STOR-i Doctoral Training Centre, Lancaster University

STOR601 Report

Mathematical programming through cones

Author:
Jamie Fairbrother

Supervisor:
Prof. Adam Letchford

Abstract

In a mathematical programme one tries to minimise (or maximise) of a function given a set of constraints. Conic programming is the special case of this where one is optimising the function over a conic section. Conic programming covers a surprisingly wide class of problems and efficient solution algorithms are allowing one to solve previously intractably problems. This report outlines the basic theory of conic programmes before presenting some of its applications.

June 28, 2011

1 Introduction

In mathematical programming one attempts to minimise (or maximise) functions subject to certain constraints, that is, one aims to evaluate expressions of the form:

$$\min_{x\in\mathbb{R}^n}\{f_0(x) : f_i(x) \leq 0,\ i = 1, \ldots, m\} \tag{1}$$

When all the functions in the above expression are linear, this is called a linear programme, and one of the great breakthroughs of mathematical programming came when in 1947 Dantzig published his *simplex algorithm* which efficiently solves linear programmes in practice . A convex programme is one in which the functions in (1) are convex (see Section 2). These programmes have desirable properties which make the problem more tractable and naturally arise. However, for a long time no general solution methods existed even for this relatively restricted case.

In 1984 Karmakar published his seminal paper on a polynomial-time method for linear programmes [Kar84] which was efficient in practice as well as in theory, and was even claimed to be faster than the long-enduring simplex algorithm. This advance led to a flurry of research into interior point methods which culminated in work by Yurii Nesterov [NN93] who pioneered general solution algorithms for convex programmes.

Of the methods conceived, the *potential reduction method* was valuable because of its ability to estimate accuracy of the solutions it produces. The preliminary step in this approach is to reformulate the convex programme into a *conic programme*, that is a programme in which we optimise a linear function over a conic section. The essence of the approach is to use penalty methods and conic duality to converge to the solution.

Although not a panacea, this conic approach yields very efficient results for special types of cone [NT94] which cover a diverse range of problems from linear programming to semi definite programming. Conic programming has an elegant theory and unifies many branches of mathematical programming; as a consequence, it has steadily been gaining in prevalence.

In this introduction we aim to present conic programmes, derive their properties and try to demonstrate their wide applicability. We will not present or discuss in any detail interior point methods for which we refer the reader to [NN93, Ch.4]. In Sections 2 and 3 we lay out the relevant theory and formulate conic programmes; in Section 4 we discuss duality in conic programmes; Sections 5 and 6 present the important branches of conic programming of Second-order cone programming and Semi definite programming as well as their applications.

2 Convex Programmes

Convexity is a universal concept in mathematics and understanding the relationship between convex programmes and conic programming is key to appreciating the universality of the latter. We commence with the definitions of convex sets and functions.

Definition 2.1. A set $X \subset \mathbb{R}^n$ is *convex* if for all $x, y \in X$ and $0 \leq \lambda \leq 1$ we have $(1-\lambda)x+\lambda y \in X$.

Although the definition is not entirely lucid, the concept of convexity is natural and very intuitive; a set is convex if for any two points in the set the line segment connecting them is also in the set. See Figure 1 for examples of convex and non-convex sets.

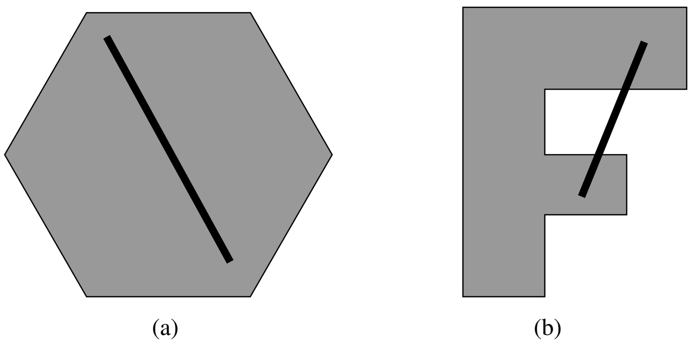

Figure 1: Examples of a convex region (a) and non-convex region (b) in $\mathbb{R}^2$

Definition 2.2. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for all $x,\ y \in \mathbb{R}^n$ and $0 \leq t \leq 1$ we have $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$.

Visually, a function is convex if its *epigraph* is convex, that is, the region above its graph is convex. Convex functions have useful properties which make them more tractable, for example they have at most one minimum. Figure 2 gives an example of a convex function.

