## CONVEX OPTIMIZATION

An optimization problem can be stated in the so-called standard form as follows:

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) : \mathbf{R} \to \mathbf{R} \\ &\text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) \leq 0, \quad \boldsymbol{g} : \mathbf{R}^m \to \mathbf{R}^n \end{aligned} \tag{NLP}$$

representing the minimization of a function $f$ of $n$ variables under constraints specified by inequalities determined by functions $\boldsymbol{g} = [g_1, g_2, \ldots, g_m]^{\mathrm{T}}$. The functions $f$ and $g_i$ are, in general, nonlinear functions. Note that $\geq$ inequalities can be handled under this paradigm by multiplying each side by $-1$, and equalities by representing them as pairs of inequalities. The maximization of an objective function $f(\boldsymbol{x})$ can be achieved by minimizing $-f(\boldsymbol{x})$. The set $\mathscr{F} = \{\boldsymbol{x} | \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}\}$ that satisfies the constraints on the nonlinear optimization problem is known as the feasible set, or the feasible region. If $\mathscr{F}$ covers all of (a part of) $\mathbf{R}^n$, then the optimization is said to be *unconstrained* (*constrained*).

Note that the above standard-form formulation may not be directly applicable to real life design problems, where often, multiple conflicting objectives must be optimized. In such a case, multicriterion optimization techniques and Pareto optimality can be used to identify noninferior solutions (1). In practice, however, techniques to map the problem to the form in Eq. (NLP) are often used.

When the objective function is a convex function and the constraint set is a convex set (both terms will be formally defined later), the optimization problem is known as a convex programming problem. This problem has the remarkable property of unimodality, i.e., any local minimum of the problem is also a global minimum. Therefore, it does not require special methods to extract the solution out of local minima in a quest to find the global minimum. While the convex programming problem and its unimodality property have been known for a long time, it is only recently that efficient algo-

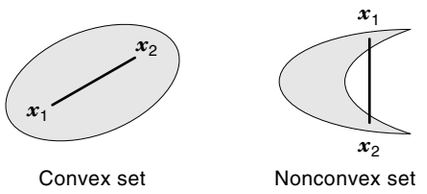


**Figure 1.** Convex and nonconvex sets.

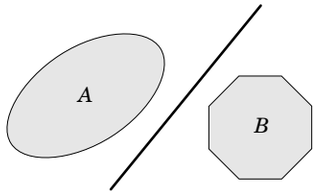


**Figure 3.** A separating hyperplane (line) in two dimensions between convex sets $A$ and $B$.

rithms for the solution of these problems have been proposed. The genesis of these algorithms can be traced back to the work of Karmarkar (2) that proposed a polynomial-time interior-point technique for linear programming, a special case of convex programming. Unlike the simplex method for linear programming, this technique was found to be naturally extensible from the problem of linear programming to general convex programming formulations. It was shown that this method belongs to the class of interior penalty methods proposed by Fiacco and McCormick (3) using barrier functions. The work of Renegar (4) showed that a special version of the method of centers for linear programming is polynomial. Gonzaga (5) showed similar results for the barrier function associated with a linear programming problem, with a proof of polynomial-time complexity. Nesterov and Nemiorvsky (6) introduced the concept of self-concordance, studying barrier methods in their context. Further improvements in the computational complexity using approximate solutions and rank-one updates were shown in the work of Vaidya (7). The work of Ye (8) used a potential function to obtain the same complexity as Renegar's work without following the central path too closely.

## DEFINITIONS OF CONVEXITY

### Convex Sets

***Definition.*** A set $C \in \mathbf{R}^n$ is said to be a *convex set* if, for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and every real number $\alpha$, $0 \leq \alpha \leq 1$, the point $\alpha\boldsymbol{x}_1 + (1 - \alpha)\boldsymbol{x}_2 \in C$.

This definition can be interpreted geometrically as stating that a set is convex if, given two points in the set, every point on the line segment joining the two points is also a member of the set. Examples of convex and nonconvex sets are shown in Fig. 1.

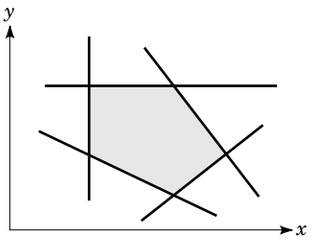


**Figure 2.** An example convex polytope in two dimensions as an intersection of five half spaces.

### Elementary Convex Sets

*Ellipsoids.* An ellipsoid $E(\boldsymbol{x}, \mathscr{B}, r) \in \mathbf{R}^n$ centered at point $\boldsymbol{x} \in \mathbf{R}^n$ is given by the equation

$$\{\boldsymbol{y} | (\boldsymbol{y} - \boldsymbol{x})^{\mathrm{T}} \mathscr{B} (\boldsymbol{y} - \boldsymbol{x}) \leq r^2\}$$

If $\mathscr{B}$ is a scalar multiple of the unit matrix, then the ellipsoid is called a *hypersphere.* The axes of the ellipsoid are given by the eigenvectors, and their lengths are related to the corresponding eigenvalues of $\mathscr{B}$.

*Hyperplanes.* A hyperplane in $n$ dimensions is given by the region

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = b, \qquad \boldsymbol{c} \in \mathbf{R}^n, \quad b \in \boldsymbol{R}$$

*Half spaces.* A half space in $n$ dimensions is defined by the region that satisfies the inequality

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq b, \qquad \boldsymbol{c} \in \mathbf{R}^n, \quad b \in \mathbf{R}$$

*Polyhedra.* A (convex) polyhedron is defined as an intersection of half spaces, and is given by the equation

$$\mathrm{P} = \{\boldsymbol{x} | A\boldsymbol{x} \leq \boldsymbol{b}), \qquad A \in \mathbf{R}^{m \times n}, \quad \boldsymbol{b} \in \mathbf{R}^m$$

corresponding to a set of $m$ inequalities $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \leq b_i$, $\boldsymbol{a}_i \in \mathbf{R}^n$. If the polyhedron has a finite volume, it is referred to as a *polytope.* An example of a polytope is shown in Fig. 2.

### Some Elementary Properties of Convex Sets

***Property.*** The intersection of convex sets is a convex set. The union of convex sets is not necessarily a convex set.

***Property (Separating hyperplane theorem).*** Given two nonintersecting convex sets $A$ and $B$, there exists a separating hyperplane $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{b}$ such that $A$ lies entirely within the half space $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \geq \boldsymbol{b}$ and $B$ lies entirely within the half space $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq \boldsymbol{b}$. This is pictorially illustrated in Fig. 3.

***Property (Supporting hyperplane theorem).*** Given a convex set $C$ and any point $\boldsymbol{x}$ on its boundary, there exists a supporting hyperplane $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{b}$ such that $C$ lies entirely within the half space $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq \boldsymbol{b}$. This is illustrated in Fig. 4.

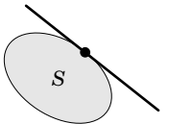


**Figure 4.** A supporting hyperplane (line) in two dimensions at the boundary point of a convex set $S$.

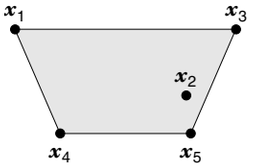


**Figure 5.** An example showing the convex hull of five points.

***Definition.*** The *convex hull* of $m$ points, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m \in \mathbf{R}^n$, denoted $\text{co}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$, is defined as the set of points $\boldsymbol{y} \in \mathbf{R}^n$ such that

$$\mathbf{y} = \sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i, \qquad \alpha_i \geq 0 \ \forall i, \quad \sum_{i=1}^{m} \alpha_i = 1$$

The convex hull is thus the smallest convex set that contains the $m$ points. An example of the convex hull of five points in the plane is shown by the shaded region in Fig. 5. If the set of points $\boldsymbol{x}_i$ is of finite cardinality (i.e., $m$ is finite), then the convex hull is a polytope. Hence, a polytope is also often described as the convex hull of its vertices.

### Convex Functions

***Definition.*** A function $f$ defined on a convex set $\Omega$ is said to be a *convex function* if, for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$ and every $\alpha$, $0 \leq \alpha \leq 1$,

$$f[\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2] \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

$f$ is said to be *strictly convex* if the above inequality is strict for $0 < \alpha < 1$.

