Chapter V: Conic Programming

1 Conic Programming

In this section, we introduce an important class of convex optimization problems, which generalizes the class of linear programs in a natural fashion.

Definition 1 (Conic Programming)**.** Let K be a proper cone. A *conic program in standard form* (relative to the cone K) is an optimization problem of the form

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\textbf{minimize}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ s.t. \quad & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \succeq_K \mathbf{0}. \end{aligned} \tag{P$_{\text{Cone}}$}$$

- When the cone K is the nonnegative orthant $\mathbb{R}^n_+$, we say that Problem (P$_{\text{Cone}}$) is a *Linear Program* (LP).
- When the cone K is a direct product of Lorrentz cones, we say that (P$_{\text{Cone}}$) is a *Second Order Cone Program* (SOCP). Some authors also use the term *conic quadratic program* (CQP). For consistency, we will adopt the following notation for the Lorrentz cone:

$$\mathbb{L}^n_+ := \{\boldsymbol{x} \in \mathbb{R}^n : \sqrt{x_1^2 + \ldots + x_{n-1}^2} \leq x_n\}.$$

- When the cone K is the semidefinite cone $\mathbb{S}^n_+$, we say that (P$_{\text{Cone}}$) is a *Semidefinite Program* (SDP).

More generally, we call LP/SOCP/SDP any optimization problem that is "trivially equivalent" to a conic problem in the standard form (P$_{\text{Cone}}$). We define this more formally next.

Definition 2. Let $\boldsymbol{x} \in \mathbb{R}^n$ be a decision variable. We say that a constraint of the form

$$\|A\boldsymbol{x} - \boldsymbol{b}\| \leq \boldsymbol{c}^T\boldsymbol{x} + d,$$

is a *second order cone inequality*. Now, let $F : \mathbb{R}^n \to \mathbb{S}^m$ be an affine function. We say that a constraint of the form

$$F(\boldsymbol{x}) \succeq 0,$$

where the inequality $\succeq$ is relative to $\mathbb{S}^m_+$, is a *linear matrix inequality*.

Now, the general idea is that

- any optimization problem that contains only linear equalities and inequalities is an LP;
- if in addition it contains some *second order cone inequalities*, it is an SOCP;
- finally, an optimization problem that contains only linear (in)equalities and *linear matrix inequalities*, is an SDP.

Definition 3. Let $K_i \subset \mathcal{V}_i$ be a proper cone, where $\mathcal{V}_i$ is a vector space (of finite dimension), and let $F_i : \mathbb{R}^n \to \mathcal{V}_i$ be an affine mapping ($\forall i \in [q]$). Any optimization problem of the form

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\textbf{minimize}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ s.t. \quad & A\boldsymbol{x} = \boldsymbol{b}, \\ & F_i(\boldsymbol{x}) \succeq_{K_i} \boldsymbol{0}, \qquad (\forall i \in [q]). \end{aligned}$$

is equivalent to a conic program in standard form ($\mathrm{P_{Cone}}$), and is hence called a *conic program.*

Note that the above definition is written with abstract vector spaces $\mathcal{V}_i$ (rather than, e.g., $\mathbb{R}^{n_i}$). This allows to handle the case where $F_i(\boldsymbol{x})$ is a matrix (for semidefinite programming). We will now show in several examples how the reduction to the standard form works.

Example:
The following optimization problem is an LP:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\textbf{minimize}} \quad & \boldsymbol{c}_0^T \boldsymbol{x} \\ s.t. \quad & F\boldsymbol{x} = \boldsymbol{f}, \\ & H\boldsymbol{x} \geq \boldsymbol{h}. \end{aligned}$$

To convert it to the standard form ($\mathrm{P_{Cone}}$), we proceed in two steps. Denote by m_1 and m_2 the dimensions of $\boldsymbol{f}$ and $\boldsymbol{h}$, respectively. First, we introduce a slack variable $\boldsymbol{z} = H\boldsymbol{x} - \boldsymbol{h} \in \mathbb{R}^{m_2}$. This yields the equivalent formulation:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n,\ \boldsymbol{z}\in\mathbb{R}^{m_2}}{\textbf{minimize}} \quad & \boldsymbol{c}_0^T \boldsymbol{x} \\ s.t. \quad & F\boldsymbol{x} = \boldsymbol{f}, \\ & H\boldsymbol{x} - \boldsymbol{z} = \boldsymbol{h}. \\ & \boldsymbol{z} \geq \boldsymbol{0}. \end{aligned}$$

