semidefinite optimization problem has the form

$$\begin{aligned} &\text{Maximize } b^T y \\ &\text{subject to } \sum_{i=1}^{m} A_i y_i \preceq C, \end{aligned} \tag{SDP-D}$$

where $b = (b_1, \ldots, b_m)$, and $y = (y_1, \ldots, y_m)$ are the dual decision variables.

2.4.2 Properties

Following along with Appendix A.1.1 of Blekherman et al. (2013), we will list some equivalent statements about positive semidefinite matrices. For a symmetric matrix A, the following are equivalent:

- The matrix A is positive semidefinite ($A \succeq 0$).
- For all $x \in \mathbf{R}^n$, $x^T A x \geq 0$.
- All eigenvalues of A are nonnegative.
- There exists a factorization $A = BB^T$.

Proof. Let A be a positive semidefinite matrix. Suppose λ is an eigenvalue of A. Then $\exists v \in \mathbf{R}^n$ such that $Av = \lambda v$. That means that

$$v^T A v = \lambda v^T v \geq 0,$$

with the inequality being a result of the definition of semidefinite matrices. Since v is an eigenvector, $v \neq 0$, so $v^T v > 0$. As such, we know that $\lambda \geq 0$. As such, all eigenvalues of A are nonnegative.

Continuing to use our positive semidefinite matrix A, since A is a symmetric matrix, it has an eigenvalue decomposition of $A = V\Lambda V^T$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and the matrix V is orthogonal, i.e., $VV^T = V^T V = I$. Taking the square root of the matrix Λ as $\sqrt{\Lambda} := (\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$, and defining $B := V\sqrt{\Lambda}$, we have

$$A = V\Lambda V^T = V\sqrt{\Lambda}\sqrt{\Lambda}V^T = V\sqrt{\Lambda}(V\sqrt{\Lambda}^T)^T = V\sqrt{\Lambda}(V\sqrt{\Lambda})^T = BB^T,$$

which shows that a positive semidefinite matrix has a factorization of form $A = BB^T$.

To show that if a matrix has the form $A := BB^T$, then it is positive semidefinite, it suffices to show that the following holds for all $x \in \mathbf{R}^n$:

$$x^T Ax = x^T BB^T x = \|B^T x\|_2^2 \geq 0.$$

As such, A must be positive semidefinite. □

2.4.3 Simple Example

The following example is from Blekherman et al. (2013, p. 11). As with the linear programming examples, we will leave out the steps for solving since we will discuss our solvers later in the chapter.

Consider the semidefinite optimization problem

$$\begin{aligned} \text{Minimize} \quad & 2x_{11} + 2x_{12} \\ \text{subject to} \quad & x_{11} + x_{22} = 1, \\ & \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \succeq 0. \end{aligned} \tag{2.21}$$

This problem has the form of SDP-P, and here $m = 1$, $b_1 = 1$,

$$C = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

For this problem, the optimal solution is

$$X^* = \begin{bmatrix} \frac{2-\sqrt{2}}{4} & -\frac{1}{2\sqrt{2}} \\ -\frac{1}{2\sqrt{2}} & \frac{2+\sqrt{2}}{4} \end{bmatrix},$$

and the optimal value is $p^* = 1 - \sqrt{2}$.

The dual problem for (2.21) is

$$\begin{aligned} \text{Maximize} \quad & y \\ \text{subject to} \quad & \begin{bmatrix} 2-y & 1 \\ 1 & -y \end{bmatrix} \succeq 0. \end{aligned} \tag{2.22}$$

The optimal solution for this problem is $y^* = 1 - \sqrt{2}$, with an optimal value of $d^* = 1 - \sqrt{2}$. In this example, we see that we end up with a zero duality gap since $p^* = d^* = 1 - \sqrt{2}$.

2.4.4 Computational Complexity

Something that is important to note is that our semidefinite programs will not always solve the exact program we are trying to solve. A natural question is why we would use methods that merely approximate the answer to our problem as opposed to directly solving the problem. The short answer is that certain applications that we can apply semidefinite programming to have been proven to be NP-hard. However, semidefinite programming can be solved to accuracy ϵ in polynomial time, as we will see when discussing our solvers.

2.5 Solvers

In order to explore our applications, we needed to pick a programming language that would give us access to solvers for semidefinite programming problems. The language that we decided to use is Julia (Bezanson, Edelman, Karpinski, & Shah, 2017). We used its modeling language of JuMP in order to interact with our solvers (Dunning, Huchette, & Lubin, 2015).

There are a few different ways to implement semidefinite programming. In order to get a sense of which is the most effective for which problem, we will use two different solvers: SCS Solver and Mosek. The aim of this section is to give a brief introduction as to what these solvers are doing.

2.5.1 SCS Solver

SCS Solver is based on using the alternating direction method of multipliers (ADMM) to solve semidefinite programming problems (O'Donoghue, Chu, Parikh, & Boyd, 2016). ADMM is a convex optimization algorithm that dates back to the early 1980s that uses a first-order operator splitting method. It has gained some extra attention as of late because of its usefulness in machine learning and image processing applications. ADMM is often simple to run in a parallel manner, which is incredibly helpful for big data problems. It is typically used for applications that do not require a high level of accuracy since it has slow tail convergence (Eckstein & Yao, 2012).

It has been shown by Nishihara, Lessard, Recht, Packard, and Jordan (2015) that ADMM has a linear convergence "when one of the objective terms is strongly convex." This leads us to believe that SCS Solver will also have this rate of convergence. The article on the SCS Solver goes into detail about how it guarantees convergence and provides detail on the algorithm's stopping criteria. The article on SCS Solver also includes information on how it handles scaling data, as well as examples and numerical experiments (O'Donoghue et al., 2016).