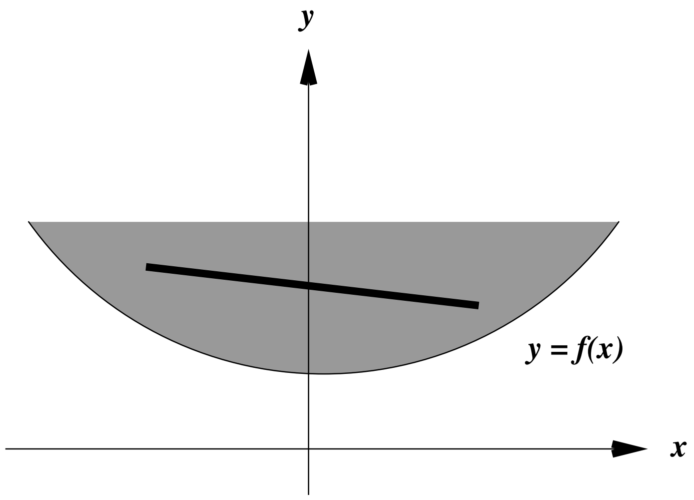

Figure 2: Example of a convex function.

Example 2.3. Quadratic Functions
A quadratic function is a function $f : \mathbb{R}^n \to \mathbb{R}$ of the form $f(x) = x^T Ax + b^T x + c$ where A

is a symmetric matrix. This is convex if A is *positive semidefinite*, that is if for all $x \in \mathbb{R}^n$ we have $x^T A x \geq 0$.

Proof. Let $f(x) = x^T A x + b^T x + c$ where A is positive semidefinite matrix. Fix $x, y \in \mathbb{R}^n$ and $0 \leq t \leq 1$. By expanding and simplifying we find that:

$$\begin{aligned} tf(x) + (1-t)f(y) - f(tx + (1-t)y) &= t(1-t)\left[x^T A x - 2x^T A y + y^T A y\right] \\ &= t(1-t)\left[(x-y)^T A (x-y)\right] \\ &\geq 0 \qquad \text{since } A \text{ is positive semidefinite.} \end{aligned}$$

Thus, f is convex. □

The feasible region of the mathematical programme in (1) is $\{x \in \mathbb{R}^n : f_i(x) \leq 0 for i = 1 \ldots n\}$, that is the set of points in $\mathbb{R}^n$ which obey all the constraints in a programme. A *convex programme* is one in which the objective function and feasible region are convex. A sufficient condition for the feasible region being convex is for all the constraint functions to be convex. Convex programmes arise very naturally, for example, since all linear functions are convex, linear programmes are convex.

Example 2.4. Portfolio Selection
Suppose we have an amount of money M and n stocks in which we wish to invest. If $r \in \mathbb{R}^n$ is the vector of expected returns and $Q \in \mathbb{R}^{n \times n}$ the covariance. Let $x_i \in \mathbb{R}$ be the amount we invest in stock i. If we wish to minimise the risk of an investment while ensuring a expected return is at least some amount R then Markowitz [Mar52] suggested solving the following optimisation problem:

$$\begin{gathered} \min_{x \in \mathbb{R}^n} x^T Q x \\ r^T x \geq R \\ \sum_{i=1}^{n} x_i \leq M \\ x \geq 0 \end{gathered}$$

Notice that since a covariance matrix is positive semidefinite that this problem is a convex programme. A problem with a quadratic objective and linear constraints is called a *quadratic programme.* □

Before moving onto conic programmes in the next section, we give convex programmes a reformulation which will prove advantageous:

Suppose the following mathematical programme is convex:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f_0(x) \\ \text{subject to} \qquad &f_i(x) \leq 0 \qquad \text{for } i = 1, \ldots, m \end{aligned}$$

Now, by introducing the auxiliary variable $t \in \mathbb{R}$, we can rewrite this programme as:

$$\begin{aligned} &\min_{(t,x)\in\mathbb{R}^{n+1}} t \\ \text{subject to} \quad & f_0(x) \leq t \\ & f_i(x) \leq 0 \qquad \text{for } i = 1, \ldots, m \end{aligned}$$

Notice that the function $F(x,t) = f_0(x) - t$ is convex and so this reformulation is still a convex programme. Thus, any convex programme can be written in the form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} c^T x \\ \text{subject to} \quad & x \in G \end{aligned}$$

where G is a convex feasible region. So, any convex programme can be considered to be the minimisation of a linear function over convex feasible region.

3 Conic Programmes

In this section we define conic programmes and relate them to convex programmes. We commence with the definition of a cone.

Definition 3.1. A subset $\mathcal{C} \subseteq \mathbb{R}^n$ is a *cone* if given $x \in \mathcal{C}$ for all $\lambda \geq 0$ we have $\lambda x \in \mathcal{C}$.

This definition is somewhat open for our purposes and, as in Figure 3, may not even conform to what our intuition tells us is a cone. The next definition provides a more useful and visually appealing formulation.

Definition 3.2. A cone $C \subseteq \mathbb{R}^n$ is *proper* if it is pointed ($C \cap -C = \{0\}$), full dimensional and convex.

Figure 3 gives an idea of what is and what is not a proper cone and from now on we will assume all cones are proper. As will be seen in the proceeding sections, this restriction will provide us with an elegant formulation and framework for conic programmes.