Now, it remains to deal with the fact that the variable $\boldsymbol{x}$ is *unconstrained*, while the standard form ($\mathrm{P_{Cone}}$) only allows for variables in the cone K. So, the idea is to replace $\boldsymbol{x}$ by the difference between two vectors of nonnegative variables: #1

$$\boldsymbol{x} \in \mathbb{R}^n \iff \exists \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n_+ : \boldsymbol{x} = \boldsymbol{x}_1 - \boldsymbol{x}_2.$$

We thus obtain the following problem:

$$\begin{aligned} \underset{\boldsymbol{x}_1,\boldsymbol{x}_2\in\mathbb{R}^n,\ \boldsymbol{z}\in\mathbb{R}^{m_2}}{\textbf{minimize}} \quad & \boldsymbol{c}_0^T (\boldsymbol{x}_1 - \boldsymbol{x}_2) \\ s.t. \quad & F(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \boldsymbol{f}, \\ & H(\boldsymbol{x}_1 - \boldsymbol{x}_2) - \boldsymbol{z} = \boldsymbol{h}. \\ & \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z} \geq \boldsymbol{0} \end{aligned}$$

Finally, we obtain a problem of the standard form, by setting

$$\boldsymbol{c}^T = [\boldsymbol{c}_0^T, -\boldsymbol{c}_0^T, \boldsymbol{0}^T]^T \in \mathbb{R}^{2n+m_2}, \qquad A = \begin{bmatrix} F & -F & O \\ H & -H & -I \end{bmatrix} \in \mathbb{R}^{(m_1+m_2)\times(2n+m_2)}, \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{h} \end{bmatrix} \in \mathbb{R}^{m_1+m_2},$$

and the cone K is the nonnegative orthant $\mathbb{R}^{2n+m_2}_+$.

Example:
The following problem is an SOCP:

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\textbf{minimize}} \quad & \boldsymbol{c}_0^T \boldsymbol{x} & (1)\\ s.t. \quad & F\boldsymbol{x} = \boldsymbol{f}, \\ & H\boldsymbol{x} \geq \boldsymbol{g}, \\ & \|A_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^T\boldsymbol{x} + d_i, \quad (\forall i \in [m]). \end{aligned}$$

The fact that the variable $\boldsymbol{x}$ is unconstrained can be handled exactly as in the previous example. Now, we need to rewrite the linear inequalities and second order cone inequalities as in the standard form ($\mathrm{P}_{\mathrm{Cone}}$). If the rows of H are $\boldsymbol{h}_1^T, \ldots, \boldsymbol{h}_p^T$, then we introduce the slack variables

$$\boldsymbol{y}_i = \begin{bmatrix} 0 \\ \boldsymbol{h}_i^T\boldsymbol{x} - g_i \end{bmatrix} \in \mathbb{R}^2,$$

#2

and for all $i \in [m]$, we introduce the slack variable

$$\boldsymbol{z}_i = \begin{bmatrix} A_i\boldsymbol{x} + \boldsymbol{b}_i \\ \boldsymbol{c}_i^T\boldsymbol{x} + d_i \end{bmatrix} \in \mathbb{R}^{n_i+1},$$

where n_i is the dimension of $\boldsymbol{b}_i$. Now, it is easy to see that $\boldsymbol{h}_i^T\boldsymbol{x} \geq g_i \iff \boldsymbol{y}_i \in \mathbb{L}_+^2$, and

$$\|A_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^T\boldsymbol{x} + d_i \iff \boldsymbol{z}_i \in \mathbb{L}_+^{n_i+1}.$$

The inequalities and second order cone inequalities of Problem (1) can hence be reformulated as a set of equalities, and the generalized conic inequality

$$(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_m) \succeq_{K'} \boldsymbol{0}, \quad \text{where} \quad K' := (\mathbb{L}_+^2)^p \times \mathbb{L}_+^{n_1+1} \times \cdots \times \mathbb{L}_+^{n_m+1}$$

For the reduction of SDPs to the standard form ($\mathrm{P}_{\mathrm{Cone}}$), we need will need the following lemma:

Lemma 1. *The block diagonal matrix* $M = \begin{pmatrix} A_1 & & \\ & \ddots & \\ & & A_n \end{pmatrix}$ *is positive semidefinite if and only if each diagonal block is positive semidefinite:*

$$M \succeq 0 \iff \forall i \in [n],\ A_i \succeq 0.$$

Proof. The direct implication follows from the fact that the principal submatrices of a positive semidefinite matrix are positive semidefinite.