Geometrically, a function is convex if the line joining two points on its graph is always above the graph. Examples of convex and nonconvex functions on $\mathbf{R}^n$ are shown in Fig. 6.

### Some Elementary Properties of Convex Functions

***Property.*** A function $f(\boldsymbol{x})$ is convex over the set $S$ if and only if

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) + [\nabla f(\boldsymbol{x}_0)]^{\mathrm{T}} (\boldsymbol{x} - \boldsymbol{x}_0) \qquad \forall \boldsymbol{x}, \boldsymbol{x}_0 \in S$$

where $\nabla f$ corresponds to the gradient of $f$ with respect to the vector $\boldsymbol{x}$. Strict convexity corresponds to the case where the inequality is strict.

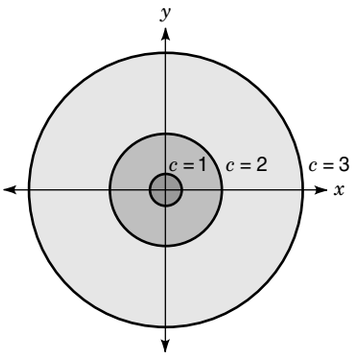


**Figure 7.** Level sets of $f(x,y) = x^2 + y^2$.

***Property.*** A function $f(\boldsymbol{x})$ is convex over the convex set $S$ if and only if

$$\nabla^2 f(\boldsymbol{x}_0) \geq 0 \qquad \forall \boldsymbol{x}_0 \in S$$

i.e., its Hessian matrix is positive semidefinite over $S$. For strict convexity, $\nabla^2 f(\boldsymbol{x}_0)$ must be positive definite.

***Property.*** If $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are two convex functions on the convex set $S$, then the functions $f + g$ and $\max(f, g)$ are convex over $S$.

***Definition.*** The *level set* of a function $f(\boldsymbol{x})$ is the set defined by $f(\boldsymbol{x}) \leq c$ where $c$ is a constant. An example of the level sets of $f(x, y) = x^2 + y^2$ is shown in Fig. 7. Observe that the level set for $f(x, y) \leq c_1$ is contained in the level set of $f(x, y) \leq c_2$ for $c_1 < c_2$.

***Property.*** If $f$ is a convex function in the space $\mathscr{S}$, then the level set of $f$ is a convex set in $\mathscr{S}$.

This is a very useful property, and many convex optimization algorithms depend on the fact that the constraints are defined by an intersection of level sets of convex functions.

***Definition.*** A function $g$ defined on a convex set $\Omega$ is said to be a *concave function* if the function $f = -g$ is convex. The function $g$ is *strictly concave* if $-g$ is strictly convex.

For a fuller treatment of convexity properties, the reader is referred to Ref. 9.

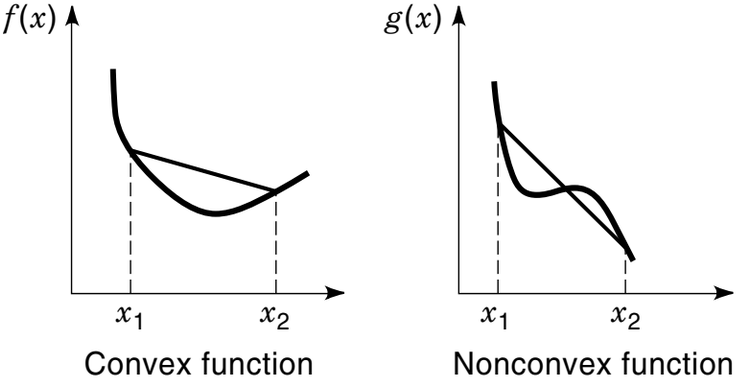


**Figure 6.** Convex and nonconvex functions.

## CONVEX OPTIMIZATION

### Convex Programming

***Definition.*** A *convex programming problem* is an optimization problem that can be stated as follows:

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{such that} \quad \boldsymbol{x} \in S \end{aligned} \tag{CP}$$

where $f$ is a convex function and $S$ is a convex set.

Such a problem has the property that any local minimum of $f$ over $S$ is a global minimum.

***Comment.*** The problem of maximizing a convex function over a convex set does not have the above property. However, it can be shown (10) that the maximum value for such a problem lies on the boundary of the convex set.

For a convex programming problem of the type (CP), we may state without loss of generality that the objective function is linear. To see this, note that the problem (CP) may equivalently be written as $\{\min t : f(\boldsymbol{x}) \leq t, \boldsymbol{g}(\boldsymbol{x}) \leq 0\}$

Linear programming is a special case of nonlinear optimization, and more specifically, a special type of convex programming problem where the objective and constraints are all linear functions. The problem is stated as

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{subject to} \quad A\boldsymbol{x} \leq \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0} \\ &\text{where} \quad A \in \mathbf{R}^{m \times n}, \quad \boldsymbol{b} \in \mathbf{R}^{m}, \quad \boldsymbol{c} \in \mathbf{R}^{n}, \quad \boldsymbol{x} \in \mathbf{R}^{n} \end{aligned} \tag{LP}$$

The feasible region for a linear program corresponds to a polyhedron in $\mathbf{R}^n$. It can be shown that an optimal solution to a linear program must necessarily occur at one of the vertices of the constraining polyhedron. The most commonly used technique for solution of linear programs, the simplex method (11), is based on this principle and operates by a systematic search of the vertices of the polyhedron. The computational complexity of this method can show exponential behavior for pathological cases, but for most practical problems it has been observed to grow linearly with the number of variables and sublinearly with the number of constraints. Algorithms with polynomial-time worst-case complexity do exist; these include Karmarkar's method (2) and the Shor–Khachiyan ellipsoidal method (12). The computational times of the latter, however, are often seen to be impractical.

In the remainder of this section, we will describe various methods used for convex programming and for mapping problems to convex programs.

### Path-Following Methods

This class of methods proceeds iteratively by solving a sequence of unconstrained problems that lead to the solution of the original problem. In each iteration, a technique based on barrier methods is used to find the optimal solution. If we denote the optimal solution in the $k$th iteration as $\boldsymbol{x}_k^*$, then the path $\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \ldots$ in $\mathbf{R}^n$ leads to the optimal solution, and hence techniques of this class are known as path-following methods.

**Barrier Methods.** The barrier technique of Fiacco and McCormick (3) is a general technique to solve any constrained nonlinear optimization problem by solving a sequence of unconstrained nonlinear optimization problems. This method may be used for the specific case of minimizing a convex function over a convex set $S$ described by an intersection of convex inequalities

$$S = \{\boldsymbol{x} | g_i(\boldsymbol{x}) \leq 0, i = 1, 2, \ldots, m\}$$

where each $g_i(\boldsymbol{x})$ is a convex function. The computation required of the method is dependent on the choice of the barrier function. In this connection, the logarithmic barrier function (abbreviated as the log barrier function) for the set of inequalities is defined as

$$\Phi(\boldsymbol{x}) = \begin{cases} -\sum_{i=1}^{n} \log[-g_i(\boldsymbol{x})] & \text{for} \quad \boldsymbol{x} \in S \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, the idea of the barrier is that any iterative gradient-based method that tries to minimize the barrier function will be forced to remain in the feasible region, due to the singularity at the boundary of the feasible region. It can be shown that $\Phi(\mathbf{x})$ is a convex function over $S$ and its value approaches infinity as $\boldsymbol{x}$ approaches the boundary of $S$. Intuitively, it can be seen that $\Phi(\boldsymbol{x})$ becomes smallest when $\boldsymbol{x}$ is, in some sense, farthest away from all of the boundaries of $S$. The value of $\boldsymbol{x}$ at which the function $\Phi(\boldsymbol{x})$ is minimized is called the *analytic center* of $S$.