2.5.2 Mosek

Mosek uses an interior-point method in order to solve semidefinite programming problems (ApS, 2015). This method involves following a "central path" in order to find its solution. After picking a point to start at, interior-point methods utilize Newton's method in order to choose a direction for the next point in the iteration, moving towards the optimal solution with each step. The step size is then determined by making sure that the step results in a

point that is within the interior of the cone. A detailed explanation of interior-point methods can be found in Nocedal and Wright (2006). According to Borchers (1999), interior-point methods can be solved in polynomial time based on the number of variables and constraints used.

CHAPTER 3

POLYNOMIAL OPTIMIZATION

The focus of this chapter will be on polynomial optimization. We will start with nonnegative polynomials and discuss how some of them may be represented as a sum -of- squares, then we will show the connection between sum -of- squares polynomials and semidefinite matrices. The motivation behind this is that we wish to formulate our optimization problems as

$$\text{Maximize } \gamma \quad \text{subject to} \quad p(x_1, \ldots, x_n) - \gamma \quad \text{is nonnegative,}$$

and our goal is to determine in what ways we may apply semidefinite programming to polynomial optimization. The bulk of this information will be adapted from Chapter 3 of Blekherman et al. (2013).

3.1 Nonnegative Polynomials

Our discussion will consider polynomials with n variables with real coefficients. A multivariate polynomial $p(x_1, \ldots, x_n)$ is *nonnegative* if it takes only nonnegative values, i.e.,

$$p(x_1, \ldots, x_n) \geq 0 \qquad \text{for all } (x_1, \ldots, x_n) \in \mathbf{R}^n. \tag{3.1}$$

There are a number of natural questions about nonnegative polynomials to consider going forward:

- How do we decide if a polynomial $p(x)$ is nonnegative?

- Is it possible to certify whether or not a polynomial is nonnegative efficiently?
- How computationally complex is it to decide if a polynomial is nonnegative?
- Are there any features of the structure of the set of nonnegative polynomials that we can exploit?

Before we answer these questions, we will first take a look at some simple special cases.

3.1.1 Univariate Polynomials

We will start with the case of polynomials that only have a single variable, i.e., when $n = 1$:

$$p(x) = p_d x^d + p_{d-1} x^{d-1} + \cdots + p_1 x + p_0. \tag{3.2}$$

We normally assume that the leading coefficient p_d is nonzero, and it sometimes can be helpful to normalize the polynomial $p(x)$ to make it *monic*; i.e., we make $p_d = 1$. We say that the *roots* of $p(x)$ are the values of x at which $p(x) = 0$. By the fundamental theorem of algebra, there is a unique factorization

$$p(x) = p_d * \prod_{i=1}^{d} (x - x_i), \tag{3.3}$$

where the (complex) roots x_i may have multiplicities, meaning that they are not all necessarily distinct.

In this case, we can clearly tell that in order for $p(x)$ to be nonnegative, it is a necessary condition that the degree of $p(x)$ be even. If the degree is odd, then $p(x)$ will become negative as either $x \to \infty$ or $x \to -\infty$.

3.1.2 Multivariate Polynomials

Let $P_{n,2d}$ be the set of nonnegative polynomials that have n variables of degree less than or equal to $2d$, i.e.,

$$P_{n,2d} = \{p \in \mathbf{R}[x]_{n,2d} \mid p(x) \geq 0, \quad \forall x \in \mathbf{R}^n\}.$$

If we identify a polynomial with its $N := \binom{n+d}{d}$ coefficients, and we notice that the constraints $p(x) \geq 0$ are affine in the coefficients of p for every fixed x, it follows directly that $P_{n,2d}$ is a *convex* set in $\mathbf{R}[x]_{n,2d} \sim \mathbf{R}^N$. We also have that the following is true.

Theorem 3. *The set of nonnegative polynomials $P_{n,2d}$ is a proper cone (closed, convex, pointed, and solid) in $\mathbf{R}[x]_{n,2d} \sim \mathbf{R}^N$.*

Proof. In order to show that $P_{n,2d}$ is a proper cone, we need to show that it is closed, convex, solid, and pointed. We should also show that it is indeed a cone.

Cone : $\forall \lambda \geq 0$, $p \in P_{n,2d}$, $\lambda p(x) \geq 0$.

Closed : Noticing that $p(x) \geq 0$ defines a closed half space for each fixed x, we can consider $P_{n,2d}$ to be the infinite intersection of closed half spaces, which means that $P_{n,2d}$ is closed.

Convex : $\forall p, q \in P_{n,2d}$, $0 \leq \lambda \leq 1$, we have that $\forall x \in \mathbf{R}^n$,

$$\lambda p(x) + (1 - \lambda) q(x) \geq 0,$$

since $0 \leq \lambda \leq 1$ and $p, q \in P_{n,2d}$. Thus, $\lambda p(x) + (1 - \lambda) q(x) \in P_{n,2d}$, so $P_{n,2d}$ is convex.

Solid : We want to show that we have a nonempty interior. The interior here would be $\text{int}(P_{n,2d}) = \{p \in \mathbf{R}[x]_{n,2d} \mid p(x) > 0,\ \forall x \in \mathbf{R}^n\}$. We see here by simple example that $p(x) = 1 + \sum_{i=1}^{n} x_i^{2d} \in \text{int}(P_{n,2d})$. Therefore, $P_{n,2d}$ is solid.

Pointed : If $p \in P_{n,2d}$ and $p \in -P_{n,2d}$, then $p(x) \geq 0$ and $-p(x) \geq 0$, which implies that $p(x) = 0$. Thus, $P_{n,2d}$ is pointed. □

3.2 Sum of Squares

A multivariate polynomial $p(x_1, \ldots, x_n)$ is a *sum -of- squares* (sos) if it can be written as the sum -of- squares of some other polynomials. Formally, a polynomial $p(x) \in \mathbf{R}[x]_{n,2d}$ is a sum -of- squares (sos) if there exist $q_1, \ldots, q_m \in \mathbf{R}[x]_{n,d}$ such that

$$p(x) = \sum_{k=1}^{m} q_k^2(x). \tag{3.4}$$

We will use $\sum_{n,2d}$ to denote the set of sos polynomials with n variables of degree less than or equal to $2d$. If a polynomial $p(x) \in \sum_{n,2d}$, then it is obvious that $p(x) \geq 0$ for all $x \in \mathbf{R}^n$. This shows that a polynomial that is sos is immediately a nonnegative polynomial, i.e., $\sum_{n,2d} \subseteq P_{n,2d}$. In general, sos decompositions are not unique. We can quickly see from this that $\sum_{n,2d}$ is a convex cone. In fact, we will show later that $\sum_{n,2d}$ is actually a proper cone.

