Japan J. Indust. Appl. Math., 26 (2009), 337–363 Area ⟨2⟩

On Verified Numerical Computations in Convex Programming

Christian Jansson

Institute for Reliable Computing, Hamburg University of Technology
Schwarzenbergstraße 95, Hamburg 21071, Germany
E-mail: jansson@tu-harburg.de

Received March 31, 2008
Revised November 21, 2008

This survey contains recent developments for computing verified results of convex constrained optimization problems, with emphasis on applications. Especially, we consider the computation of verified error bounds for non-smooth convex conic optimization in the framework of functional analysis, for linear programming, and for semidefinite programming. A discussion of important problem transformations to special types of convex problems and convex relaxations is included. The latter are important for handling and for reliability issues in global robust and combinatorial optimization. Some remarks on numerical experiences, including also large-scale and ill-posed problems, and software for verified computations concludes this survey.

Key words: linear programming, semidefinite programming, conic programming, convex programming, combinatorial optimization, rounding errors, ill-posed problems, interval arithmetic, branch-bound-and-cut

1. Introduction

Mathematical models of real world problems are incomplete in many cases and therefore are flexible to some extent. Frequently, this flexibility can be used to build a tractable model. In contrast to traditional optimization which did not pay much attention to complexity, and which focused on asymptotical convergence results by assuming certain smoothness properties, it turned out that convexity plays a crucial role from the point of view of applications and tractability.

During the last decade, major developments in convex optimization were focusing on conic programming, a natural non-smooth extension of linear programming. There, the problem is to minimize a linear function over a feasible set given as the intersection of an affine subspace and a convex cone. Conic programming is in fact a universal form of convex programming, since each convex set is the intersection of a hyperplane with an appropriate cone in a higher-dimensional space. The advantage of the conic form is that nearly all convex problems with interesting applications can be reformulated as conic problems solvable in polynomial time, although they are non-smooth in many cases (see Nesterov and Nemirovski [47]). Most of the applications are covered by linear programming and semidefinite programming (SDP).

In practice, we have to live with the fact that for almost all nonlinear problems we never have an exactly representable optimal solution. The current optimization solvers simply implement algorithms valid in exact arithmetic, but do not care about

roundoff errors due to floating-point arithmetic. Even for combinatorial problems with a finite number of integral solutions roundoff errors may yield nonsensical results. Indeed, Neumaier and Shcherbina [53] have shown that for a small innocent-looking linear integer problem many well-known state-of-the-art solvers comprising CPLEX 8.0 did not find the optimal solution and declared the problem infeasible, because intermediate computations for solving ill-conditioned linear relaxations lead to erroneous decisions.

Forward and backward error analysis together with a detailed discussion of roundoff errors and condition numbers for matrix problems were first described in the outstanding papers published sixty years ago by von Neumann and Goldstine [54] and Turing [71]. Today, in this context frequently the notion *verified* or *rigorous error bounds* is used, and these two papers can be viewed as the pioneering work in the field of verification methods, a part of numerical analysis. Rigorous forward error bounds are propagated in interval arithmetic, too; see the books Alefeld and Herzberger [1], Hansen [21], Kearfott [36], Moore [44], and Neumaier [50], [51].

This survey wants to present the fundamental concepts of conic programming, duality, and a part of the large variety of its applications. Special attention is payed to verified results for conic optimization problems including a discussion of ill-conditioned and ill-posed problems up to large scale. The basic results are formulated in the framework of functional analysis; that is, the underlying vector spaces are in general infinite-dimensional.

Non-convex optimization is much more difficult than convex programming, since many local minimizers may occur which are not global. Frequently these problems are solved by using relaxations in a sequential manner. Most important are branch-bound-and-cut algorithms that solve relaxations at the nodes of the branch-and-bound tree. A relaxation is a tractable auxiliary modification of the original problem such that each optimal solution of the original problem is feasible for the modification. Nearly all relaxations used in practice are conic problems, or can be reformulated in this form. One consequence is that verified results for conic optimization problems yield verified results for non-convex global optimization problems. An excellent survey by Neumaier [52] covers the state of the art of techniques for solving constrained global optimization problems and combinatorial problems, the use of relaxations, important problem transformations, and also a discussion of roundoff errors for bounded linear programming problems. The reader is referred to this survey and the literature cited there for verified results in non-convex optimization.