***Example.*** For a linear programming problem of the type described in Eq. (LP), with constraint inequalities described by $a_i^{\mathrm{T}}\boldsymbol{x} \leq b_i$, the barrier function in the feasible region is given by

$$\Phi(\boldsymbol{x}) = -\sum_{i=1}^{n} \log(b_i - \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x})$$

The value of $b_i - \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}$ represents the slack in the $i$th inequality, i.e., the distance between the point $\boldsymbol{x}$ and the corresponding constraint hyperplane. The log barrier function, therefore, is a measure of the product of the distances from a point $\boldsymbol{x}$ to each hyperplane, as shown in Fig. 8(a). The value of $\Phi(\boldsymbol{x})$ is minimized when $\Pi_{i=1}^{n}(b_i - \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x})$ is maximized. Coarsely speaking, this occurs when the distance to each constraint hyperplane is sufficiently large.

As a cautionary note, we add that while the analytic center is a good estimate of the center in the case where all con-

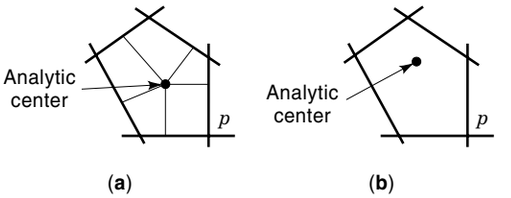


**Figure 8.** (a) Physical meaning of the barrier function for the feasible region of a linear program; (b) the effect of redundant constraints on the analytic center.

straints present an equal contribution to the boundary, the presence of redundant constraints can shift the analytic center. The effect on the analytic center of repeating the constraint $p$ five times is shown in Fig. 8(b).

We will now consider the convex programming problem specified as

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{such that} \quad g(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m \end{aligned}$$

where each $g_i(\boldsymbol{x})$ is a convex function. The traditional barrier method (3) used to solve this problem formulates the corresponding unconstrained optimization problem

$$\text{minimize} \quad B(\alpha) = \alpha f(\boldsymbol{x}) + \Phi(\boldsymbol{x})$$

and solves this problem for a sequence of increasing (constant) values of the parameter $\alpha$. When $\alpha$ is zero, the problem reduces to finding the center of the convex set constraining the problem. As $\alpha$ increases, the twin objectives of minimizing $f(\boldsymbol{x})$ and remaining in the interior of the convex set are balanced. As $\alpha \to \infty$, the problem amounts to the minimization of the original objective function $f$.

In solving this sequence of problems, the outer iteration consists in increasing the values of the parameter $\alpha$. The inner iteration is used to minimize the value of $B(\alpha)$ at that value of $\alpha$, using the result of the previous outer iteration as an initial guess. The inner iteration can be solved using Newton's method (10). For positive values of $\alpha$, it is easy to see that $B(\alpha)$ is a convex function. The notion of a central path for a linearly constrained optimization problem is shown in Fig. 9.

**Method of Centers.** Given a scalar value $t > f(\boldsymbol{x}^*)$, the method of centers finds the analytic center of the set of inequalities $f(\boldsymbol{x}) \leq t$ and $g_i(\boldsymbol{x}) \leq \boldsymbol{0}$ by minimizing the function

$$-\log[t - f(\boldsymbol{x})] + \Phi(\boldsymbol{x})$$

where $\Phi(\boldsymbol{x})$ is the log barrier function defined earlier. The optimal value $\boldsymbol{x}^*$ associated with solving the optimization problem associated with finding the analytic center for this barrier function is found, and the value of $t$ is updated to be a convex combination of $t$ and $f(\boldsymbol{x}^*)$ as

$$t \leftarrow \theta t + (1 - \theta) f(\boldsymbol{x}^*), \qquad \theta > 0$$

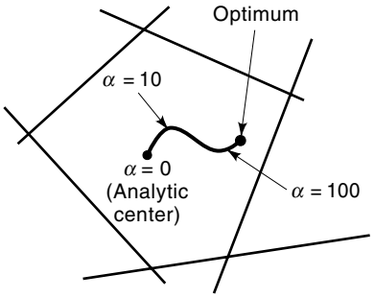


**Figure 9.** Central path for a linearly constrained convex optimization problem.

**The Self-Concordance Property and Step Length.** The value of $\theta$ above is an adjustable parameter that will affect the number of Newton iterations required to find the optimum value of the analytic center. Depending on the value of $\theta$ chosen, the technique is classified as a short-step, medium-step, or long-step (possibly even with $\theta > 1$) method. For a short-step method, one Newton iteration is enough, while for longer steps, further Newton iterations may be necessary.

Nesterov and Nemirovskii (6) introduced the following idea of the self-concordance condition:

***Definition.*** A convex function $\phi : S \to \mathbf{R}$ is *self-concordant* with parameter $a \geq 0$ ($a$-self-concordant) on $S$ if $\phi$ is three times continuously differentiable on $S$ and for all $\boldsymbol{x} \in S$ and $\boldsymbol{h} \in \mathbf{R}^m$, the following inequality holds:

$$|D^3\varphi(\boldsymbol{x})[\boldsymbol{h}, \boldsymbol{h}, \boldsymbol{h}]| \leq 2a^{-1/2}\{D^2\varphi(\boldsymbol{x})[\boldsymbol{h}, \boldsymbol{h}]\}^{3/2}$$

where $D^k\phi(\boldsymbol{x})[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k]$ denotes the value of the $k$th differential of $\phi$ taken at $\boldsymbol{x}$ along the collection of directions $[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k]$.

By formulating logarithmic barrier functions that are self-concordant, proofs of the polynomial-time complexity of various interior point methods have been shown. An analysis of the computational complexity in terms of the number of outer and inner (Newton) iterations is presented in Refs. 13 and 6.

## Other Interior-Point Methods

**Affine Scaling Methods.** For a linear programming problem, the nonnegativity constraints $\boldsymbol{x} \geq \boldsymbol{0}$ are replaced by constraints of the type $\|X^{-1}(\boldsymbol{x} - \boldsymbol{x}_c)\| < \beta \leq 1$, representing an ellipsoid centered at $\boldsymbol{x}_c$. The linear program is then relaxed to the following form, whose feasible region is contained in that of the original linear program:

$$\min\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \|X^{-1}(\boldsymbol{x} - \boldsymbol{x}_c)\| < \beta \leq 1\}$$

Note that the linear inequalities in Eq. (LP) have been converted to equalities by the addition of slack variables. This form has the following closed-form solution:

$$\boldsymbol{x}(\beta) = \boldsymbol{x} - \beta \frac{XP_{AX}X\boldsymbol{c}}{\|P_{AX}X\boldsymbol{c}\|}$$

where $P_{AX} = I - XA^T(AX^2A^T)^{-1}AX$. The updated value of $\boldsymbol{x}$ is used in the next iteration, and so on. The search direction $XP_{AX}X\boldsymbol{c}$ is called the primal affine scaling direction and corresponds to the scaled projected gradient with respect to the objective function, with scaling matrix $X$. Depending on the value of $\beta$, the method may be a short-step or a long-step (with $\beta > 1$) method, and convergence proofs under various conditions are derived. For details of the references, the reader is referred to Ref. 13.

We consider a general convex programming problem of the type (note that the linear objective function form is used here)

$$\{\min f(\boldsymbol{y}) = \boldsymbol{b}^T\boldsymbol{y} : g_i(\boldsymbol{y}) \leq 0\}$$

The constraint set here is similarly replaced by the ellipsoidal constraint $(\boldsymbol{y} - \boldsymbol{y}_c)^T H (\boldsymbol{y} - \boldsymbol{y}_c) \leq \beta^2$, where $\boldsymbol{y}_c$ is the center of

the current ellipsoid, $\boldsymbol{y}$ is the variable over which the minimization is being performed, and $H$ is the Hessian of the log-barrier function $-\sum_{i=1}^{n} \log[-g_i(\boldsymbol{y})]$. The problem now reduces to

$$\{\min \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} : (\boldsymbol{y}-\boldsymbol{y}_{\mathrm{c}})^{\mathrm{T}}H(\boldsymbol{y}-\boldsymbol{y}_{\mathrm{c}}) \leq \beta^2\}$$

which has a closed-form solution of the form

$$\boldsymbol{y}(\beta) = \boldsymbol{y} - \beta\frac{H^{-1}\boldsymbol{b}}{\sqrt{\boldsymbol{b}^{\mathrm{T}}H^{-1}\boldsymbol{b}}}$$

This is used as the center of the ellipsoid in the next iteration. The procedure continues iteratively until convergence.