3.2.1 When Is Nonnegativity Equal to Sum -of- Squares?

Now that we know that $p(x) \in \sum_{n,2d} \implies p(x) \in P_{n,2d}$, it is natural to ask when the converse holds true, i.e., when a nonnegative polynomial can be expressed as a sum -of-

squares. According to Blekherman et al. (2013), David Hilbert proved that $P_{n,2d} = \sum_{n,2d}$ only in the following three cases:

- Univariate polynomials (i.e., $n = 1$).
- Quadratic polynomials ($2d = 2$).
- Bivariate quartics ($n = 2, 2d = 4$).

For all other cases, there always exist nonnegative polynomials that are *not* sums of squares. We will prove the first two cases.

Univariate nonnegative polynomials are sos. Suppose we have a univariate nonnegative polynomial $p(x) = p_{2d}x^{2d} + \cdots + p_1x + p_0$. Utilizing the fundamental theorem of algebra, this polynomial has a factorization of the form

$$p(x) = p_{2d} * \prod_j (x - r_j)^{n_j} * \prod_k [(x - z_k)(x - z_k^*)]^{m_k},$$

where r_j and (z_k, z_k^*) are the real and complex roots of $p(x)$. Because $p(x)$ is a univariate nonnegative polynomial, we know that $p_{2d} > 0$ because of the necessary end behavior in order to guarantee nonnegativity. For our complex roots, we can write $z_k = a_k + ib_k$ and $z_k^* = a_k - ib_k$. This allows us to rewrite part of the above formulation of $p(x)$ as

$$(x-z_k)(x-z_k^*) = (x-a_k-ib_k)(x-a_k+ib_k) = x^2-2a_kx+a_k^2+b_k^2 = (x-a_k)^2+b_k^2.$$

Thus, that portion of the polynomial is a sum -of- squares and as such is positive. Next we will show that the n_j's must be even. Assume that there exists an n_j which is odd. If we focus on this particular exponent, which we will call $n_{j_{\text{odd}}}$,

then the root $r_{j_{\text{odd}}}$ must cause our polynomial to change signs, which contradicts our nonnegativity condition for $p(x)$. Thus, each of the n_j's must be even.

Finally, we will show that our polynomial is a sum -of- squares. Suppose that there exist two polynomials $q(x)$ and $t(x)$ that are sums of squares. Thus, they can be written in the forms $q(x) = \sum_{c=1}^{u} f_c^2(x)$ and $t(x) = \sum_{d=1}^{v} g_d^2(x)$. If we take their product, we obtain

$$\begin{aligned} q(x)t(x) = \left(\sum_{c=1}^{u} f_c^2(x)\right)\left(\sum_{d=1}^{v} g_d^2(x)\right) &= \sum_{c=1}^{u}\left[f_c^2(x)\left(\sum_{d=1}^{v} g_d^2(x)\right)\right] \\ &= \sum_{c=1}^{u}\left[\sum_{d=1}^{v} f_c^2(x)g_d^2(x)\right] \\ &= \sum_{c=1}^{u}\left[\sum_{d=1}^{v} \left(f_c(x) * g_d(x)\right)^2\right]. \end{aligned}$$

As we can see, this means that a product of a sum -of- squares will also be a sum -of- squares. Since we are multiplying squares and sums of squares in order to form $p(x)$, it is indeed a sum -of- squares. □

Quadratic nonnegative polynomials are sos. Let $p(x)$ be a quadratic polynomial in n variables. According to Blekherman et al. (2013, p. 51), every quadratic polynomial can be represented as

$$p(x) = x^T Ax + 2b^T x + c,$$

where $A \in \mathcal{S}^n$ is a symmetric matrix. We can factor this as

$$p(x) = x^T Ax + 2b^T x + c = \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} A & b \\ b^T & c \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = v^T Qv.$$

Also according to Blekherman et al. (2013, p. 51), since $p(x)$ is nonnegative, we are guaranteed that

$$\begin{bmatrix} A & b \\ b^T & c \end{bmatrix} = Q \succeq 0.$$

This implies that we can rewrite $Q = M^T M$, and as such

$$p(x) = \begin{bmatrix} x \\ 1 \end{bmatrix}^T M^T M \begin{bmatrix} x \\ 1 \end{bmatrix} = ||M \begin{bmatrix} x \\ 1 \end{bmatrix} ||_2^2,$$

which shows that $p(x)$ is indeed a sos polynomial. □

3.3 Sum -of- Squares and Semidefinite Matrices

At this point, a natural question is how one would decide if a polynomial $p(x)$ belongs to the cone $\sum_{n,2d}$. As we will explore, this is related to its connection to semidefinite programming.

3.3.1 Univariate Polynomials

We will begin with the univariate case. Consider a univariate polynomial $p(x)$ of degree $2d$:

$$p(x) = p_{2d}x^{2d} + p_{2d-1}x^{2d-1} + \cdots + p_1 x + p_0. \tag{3.5}$$

Assume that $p(x)$ is a sum -of- squares, i.e., it can be expressed as in (3.4). It is important to note that the degree of the polynomials q_k used for the sum -of- squares representation can be of degree of at most d. We can then write

$$\begin{bmatrix} q_1(x) \\ q_2(x) \\ \vdots \\ q_m(x) \end{bmatrix} = V \begin{bmatrix} 1 \\ x \\ \vdots \\ x^d \end{bmatrix}, \tag{3.6}$$

where $V \in \mathbf{R}^{m \times (d+1)}$, and the kth row of V contains the coefficients for the polynomial q_k. Define $[x]_d$ as the vector of monomials on the right-hand side of (3.6), and define the matrix $Q := V^T V$. Then we have

$$p(x) = \sum_{k=1}^{m} q_k^2(x) = (V[x]_d)^T (V[x]_d) = [x]_d^T V^T V [x]_d = [x]_d^T Q [x]_d.$$

This result leads to the following lemma.