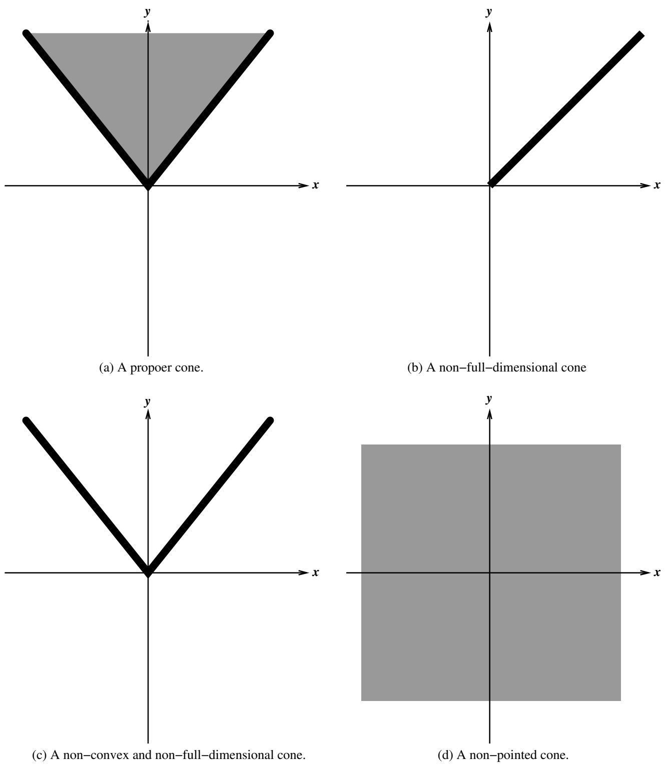
y
x
(a) A propoer cone.
y
x
(b) A non−full−dimensional cone
y
x
(c) A non−convex and non−full−dimensional cone.
y
x
(d) A non−pointed cone.

Figure 3: Examples of cones.

Example 3.3. The Lorentz cone is a cone in $\mathbb{R}^3$ and is what one would probably first think of when talking about cones. It is straight-forward to verify that it is a proper.

$$\mathcal{L}^3 = \{(x, y, z) \in \mathbb{R}^3 : z \geq \sqrt{x^2 + y^2}\}$$

The first useful property of a proper cone is that it defines a partial ordering on Euclidean space:

Definition 3.4. A proper cone $C \subset \mathbb{R}^n$ defines the following *partial ordering* on $\mathbb{R}^n$.

$$x \succeq y \iff x - y \succeq 0 \iff x - y \in C$$

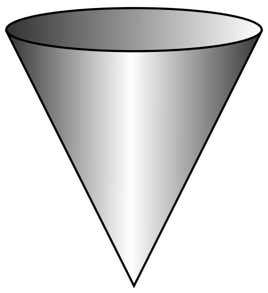

Figure 4: The Lorentz Cone in $\mathbb{R}^3$.

This property is not just a mathematical curiosity but allows, as can be seen in the next definition, an elegant formulation of conic programmes reminiscent of linear programmes.

Definition 3.5. A conic programme is a mathematical programme of the form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} c^T x \\ \text{subject to} \quad & Ax = b \\ & x \succeq_{\mathcal{C}} 0 \qquad (x \in \mathcal{C}) \end{aligned}$$

This, in effect, says that a conic programme is just the minimisation (or maximisation) of a linear function over a conic section (that is, the intersection of a cone and an affine subspace).

Now, as mentioned in the introduction, a powerful property of conic programmes is that any convex programme can be reformulated as one:

Proposition 3.6. Any convex programme can be written as a conic programme.

Proof. From the end of Section 2 we know that convex programmes can be written in the following form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \ c^T x \\ \text{subject to} \quad & x \in G \end{aligned}$$

where G is a convex feasible region. Now embed $\mathbb{R}^n$ into $\mathbb{R}^{n+1}$ as the hyperplane $\Lambda = \{1\}\times\mathbb{R}^n \subset \mathbb{R}^{n+1}$ and define a proper cone in $\mathbb{R}^{n+1}$ as follows:

$$K = cl(\{(t,x) \in \mathbb{R}^{n+1} : \frac{x}{t} \in G\})$$

where *cl* stands for the standard Euclidean closure. Notice that $x \in G \Leftrightarrow (1,x) \in K$. Now setting $d = \binom{0}{c}$, we can write the above convex programme as the following conic programme:

$$\begin{aligned} &\min_{(t,x)\in\mathbb{R}^{n+1}} d^T x \\ \text{subject to} \quad & x \in \Lambda \cap K \end{aligned}$$

$\square$

4 Duality

Duality in conic programmes is an extremely important property because, as will be seen, it allows us to estimate the accuracy of our solution. We start off with a description of duality in general mathematical programmes before deriving the dual of a conic programme.