For the reverse implication, observe that

$$[\boldsymbol{u}_1^T, \ldots, \boldsymbol{u}_n^T]M \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_n \end{bmatrix} = \sum_{i=1}^n \boldsymbol{u}_i^T A_i \boldsymbol{u}_i.$$

When the blocks A_i are positive semidefinite, the above sum has only nonnegative terms, so it is ≥ 0. Since the vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ were arbitrary, this proves $M \succeq 0$. □

Example:
The following problem is an SDP:

$$\begin{aligned}
\underset{\boldsymbol{x}\in\mathbb{R}^n,\ X\in\mathbb{S}^n}{\textbf{minimize}} \quad & \langle C, X\rangle + \boldsymbol{c}_0^T\boldsymbol{x} && \text{(2a)}\\
s.t. \quad & \langle F_i, X\rangle + \boldsymbol{f}_i^T\boldsymbol{x} = g_i, \quad (\forall i\in[p]) && \text{(2b)}\\
& \langle H_i, X\rangle + \boldsymbol{h}_i^T\boldsymbol{x} \geq \ell_i, \quad (\forall i\in[q]) && \text{(2c)}\\
& BXB^T + DX + X^TD \succeq R_1 && \text{(2d)}\\
& \sum_{i=1}^m x_i Q_i \succeq R_2. && \text{(2e)}
\end{aligned}$$

#3

The fact that the variables $\boldsymbol{x}$ and X are unconstrained can be handled as in Example #1. We assume that $R_1\in\mathbb{S}^{r_1}$ and $R_2\in\mathbb{S}^{r_2}$. We define the slack matrix

$$Z = \begin{pmatrix} \mathrm{Diag}(\boldsymbol{u}) & O & O\\ O & BXB^T + DX + X^TD - R_1 & O \\ O & O & \sum_{i=1}^m x_iQ_i - R_2\end{pmatrix} \in \mathbb{S}^{q+r_1+r_2},$$

where $u_i = \langle H_i, X\rangle + \boldsymbol{h}_i^T\boldsymbol{x} - \ell_i$ $(\forall i\in[q])$. Now, by Lemma 1, the constraints (2c)-(2d)-(2e) are equivalent to $Z\succeq 0$, and we have only used linear equalities to introduce the slack variable Z.

2 What can be expressed by Linear Programming ?

In this section, we always consider a variable $\boldsymbol{x}\in\mathbb{R}^n$ and a scalar variable $t\in\mathbb{R}$. We want to review some common nonlinear functions f such that the constraint

$$f(\boldsymbol{x}) \leq t$$

can be reformulated as a set of equivalent linear (in)equalities.

(a) Inequalities involving convex piecewise linear functions.

A convex piecewise linear function f can always be written as $f(\boldsymbol{x}) = \max_{i\in[m]} \boldsymbol{a}_i^T\boldsymbol{x} + b_i$ for some vectors $\boldsymbol{a}_i$'s and scalars b_i's. Then, the inequality $f(\boldsymbol{x})\leq t$ is equivalent to

$$\boldsymbol{a}_i^T\boldsymbol{x} + b_i \leq t \quad (\forall i\in[m]).$$

(b) Inequalities involving the ℓ_1-norm of a vector:

$$\|\boldsymbol{x}\|_1 \leq t \iff \exists \boldsymbol{u}\in\mathbb{R}^n : \begin{cases} -\boldsymbol{u}\leq\boldsymbol{x}\leq\boldsymbol{u};\\ \mathbf{1}^T\boldsymbol{u}\leq t.\end{cases}$$

(c) Inequalities involving the $\ell_\infty-$norm of a vector:

$$\|\boldsymbol{x}\|_\infty \leq t \iff (-t\leq x_i\leq t, \quad \forall i\in[n]).$$

(d) Sum of the k largest elements of a vector.

Denote by $s_k(\boldsymbol{x})$ the sum of the k largest elements of $\boldsymbol{x}$. The function $\boldsymbol{x}\mapsto s_k(\boldsymbol{x})$ is convex. Indeed, it can be rewritten as a pointwise maximum of linear functions:

$$s_k(\boldsymbol{x}) = \max_{1\leq i_1<i_2<\ldots<i_k\leq n} x_{i_1} + x_{i_2} + \ldots + x_{i_k}.$$

We could hence use the point (a) above to rewrite $s_k(\boldsymbol{x}) \leq t$ as a set of linear inequalities. However, this representation involves the maximum of $\binom{n}{k}$ functions. Below, we give a much more compact representation, thanks to the use of $n+1$ additional variables. Geometrically, we are in fact expressing the polytope $\{(\boldsymbol{x}, t) : s_k(\boldsymbol{x}) \leq t\} \subseteq \mathbb{R}^{n+1}$ as the projection of another polytope in $\mathbb{R}^{2(n+1)}$ over the $(\boldsymbol{x}, t)$-space:

$$s_k(\boldsymbol{x}) \leq t \iff \exists u \in \mathbb{R}, \exists \boldsymbol{v} \in \mathbb{R}^n : \begin{cases} ku + \mathbf{1}^T\boldsymbol{v} \leq t; \\ \boldsymbol{v} \geq \mathbf{0} \\ \boldsymbol{v} \geq \boldsymbol{x} - u\mathbf{1}. \end{cases}$$

Proof. We only prove the point (d), the other points are rather trivial. Denote by $x_{(1)} \geq x_{(2)} \geq \ldots \geq x_{(n)}$ the sorted elements of $\boldsymbol{x}$.