Lemma 1. *Let $p(x)$ be a univariate polynomial of degree $2d$. Then $p(x)$ is a sum -of- squares if and only if there exists a symmetric matrix $Q \in \mathcal{S}^{d+1}$ that satisfies*

$$p(x) = [x]_d^T Q [x]_d, \qquad Q \succeq 0. \tag{3.7}$$

We typically call the matrix Q the *Gram matrix* of the sum -of- squares representation. One part of this lemma follows from our properties of semidefinite matrices, as constructing the matrix Q as $V^T V$ immediately gives us that Q is positive semidefinite. For the other direction, assume that there exists a positive semidefinite matrix Q that satisfies (3.7). From there, we can factorize $Q = V^T V$ by using either Cholesky or square root factorization. This will provide us with our sum -of- squares decomposition of $p(x)$.

The condition (3.7) provides us with a semidefinite program. To see this, notice that the constraint $p(x) = [x]_d^T Q [x]_d$ is *affine* in the matrix Q, and as such the set of possible Gram matrices Q is given by the intersection of an affine subspace and the cone of positive semidefinite matrices. To obtain explicit equations for this semidefinite program, we will index the rows and columns of Q by $\{0, \ldots, d\}$ as

$$[x]_d^T Q [x]_d = \sum_{i=0}^{d} \sum_{j=0}^{d} Q_{ij} x^{i+j} = \sum_{k=0}^{2d} \left(\sum_{i+j=k} Q_{ij} \right) x^k.$$

So for this expression to be equal to $p(x)$, we must have that

$$p_k = \sum_{i+j=k} Q_{ij}, \qquad k = 0, \ldots, 2d. \tag{3.8}$$

This provides us with a system of $2d+1$ linear equations between the entries of Q and the coefficients of $p(x)$. Therefore, since Q is constrained to be positive semidefinite while also belonging to the affine subspace that is defined by our system of linear equations, we have that our sos condition is equivalent to a semidefinite programming problem. This means that we have shown the following lemma.

Lemma 2. *A univariate polynomial $p(x) = \sum_{k=0}^{2d} p_k x^k$ is a sum -of- squares if and only if there exists a positive semidefinite matrix $Q \in \mathcal{S}^{d+1}$ satisfying* (3.8).

Note that we have previously shown that nonnegativity and sum -of- squares are equivalent conditions in the univariate case. Thus, Lemma 2 completely characterizes the set of univariate nonnegative polynomials and shows that the sets $P_{1,2d}$ and $\sum_{1,2d}$ are equal.

3.3.2 Multivariate Polynomials

We will now discuss the multivariate case. Consider a polynomial $p(x_1, x_2, \ldots, x_n)$ of degree $2d$ in n variables. The number of coefficients for p is $\binom{n+2d}{2d}$. We will let $p(x) = \sum_\alpha p_\alpha x^\alpha$, where α are tuples of exponents $\alpha \in \{(\alpha_1, \alpha_2, \ldots, \alpha_n) \mid \alpha_1 + \cdots + \alpha_n \leq 2d, \alpha_i \geq 0 \quad \forall i = 1, \ldots, n\}$.

Let $[x]_d := [1, x_1, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^d]^T$ be the vector of all $\binom{n+d}{d}$ monomials in $x_1, \ldots, x_n$ of degree less than or equal to d, and consider the equation

$$p(x) = [x]_d^T Q [x]_d, \tag{3.9}$$

where Q is an $\binom{n+d}{d} \times \binom{n+d}{d}$ symmetric matrix. Following the process used to arrive at (3.8) - that is, indexing the matrix Q by the $\binom{n+d}{d}$ monomials in n variables with their associated exponent tuples - allows us to obtain the following conditions:

$$p_\alpha = \sum_{\beta+\gamma=\alpha} Q_{\beta\gamma}, \qquad Q \succeq 0. \tag{3.10}$$

This provides us with a system of $\binom{n+2d}{2d}$ linear equations, one for each coefficient of $p(x)$. As before, these equations are affine conditions that relate the entries of Q to the coefficients of $p(x)$. This allows us to decide if a polynomial is a member of, or optimize over, the set of sos polynomials by solving an SDP program. This relationship between positive semidefinite matrices and sum -of- squares is summarized by the following theorem:

Theorem 4. *A multivariate polynomial $p(x) = \sum_\alpha p_\alpha x^\alpha$ in n variables and degree $2d$ is a sum -of- squares if and only if there exists $Q \in \mathcal{S}_+^{\binom{n+d}{d}}$ that satisfies* (3.10)*. As a consequence, membership in $\sum_{n,2d}$ can be decided via semidefinite programming.*

The size of the matrix of the semidefinite program that appears in (4) is $\binom{n+d}{d}$, which grows polynomially in the number of variables n for a fixed degree d. If n is fixed, it also grows polynomially in d since $\binom{n+d}{d} = \binom{n+d}{n}$.

Now that we have shown the connection between semidefinite matrices and sos polynomials, we can more easily show that the set of sos polynomials forms a proper cone.

Theorem 5. *The set of sos polynomials $\sum_{n,2d}$ is a proper cone (closed, convex, pointed, and solid) in $\mathbf{R}[x]_{n,2d} \sim \mathbf{R}^N$.*

Proof. Consider a polynomial $p(x) \in \sum_{n,2d}$. We have shown previously that there exists a representation of $p(x)$ as $[x]_d^T Q[x]_d$, where $Q \succeq 0$. Because we can represent all possible $p(x)$ as a positive semidefinite matrix, and we have shown that $\mathcal{S}_+^n$ is a proper cone, it follows that $\sum_{n,2d}$ is also a proper cone. We will elaborate further.