Dual of a General Mathematical Programme Consider the general mathematical programme which we will call the *primal*:

$$\min_{x\in\mathbb{R}^n}\{f_0(x) : f_k(x) \leq 0,\ i = 1, \dots, m\} \tag{2}$$

Notice for any fixed $y \in \mathbb{R}^m$ the optimal solution for the following relaxed programme is a lower bound for the original problem (2).

$$\inf_{x\in\mathbb{R}^n} f_o(x) + \sum_{i=1}^{m} y_i f_i(x) \tag{3}$$

It is this observation that motivates the definition of the Lagrangian dual. Note that the expression $L(x, y) = f_0(x) + \sum_{i=1}^{m} y_i f_i(x)$ is called the *Lagrangian.* We also write $L^*(y) = \inf_{x\in\mathbb{R}^n} f_0(x) + \sum_{i=1}^{m} y_i f_i(x)$ for the lower bound given by $y \in \mathbb{R}^m$.

Definition 4.1. The *Lagrangian Dual* to the mathematical programme (2) is defined as follows:

$$\sup_{y\in\mathbb{R}^m} L^*(y) = \sup_{y\in\mathbb{R}^m} \{ \inf_{x\in\mathbb{R}^n} f_o(x) + \sum_{i=1}^{m} y_i f_i(x)\}$$

When the solution to this exists, it provides in a sense the best possible lower bound. The existence of this lower bound is called *weak duality.* In certain cases the solutions to the primal and dual problems are equal, and in this case we are said to have *strong duality.* Strong duality is guaranteed in certain conditions:

Theorem 4.2. Strong Duality via Slater's Condition
Suppose the primal problem (2) is convex and there exists a strictly feasible solution x, that is $f(x_1) < 0, \dots, f(x_m) < 0$, then strong duality holds.

Dual of a Conic Programme Before proceeding with the derivation of a dual conic programme, we introduce the concept of a *dual cone.*

Definition 4.3. Given a cone $C \subset \mathbb{R}^n$ we define its *dual cone* to be:

$$C^* := \{y \in \mathbb{R}^n : x \cdot\ y \geq 0\ \forall\ x \in C\}$$

Recall that the primal conic problem is of the form:

$$\begin{aligned} &\min c^T x \\ \text{subject to}\quad &Ax = b \\ &x \in \mathcal{C} \end{aligned}$$

Now,

$$L(x, y) = c^T x + y^T (Ax - b)$$

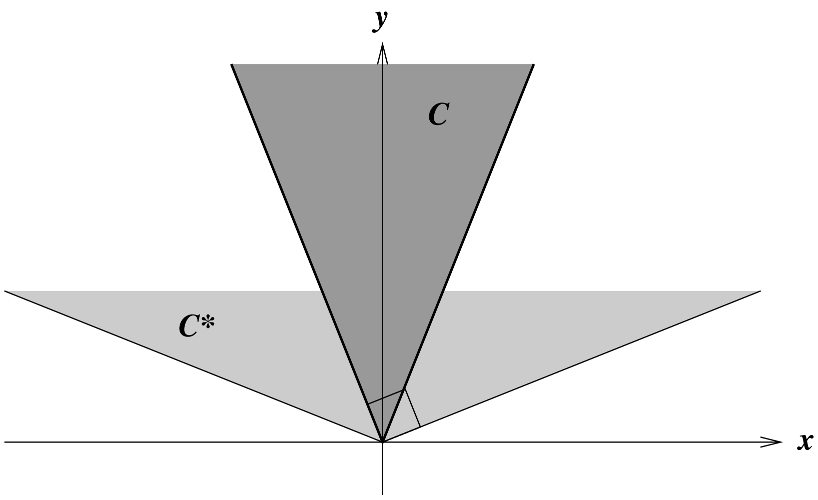

Figure 5: A dual cone.

This can be rewritten as:

$$L(x, y) = < b, y > + < c - y^T A, x >$$

So,

$$L^*(y) = \inf_{x \succeq_{\mathcal{C}} 0} < b, y > + < c - y^T A, x >$$

Notice here that for $L^*(y)$ to be a lower bound on the primal solution we need only take the infinum over the cone $\mathcal{C}$ rather than $\mathbb{R}^n$. Now if for some $x \in \mathcal{C}$ we have $< c - y^T A, x > < 0$ then $L^*(y) = -\infty$ which is not a very informative lower bound. To avoid this we must have $< c - y^T A, x > \geq 0$ for all $x \in \mathcal{C}$; in other words $c - y^T A \in \mathcal{C}^*$. In this case, the minimum of $< c - y^T A, x >$ is zero and the dual programme becomes:

$$\begin{array}{rl} & \sup_{y \in \mathbb{R}^m} < b, y > \\ \text{subject to} & c - y^T A \succeq_{\mathcal{C}^*} 0 \end{array}$$

Now, if we have two strictly feasible solutions x and y for the primal and dual problems respectively, then our we know that that the optimal solution for the primal x^* is within a range of $y - x$ of x. Recall that a conic programme is convex and so we can use Theorem 4.2. Thus, if we are given a problem which is strictly feasible then we get the same solution by solving the dual problem.