**Potential-Based Methods.** These methods formulate a potential function that provides a coarse estimate of how far the solution at the current iterate is from the optimal value. At each step of a potential reduction method, a direction and step size are prescribed; however, the potential may be minimized further by the use of a line search (large steps). This is in contrast with a path-following method that must maintain proximity to a prescribed central path. An example of a potential-based technique is one that utilizes a weighted sum of the gap between the value of the primal optimization problem and its dual, and of the log barrier function value as the potential. For a more detailed description of this and other potential-based methods, the reader is referred to Refs. 6 and 13.

### Localization Approaches

**Polytope Reduction.** This method begins with a polytope $\mathscr{P} \in \mathbf{R}^n$ that contains the optimal solution, $\boldsymbol{x}_{\mathrm{opt}}$. The polytope $\mathscr{P}$ may, for example, be initially selected to be an $n$-dimensional box described by the set

$$\{\boldsymbol{x} | x_{\min} \leq x(i) \leq x_{\max}\}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of each variable, respectively. In each iteration, the volume of the polytope is shrunk while keeping $\boldsymbol{x}_{\mathrm{opt}}$ within the polytope, until the polytope becomes sufficiently small. The algorithm proceeds iteratively as follows.

*Step 1.* A *center* $\boldsymbol{x}_{\mathrm{c}}$ deep in the interior of the polytope $\mathscr{P}$ is found.

*Step 2.* The feasibility of the center $\boldsymbol{x}_{\mathrm{c}}$ is determined by verifying whether all of the constraints of the optimization problem are satisfied at $\boldsymbol{x}_{\mathrm{c}}$. If the point $\boldsymbol{x}_{\mathrm{c}}$ is infeasible, it is possible to find a separating hyperplane passing through $\boldsymbol{x}_{\mathrm{c}}$ that divides $\mathscr{P}$ into two parts, such that the feasible region lies entirely in the part satisfying the constraint

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \geq \beta$$

where $\boldsymbol{c} = -[\nabla g_{\mathrm{p}}(\boldsymbol{x})]^{\mathrm{T}}$ is the negative of the gradient of a violated constraint $g_{\mathrm{p}}$, and $\beta = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{c}}$. The separating hyperplane above corresponds to the tangent plane to the violated constraint. If the point $\boldsymbol{x}_{\mathrm{c}}$ lies within the feasible region, then there exists a hyperplane $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \geq \beta$ that divides the polytope into two parts such that $\boldsymbol{x}_{\mathrm{opt}}$ is contained in one of them, with $\boldsymbol{c} = -[\nabla f(\boldsymbol{x})]^{\mathrm{T}}$ being the negative of the gradient of the objective function, and $\beta$ being defined as $\beta = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{c}}$ once again. This hyperplane is the supporting hyperplane for the set $f(\boldsymbol{x}) \leq f(\boldsymbol{x}_{\mathrm{c}})$ and thus eliminates from the polytope a set of points whose value is larger than the value at the current center. In either case, a new constraint $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \geq \beta$ is added to the current polytope to give a new polytope that has roughly half the original volume.

*Step 3.* Go to step 1 and repeat the process until the polytope is sufficiently small.

Note that the center in step 1 is ideally the center of gravity of the current polytope, since a hyperplane passing through the center of gravity is guaranteed to reduce the volume of the polytope by a factor of $1 - 1/e$ in each iteration. However, since finding the center of gravity is prohibitively expensive in terms of computation, the analytic center is an acceptable approximation.

***Example.*** The algorithm is illustrated by using it to solve the following problem in two dimensions:

$$\begin{aligned} &\text{minimize} \quad f(x_1, x_2) \\ &\text{such that} \quad (x_1, x_2) \in S \end{aligned}$$

where $S$ is a convex set and $f$ is a convex function. The shaded region in Fig. 10(a) is the set $S$, and the dashed lines show the level curves of $f$. The point $\boldsymbol{x}_{\mathrm{opt}}$ is the solution to this problem. The expected solution region is first bounded by a rectangle with center $\boldsymbol{x}_{\mathrm{c}}$, as shown in Fig. 10(a). The feasibility of $\boldsymbol{x}_{\mathrm{c}}$ is then determined; in this case, it can be seen that $\boldsymbol{x}_{\mathrm{c}}$ is infeasible. Hence, the gradient of the constraint function is used to construct a hyperplane through $\boldsymbol{x}_{\mathrm{c}}$ such that the polytope is divided into two parts of roughly equal volume, one of which contains the solution $\boldsymbol{x}_{\mathrm{c}}$. This is illustrated in Fig. 10(b), where the region enclosed in darkened lines corresponds to the updated polytope. The process is repeated on the new,

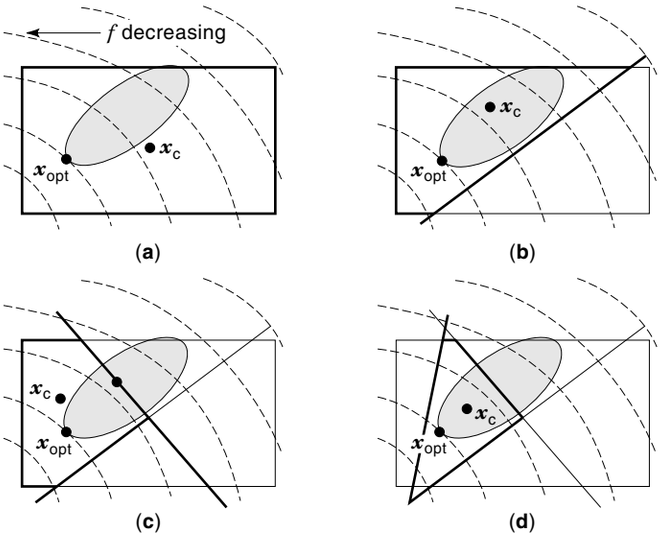


**Figure 10.** Polytope reduction approach.

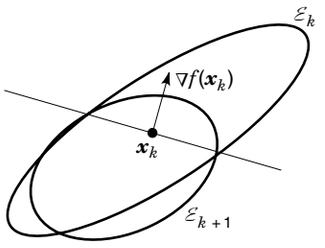


**Figure 11.** The ellipsoidal method.

smaller polytope. Its center lies inside the feasible region, and hence the gradient of the objective function is used to generate a hyperplane that further shrinks the size of the polytope, as shown in Fig. 10(c). The result of another iteration is illustrated in Fig. 10(d). The process continues until the polytope has been shrunk sufficiently.

**Ellipsoidal Method.** The ellipsoidal method begins with a sufficiently large ellipsoid that contains the solution to the convex optimization problem. In each iteration, the size of the ellipsoid is shrunk, while maintaining the invariant that the solution lies within the ellipsoid, until the ellipsoid becomes sufficiently small. The process is illustrated in Fig. 11.

The $k$th iteration consists of the following steps, starting from the fact that the center $\mathbf{x}_k$ of the ellipsoid $\mathcal{E}_k$ is known:

*Step 1.* In case the center is not in the feasible region, the gradient of the violated constraint is evaluated; if it is feasible, the gradient of the objective function is found. In either case, we will refer to the computed gradient as $\nabla h(\boldsymbol{x}_k)$.

*Step 2.* A new ellipsoid containing the half ellipsoid given by

$$\mathcal{E}_k \cap \{\boldsymbol{x} | \nabla h(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{x} \leq \nabla h(\boldsymbol{x}_k)^{\mathrm{T}} \boldsymbol{x}_k\}$$

is computed. This new ellipsoid, $\mathcal{E}_{k+1}$, and its center $\boldsymbol{x}_{k+1}$, are given by the following relations:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{n+1} A_k \tilde{\boldsymbol{g}}_k$$

$$A_{k+1} = \frac{n^2}{n^2-1}\left(A_k - \frac{2}{n+1} A_k \tilde{\boldsymbol{g}}_k \tilde{\boldsymbol{g}}_k^{\mathrm{T}} A_k\right)$$

where

$$\tilde{\boldsymbol{g}}_k = \frac{\nabla h(\boldsymbol{x}_k)}{\sqrt{\nabla h(\boldsymbol{x}_k)^{\mathrm{T}} A_k \nabla h(\boldsymbol{x}_k)}}$$

*Step 3.* Repeat the iterations in steps 1 and 2 until the ellipsoid is sufficiently small.