Convex : Let $p(x) = \sum_{i=1}^a q_i^2(x) \in \sum_{n,2d}$ and $r(x) = \sum_{j=1}^b s_j^2(x) \in \sum_{n,2d}$. Then for $0 \leq \lambda \leq 1$, we have

$$\lambda p(x)+(1-\lambda)r(x) = \lambda \sum_{i=1}^{a} q_i^2(x)+(1-\lambda)\sum_{j=1}^{b} s_j^2(x) = \sum_{i=1}^{a}(\sqrt{\lambda}q_i(x))^2+\sum_{j=1}^{b}(\sqrt{1-\lambda}s_j(x))^2 \in \sum_{n,2d}.$$

Thus, $\sum_{n,2d}$ is convex.

Pointed : If $p \in \sum_{n,2d}$ and $p \in -\sum_{n,2d}$, then $p(x) \geq 0$ and $-p(x) \geq 0$, which implies that $p(x) = 0$. Thus, $\sum_{n,2d}$ is pointed.

Solid : Choose Q to be the identity matrix. This gives us a representation for a sos polynomial, and because of our properties of $\mathcal{S}_n^+$ we know that it is in the interior of the set of sos polynomials. Thus, $\sum_{n,2d}$ is solid.

Closed : Noticing that $p(x) \geq 0$ defines a closed half space for each fixed x, we can consider $\sum_{n,2d}$ to be the infinite intersection of closed half spaces, which means that $\sum_{n,2d}$ is closed. □

3.4 Polynomial Optimization with Sum -of- Squares Programming

In this section, we will discuss how we can use sum -of- squares and semidefinite programming in order to solve both unconstrained and constrained optimization problems for univariate and multivariate polynomials.

3.4.1 Unconstrained Univariate Polynomial Optimization

Our first application will be our simplest, as we will demonstrate how to perform global optimization of a univariate polynomial $p(x)$. While there are many other methods that could handle this task, walking through it will help us outline certain features for our more complicated applications.

Instead of computing a minimizer x_* that results in a global minimum $p(x_*)$, we will instead frame our approach as obtaining a best possible lower bound on the optimal value of $p(x)$. Clearly a number γ is a global lower bound for $p(x)$ if and only if the polynomial $p(x) - \gamma$ is nonnegative. In other words, for all $x \in \mathbf{R}$,

$$p(x) \geq \gamma \iff p(x) - \gamma \geq 0.$$

This naturally leads to the following optimization problem:

$$\text{Maximize } \gamma \quad \text{subject to} \quad p(x) - \gamma \quad \text{is nonnegative.} \tag{OPT-NN}$$

Because this problem is defined by an infinite number of linear inequalities (one for each value of x), this is a convex optimization problem. Its optimal solution γ_* is equal to the global minimum of the polynomial $p(x_*)$. Now let us consider the following optimization problem:

$$\text{Maximize } \gamma \quad \text{subject to} \quad p(x) - \gamma \quad \text{is sos.} \qquad \text{(OPT-SOS)}$$

The only difference between the two above formulas comes from the replacement of the nonnegativity condition with a sum -of- squares condition. As we have shown previously, these two conditions are equivalent for the univariate case. The form OPT-SOS is in the form of a sos program, which means that it is equivalent to a semidefinite program for us to solve. This results in us being able to use semidefinite programming to find the global minimum value for $p(x)$. Notice that even if $p(x)$ is highly nonconvex, we can still find its global minimum with convex optimization.

3.4.2 Multivariate Unconstrained Polynomial Optimization

We will now consider the multivariate case of unconstrained polynomial optimization. Following our framework for the formulation of the univariate case, we can formulate the global minimum of a multivariate polynomial $p(x_1, \ldots, x_n)$ as follows:

$$\text{Maximize } \gamma \quad \text{subject to} \quad p(x_1, \ldots, x_n) - \gamma \quad \text{is nonnegative.} \qquad \text{(MOPT-NN)}$$

Since we do not wish for our constraint set to be nonnegative polynomials and would instead prefer to work with sos constraints, we can relax the global minimum as

$$\text{Maximize } \gamma \quad \text{subject to} \quad p(x_1, \ldots, x_n) - \gamma \quad \text{is sos.} \qquad \text{(MOPT-S0S)}$$

Note that this time, we are not guaranteed an exact solution by solving MOPT-SOS because we cannot guarantee equality between nonnegativity and sos in the multivariate case. However, we can use it to establish a bound on our desired value. Let p_* be the optimal value of MOPT-NN and p_{sos} be the optimal value of MOPT-SOS. Because we are not guaranteed equality, we have the inequality

$$p_{sos} \leq p_*,$$

providing us with a lower bound on the global minimum of $p(x_1, \ldots, x_n)$. This is expected since multivariate polynomial optimization is NP-hard while semidefinite programming is solvable in polynomial -time to a given accuracy.

3.4.3 Constrained Optimization

It is natural to question how we may rewrite our optimization problems if we only are concerned with a polynomial $p(x)$ being nonnegative over a subset $S \subset \mathbf{R}^n$. In other words, is there a way to consider constraints in our optimization formulation? We will consider separately sets S defined by equalities, inequalities, and a combination of them.

We will start by considering equality constraints. Consider a set S described by polynomial equations, i.e., of the form

$$S = \{x \in \mathbf{R}^n \mid q_1(x) = 0, \ldots, q_m(x) = 0\}.$$

Drawing parallels to weak duality and Lagrange multipliers, we will write our constraints as the following condition:

$$p(x) + \sum_{i=1}^{m} \lambda_i(x) q_i(x) \quad \text{is sos}, \tag{3.11}$$

where $\lambda_i(x)$ are arbitrary polynomials. This condition obviously implies that $p(x)$ is nonnegative on S, since our summation disappears and we are left with a sos (and thus nonnegative) $p(x)$. If we restrict the degrees of the λ_i's, our condition has the form of a sos program.

