# 4 Conic programming

**Definition 4.1.** A cone $K \subseteq \mathbb{R}^n$ is called *proper* if it is closed, convex, pointed and has nonempty interior.

From Theorem 2.2 and Exercise 2.1 we know that the dual of a proper cone is also proper. We also saw in Lecture 3 that the positive semidefinite cone is a proper cone.

Let $K \subseteq \mathbb{R}^n$ be a proper cone. A *conic program* over $K$ is an optimisation problem of the form:

$$\begin{array}{ll} \text{minimise} & \langle c, x\rangle \\ \text{subject to} & Ax = b \\ & x \in K \end{array} \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The optimisation variable here is $x \in \mathbb{R}^n$. The *feasible set* is the set of $x \in \mathbb{R}^n$ that satisfy the constraints $x \in K$ and $Ax = b$. The feasible set is the intersection of the cone $K$ with an affine space $\{x \in \mathbb{R}^n : Ax = b\}$ and thus is a closed convex set as an intersection of closed convex sets.

## 4.1 Linear programming

A *linear program* is a conic program over the cone $K = \mathbb{R}^n_+$ (nonnegative orthant). The constraint $x \in \mathbb{R}^n_+$ means that $x_i \geq 0$ for $i = 1, \ldots, n$. We will often use the abbreviation $x \geq 0$ to denote that $x_i \geq 0$ for $i = 1, \ldots, n$ (here $x \in \mathbb{R}^n$). So a linear program is a problem of the form:

$$\begin{array}{ll} \text{minimise} & \langle c, x\rangle \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{2}$$

For example the following optimisation problem is a linear program:

$$\text{minimise} \quad 2x_1 + x_2 \quad \text{s.t.} \quad 3x_1 - x_2 = 1, \;\; x_1 \geq 0, \;\; x_2 \geq 0.$$

This optimisation problem is an instance of (1) where $K = \mathbb{R}^2_+$, the cost vector is $c = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, the matrix $A$ is $1 \times 2$ given by $A = \begin{bmatrix} 3 & -1 \end{bmatrix}$ and $b = 1$.

Despite their apparent simplicity, linear programs have applications in many areas of applied sciences, engineering and economics. What makes linear programming appealing is that there are efficient algorithms to solve such optimisation problems. Problems with thousands (even millions) of constraints can be easily solved on a personal computer using current algorithms.

*Note: Historically, linear programming appeared in 1940s, much earlier than conic programs. Conic programs were introduced in 1990s as a generalisation of linear programming and were shown to enjoy some of the nice theoretical (and sometimes computational) properties of linear programming. For more historical information, see the bibliography section of Chapter 4 in Boyd & Vandenberghe.*

**An example from signal processing** We now give a simple example of a linear program that has attracted a lot of attention in the signal processing community. Let $M \in \mathbb{R}^{m\times n}$ and $d \in \mathbb{R}^m$ with $m < n$. We are interested in finding a solution to $Mx = d$ that has the smallest $\ell_1$ norm. Recall that the $\ell_1$ norm of a vector is given by $\|x\|_1 = \sum_{i=1}^n |x_i|$. In other words, we want to solve the optimisation problem

$$\underset{x\in\mathbb{R}^n}{\text{minimise}} \quad \|x\|_1 \quad \text{s.t.} \quad Mx = d. \tag{3}$$

Problem (3), as written, is *not* a linear program since the cost function is not linear. We will see however that by adequately introducing new variables we can express it as a linear program. We first claim that (3) is "equivalent" to the following problem:

$$\underset{x,y\in\mathbb{R}^n}{\text{minimise}} \quad \sum_{i=1}^n y_i \quad \text{s.t. } Mx = d, y + x \geq 0, y - x \geq 0. \tag{4}$$

What we will show is that any solution to (3) can be converted to a solution of (4) and vice-versa.