We first show that

$$s_k(\boldsymbol{x}) = \min_{u\in\mathbb{R}} \ ku + \sum_{i=1}^{n} \max(0, x_i - u). \tag{3}$$

Let u be any real number in the interval $[x_{(k+1)}, x_{(k)}]$. Then,

$$ku + \sum_{i=1}^{n} \max(0, x_i - u) = ku + \sum_{j=1}^{k} \big(x_{(j)} - u\big) = s_k(\boldsymbol{x}),$$

where the first equality comes from the fact that $j > k \implies x_{(j)} - u < 0$. This already shows that the optimal value $p^*(\boldsymbol{x})$ of the minimization problem in the right hand side of (3) satisfies $p^*(\boldsymbol{x}) \leq s_k(\boldsymbol{x})$.

On the other hand, for all $u \in \mathbb{R}$,

$$s_k(\boldsymbol{x}) = \sum_{j=1}^{k} x_{(j)} = \sum_{j=1}^{k} (x_{(j)} - u + u) = ku + \sum_{j=1}^{k} (x_{(j)} - u) \leq ku + \sum_{j=1}^{k} \max(0, x_{(j)} - u) \leq ku + \sum_{i=1}^{n} \max(0, x_i - u).$$

Therefore, $s_k(\boldsymbol{x}) \leq p^*(\boldsymbol{x})$. Finally, the representation of point (d) is obtained by introducing an auxiliary variable $v_i \geq \max(0, x_i - u)$ for all $i \in [n]$. □

Example:
We show how to reformulate the nonlinear optimization problem

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\textbf{minimize}} \quad & 3\,\|\boldsymbol{x}\|_1 + 7\,s_4(\boldsymbol{x}) \\ s.t. \quad & A\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

to an equivalent LP with $3n+1$ variables. We first use the epigraph representation

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n,\lambda_1,\lambda_2\in\mathbb{R}}{\textbf{minimize}} \quad & 3\lambda_1 + 7\lambda_2 \\ s.t. \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \|\boldsymbol{x}\|_1 \leq \lambda_1 \\ & s_4(\boldsymbol{x}) \leq \lambda_2. \end{aligned}$$

Now, we use the points (b) and (d) above, which yields

$$\begin{aligned} \underset{\substack{\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}\in\mathbb{R}^n \\ t,\lambda_1,\lambda_2\in\mathbb{R}}}{\textbf{minimize}} \quad & 3\lambda_1 + 7\lambda_2 && \text{(4a)} \\ s.t. \quad & A\boldsymbol{x} \leq \boldsymbol{b} && \text{(4b)} \\ & -\boldsymbol{u} \leq \boldsymbol{x} \leq \boldsymbol{u} && \text{(4c)} \\ & \mathbf{1}^T\boldsymbol{u} \leq \lambda_1 && \text{(4d)} \\ & \boldsymbol{v} \geq \mathbf{0} && \text{(4e)} \\ & \boldsymbol{v} \geq \boldsymbol{x} - t\mathbf{1} && \text{(4f)} \\ & 4t + \mathbf{1}^T\boldsymbol{v} \leq \lambda_2 && \text{(4g)} \end{aligned}$$

#4

Finally, it is possible to eliminate λ_1 and λ_2 from this formulation, by putting the LHS of constraints (4d) and (4g) directly in the objective function:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}\in\mathbb{R}^n,\ t\in\mathbb{R}}{\textbf{minimize}} \quad & 3\cdot(\mathbf{1}^T\boldsymbol{u}) + 7\cdot(4t + \mathbf{1}^T\boldsymbol{v}) \\ s.t. \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & -\boldsymbol{u} \leq \boldsymbol{x} \leq \boldsymbol{u} \\ & \boldsymbol{v} \geq \mathbf{0} \\ & \boldsymbol{v} \geq \boldsymbol{x} - t\mathbf{1}. \end{aligned}$$

3 What can be expressed by Second Order Cone Programming ?

As in the previous section, $\boldsymbol{x}$ always denote a vector of dimension n. We review some nonlinear constraints involving the variable $\boldsymbol{x} \in \mathbb{R}^n$, as well as some scalar variables $t, y, z \in \mathbb{R}$, which can be handled by SOCP.