Now let us consider sets S that are formed using inequalities. Consider a set S described by polynomial inequalities, i.e., of the form

$$S = \{x \in \mathbf{R}^n \mid r_1(x) \geq 0, \ldots, r_m(x) \geq 0\}.$$

Following a similar approach to the one that gave us (3.11), we can consider expressing $p(x)$ as

$$p(x) = s_0(x) + \sum_{i=1}^{m} s_i(x) r_i(x), \tag{3.12}$$

where $s_0(x)$ and $s_i(x)$ are sos polynomials. By doing this, we guarantee that $p(x)$ will be nonnegative on the set S since all parts of the representation will be positive. This gives us a way of incorporating constraints in our optimization problem.

It is natural to ask how we may certify nonnegativity on a set S defined by both polynomial equations and inequalities, i.e., of the form

$$S = \{x \in \mathbf{R}^n \mid q_1(x) = 0, \ldots, q_k(x) = 0, r_1(x) \geq 0, \ldots, r_m(x) \geq 0\}.$$

Simply put, we would combine (3.11) and (3.12) in order to have conditions of the form

$$p(x) + \sum_{i=1}^{m} \lambda_i(x) q_i(x) \quad \text{is sos}, \tag{3.13}$$

where $\lambda_i(x)$ are arbitrary polynomials and

$$p(x) = s_0(x) + \sum_{j=1}^{m} s_j(x) r_j(x), \tag{3.14}$$

where $s_0(x)$ and $s_j(x)$ are sos polynomials. These conditions would certify that $p(x)$ was nonnegative on the set S.

3.5 Numerical Results and Applications

Now that we have a framework for using sos polynomials and semidefinite programming in order to solve polynomial optimization problems, we can set up numerical experiments and application problems in order to test our two solvers, SCS Solver (which is using ADMM) and Mosek (which is using an interior-point method).

3.5.1 Numerical Experiments

In order to get a sense of how these two different methods of solving compare, we can run each of them through a set of benchmark problems. This will help us determine when one solver may be at an advantage over the other. Our first benchmark problem comes from Blekherman et al. (2013), with the rest being randomly created custom problems. The results of our benchmarking are listed in Table 3.1. For ease of reading, the CPU times are rounded to the nearest hundredth of a second. To see the code that we implemented for this table, visit our GitHub page. [1]

[1] https://github.com/Dracoback/Polynomial-Optimization-using-Semidefinite-Programming/tree/master/Benchmark%20Problems

Table 3.1: Numerical Experiments Run on Both Solvers

Problem	n	m	2d	SCS CPU	Mosek CPU
Blekherman Exer. 3.62	1	1	4	0.00	0.00
Custom Problem 01	2	1	4	0.03	0.00
Custom Problem 02	3	1	4	0.29	0.03
Custom Problem 03	4	1	4	0.16	0.02
Custom Problem 04	4	2	4	0.13	0.02
Custom Problem 05	4	3	4	0.40	0.02
Custom Problem 06	4	4	4	0.23	0.02
Custom Problem 07	4	4	6	1.60	0.11
Custom Problem 08	5	4	6	0.66	0.86
Custom Problem 09	6	4	6	6.97	5.47
Custom Problem 10	5	5	6	0.62	0.78
Custom Problem 11	6	2	6	7.99	7.48
Custom Problem 12	5	4	8	3.29	39.80
Custom Problem 13	3	1	6	0.39	0.03
Custom Problem 14	3	1	8	0.66	0.13
Custom Problem 15	4	4	8	2.98	2.56
Custom Problem 16	4	4	10	3.79	38.36
Custom Problem 17	5	2	8	0.25	41.83
Custom Problem 18	3	2	10	1.95	0.81
Custom Problem 19	3	3	14	2.82	26.55

The table points out some interesting trends. The number of variables and the number of constraints do not seem to have a profound impact on the time of computation on their own. The degree of the polynomial seems to have the greatest impact on the CPU time, especially for Mosek.

3.5.2 Application: Wireless Coverage Using Minimum Transmission

The work here recreates one of the application problems posed by Ahmadi and Majumdar (2016).

In this problem, we are given n wireless electromagnetic transmitters whose positions $(\bar{x}_i, \bar{y}_i), \imath = 1, \ldots, n$ are known, and we are told that we would like to achieve a certain coverage level over certain areas around these transmitters while minimizing the overall transmission rates coming from the transmitters. Each transmitter emits waves in all directions with equal intensity. We are provided with an equation for determining the amount of energy E_i coming from each transmitter from the laws of electromagnetics:

$$E_i(x, y) = \frac{c_i \lambda}{(x - \bar{x}_i)^2 + (y - \bar{y}_i)^2},$$

where c_i is the transmission rate from the ith device and λ is some propagation constant, which going forward we will set to 1 without loss of generalization. Our goal is to make sure that certain regions β_j are guaranteed to receive a total energy of at least C units while minimizing the total transmission power of the transmitters. In addition, each transmitter has an upper limit on its transmission rate γ_i.

To summarize, we are given the following values as input: C (required coverage level), γ_i (upper bounds on transmission rates), $(\bar{x}_i, \bar{y}_i)$ (location of our transmitters), and β_j, $j = 1, \ldots, m$ (basic semialgebraic sets describing regions to be covered). Our goal is to find transmission rates c_i to solve the following optimization problem:

$$\begin{aligned} &\text{Minimize} && \sum_{i=1}^{n} c_i \\ &\text{subject to} && c_i \leq \gamma_i, \quad \forall i = 1 \ldots n, \\ E(x,y) := &\sum_{i=1}^{n} \frac{c_i}{(x-\bar{x}_i)^2 + (y-\bar{y}_i)^2} \geq C, && \forall (x,y) \in \beta_j, j = 1, \ldots m. \end{aligned} \tag{3.15}$$