## RELATED TECHNIQUES

### Quasiconvex Optimization

***Definition.*** A function $f : S \to \mathbf{R}$, where $S$ is a convex set, is *quasiconvex* if every level set $L_a = \{\boldsymbol{x} | f(\boldsymbol{x}) \leq a\}$ is a convex set. A function $g$ is *quasiconcave* if $-g$ is quasiconvex over $S$. A function is *quasilinear* if it is both quasiconvex and quasiconcave.

Some elementary examples of such functions are:

1. Clearly, any convex (concave) function is also quasiconvex (quasiconcave).
2. Any monotone function $f : \mathbf{R} \to \mathbf{R}$ is quasilinear.
3. The linear fractional function $f(\boldsymbol{x}) = (\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b)/(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + d)$ where $\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{x} \in \mathbf{R}^n$, is quasilinear over the half space $\{\boldsymbol{x} | \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + d > 0\}$.

**Other Characterizations of Quasiconvexity.** A function $f$ defined on a convex set $\Omega$ is quasiconvex if, for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Omega$:

1. For every $\alpha$, $0 \leq \alpha \leq 1$, $f(\alpha\ \boldsymbol{x}_1 + (1 - \alpha)\boldsymbol{x}_2) \leq \max[f(\boldsymbol{x}_1), f(\boldsymbol{x}_2)]$.
2. If $f$ is differentiable, $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_2) \Rightarrow (\boldsymbol{x}_2 - \boldsymbol{x}_1)^{\mathrm{T}} \nabla f(\boldsymbol{x}_1) \leq 0$.

***Property.*** If $f, g$ are quasiconvex over $\Omega$, then the functions $\alpha f$ for $\alpha > 0$ and $\max(f, g)$ are also quasiconvex over $\Omega$. The composed function $g(f(\boldsymbol{x}))$ is quasiconvex provided $g$ is monotone increasing. In general, the function $f + g$ is *not* quasiconvex over $\Omega$.

As in the case of convex optimization, the gradient of a quasiconvex objective can be used to eliminate a half space from consideration. The work in Ref. 14 presents an adaptation of the ellipsoidal method to solve quasiconvex optimization problems.

### Semidefinite Programming

The problem of semidefinite programming (15) is stated as follows:

$$\begin{aligned} &\text{minimize} && \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{subject to} && F(\boldsymbol{x}) \geq 0 \\ &\text{where} && F(\boldsymbol{x}) \in \mathbf{R}^{m \times m}, \quad \boldsymbol{c}, \boldsymbol{x} \in \mathbf{R}^n \end{aligned} \tag{SDP}$$

Here, $F(\boldsymbol{x}) = F_0 + F_1\, x_1 + \ldots + F_n\, x_n$ is an affine matrix function of $\boldsymbol{x}$, and the constraint $F(\boldsymbol{x}) \geq 0$ represents the fact that this matrix function must be positive semidefinite, i.e., $\boldsymbol{z}^{\mathrm{T}} F(\boldsymbol{x})\, \boldsymbol{z} \geq 0$ for all $\boldsymbol{z} \in \mathbf{R}^n$. The constraint is referred to as a linear matrix inequality. The objective function is linear and hence convex, and the feasible region is convex, since if $F(\boldsymbol{x}) \geq 0$ and $F(\boldsymbol{y}) \geq 0$, then for all $0 \leq \lambda \leq 1$ it can be readily seen that $\lambda\, F(\boldsymbol{x}) + (1 - \lambda)\, F(\boldsymbol{y}) \geq 0$.

A linear program is a simple case of a semidefinite program. To see this, we can rewrite the constraint set $A\, \boldsymbol{x} \geq \boldsymbol{b}$ (note that the $\geq$ here is a componentwise inequality, and it is not related to positive semidefiniteness) as $F(\boldsymbol{x}) = \mathrm{diag}(A\, \boldsymbol{x} - \boldsymbol{b})$, i.e., $F_0 = \mathrm{diag}(\boldsymbol{b})$, $F_j = \mathrm{diag}(\boldsymbol{a}_j)$, $j = 1, \ldots, n$, where $A = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \ldots\ \boldsymbol{a}_n] \in \mathbf{R}^{m \times n}$.

Semidefinite programs may also be used to represent nonlinear optimization problems. As an example, consider the problem

$$\begin{aligned} &\text{minimize} && \frac{(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x})^2}{\boldsymbol{d}^{\mathrm{T}}\boldsymbol{x}} \\ &\text{subject to} && A\boldsymbol{x} + \boldsymbol{b} \geq \boldsymbol{0} \end{aligned}$$

where we assume that $\boldsymbol{d}^{\mathrm{T}}\, \boldsymbol{x} > 0$ in the feasible region. Note that the constraints here represent, as in the case of a linear program, componentwise inequalities. The problem is first re-

written as minimizing an auxillary variable $t$ subject to the original set of constraints and a new constraint

$$\frac{(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x})^2}{\boldsymbol{d}^{\mathrm{T}}\boldsymbol{x}} \leq t$$

The problem may be cast in the form of a semidefinite program as

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to} \quad \begin{bmatrix} \operatorname{diag}(A\boldsymbol{x}+\boldsymbol{b}) & 0 & 0 \\ 0 & t & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ 0 & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \boldsymbol{d}^{\mathrm{T}}\boldsymbol{x} \end{bmatrix} \geq 0 \end{aligned}$$

The first constraint row appears in a manner similar to the linear programming case. The second and third rows use Schur complements (16) to represent the nonlinear convex constraint above as the $2 \times 2$ matrix inequality

$$\begin{bmatrix} t & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \boldsymbol{d}^{\mathrm{T}}\boldsymbol{x} \end{bmatrix} \geq 0$$

The two tricks shown here, namely, the reformulation of linear inequations and the use of Schur complements, are often used to formulate optimization problems as semidefinite programs.

### Geometric Programming

***Definition.*** A *posynomial* is a function $h$ of a positive variable $\boldsymbol{x} \in \mathbf{R}^m$ that has the form

$$h(\boldsymbol{x}) = \sum_j \gamma_j \prod_{i=1}^{n} x_i^{\alpha(i,j)}$$

where the exponents $\alpha(i,j) \in \mathbf{R}$ and the coefficients $\gamma_j > 0$, $\gamma_j \in \mathbf{R}$.

For example, the function $f(x,y,z) = 7.4x + 2.6y^{3.18}z^{-4.2} + \sqrt{3}\ x^{-2}y^{-1.4}z^{\sqrt{5}}$ is a posynomial in the variables $x$, $y$, and $z$. Roughly speaking, a posynomial is a function that is similar to a polynomial, except that the coefficients $\gamma_j$ must be positive, and an exponent $\alpha(i,j)$ can be any real number, not necessarily a positive integer.

A posynomial has the useful property (17) that it can be mapped onto a convex function through an elementary variable transformation, $x(i) = e^{z(i)}$. Such functional forms are useful because in the case of an optimization problem where the objective function and the constraints are posynomial, the problem can easily be mapped onto a convex programming problem.

For some geometric programming problems, simple techniques based on the use of the arithmetic–geometric inequality may be used to obtain simple closed-form solutions to the optimization problems (18). The arithmetic–geometric inequality states that if $u_1, u_2, \ldots, u_n > 0$, then their arithmetic mean is no smaller than their geometric mean, i.e.,

$$\frac{u_1 + u_2 + \cdots + u_n}{n} \geq (u_1 u_2 \ldots u_n)^{1/n}$$

with equality occurring if and only if $u_1 = u_2 = \ldots = u_n$.