Symmetric Cones

Definition 4.4. A cone $\mathcal{C} \subset \mathbb{R}^n$ is *symmetric* if it is *self-dual* ($\mathcal{C} = \mathcal{C}*$) and homogeneous.

Efficient algorithms for symmetric conic programmes were shown to exist in [NT94, TN97]. (Homogeneity is a technical condition which requires that the group of linear automorphisms of the cone is transitive; in other words for every u, v in the cone there is a linear transformation A mapping the cone to itself such that $Au = v$.)

In the next two sections we discuss two major branches of conic programming second-order cone programming and semidefinite programming which consider problems over second-order cones and semidefinite cones. Both of these are examples of symmetric cones, and this fact goes some way of explaining why research in these areas has been so fruitful.

5 Second Order Cone Programming

The general $(n+1)$-dimensional second-order cone, or Lorentz cone, is defined as follows:

$$\mathcal{L}^{n+1} = \{x \in \mathbb{R}^{n+1} : x_o \geq \sqrt{x_1 + \ldots + x_n}\}$$

A *second-order cone programme* (SOCP) is a conic programme over the direct product of a finite number of second order cones. By convention we define $\mathcal{L}^1 = \mathbb{R}_+$ and so we have $(\mathcal{L}^1)^n = \mathbb{R}^n_+$. Thus, any linear programme can be written as an SOCP. Notice that this cone is self-dual, that is $\mathcal{L}^{n\,*} = \mathcal{L}^n$. In fact it is also symmetric (see Section 4) which means that there exist efficient algorithms to solve them. We now present the dual of an SOCP. For notational convenience we assume that our SOCP is over one cone and not the direct product of several. Assume we have the SOCP:

$$\begin{gathered}\min_{x \in \mathcal{L}^{n+1}} < c, x > \\ Ax = b\end{gathered}$$

From Section 4, we can write the dual of this programme as:

$$\begin{gathered}\sup_{y \in \mathbb{R}^m} b^T y \\ c^T - y^T A \in \mathcal{L}^n\end{gathered}$$

Through a little algebraic manipulation, one finds that the constraint $c^T - y^T A \in \mathcal{L}^{n+1}$ in this programme can be written in the form $\|Cy + d\| \leq \alpha^T y + \beta$. Thus, the dual becomes:

$$\begin{gathered}\sup_{y \in \mathbb{R}^m} b^T y \\ \|Cy + d\| \leq \alpha^T y + \beta\end{gathered}$$

In general, the dual to a primal SOCP is of the form:

$$\begin{aligned}&\sup_{y \in \mathbb{R}^m} b^T y \\ &\|C^i y + d^i\| \leq \alpha^{i\,T} y + \beta^i, \quad \text{for } i = 1, \ldots, p\end{aligned}$$

Recall that strong duality (which we have in practice in conic programmes) means that the problem of solving the primal problem is equivalent to solving the dual. Because of this, and the fact that in applications it is very natural to model by the dual, many surveys (for example [VMBL98]) consider the above expression to be the standard form of an SOCP.

Many types of problems can be cast as an SOCP including convex (quadratically constrained) quadratic programmes, norm minimisation problems and problems with hyperbolic constraints. SOCP therefore has a diverse array of applications from portfolio optimisation (see Example 2.4) to engineering and robotics. See [VMBL98] and [AG03] for thorough surveys.

We conclude this section with two examples:

Example 5.1. Minimising sums of norms
Consider the problem in which we are given a set of points $\{x^1, \ldots, x^m\} \subset \mathbb{R}^n$ and we want to find the point $y \in \mathbb{R}^n$ for which the sum of distance from y to each x^i . This can be written as the following unconstrained mathematical programme:

$$\inf_{y \in \mathbb{R}^n} \sum_{i=1}^{m} \|y - x^i\|$$

By introducing a set of auxiliary variables $\{t_1, \ldots, t_m\}$ this can be cast as the dual of an SOCP:

$$\inf \sum_{i=1}^{m} t_i$$
$$\|y - x^i\| \leq t_i, \quad \text{for } i = 1, \ldots, m$$

□

Example 5.2. Robust Linear Programming [VMBL98, pp.204-205]
In mathematical programming it may occur that we do not have precise information about the parameters for the system we are modelling. For instance, we may only know them up they are contained in some region or that they follow some distribution. In this case, using point estimates may be inappropriate as the solution this would give may not be valid to the real system. Robust optimisation attempts to remedy this by including information we do have about parameters into the programme. For a survey of robust optimisation see [BtN98].