(a) Squared norm less than product of nonnegative variables

Inequalities of the form $\begin{cases} \|\boldsymbol{x}\|^2 & \leq & y \cdot z; \\ y & \geq & 0; \\ z & \geq & 0. \end{cases}$

The above system is equivalent to the SOC inequality

$$\left\| \begin{bmatrix} 2\boldsymbol{x} \\ y - z \end{bmatrix} \right\| \leq y + z.$$

Proof. This follows from

$$\left\| \begin{bmatrix} 2\boldsymbol{x} \\ y - z \end{bmatrix} \right\| \leq y + z \iff \begin{cases} \left\| \begin{bmatrix} 2\boldsymbol{x} \\ y - z \end{bmatrix} \right\|^2 & \leq & (y+z)^2; \\ y + z & \geq & 0. \end{cases}$$

$$\iff \begin{cases} 4\|\boldsymbol{x}\|^2 & \leq & (y+z)^2 - (y-z)^2; \\ y + z & \geq & 0. \end{cases} \qquad \iff \begin{cases} 4\|\boldsymbol{x}\|^2 & \leq & 4\, y \cdot z; \\ y, z & \geq & 0. \end{cases}$$

□

(b) Convex quadratic inequalities $\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{a}^T \boldsymbol{x} + b \leq t$

By using the last point, we can reformulate *any* convex quadratic constraint as a SOC constraint. Indeed, the function $f : \boldsymbol{x} \mapsto \boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{a}^T \boldsymbol{x} + b$ is convex iff $Q \succeq 0$. It follows that we can compute a decomposition of the form $Q = H^T H$ (for example, by using a Cholesky decomposition). Then, we have $f(\boldsymbol{x}) = \|H\boldsymbol{x}\|^2 + \boldsymbol{a}^T \boldsymbol{x} + b$, and

$$f(\boldsymbol{x}) \leq t \iff \|H\boldsymbol{x}\|^2 \leq t - \boldsymbol{a}^T \boldsymbol{x} - b \iff \left\| \begin{bmatrix} 2H\boldsymbol{x} \\ t - \boldsymbol{a}^T \boldsymbol{x} - b - 1 \end{bmatrix} \right\| \leq t - \boldsymbol{a}^T \boldsymbol{x} - b + 1.$$

(c) Geometric and Harmonic means.

The geometric and harmonic means of a nonnegative vector are concave functions. We will show in the exercises that the inequalities

$$G(\boldsymbol{x}) = \prod_{i=1}^n x_i^{1/n} \geq t$$

and

$$H(\boldsymbol{x}) = \frac{n}{\sum_{i=1}^n \frac{1}{x_i}} \geq t$$

can be represented by an equivalent system of SOC inequalities. (The harmonic mean is defined by continuity over all $\mathbb{R}^n_+$, with $H(\boldsymbol{x}) = 0$ whenever some $x_i = 0$.)

(d) Rational powers

The inequality $y^p \leq t$, where $p \in \mathbb{Q}, p \geq 1$ can be represented by a system of SOC inequalities. Indeed, let $p = \frac{\alpha}{\beta}$, with $\alpha \geq \beta \in \mathbb{N}$. Then,

$$x^{\frac{\alpha}{\beta}} \leq t \iff x^\alpha \leq t^\beta \iff x \leq G([\underbrace{t, \ldots, t}_{\beta \text{ times}}, \underbrace{1, \ldots, 1}_{(\alpha-\beta) \text{ times}}]).$$

Note: The point (b) shows that every convex QCQP (*quadratically constrained quadratic program*), that is, an optimization problem in which the objective function and the constraints are convex quadratic functions, can be cast as a SOCP.

4 What can be expressed by Semidefinite Programming ?

Before we start to enumerate nonlinear functions that can be handled by SDP, we state a very important lemma, which is at the heart of most SDP representations. We will prove this result in the exercises.