A simple illustration of this technique is in minimizing the outer surface area of an open box of fixed volume (say, 4 units) and sides of length $x_1$, $x_2$, $x_3$. The problem can be stated as

$$\begin{aligned} &\text{minimize} \quad x_1x_2 + 2x_1x_3 + 2x_1x_3 \\ &\text{subject to} \quad x_1x_2x_3 = 4 \end{aligned}$$

By setting $u_1 = x_1x_2$, $u_2 = 2x_1x_3$, $u_3 = 2x_1x_3$, and applying the condition listed above, the minimum value of the objective function is $3(u_1u_2u_3)^{1/3} = 3(4x_1^2x_2^2x_3^2) = 12$. It is easily verified that this corresponds to the values $x_1 = 2$, $x_2 = 2$, $x_3 = 1$.

We add a cautionary note that some, but not all, posynomial programming problems may be solved using this simple solution technique. For further details, the reader is referred to Ref. 18.

### Optimization Involving Logarithmic Concave Functions

A function $f$ is a logarithmic concave (log-concave) function if $\log f$ is a concave function. Log-convex functions are similarly defined. The maximization of a log-concave function over a convex set is therefore a unimodal problem, i.e., any local minimum is a global minimum. Log-concave functional forms are seen among some common probability distributions on $\mathbf{R}^n$, for example:

1. The Gaussian or normal distribution

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det \Sigma}}\, e^{-0.5(\mathbf{x}-\mathbf{x}_c)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mathbf{x}_c)}$$

   where $\Sigma \geq 0$.
2. The exponential distribution

$$f(\boldsymbol{x}) = \left(\prod_{i=1}^{n} \lambda(\mathrm{i})\right) e^{-(\lambda(1)\mathrm{x}(1)+\lambda(2)\mathrm{x}(2)+\cdots+\lambda(\mathrm{n})\mathrm{x}(\mathrm{n}))}$$

The following properties are true of log-concave functions:

1. If $f$ and $g$ are log-concave, then their product $fg$ is log-concave.
2. If $f(\boldsymbol{x}, \boldsymbol{y})$ is log-concave, then the integral $\int_S f(\boldsymbol{x}, \boldsymbol{y})\, d\boldsymbol{x}$ is log-concave provided $S$ is a convex set.
3. If $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are log-concave, then the convolution $\int_S f(\boldsymbol{x} - \boldsymbol{y})\, g(\boldsymbol{y})\, d\boldsymbol{y}$ is log-concave if $S$ is a convex set (this follows from properties 1 and 2).

## ENGINEERING PROBLEMS AS CONVEX OPTIMIZATION PROBLEMS

There has been an enormous amount of recent interest in applying convex optimization to engineering problems, particularly as the optimizers have grown more efficient. The reader is referred to Boyd and Vandenberghe's lecture notes (19) for a treatment of this subject. In this section, we present a sampling of engineering problems that can be posed as convex programs to illustrate the power of the technique in practice.

### Design Centering

While manufacturing any system, it is inevitable that process variations will cause design parameters, such as component values, to waver from their nominal values. As a result, after manufacture, the system may no longer meet some behavioral specifications, such as requirements on the delay, gain, and bandwidth, that it has been designed to satisfy. The procedure of design centering attempts to select the nominal values of design parameters to ensure that the behavior of the system remains within specifications with the greatest probability and thus maximize the manufacturing yield.

The random variations in the values of the design parameters are modeled by a probability density function $\Psi(\boldsymbol{x},\boldsymbol{x}_c)$: $\mathbf{R}^n \rightarrow [0,1]$, with a mean corresponding to the nominal value of the design parameters. The yield of the manufacturing process, $Y$, as a function of the mean $\boldsymbol{x}_c$ is given by

$$Y(\boldsymbol{x}_c) = \int_{x \in \mathscr{F}} \Psi(\boldsymbol{x}, \boldsymbol{x}_c)\, d\boldsymbol{x}$$

where $\mathscr{F}$ corresponds to the feasible region where all design constraints are satisfied.

A common assumption made by geometrical design centering algorithms for integrated circuit applications is that $\mathscr{F}$ is a convex bounded body. Techniques for approximating this body by a polytope $\mathscr{P}$ have been presented in Ref. 20. When the probability density functions that represent variations in the design parameters are Gaussian in nature, the design centering problem can be posed as a convex programming problem. The design centering problem is formulated as (21)

$$\text{maximize} \quad Y(\boldsymbol{x}_c) = \int_{x \in \mathscr{P}} \Psi(\boldsymbol{x}, \boldsymbol{x}_c)\, d\boldsymbol{x}$$

where $\mathscr{P}$ is the polytope approximation to the feasible region $\mathscr{F}$. Since the integral of a log-concave function over a convex region is also a log-concave function (22), the yield function $Y(\boldsymbol{x})$ is log-concave, and the above problem reduces to a problem of maximizing a log-concave function over a convex set. Hence, this can be transformed into a convex programming problem.

### Robust Optimal Control

Consider a single-input single-output discrete-time linear dynamic system with a finite impulse response described by

$$y(t) = h_0 u(t) + h_1 u(t-1) + \cdots + h_k u(t-k)$$

where $\boldsymbol{u}$ is the input sequence, $y$ is the output sequence, and $h_t$ is the $t$th impulse response coefficient. Given a desired response $y_d(t)$, the problem is to find an finite bounded input sequence $u(t)$ for which the output $y(t)$ most closely tracks the desired response, subject to constraints on the slew rate of the input signal. The problem may be stated as

$$\begin{aligned} \text{minimize} \quad & \text{error} = \max_{t=1,2,\ldots,M} |y(t) - y_d(t)| \\ \text{subject to} \quad & U_{\text{low}} \le u(t) \le U_{\text{high}}, \quad t = 1, 2, \ldots, N \\ & |u(t+1) - u(t)| \le S, \quad t = 1, 2, \ldots, N-1 \end{aligned}$$

If the $h_k$'s are entirely known, then the problem is a linear program. However, if they are not known exactly, given the uncertainty in the $h_k$'s, then it may be possible to say that their values lie within the ellipsoid

$$\mathscr{H} = \{\boldsymbol{h} | \boldsymbol{h} = \boldsymbol{h}_c + F\boldsymbol{p}, \|\boldsymbol{p}\| \le 1\}$$

where $\|\boldsymbol{p}\| = (\boldsymbol{p}^T\boldsymbol{p})^{1/2}$, $\boldsymbol{h} \in \mathbf{R}^{k+1}$, $F \in \mathbf{R}^{(k+1)\times q}$.

The robust version of the optimization problem above must ensure that the error is minimized over all possible values of $\boldsymbol{h}$ within the ellipsoid. To consider the worst-case tracking error, the optimization problem may be written as

$$\begin{aligned} \text{minimize} \quad & \text{worst-case error} = \max_{h \in \mathscr{H}} \max_{t=1,2,\ldots,M} |\, y(t) - y_d(t)| \\ \text{subject to} \quad & U_{\text{low}} \le u(t) \le U_{\text{high}}, \quad t = 1, 2, \ldots, N \\ & |u(t+1) - u(t)| \le S, \quad t = 1, 2, \ldots, N-1 \end{aligned}$$

The value of $\boldsymbol{p}$ at which the worst-case error is maximized may be derived analytically (23), and the corresponding worst-case tracking error at that point is

$$\max_{t=1,2,\ldots,M} [\|F^T D_t \boldsymbol{u}\| + |\boldsymbol{h}^T D_t \boldsymbol{u} - y_d(t)|]$$

The problem is therefore described as a specific form of convex programming problem, called a second-order cone programming (SOCP) problem as follows:

$$\begin{aligned} \text{minimize} \quad & \gamma \\ \text{subject to} \quad & U_{\text{low}} \le u(t) \le U_{\text{high}}, \quad t = 1, 2, \ldots, N \\ & -S \le u(t+1) - u(t) \le S, \quad t = 1, 2, \ldots, N-1 \\ & \|F^T D_t \boldsymbol{u}\| + [\boldsymbol{h}^T D_t \boldsymbol{u} - y_{\text{des}}(t)] \le \gamma \\ & \|F^T D_t \boldsymbol{u}\| - [\boldsymbol{h}^T D_t \boldsymbol{u} - y_{\text{des}}(t)] \le \gamma \end{aligned}$$