In this example we consider linear programming in a statistical framework, and show that this yields an SOCP. Suppose that we have parameters A_i which are independent Normal random vectors with mean μ_i and covariance matrix Σ_i, and that we require our variable $x \in \mathbb{R}^n$ to satisfy the constraints $A_i^T x \leq b_i$ with probability ν, in other words:

$$\mathbb{P}(A_i^T x \leq b_i) \geq \nu$$

Noting that $A_i^T x$ are normal random variables with mean $\mu_i^T x$ and variance $x^T \Sigma_i \, x = \|\Sigma_i^{\frac{1}{2}} x\|^2$ the above constraints become:

$$\Phi \left(\frac{b_i - \mu_i^T x}{\|\Sigma_i^{\frac{1}{2}} x\|} \right) \geq \nu$$

Now the standard Linear Programme becomes:

$$\min_{x \in \mathbb{R}^n} c^T x$$
$$\mu_i^T x + \Phi^{-1}(\nu) \|\Sigma_i^{\frac{1}{2}} x\| \leq b_i$$

which is the dual SOCP. □

6 Semidefinite Programming

An $n \times n$ real symmetric matrix A is said to be positive semidefinite if it satisfies one of the following equivalent properties:

- For all $x \in \mathbb{R}^n$, $x^T Ax \geq 0$.
- All Eigenvalues of A are positive.

Positive semidefinite matrices crop up in many different areas of mathematics; for example in statistics the covariance matrix of a random vector is positive semidefinite, and in multivariate calculus they are used to classify extrema via the Hessian. The set of $n \times n$ positive semidefinite matrices is denoted $\mathcal{S}^n_+$ and is in fact a proper cone.

Proposition 6.1. The set $\mathcal{S}^n_+$ is a proper cone considered as a subset of the $n \times n$ real symmetric matrices Sym_n.

Proof. To show that $\mathcal{S}^n_+$ is a proper cone one must show that it is a cone, and that it is pointed, convex and full-dimensional. We just show full-dimensionality as the other criteria are immediate from the above definitions of positive semidefiniteness. Denote by $e_{i,j}$ the $n \times n$ matrix with all zero entries expect (i, j), (j, i), (i, i), and (j, j) which are 1. Now the set $\{e_{i,j}\}_{1 \leq i,j \leq n}$ forms a basis over the vector space of $n \times n$ symmetric matrices. Now notice that for all $x \in \mathbb{R}^n$ we have:

$$\begin{aligned} x^T e_{i,j} x &= (0, \ldots, \underbrace{x_i + x_j}_{\text{i-th position}}, \ldots, \underbrace{x_i + x_j}_{\text{j-th position}}, \ldots, 0)\ x \\ &= x_i^2 + 2x_i x_j + x_j^2 \\ &= (x_i + x_j)^2 \geq 0 \end{aligned}$$

In addition,

$$x^T e_{i,i} x = x_i^2 \geq 0$$

Since all members of this basis of symmetric matrices are positive semidefinite, $\mathcal{S}^n_+$ is full-dimensional. □

Now, a Semidefinite programme (SDP) is just a conic programme over the direct product over a finite number of semidefinite cones. To keep our notation simple, all SDPs considered in this section shall be assumed to be over just one semidefinite cone. The following is the general form of SDP we consider:

$$\begin{aligned} &\min_{X \in \mathrm{Sym}_n} \ < C, X > \\ \text{subject to} \quad & \mathcal{A}X = b \\ & X \succeq_{\mathcal{S}^n_+} 0 \end{aligned}$$

where $<,>$ represents the regular dot product and $\mathcal{A} : \mathrm{Sym}_n \to \mathbb{R}^m$ is a linear mapping.

Any SOCP can in fact be reformulated as an SDP. To demonstrate this we must first prove two technical statements.

Lemma 6.2. Let $t \in \mathbb{R}$ and $y \in \mathbb{R}^m$ then $\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix} \succeq_{\mathcal{S}^{n+1}_+} 0 \Leftrightarrow \begin{pmatrix} t \\ y \end{pmatrix} \succeq_{\mathcal{L}^{n+1}} 0$.

Proof. We first compute the Eigenvalues of $A = \begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix}$:

$$\begin{aligned}
\chi_A(x) &= \begin{vmatrix} t-x & y_1 & \dots & y_n \\ y_1 & t-x & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ y_n & 0 & \dots & t-x \end{vmatrix} \\
&= \begin{vmatrix} t-x-\frac{\sum_{i=1}^n y_i^2}{t-x} & y_1 & \dots & y_n \\ 0 & t-x & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & t-x \end{vmatrix} \\
&= (t-x)^{n-1}((t-x)^2 - \sum y_i^2)
\end{aligned}$$

Thus, the eigenvalues of A are t and $t \pm \|y\|_2$. Now,

$$\begin{aligned}
(t,y) \in \mathcal{L}^{n+1} &\Leftrightarrow 0 \leq \|y\|_2 \leq t \\
&\Leftrightarrow t \pm \|y\|_2 \geq 0 \text{ and } t \geq 0 \\
&\Leftrightarrow \text{All eigenvalues of } \begin{pmatrix} t & y^T \\ y & tI_k \end{pmatrix} \text{ are positive (by above).} \\
&\Leftrightarrow \begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix} \in \mathcal{S}_+^{n+1}
\end{aligned}$$