Lemma 2 (Schur Complement)**.** *Let M be a symmetric matrix paritioned in blocks as $M = \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}$. The following holds:*

$$M \succ 0 \iff \begin{cases} B \succ 0 \\ A - CB^{-1}C^T \succ 0 \end{cases} \iff \begin{cases} A \succ 0 \\ B - C^T A^{-1} C \succ 0 \end{cases}$$

Moreover, if $B \succ 0$, then

$$M \succeq 0 \iff A - CB^{-1}C^T \succeq 0;$$

And similarly, if $A \succ 0$, then

$$M \succeq 0 \iff B - C^T A^{-1} C \succeq 0.$$

Note: When the matrix B is singular, there is an extended version of the lemma which characterizes the positive semidefiniteness of M in terms of the Schur complement $A - CB^\dagger C^T$, where $B^\dagger$ is the Moore-Penrose pseudo inverse of B; In this case, an additional technical condition is required:

$$\begin{aligned} M \succeq 0 &\iff B \succeq 0 \quad \text{and} \quad \mathrm{A - CB^\dagger C^T} \succeq 0 \quad \text{and} \quad \mathbf{Im}\, \mathrm{C^T} \subseteq \mathbf{Im}\, \mathrm{B}. \\ &\iff A \succeq 0 \quad \text{and} \quad \mathrm{B - C^T A^\dagger C} \succeq 0 \quad \text{and} \quad \mathbf{Im}\, \mathrm{C} \subseteq \mathbf{Im}\, \mathrm{A}. \end{aligned}$$

In what follows, $X \in \mathbb{S}^n$ is a matrix variable, $\boldsymbol{x} \in \mathbb{R}^m$ is a vector of variable, and $t \in \mathbb{R}$ is a scalar variable. We review some nonlinear functions such that $f(X) \leq t$ or $g(\boldsymbol{x}) \leq t$ can be represented by a LMI (linear matrix inequality).

(a) Second Order Cone Programming

The SOC inequality $\|\boldsymbol{x}\| \leq t$ can be cast as a LMI. Indeed,

$$\|\boldsymbol{x}\| \leq t \iff \begin{cases} \|\boldsymbol{x}\|^2 \leq t^2 \\ t \geq 0 \end{cases} \iff \begin{cases} \boldsymbol{x}^T (tI_n)^{-1} \boldsymbol{x} \leq t \\ t \geq 0 \end{cases} \iff \begin{bmatrix} t\, I_n & \boldsymbol{x} \\ \boldsymbol{x}^T & t \end{bmatrix} \succeq 0.$$

This shows that any SOCP can be cast as a SDP. This is often not a good idea in practice, but this shows that SDP is a *superclass* of SOCP !

(b) Let $\boldsymbol{c} \in \mathbb{R}^n$, and let $K : \mathbb{R}^m \to \mathbb{S}^n_{++}$ be an affine function. Then, the Schur complement lemma tells us that

$$\boldsymbol{c}^T K(\boldsymbol{x})^{-1} \boldsymbol{c} \leq t \iff \begin{bmatrix} K(\boldsymbol{x}) & \boldsymbol{c} \\ \boldsymbol{c}^T & t \end{bmatrix} \succeq 0.$$

(c) Largest eigenvalue of a matrix.

Let $M:\mathbb{R}^m\to\mathbb{S}^n$ be an affine function, that is, $M(\boldsymbol{x})=M_0+\sum_{i=1}^m x_iM_i$ for some $M_i\in\mathbb{S}^n$. Then,

$$\lambda_{\max}(M(\boldsymbol{x}))\leq t \iff M(\boldsymbol{x})\preceq t\,I_n.$$

By the way, this shows that $\lambda_{\max}$ is a convex function over $\mathbb{S}^n$.

(d) Smallest eigenvalue of a matrix.

Similarly, $\lambda_{\min}$ is a concave function over $\mathbb{S}^n$, and

$$\lambda_{\min}(M(\boldsymbol{x}))\geq t \iff M(\boldsymbol{x})\succeq t\,I_n.$$

(e) nth root of determinant.

The nth root of the determinant is a concave function over $\mathbb{S}^n_{++}$. We have the following representation with LMIs:

$$(\det X)^{\frac{1}{n}}\geq t \iff \exists L\in\mathbb{R}^{n\times n},\boldsymbol{u}\in\mathbb{R}^n: \begin{cases}\begin{bmatrix} X & L\\ L^T & \operatorname{Diag}\boldsymbol{u}\end{bmatrix}\succeq 0;\\ L \text{ is Lower triangular;}\\ \mathbf{diag}\, L=\boldsymbol{u};\\ G(\boldsymbol{u})\geq t,\end{cases}$$

where $G(\boldsymbol{u})$ is the geometric mean of the vector $\boldsymbol{u}$ (cf. Section 3, Point (c)).

Proof. (a) and (b) are already proved, and (d) is similar to (c). We only need to prove (c) and (e).