We can manipulate the last line of constraints in order to obtain a polynomial that must be nonnegative on our β_j's. Then $\forall (x,y) \in \beta_j, j = 1, \ldots, m$, we have

$$p(x,y) := -C\Pi_{i=1}^{n}[(x-\bar{x}_i)^2 + (y-\bar{y}_i)^2] + \sum_{i=1}^{n} c_i \Pi_{k \neq i}[(x-\bar{x}_k)^2 + (y-\bar{y}_k)^2] \geq 0. \tag{3.16}$$

The degree of this polynomial is two times the number of transmitters. We will let each set β_j be defined as

$$\beta_j = \{x : r_{j,1}(x,y) \geq 0, \ldots, r_{j,k_j}(x,y) \geq 0\},$$

for some bivariate polynomials $r_{j,1}, \ldots, r_{j,k_j}$. Basing the reformulation of our problem on (3.12), we end up with the following optimization problem:

$$\begin{aligned} &\text{Minimize} && \sum_{i=1}^{n} c_i \\ &\text{subject to } p = && \sigma_{j,0} + \sum_{i=1}^{k_j} \sigma_{j,i} r_{j,i}, \qquad j = 1 \ldots m, \\ & && \sigma_{j,0}, \sigma_{j,i}, \qquad\qquad \text{sos}, \end{aligned} \tag{3.17}$$

where p is the same as in (3.16) and $\sigma_{j,0}, \sigma_{j,i}$ are bivariate polynomials whose degree is upper bounded by some even integer d. This provides us with a semidefinite program to solve.

Our code for the following problems is available on our GitHub page. [2]

Example 1 - Two Transmitters and Five Regions

We now use this to solve a concrete example. We are given two transmitters whose upper bounds for their transmission rates are both 11. The first transmitter is located at $(1, 1.5)$ while the second transmitter is located at $(2, 1)$. We are given five ellipsoidal regions to cover:

$$\beta_1 = \{(x, y) : 0.01 - 3(x - 1.1)^2 - 2(x - 1.1)(y - 1.75) - (y - 1.75)^2 \geq 0\},$$

$$\beta_2 = \{(x, y) : 0.01 - (x - 1.25)^2 - 3(y - 2)^2 \geq 0\},$$

$$\beta_3 = \{(x, y) : 0.01 - (x - 1.5)^2 - (y - 1.75)^2 \geq 0\},$$

$$\beta_4 = \{(x, y) : 0.01 - (x - 1.8)^2 + 2(x - 1.8)(y - 1.8) - 3(y - 1.8)^2 \geq 0\},$$

$$\beta_5 = \{(x, y) : 0.02 - 5(x - 2)^2 - (y - 1.4)^2 \geq 0\}.$$

Each of these areas has a required energy level of 10.

A natural question to consider is if both transmitters are even necessary. If we remove one transmitter at a time from our problem, we see that transmitter 1 (located at $(1, 1.5)$) would need a transmission rate of 11.446 and transmitter 2 (located at $(2, 1)$) would need a transmission rate of 17.594. As both of these numbers are above our upper bounds for transmission rates, we can clearly see that both transmitters will be necessary. When we include both transmitters and set our sos polynomials to degree 2, we end up with an optimal solution of $c_1 = 2.554$ and $c_2 = 5.564$, for a cumulative transmission rate of 8.118.

See Figure 3.1. When we ran this example with both of our solvers, Mosek had no problems finding our optimal solution very quickly, with a CPU time of 0.02 seconds. SCS

[2] https://github.com/Dracoback/Polynomial-Optimization-using-Semidefinite-Programming/tree/master/Application%20-%20Wireless%20Transmission%20Problems

Solver, on the other hand, was unable to obtain the optimal solution. Even after increasing the number of iterations allowed, SCS Solver was unable to determine the optimal solution.

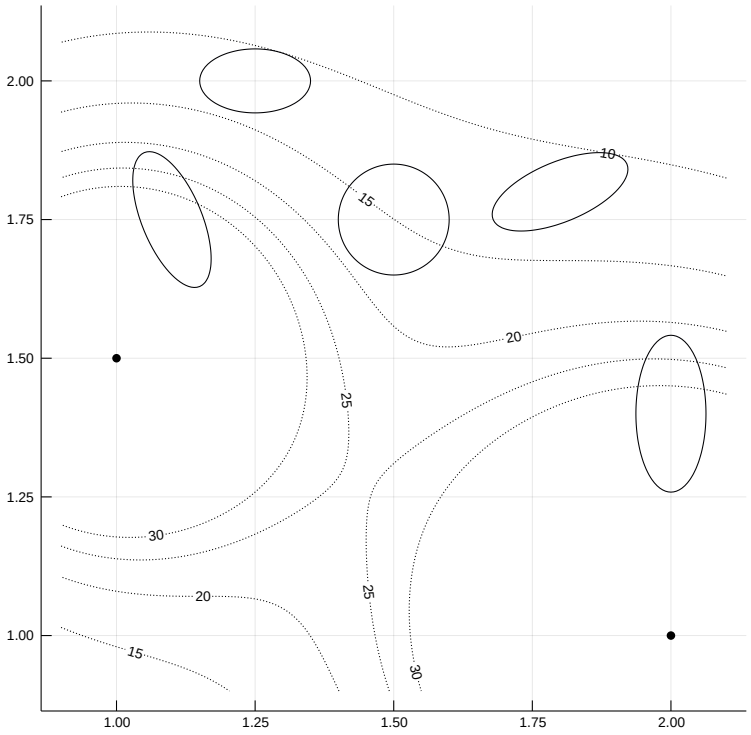

Figure 3.1: Two transmitters with five regions.

Example 2 - Three Transmitters and Eight Regions

Now our company has acquired a third transmitter in this area, located at $(1.5, 2.5)$. In addition, we are given three more ellipsoidal regions to cover:

$$\beta_6 = \{(x,y) : 0.02 - 4(x-1.6)^2 + 2(x-1.6)(y-2.3) - 2(y-2.3)^2 \geq 0\},$$
$$\beta_7 = \{(x,y) : 0.01 - 3(x-0.9)^2 - (y-2.3)^2 \geq 0\},$$
$$\beta_8 = \{(x,y) : 0.02 - 2(x-1.55)^2 + 2(x-1.55)(y-2.8) - 2(y-2.8)^2 \geq 0\}.$$

Each of these areas also has a required energy level of 10.