### Optimizing Structural Dynamics

Consider a linear elastic structure consisting of a stack of $k$ linear elastic bars connecting a set of $p$ nodes. The topology and lengths of the bars and their material are fixed, and the appropriate cross-sectional widths of the bars are to be determined. The elastic stored energy of this system is given by $\boldsymbol{f}^T\boldsymbol{d}$, where $\boldsymbol{f}$ is the vector of load forces and $\boldsymbol{d}$ is the vector of (small) node displacements. The relation between $\boldsymbol{f}$ and $\boldsymbol{d}$ is given by $\boldsymbol{f} = A(\boldsymbol{x})\,\boldsymbol{d}$, where $A(\boldsymbol{x})$, called the stiffness matrix, is an affine sum of the variables $x_i$, given by $A(\boldsymbol{x}) = \sum_{i=1}^{k} x_i A_i$ with the matrices $A$ being all symmetric positive semidefinite. The optimization problem of minimizing the elastic stored energy (24) can then be stated as follows:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{f}^T\boldsymbol{d} \\ \text{subject to} \quad & \sum_{j=1}^{k} l_j x_j \le v \\ & \boldsymbol{f} = A(\boldsymbol{x})\boldsymbol{d} \\ & x_{j,\min} \le x_j \le x_{j,\max} \quad \text{for } j = 1, 2, \ldots, k \end{aligned}$$

Here $v$ is the maximum volume, and $z_j$ the length of the $j$th bar. The last constraint places simple bounds on the values of the $\boldsymbol{x}$ variables (clearly, all of these variables must be positive, since they correspond to physical lengths). We can then

rewrite the problem by eliminating the $\boldsymbol{d}$ variables by substitution, as follows:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{f}^{\mathrm{T}} A(\boldsymbol{x})^{-1} \boldsymbol{f} \\ \text{subject to} \quad & \sum_{j=1}^{k} l_j x_j \le v \\ & x_{j,\min} \le x_j \le x_{j,\max} \quad \text{for } j = 1, 2, \ldots, k \end{aligned}$$

Using Schur complements, this leads to the semidefinite programming formulation in $\boldsymbol{x}$ and $t$ given by

$$\begin{aligned} \text{minimize} \quad & t \\ \text{subject to} \quad & \begin{bmatrix} t & \boldsymbol{f}^{\mathrm{T}} \\ \boldsymbol{f} & A(\mathbf{x}) \end{bmatrix} \ge 0 \\ & \sum_{j=1}^{k} l_j x_j \le v \\ & x_{j,\min} \le x_j \le x_{j,\max} \quad \text{for } j = 1, 2, \ldots, k \end{aligned}$$

### VLSI Transistor and Wire Sizing

**Convex Optimization Formulation.** Circuit delays in integrated circuits often have to be reduced to obtain faster response times. Given the circuit topology, the delay of a combinational circuit can be controlled by varying the sizes of transistors, giving rise to an optimization problem of finding the appropriate area–delay tradeoff. The formal statement of the problem is as follows:

$$\begin{aligned} \text{minimize} \quad & \text{area} \\ \text{subject to} \quad & \text{delay} \le T_{\text{spec}} \end{aligned} \tag{TS}$$

The circuit area is estimated as the sum of transistor sizes, i.e.,

$$\text{area} = \sum_{i=1}^{n} x_i$$

where $x_i$ is the size of the $i$th transistor and $n$ is the number of transistors in the circuit. This is easily seen to be a posynomial function of the $x_i$'s. The circuit delay is estimated using the Elmore delay estimate (25), which calculates the delay as a maximum of path delays. Each path delay is a sum of resistance–capacitance products. Each resistance term is of the form $a/x_i$, and each capacitance term is of the type $\sum b_i x_i$, with the constants $a$ and $b_i$ being positive. As a result, the delays are posynomial functions of the $x_i$'s, and the feasible region for the optimization problem is an intersection of constraints of the form

$$(\text{posynomial function in } x_i\text{'s}) \le T_{\text{spec}}$$

Since the objective and constraints are both posynomial functions in the $x_i$'s, the problem is equivalent to a convex programming problem. Various solutions to the problem have been proposed, for instance, in Refs. 26 and 27.

**Semidefinite Programming Formulation.** In the problem (TS) above, the circuit delay may alternatively be determined from the dominant eigenvalue of a matrix $G^{-1}C$, where $G$ and $C$ are, respectively, matrices representing the conductances (corresponding to the resistances) and the capacitances referred to above. The entries in both $G$ and $C$ are affine functions of the $x_i$'s. The dominant time constant can be calculated as the negative inverse of the largest zero of the polynomial $\det(sC + G)$. It is also possible to calculate it using the following linear matrix inequality:

$$T^{\text{dom}} = \min\{T | TG - C \ge 0\}$$

The "$\ge 0$" here refers to the fact that the matrix must be positive definite. To ensure that $T^{\text{dom}} \le T_{\max}$ for a specified value of $T_{\max}$, the linear matrix inequality $T_{\max} G(\boldsymbol{x}) - C(\boldsymbol{x}) \ge 0$ must be satisfied. This sets up the problem in the form of a semidefinite program as follows (28):

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} l_i x_i \\ \text{subject to} \quad & T_{\max} G(\boldsymbol{x}) - C(\boldsymbol{x}) \ge 0 \\ & \boldsymbol{x}_{\min} \le \boldsymbol{x} \le \boldsymbol{x}_{\max} \end{aligned}$$

### Largest Inscribed Ellipsoid in a Polytope

Consider a polytope in $\mathbf{R}^n$ given by $\mathscr{P} = \{\boldsymbol{x} | \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{x} \le b_i,\ i = 1, 2, \ldots, L\}$ into which the largest ellipsoid $\mathscr{E}$, described as follows, is to be inscribed:

$$\mathscr{E} = \{B\boldsymbol{y} + \boldsymbol{d} | \|\boldsymbol{y}\| \le 1\}, \qquad B = B^{\mathrm{T}} > 0$$

The center of this ellipsoid is $\boldsymbol{d}$, and its volume is proportional to $\det B$. The objective here is to find the entries in the matrix $B$ and the vector $\boldsymbol{d}$. To ensure that the ellipsoid is contained within the polytope, it must be ensured that for all $\boldsymbol{y}$ such that $\|\boldsymbol{y}\| \le 1$,

$$\boldsymbol{a}_i^{\mathrm{T}} (B\boldsymbol{y} + \boldsymbol{d}) \le b_i$$

Therefore, it must be true that $\sup_{\|\boldsymbol{y}\| \le 1} (\boldsymbol{a}_i^{\mathrm{T}} B\boldsymbol{y} + \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{d}) \le b_i$, or in other words, $\|B\boldsymbol{a}_i\| \le b_i - \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{d}$. The optimization problem may now be set up as

$$\begin{aligned} \text{maximize} \quad & \log\det B \\ \text{subject to} \quad & B = B^{\mathrm{T}} > 0 \\ & \|B\boldsymbol{a}_i\| \le b_i - \boldsymbol{a}_i^{\mathrm{T}} \boldsymbol{d}, \quad i = 1, 2, \ldots, L \end{aligned}$$

This is a convex optimization problem (6) in the variables $B$ and $\boldsymbol{d}$, with a total dimension of $n(n + 1)/2$ variables corresponding to the entries in $B$ and $n$ variables corresponding to those in $\boldsymbol{d}$.