The paper is organized as follows. After introducing some notation and basic definitions in Section 2, we consider in the next section conic optimization problems, duality, and sensitivity analysis. Then in Section 4 we will focus on verified error bounds for the exact optimal value, verified error bounds for optimal and almost optimal solutions, verifying strong duality, and verified certificates for proving infeasibility. In Section 5 several applications are presented. Moreover, a

rigorous calculus of conic representable sets and functions is described, evidencing the abilities to reformulate many applications as conic programming problems. Then in Section 6 we list some codes for solving conic programs approximately and rigorously. Furthermore, some remarks or numerical results for well-known benchmark problems are given. We end this survey with some conclusions.

2. Notation and preliminaries

Let $\mathcal{X}$ be a real vector space equipped with a norm $\|\cdot\|$, and let $\mathcal{K} \subseteq \mathcal{X}$ be a *convex cone*, i.e., $\mathcal{K} + \mathcal{K} \subseteq \mathcal{K}$, $\alpha\mathcal{K} \subseteq \mathcal{K}$ for $\alpha \in \mathbf{R}_+$, where $\mathbf{R}_+$ denotes the set of nonnegative real numbers. A convex cone $\mathcal{K}$ induces a *partial ordering* $x \leq y$ iff $y - x \in \mathcal{K}$, which is a transitive and reflexive binary relation on $\mathcal{X}$ compatible with addition and scalar multiplication. Conversely, each partial ordering determines a convex cone, namely the *positive cone* $\mathcal{K} := \{x \in \mathcal{X} \colon x \geq 0\}$. A vector space $\mathcal{X}$ equipped with a partial ordering is called a *partially ordered vector space*. Given a partial ordering the set $[\underline{x}, \overline{x}] := \{x \in \mathcal{X} \colon \underline{x} \leq x \leq \overline{x}\} = (\underline{x} + \mathcal{K}) \cap (\overline{x} - \mathcal{K})$ is called an *interval*. For a subset $\mathcal{M}$ of a partially ordered vector space $\mathcal{X}$ a vector $\underline{x}$ is called a *lower bound* of $\mathcal{M}$, if $\underline{x} \leq m$ for all $m \in \mathcal{M}$, and in this case we write $\underline{x} \leq \mathcal{M}$. The lower bound $\underline{x}$ is called *infimum* of $\mathcal{M}$ if every other lower bound $\underline{y}$ of $\mathcal{M}$ satisfies $\underline{y} \leq \underline{x}$. Analogously, *upper bounds* and *supremum* are defined. $\mathcal{X}$ is said to be a *vector lattice* for the partial ordering $\leq$ if for all $x, y \in \mathcal{X}$ the supremum $\sup\{x, y\}$ and the infimum $\inf\{x, y\}$ exists and is contained in $\mathcal{X}$, respectively. In a vector lattice the operations $x^+ := \sup\{x, 0\}$, $x^- := \inf\{x, 0\}$ and $|x| := \sup\{x, -x\}$ are defined, and the properties $|x| = x^+ - x^-$, $x = x^+ + x^-$, $|x| = 0$ iff $x = 0$, $|\lambda x| = |\lambda|\,|x|$ for real λ, and $|x + y| \leq |x| + |y|$ are satisfied.

Let $\mathcal{X}^*$ denote the *dual space* of $\mathcal{X}$, that is the space of continuous linear functionals endowed with the operator norm. The set $\mathcal{K}^*$ of all positive linear functionals, i.e.,

$$\mathcal{K}^* = \{y \in \mathcal{X}^* \colon \langle y, x\rangle := y(x) \geq 0 \text{ for all } x \in \mathcal{K}\}, \tag{2.1}$$

is a convex cone in $\mathcal{X}^*$ defining a partial ordering in the dual space.