□

Lemma 6.3. Any vector subspace W of a finite-dimensional vector space V can be written as the kernel of a linear mapping. That is, there exists a linear mapping $\mathcal{B} : V \to U$ between finite vector spaces such that:

$$\text{kern } \mathcal{B} := \{X \in V : \mathcal{B}X = 0\} = W$$

Proof. Let $W^\perp$ be the orthogonal complement to W in V. Since V is finite-dimensional so is $W^\perp$; let $F_1, \dots, F_k$ be a basis for $W^\perp$ and define $\mathcal{B}$ as follows:

$$\mathcal{B} : X \longmapsto (< F_1, X >, \dots, < F_k, X >)$$

We claim that $W = \text{kern } \mathcal{B} = \{X \in V : \mathcal{B}X = 0\}$.

Suppose $X \in W$. Now,

$$\begin{aligned}
\mathcal{B}(X) &= (< F_1, X >, \dots, < F_k, X >) \\
&= (0, \dots, 0) \qquad \text{since } F_i \in W^\perp
\end{aligned}$$

Therefore $X \in \text{kern } \mathcal{B}$ and hence $W \subset \text{kern } \mathcal{B}$.

Suppose $X \in \text{kern } \mathcal{B}$. Then $< F_i, X >= 0$ for all $1 \leq i \leq k$ and so $X \in (W^\perp)^\perp$. But since W is finite-dimensional we have $(W^\perp)^\perp = W$ and so $X \in W$. Hence, $\text{kern } \mathcal{B} \subset W$.

We conclude that $W = \text{kern } \mathcal{B}$ as required. □

Proposition 6.4. Any SOCP problem can be reformulated as an SDP problem.

Proof. Let $t \in \mathbb{R}$ and $y \in \mathbb{R}^n$ and suppose we have the following SOCP:

$$\begin{aligned} \min_{(t,y)\in\mathbb{R}^{n+1}} \quad & c^T \begin{pmatrix} t \\ y \end{pmatrix} \\ \text{subject to} \quad & A \begin{pmatrix} t \\ y \end{pmatrix} = b \\ & \begin{pmatrix} t \\ y \end{pmatrix} \in \mathcal{L}^{n+1} \end{aligned}$$

Recall that an SDP is a mathematical programme in which our variable is symmetric matrix. By forcing this matrix to take a specific form and using an appropriate objective and constraints we can reconstruct the above programme as an SDP which is mathematically equivalent. Our strategy is as follows:

1. Constrain our programme to matrices of the form $\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix}$.
2. Rewrite the objective function in terms of $\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix}$.
3. Rewrite the linear constraint $A \begin{pmatrix} t \\ y \end{pmatrix} = b$ in terms of $\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix}$.
4. Constrain programme to the semidefinite cone $\mathcal{S}_+^{n+1}$.

1.
Notice that matrices of the form $\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix}$ form a vector subspace W of Sym_{n+1}. By Proposition 6.3 there exists a linear mapping $\mathcal{B}$ such that $W = \{X \in \mathrm{Sym}_{n+1} : \mathcal{B}X = 0\}$. Thus, using the linear constraint $\mathcal{B}X = 0$ in our programme will force our symmetric matrices to be of the appropriate form.

2.
Suppose $c = (c_1, \ldots, c_{n+1})$ and set $C = \begin{pmatrix} c_1 & \cdots & c_{n+1} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} c \\ 0_{n\times(n+1)} \end{pmatrix}$. Then $< C, \begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix} >= c^T \begin{pmatrix} t \\ y \end{pmatrix}$.

3.
Suppose $A = \begin{pmatrix} A_1 \\ \vdots \\ A_m \end{pmatrix}$ and $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$, where, the A_i represent the rows of the matrix A.
Define a linear mapping $\mathcal{A} : \mathrm{Sym}_{n+1} \longrightarrow \mathbb{R}^m$ as follows:

$$X \longmapsto \left(< \begin{pmatrix} A_1 \\ 0_{n\times(n+1)} \end{pmatrix}, X >, \ldots, < \begin{pmatrix} A_m \\ 0_{n\times(n+1)} \end{pmatrix}, X > \right)$$

Notice that

$$\mathcal{A}\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix} = \begin{pmatrix} A_1\begin{pmatrix} t \\ y \end{pmatrix} \\ \vdots \\ A_m\begin{pmatrix} t \\ y \end{pmatrix} \end{pmatrix}$$

and so

$$\mathcal{A}\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix} = b \Longleftrightarrow A\begin{pmatrix} t \\ y \end{pmatrix} = b$$

.