(c) The largest eigenvalue of a matrix satisfies the variational characterization

$$\lambda_{\max}(X)=\sup_{\boldsymbol{u}\in\mathbb{R}^n\setminus\{\boldsymbol{0}\}}\frac{\boldsymbol{u}^TX\boldsymbol{u}}{\boldsymbol{u}^T\boldsymbol{u}}.$$

Hence,

$$\begin{aligned}\lambda_{\max}(X)\leq t &\iff \forall\boldsymbol{u}\in\mathbb{R}^n\setminus\{\boldsymbol{0}\},\quad \frac{\boldsymbol{u}^TX\boldsymbol{u}}{\boldsymbol{u}^T\boldsymbol{u}}\leq t\\ &\iff \forall\boldsymbol{u}\in\mathbb{R}^n,\quad \boldsymbol{u}^TX\boldsymbol{u}\leq t\boldsymbol{u}^T\boldsymbol{u}\\ &\iff \forall\boldsymbol{u}\in\mathbb{R}^n,\quad \boldsymbol{u}^T(X-tI_n)\boldsymbol{u}\leq 0\\ &\iff (X-tI_n)\preceq 0.\end{aligned}$$

(e) $\implies$. Let $X\succ 0$, and let $X=JJ^T$ be a Cholesky decomposition of X. Let D be the diagonal matrix with diagonal elements u_i, where $u_i:=J_{ii}^2$ is the square of the ith diagonal element of J, and let $L=JD^{1/2}$, so $L_{ii}=J_{ii}^2=D_{ii}=u_i$. We have $X=LD^{-1}L^T$, so the Schur-complement lemma tells us that

$$\begin{bmatrix} X & L\\ L^T & D\end{bmatrix}\succeq 0.$$

By construction, L is lower triangular, and the diagonal elements of L are the same as the diagonal elements u_i of D. Finally, since the determinant of a triangular matrix is equal to the product of its diagonal elements, we have

$$\det X=\det L\det D^{-1}\det L=\left(\prod u_i\right)\left(\prod u_i\right)^{-1}\left(\prod u_i\right)=\left(\prod u_i\right).$$

Hence, $(\det X)^{\frac{1}{n}}\geq t\implies G(\boldsymbol{u})\geq t$.

$\impliedby$. We first prove the intermediate result $A\succeq B\succeq 0\implies \det A\geq\det B$. First note that the statement is trivial whenever B is singular. So we assume $B\succ 0$. If $M\succeq I$, then we know from point (d) that the eigenvalues of M are larger than 1, so $\det M\geq 1$. Applying this to the matrix $M=B^{-1/2}AB^{-1/2}$ yields $A\succeq B\implies B^{-1/2}AB^{-1/2}\succeq I\implies \det B^{-1}\det A\geq 1$, hence the desired result.

Now, if the system of constraints holds and $t>0$, then by the Schur complement lemma, we have $X\succeq L\operatorname{Diag}\boldsymbol{u}^{-1}L^T$. So, using the implication $A\succeq B\implies\det A\geq\det B$, we obtain $\det X\geq(\prod u_i)(\prod u_i)^{-1}(\prod u_i)=(\prod u_i)$, and $(\det X)^{\frac{1}{n}}\geq G(\boldsymbol{u})\geq t$. In the trivial case $t\leq 0$, the LMI tells us that $X\succeq 0$, hence $\det X\geq 0\geq t$. □

5 A very quick guide through the PICOS interface

The python interfaces PICOS and CVXPY allow the users to easily implement conic programming problems in python, and to solve them with state-of-the-art solvers.

We give a few details on the PICOS syntax below:

- To create a problem instance:

```
import picos as pic
P = pic.Problem()
```

- Variables can be added to the problem as follows:

```
x = P.add_variable(name, size=(1,1), type="continuous")
```

  Here, `<name>` is a string, `<size>` is an integer (for vectors) or a pair of integers (for matrices), and `<type>` can be used to indicate that the variable has a special type, for example `type="symmetric"` indicates that the variable is a symmetric matrix. Other interesting types are `"binary"` and `"integer"` for (mixed)-integer optimization, but this goes out of the scope of this lecture.

- The objective function should be specified by

```
P.set_objective(direction, objective)
```

  where `<direction>` is either `"max"` or `"min"`, and `<objective>` is an *affine expression* formed with the variables of the problem.

- Constraints are added with the syntax

```
P.add_constraint(cons)
```

  where `<cons>` is a *constraint* obtained by comparing two *expressions*, for example

```
P.add_constraint(expression1 <= expression2)
P.add_constraint(expression1 == expression2)
```

  for inequality and equality constraints, respectively.

- When the problem is fully implemented, we can solve it with

```
P.solve()
```

  This calls the most appropriate solver, solves the problem, and stores the optimal value of the variables in their `value` attribute. Optimal dual variables of the constraints are computed, too (cf. next chapter), and stored in the `dual` attribute of the constraints.