Running our solver on the updated problem, we see that it is indeed possible to obtain our goal of obtaining the required coverage over all of the regions. See Figure 3.2. In fact, with our new transmitter, we end up having a smaller overall transmission rate! Our new solution was $c_1 = 0.305$, $c_2 = 1.830$, and $c_3 = 4.451$, resulting in a cumulative transmission rate of 6.586.

In fact, the numbers are so low that it is natural to ask if we even need all three towers. Can our newest transmission tower handle all of the regions on its own? In short, no. Without the other towers, we would need $c_3 = 17.969$. We should consider if only two of the transmitters can handle the job. Running some experiments, we determine that we can remove either of the first two transmitters and still be okay. However, it is impossible to cover all eight regions if we remove the newest transmitter.

When we ran this example with both of our solvers, Mosek once again had no problems finding our optimal solution very quickly, with a CPU time of 0.06 seconds. SCS Solver, on the other hand, obtained an incorrect solution where each of the c_i's was roughly 0.001.

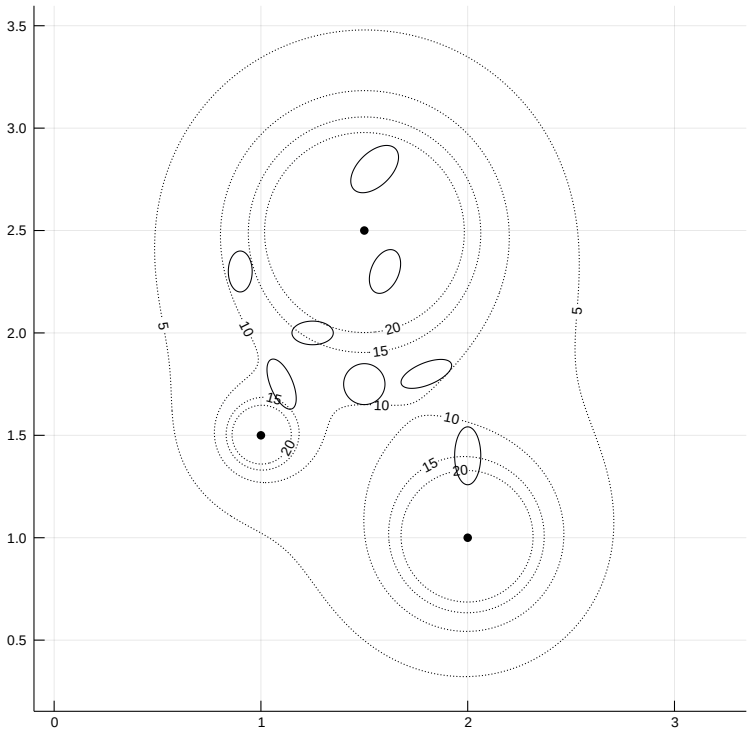

Figure 3.2: Three transmitters with eight regions.

CHAPTER 4

CONCLUSION

The purpose of this thesis was to discuss how to solve polynomial optimization problems. We did this by first discussing optimization problems in general and focused on the case of semidefinite programming. From there, we discussed how we can find the optimal value for any polynomial by rewriting it with a nonnegativity constraint, which we then rewrote as a sos constraint. This allowed us to apply our semidefinite programming to polynomial optimization problems, which meant that we could quickly solve them.

As much as we have covered in this thesis, there are still several areas we would like to explore that are related to what we covered here.

1. **ADMM and Interior -Point Methods** - While we briefly discussed the methods that our solvers utilized in order to solve semidefinite programming problems, we did not get to truly dig into the details and determine why our numerical experiments obtained the results they did. It would be interesting to explore these methods and see if they could be improved upon.

2. **Other Applications** - Needless to say, there is more than one application of polynomial optimization. In Ahmadi and Majumdar (2016) alone, there are also applications involving Lyapunov theory as well as nonlinear control design for quadrotors. These applications would be fascinating to explore.

3. **The Wireless Transmitter Placement Problem** - While our application was interested in determining transmission rates for transmitters with given placements, it did

not explore the possibility of placing transmitters in the optimal locations. It would be interesting to explore how this changes the problem formulation.

REFERENCES

Ahmadi, A. A., & Majumdar, A. (2016). Some applications of polynomial optimization in operations research and real-time decision making. *Optimization Letters*, *10*(4), 709–729.

Anderson, D. R., Sweeney, D. J., Williams, T. A., Camm, J. D., & Martin, R. K. (2011). *An introduction to management science: Quantitative approaches to decision making, revised*. Cengage Learning.

ApS, M. (2015). Interface to the Mosek solver in Julia [Computer software manual]. Retrieved from `https://github.com/JuliaOpt/Mosek.jl`

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. Retrieved from `http://julialang.org/publications/julia-fresh-approach-BEKS.pdf` doi: 10.1137/141000671

Blekherman, P. P. A. . T. R. R., G. (2013). *Semidefinite optimization and convex algebraic geometry*. Philadelphia, Pennsylvania: SIAM.

Borchers, B. (1999). CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, *11*(1-4), 613-623.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Dunning, I., Huchette, J., & Lubin, M. (2015). JuMP: A modeling language for mathematical optimization. *arXiv:1508.01982 [math.OC]*.

Eckstein, J., & Yao, W. (2012). Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*.

Nishihara, R., Lessard, L., Recht, B., Packard, A., & Jordan, M. I. (2015). A general analysis of the convergence of admm. *arXiv preprint arXiv:1502.02009*.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization, second edition*. World Scientific.

O'Donoghue, B., Chu, E., Parikh, N., & Boyd, S. (2016, jun). Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding. *Journal of Optimization Theory and Applications*, *169*(3), 1042–1068. Retrieved from `http://stanford.edu/ boyd/papers/scs.html`