4.
By Proposition 6.2 we have $\begin{pmatrix} t & y^T \\ y & tI_n \end{pmatrix} \in \mathcal{S}_+^{n+1} \Longleftrightarrow \begin{pmatrix} t \\ y \end{pmatrix} \in \mathcal{L}^{n+1}$ and so finally the original SOCP can be written as the following SDP:

$$\begin{aligned} \min_{X \in \mathrm{Sym}_{n+1}} \quad & < C, X > \\ \text{subject to} \quad & \mathcal{A}X = b \\ & \mathcal{B}X = 0 \\ & X \in \mathcal{S}_+^{n+1} \end{aligned}$$

□

Thus, SDP is a generalisation of SOCP. However, one should not solve an SOCP as SDP as solution algorithms for the former are a lot more efficient [NN93, p.223, p.236]. This is why SOCP has developed as separate branches of conic programming.

The dual of an SDP can be derived as easily as has been done in the previous sections. The general form of the dual of the above SDP is as follows:

$$\begin{aligned} & \sup_{y \in \mathbb{R}^m} b^T y \\ & F(y) \succeq_{\mathcal{S}_+^n} 0 \end{aligned}$$

where F is an affine function $F : \mathbb{R}^m \to \mathrm{Sym}_n$. As with second order cone programming, this dual is often considered to be the standard form of an SDP because of propensity for modelling problems. We illustrate this with an example.

Example 6.5. Eigenvalue Optimisation[Hel02] Eigenvalue optimisation is a field which has many applications (see [BFK93] and [Ove92] for surveys). One example of an Eigenvalue optimisation problem is that of minimising the maximum Eigenvalue over an affine mapping $\mathbb{R}^m \longrightarrow M_{n,n}$:

$$\min_{y \in \mathbb{R}^m} \lambda_{\max}(C - \mathcal{A}y) + b^T y$$

If we have matrix A and a diagonal matrix λI_n then the Eigenvalues of the matrix $\lambda I_n + A$ are the Eigenvalues of A plus λ. Thus, recalling the definition of a positive semidefinite matrix, the inequality $\lambda I_n \succeq_{S_+^n} A$ says that all Eigenvalues of A are less than λ. Now the optimisation problem above can be reformulated as the following SDP problem:

$$\min_{\lambda\in\mathbb{R},\ y\in\mathbb{R}^m} \lambda + b^T y \text{ such that } \lambda I_n \succeq_{S^n_+} C - \mathcal{A}y.$$

Problems in SDP arise in many areas including statistics, combinatorial optimisation, control theory and structural optimisation. See [Hel02], [VB96], [WM02] for more thorough surveys.

7 Concluding Remarks

In this report we have outlined the basic theory of conic programming, described the main branches of it and gave some simple examples and applications. Here we mention some aspects of conic programming we did not consider and refer the reader to appropriate references. Throughout the report, the linear constraint in our conic programmes involved only equalities. This was simply done to avoid introducing cumbersome notation; one can easily deal with linear inequalities by adding slack variables, for example see [Hel02]. As mentioned in Section 4, conic programmes over symmetric cones can be solved efficiently. Therefore, the question as to which convex programmes can be written as symmetric conic programmes naturally arises and this has been the focus of some research ([NN93, Section 6.2.3]). Finally, software implementing efficient solution algorithms for conic programmes is readily available, see [Mit10] for a survey on this.

References

[AG03] F Alizadeh and D Goldfarb. Second-order cone programming. *Mathematical Programming B*, 95:pp. 3–51, 2003.

[BFK93] R Brualdi, S Friedland, and K Klee. Eigenvalues in combinatorial optimization. *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, pages pp. 107–151, 1993.

[BtN98] A. Ben-tal and A Nemirovskii. Robust convex optimisation. *Mathematics of Operations Research*, 23:769–805, 1998.

[Hel02] C Helmberg. Semidefinite programming. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, 137(3):461–482, March 2002.

[Kar84] N Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[Mar52] H Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, March 1952.

[Mit10] H Mittelmann. The state-of-the-art in conic optimization software, 2010.

[NN93] Y Nesterov and A Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13 of *Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1993.

[NT94] Y Nesterov and M. J. Todd. Self-scaled cones and interior-point methods in nonlinear programming. In *Working Paper, CORE, Catholic University of Louvain, Louvain-la-Neuve*, 1994.

[Ove92] M Overton. Large-scale optimization of eigenvalues. *SIAM Journal of Optimization*, pages pp. 88–120, 1992.

[TN97] M. J. Todd and Y Nesterov. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1):1–42, February 1997.

[VB96] L Vandenberg and S Boyd. Semidefinite programming. *SIAM REVIEW*, 38(1):49–95, March 1996.

[VMBL98] L Vandenberg, Lobo MS, S Boyd, and H Lebret. Applications of second-order cone programming. *LINEAR ALGEBRA AND ITS APPLICATIONS*, 284(1-3):193–228, November 1998.

[WM02] H Wolkowicz and Anjos MF. Semidefinite programming for discrete optimization and matrix completion problems. *DISCRETE APPLIED MATHEMATICS*, 123(1-3):513–577, November 2002.