- To enter constraints, the following operators are useful:
  - The standard python operators for the usual operations are `+`, `-`, `*`, `/` and `**` for exponentiation.
  - The operators `<<` and `>>` are used to denote the conic order relative $\mathbb{S}_+^n$.
  - The operator `|` stands for the scalar product. For example, `1|x` is understood as the sum of all elements of `x`.
  - The operator `abs()` stands for the Euclidean (or Frobenius) norm of its argument.
  - The operator `&` can be used to concatenate matrix blocks, horizontally.
  - The operator `//` can be used to concatenate matrix blocks, vertically.
  - The operator `pic.diag` applied to a vector $\boldsymbol{u}$ returns the matrix $\mathrm{Diag}(\boldsymbol{u})$.
  - The operator `pic.diag_vect` applied to a matrix X returns the vector $\mathbf{diag}\,(X)$.
  - The property `.T` transposes a vector/matrix.
  - The operator `pic.sum` applied to a *list of affine expressions*, returns the sum of all expressions in the list.

- PICOS natively supports most of the nonlinear functions listed in this chapter. For example, the constraint $x^{2.7} <= t$ can be entered *as is* in PICOS, which automatically translates `x**2.7 <= t` as a set of equivalent SOC-constraints (cf. Section 3, point (d)). Also, the constraints $\|\boldsymbol{x}\|^2 \leq yz,\ y, z \geq 0$ can be entered in picos directly as

  ```
  P.add_constraint(abs(x)**2 <= y*z)
  ```

  For this special case, PICOS understands that we mean the implicit constraints $y \geq 0, z \geq 0$, and automatically reformulates the constraint as $\left\| \begin{bmatrix} 2\boldsymbol{x} \\ y - z \end{bmatrix} \right\| \leq y + z$. In other words, the above code fragment is equivalent to:

  ```
  P.add_constraint(abs( ((2*x) // (y-z)) ) <= y*z )
  ```

- Here are a few examples of nonlinear functions implemented in PICOS:
  - `pic.sum_k_largest(x,k) <= t` represents the constraint $s_k(\boldsymbol{x}) \leq t$;
  - `pic.norm(x,p) <= t` represents the constraint $\|\boldsymbol{x}\|_p \leq t$;
  - `pic.geomean(x) >= t` represents the constraint $G(\boldsymbol{x}) = \prod x_i^{1/n} \geq t$;
  - `pic.detrootn(X) >= t` represents the constraint $(\det X)^{1/n} \geq t$;
  - `pic.lambda_max(X) <= t` represents the constraint $\lambda_{\max}(X) \leq t$.

Example:
To illustrate the above rules, let us implement the following (dummy) problem in PICOS

$$\begin{aligned} \min_{X \in \mathbb{S}^n, \boldsymbol{x} \in \mathbb{R}^m, t \in \mathbb{R}} \quad & \langle A, X \rangle + \boldsymbol{b}^T \boldsymbol{x} + 3t \\ s.t. \quad & \begin{bmatrix} X & \boldsymbol{x} \\ \boldsymbol{x}^T & t \end{bmatrix} \succeq 0 \\ & \mathrm{Diag}(X) = \boldsymbol{x} \\ & \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{1} \\ & X_{3,3} + X_{1,2} + X_{2,1} + 2\,x_6 \leq x_8 \\ & \|C\boldsymbol{x} - \boldsymbol{d}\| \leq t \end{aligned}$$

We assume that the data $A \in \mathbb{S}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, $C \in \mathbb{R}^{r \times m}$ and $\boldsymbol{d} \in \mathbb{R}^r$ is already loaded in memory, and stored in objects `A,b,C,d` of the class `cvxopt.matrix` (the package `cvxopt` is a dependency of PICOS).

```
#Define the problem and the variables
P = pic.Problem()
X = P.add_variable('X',(n,n),'symmetric')
x = P.add_variable('x',m)
t = P.add_variable('t',1)

#add the constraints
P.add_constraint( ( (X & x)//(x.T & t) ) >> 0)
P.add_constraint( pic.diag_vect(X) == x )
P.add_constraint( x >= 0)                #(here, PICOS understands >= is elementwise)
P.add_constraint( x <= 1)
P.add_constraint( X[2,2] + X[0,1] + X[1,0] + 2*x[5] <= x[7] )  #(indices start from 0)
P.add_constraint( abs(C*x-d) <= t)

#set the objective function and solve the problem
P.set_objective('min',(A|X) + (b.T*x) + 3*t)
P.solve()
```

#5