**Claim 1.** *If $x \in \mathbb{R}^n$ satisfies $Mx = d$ then there is $y \in \mathbb{R}^n$ that satisfies the constraints of (4) and such that $\sum_{i=1}^n y_i \leq \|x\|_1$. Conversely if $x, y \in \mathbb{R}^n$ satisfy the constraints of (4) then $\|x\|_1 \leq \sum_{i=1}^n y_i$. As a consequence the optimal values of (3) and (4) are equal.*

*Proof.* For the first direction take $y_i = |x_i|$ and note that $y_i + x_i \geq 0$ and $y_i - x_i \geq 0$ and $\sum_{i=1}^n y_i = \|x\|_1$. For the other direction note that if $x, y \in \mathbb{R}^n$ satisfy the constraints of (4) then $|x_i| = \max(x_i, -x_i) \leq y_i$ and so $\|x\|_1 \leq \sum_{i=1}^n y_i$. □

Problem (4) is now much closer to being a linear program in the form (2), however it is not yet exactly in the form (2). We now show how to put it exactly in the form (2). If we define $u = y + x$ and $v = y - x$ then problem (4) can be rewritten as

$$\underset{u,v,x,y\in\mathbb{R}^n}{\text{minimise}} \quad \sum_{i=1}^n y_i \quad \text{s.t.} \quad \begin{cases} Mx = d \\ u = y + x \\ v = y - x \\ u \geq 0,\ v \geq 0 \end{cases} \tag{5}$$

Problem (5) is almost of the form (2) except for a small difference: in (2) the variables are all constrained to be nonnegative, whereas in (5) only the variables $u, v$ are nonnegative (and $x$ and $y$ can take arbitrary signs). One way to address this is to note that any vector $x$ can be expressed as a difference of two nonnegative vectors $x = x_1 - x_2$ where $x_1, x_2 \geq 0$. Similarly we can do the same decomposition for $y$. Finally this yields the following linear program:

$$\underset{u,v,x_1,x_2,y_1,y_2\in\mathbb{R}^n}{\text{minimise}} \quad \sum_{i=1}^n (y_1)_i - (y_2)_i \quad \text{s.t.} \quad \begin{cases} M(x_1 - x_2) = d \\ u = y_1 - y_2 + x_1 - x_2 \\ v = y_1 - y_2 - x_1 + x_2 \\ u, v, x_1, x_2, y_1, y_2 \geq 0. \end{cases} \tag{6}$$

Problem (6) is now in the form (2) with an appropriate choice of matrix $A$ and vectors $b$ and $c$,

namely:

$$A = \begin{bmatrix} 0_{m\times n} & 0_{m\times n} & M & -M & 0_{m\times n} & 0_{m\times n} \\ I_{n\times n} & 0_{n\times n} & -I_{n\times n} & I_{n\times n} & -I_{n\times n} & I_{n\times n} \\ 0_{n\times n} & I_{n\times n} & I_{n\times n} & -I_{n\times n} & I_{n\times n} & -I_{n\times n} \end{bmatrix} \in \mathbb{R}^{(m+2n)\times 6n}$$

$$b = \begin{bmatrix} d \\ 0_{n\times 1} \\ 0_{n\times 1} \end{bmatrix} \in \mathbb{R}^{m+2n} \qquad c = \begin{bmatrix} 0_{n\times 1} \\ 0_{n\times 1} \\ 0_{n\times 1} \\ 0_{n\times 1} \\ \mathbf{1}_{n\times 1} \\ -\mathbf{1}_{n\times 1} \end{bmatrix} \in \mathbb{R}^{6n}$$

Note that this linear program has $6n$ variables $(u, v, x_1, x_2, y_1, y_2)$ and $m + 2n$ linear constraints. The notation $\mathbf{1}$ indicates a vector whose components are all ones, and $I_{n\times n}$ denotes the $n \times n$ identity matrix.

**Remark 1.** *As was mentioned during the lecture (thanks to the person who pointed this out), one can obtain a smaller linear program from (5) directly by eliminating $x$ and $y$ since we have $x = (u - v)/2$ and $y = (u + v)/2$. The problem (5) is thus equivalent to*

$$\underset{u,v\in\mathbb{R}^n}{\textit{minimise}} \quad \frac{1}{2}\sum_{i=1}^{n}(u_i + v_i) \qquad \textit{s.t.} \qquad M(u - v) = 2d,\ u \geq 0,\ v \geq 0. \tag{7}$$

*This is a linear program in standard form (2) and has $2n$ variables and $m$ equality constraints.*

The conversion from a problem (4) to its LP standard form can be a bit tedious. From now on, we will not do this conversion anymore and it will be taken for granted that problem (4) for example is a linear program (the step of going from (3) to (4) however is less trivial and so cannot be taken for granted in general). It will be assumed that the reader can do the "mechanic" conversion from (4) to a standard linear programming form (2) if needed.

**Exercise 4.1.** *Let $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Show how to put the optimisation problem*

$$\textit{minimise} \quad \langle c, x\rangle \qquad \textit{s.t.} \qquad b - Ax \geq 0$$

*into standard linear programming form (2).*