### Beamforming

Antenna arrays are often used to detect and process signals arriving from different directions. Each sensor is associated with a parameter called its weight, typically a complex number, and the values of these weights determine the beam pattern. For a planar array with $N$ elements, the beam pattern is given by the expression

$$G(\theta) \sum_{i=1}^{N} w_i g_i(\theta) \exp\left(j \frac{2\pi}{\lambda} (x_i \cos\theta + y_i \sin\theta)\right)$$

The problem of optimal beamforming is to choose the weights in such a way that the pattern level in given areas $\theta_i$ are minimized, subject to upper bound constraints $\theta_j$ for other angles and a maximum level on the weights. The optimization problem is formally written as (29)

$$\begin{aligned} \text{minimize} \quad & |G(\theta_i)| \\ \text{subject to} \quad & |G(\theta_i)| \le U_i \quad \text{for } j = 1, 2, \ldots, M \\ & |w_k| \le W \quad \text{for } k = 1, 2, \ldots, N \\ & G(\theta_0) = 1 \end{aligned}$$

This may be rewritten as

$$\begin{aligned} \text{minimize} \quad & t \\ \text{subject to} \quad & |G(\theta_i)| \le t, \quad i = 1, 2, \ldots, L \\ & |G(\theta_i)| \le U_j \quad \text{for } j = 1, 2, \ldots, M \\ & |w_k| \le W \quad \text{for } k = 1, 2, \ldots, N \\ & G(\theta_0) = 1 \end{aligned}$$

The last constraint here is a normalization constraint and can be handled by decreasing the number of variables by one. Recalling that each $w_j$ is complex, we may choose the vector $\boldsymbol{x}$ as

$$\boldsymbol{x} = [\operatorname{Re} w_1, \operatorname{Im} w_1, \operatorname{Re} w_2, \operatorname{Im} w_2, \ldots, \operatorname{Re} w_{N-1}), \operatorname{Im} w_{N-1}, f\,]$$

and rewrite the problem in the corresponding convex form:

$$\text{subject to} \quad \|A_i \boldsymbol{x} + \boldsymbol{b}_i\| < \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{x} + \boldsymbol{d}_i, \quad i = 1, 2, \ldots, L + M + N$$

Here, the amplitude of $G(\theta_i)$ is represented as the norm of a vector $A_i\boldsymbol{x} + \boldsymbol{b}_i$ where $\boldsymbol{x} \in \mathbf{R}^{2N-1}$, $A_i \in \mathbf{R}^{2\times(2N-1)}$, $\boldsymbol{b}_i \in \mathbf{R}^2$. The two components of $A_i$ are the real and imaginary parts of $G(\theta_i)$. The vectors $\boldsymbol{c}_i$ and $\boldsymbol{d}_i$ are defined as follows:

$$\begin{aligned} \boldsymbol{c}_i = \{ & [0, 0, \ldots, 0, 1] \quad \text{for} \quad i = 1, 2, \ldots, L \\ & [0, 0, 0, \ldots, 0] \quad \text{for} \quad i = L + 1, \ldots, L + M + N \\ \boldsymbol{d}_i = \{ & [0, 0, \ldots, 0, 0] \quad \text{for} \quad i = 1, 2, \ldots L \\ & U_{i-L} \quad \text{for} \quad i = L + 1, \ldots, M \\ & W \quad \text{for} \quad i = L + M + 1, \ldots, L + M + N \end{aligned}$$

Note that the objective here is linear, and the constraints quadratic.

## CONCLUSION

This overview has presented an outline of convex programming. The use of specialized techniques that exploit the convexity properties of the problem have led to rapid recent advances in efficient solution techniques for convex programs, which have been outlined here. The applications of convex optimization to real problems of engineering design have been illustrated. Convex optimization techniques are used widely in control, for example, in Youla-based design and in design by linear matrix inequalities (LMIs). For Youla-based design, the reader is referred to Refs. 30 and 31. A good sourcebook for design by LMIs is Ref. 32 and a useful practical design tool is the LMI Control Toolbox (33).

## BIBLIOGRAPHY


1. W. Stadler, *Multicriteria Optimization in Engineering and in the Sciences,* New York: Plenum, 1988.
2. N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica,* **4**: 373–395, 1984.
3. A. V. Fiacco and G. P. McCormick, *Nonlinear Programming,* New York: Wiley, 1968.
4. J. Renegar, A polynomial time algorithm, based on Newton's method, for linear programming, *Math. Programming,* **40**: 59–93, 1988.
5. C. C. Gonzaga, An algorithm for solving linear programming problems in $O(nL)$ operations, in *Progress in Mathematical Programming: Interior Point and Related Methods,* N. Meggido (ed.), New York: Springer-Verlag, 1988, pp. 1–28.
6. Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming,* SIAM Studies in Applied Mathematics, Philadelphia: Society for Industrial and Applied Mathematics, 1994.
7. P. M. Vaidya, An algorithm for linear programming which requires $O(((m + n)n^2 + (m + n)^{1.5}n)L)$ arithmetic operations, *Math. Programming,* **47**: 175–201, 1990.
8. Y. Ye, An $O(n^3L)$ potential reduction algorithm for linear programming, *Math. Programming,* **50**: 239–258, 1991.
9. R. T. Rockafellar, *Convex Analysis,* Princeton, NJ: Princeton University Press, 1970.
10. D. G. Luenberger, *Linear and Nonlinear Programming,* Reading, MA: Addison-Wesley, 1984.
11. P. E. Gill, W. Murray, and M. H. Wright, *Numerical Linear Algebra and Optimization,* Reading, MA: Addison-Wesley, 1991.
12. A. J. Schrijver, *Theory of Linear and Integer Programming,* New York: Wiley, 1986.
13. D. den Hertog, *Interior Point Approach to Linear, Quadratic and Convex Programming,* Boston: Kluwer Academic, 1994.
14. J. A. dos Santos Gromicho, *Quasiconvex Optimization and Location Theory,* Amsterdam: Thesis Publishers, 1995.
15. L. Vandenberghe and S. Boyd, Semidefinite programming, *SIAM Rev.,* **38** (1): 49–95, 1996.
16. G. H. Golub and C. F. Van Loan, *Matrix Computations,* 2nd ed., Baltimore, MD: The Johns Hopkins University Press, 1989.
17. J. G. Ecker, Geometric programming: methods, computations and applications, *SIAM Rev.,* **22** (3): 338–362, 1980.
18. R. J. Duffin and E. L. Peterson, *Geometric Programming: Theory and Application,* New York: Wiley, 1967.
19. S. Boyd and L. Vandenberghe, *Introduction to Convex Optimization with Engineering Applications,* Lecture Notes, Electrical Engineering Department, Stanford University, 1995. Available from http://www-isl.stanford.edu/~boyd.
20. S. W. Director and G. D. Hachtel, The simplicial approximation approach to design centering, *IEEE Trans. Circuits Syst.,* **CAS-24**: 363–372, 1977.
21. S. S. Sapatnekar, P. M. Vaidya, and S. M. Kang, Convexity-based algorithms for design centering, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* **13**: 1536–1549, 1994.
22. A. Prekopa, Logarithmic concave measures and other topics, in *Stochastic Programming,* M. Dempster (ed.), London: Academic Press, 1980, pp. 63–82.
23. S. Boyd, C. Crusius, and A. Hansson, Control applications of nonlinear convex programming, Electrical Engineering Department, Stanford University, 1997; *J. Process Control,* in press.
24. A. Ben-Tal and M. P. Bendsoe, A new method for optimal truss topology design, *SIAM J. Optimiz.,* **3**: 322–358, 1993.

25. S. S. Sapatnekar and S. M. Kang, *Design Automation for Timing-Driven Layout Synthesis,* Boston: Kluwer Academic, 1993.

26. J. Fishburn and A. E. Dunlop, TILOS: A posynomial programming approach to transistor sizing, in *Proc. IEEE Int. Conf. Comput.-Aided Des.,* 1985, pp. 326–328.

27. S. S. Sapatnekar et al., An exact solution to the transistor sizing problem using convex optimization, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* **12**: 1621–1632, 1993.

28. L. Vandenberghe, S. Boyd, and A. El Gamal, Optimal wire and transistor sizing for circuits with non-tree topology, in *Proc. IEEE Int. Conf. Comput.-Aided Des.,* 1997, pp. 252–259.

29. H. Lebret, Optimal beamforming via interior point methods, *J. VLSI Signal Process.,* **14** (1): 29–41, 1996.

30. S. P. Boyd and C. H. Baratt, *Linear Controller Design: Limits of Performance,* Englewood Cliffs, NJ: Prentice-Hall, 1991.

31. M. A. Dahleh and I. J. Diaz-Bobillo, *Control of Uncertain Systems: A Linear Programming Approach,* Englewood Cliffs, NJ: Prentice-Hall, 1995.

32. S. P. Boyd et al., *Linear Matrix Inequalities in System and Control Theory,* SIAM Studies in Applied Mathematics, Philadelphia, PA: SIAM, 1994.

33. P. Gahinet et al., *LMI Control Toolbox,* Natick, MA: The MathWorks, 1995.

SACHIN S. SAPATNEKAR
University of Minnesota