The basic properties and relations for vector lattices as well as examples can be found in Birkhoff [10] and Bourbaki [13]; see also Peressini [58] and Schaefer [67]. We use the same notation $\|\cdot\|$ and $\leq$ for all norms and partial orderings. It will always be clear from the context which norm and which cone is referred to. Hence, if $x \in \mathcal{X}$ then $x \geq 0$ means $x \in \mathcal{K}$, and if $y \in \mathcal{X}^*$ then $y \geq 0$ denotes $y \in \mathcal{K}^*$. Observe that we do not write y^* for a continuous linear functional in $\mathcal{X}^*$ because from the position in $\langle y, x\rangle$ the meaning is clear, and we can omit the star. This notion is closely related to Hilbert spaces and the Theorem of Riesz which states that the continuous linear functions can be represented by the inner product $\langle y, x\rangle$, where y is a vector in the Hilbert space.

In the following a few illustrative and well-known examples of normed vector lattices are shown. The real finite dimensional space $\mathcal{X} = \mathbf{R}^n$ equipped with the Euclidean inner product and the Euclidean norm $\|\cdot\|$ can be ordered by the positive orthant

$$\mathcal{K} := \mathbf{R}^n_+ = \{x \in \mathbf{R}^n \colon x_i \geq 0 \text{ for } i = 1, \ldots, n\}. \tag{2.2}$$

This cone is *self-dual* (i.e., $\mathcal{K} = \mathcal{K}^*$) and implies the lattice operations

$$x_i^+ = \max\{0, x_i\}, \quad x_i^- = \min\{0, x_i\}, \quad |x_i| = x_i^+ - x_i^- \tag{2.3}$$

for $i = 1, \ldots, n$. This vector lattice is used in *linear programming* (LP).

In *second order cone programming* (SOCP) the same normed space $\mathcal{X} = \mathbf{R}^n$ is equipped with the partial ordering defined by the convex *ice-cream* or *Lorenz cone*

$$\mathcal{K} := \left\{ x = \begin{pmatrix} x_: \\ x_n \end{pmatrix} \in \mathbf{R}^n \colon x_n \geq \|x_:\| \right\}, \tag{2.4}$$

where $x_: := (x_1, \ldots, x_{n-1})^{\mathrm{T}}$. This cone is also self-dual, and further properties, like the lattice vector operations, are described in [31].

In *semidefinite programming* (SDP) the real linear space is $\mathcal{X} = \mathbf{R}^{n(n+1)/2}$, which is identified with the set of real symmetric $n \times n$ matrices X. The inner product of two matrices X, Y is defined by $\langle X, Y \rangle := \operatorname{trace} X^{\mathrm{T}} Y = \sum_{ij} X_{ij} Y_{ij}$, and the induced norm $\|X\| := (\operatorname{trace} X^{\mathrm{T}} X)^{1/2}$ is the *Frobenius norm*. This space is a Hilbert space, thus self-dual, and it is equipped with the self-dual cone of positive semidefinite matrices

$$\mathcal{K} := S^n_+ = \{X \in \mathcal{X} \colon X \text{ is positive semidefinite}\}. \tag{2.5}$$

Using the eigenvalue decomposition $X = Q^{\mathrm{T}} \Lambda Q$ of a real symmetric matrix, it follows that the lattice operations are

$$X^- = Q^{\mathrm{T}} \Lambda^- Q, \quad X^+ = Q^{\mathrm{T}} \Lambda^+ Q, \quad |X| = Q^{\mathrm{T}} |\Lambda| Q, \tag{2.6}$$

where Λ^-, Λ^+, and $|\Lambda|$ denote the diagonal matrices with nonpositive, nonnegative, and modulus of the eigenvalues of X on the diagonal, respectively.

For any compact Hausdorff space Ω, the vector space $\mathcal{X} := C(\Omega)$ of real-valued functions is a normed vector lattice with norm and ordering cone

$$\|f\|_{C(\Omega)} := \sup_{x \in \Omega} \{|f(x)|\}, \quad \mathcal{K} := \{f \in C(\Omega) \colon f(x) \geq 0 \text{ for all } x \in \Omega\}.$$

3. Conic programming

In this section we present a brief overview of important concepts in conic optimization. Most of these well-known results are described using the framework of functional analysis. For more detailed expositions refer to Ben-Tal and Nemirovski [7], Nemirovski and Nesterov [47], Renegar [59] and [60], and the literature cited there.

3.1. The standard form

The *conic optimization problem in standard form* is defined as

$$\text{minimize } \langle c, x\rangle \quad \text{s.t. } Ax = b,\ x \in \mathcal{K}, \tag{3.1}$$

where $\mathcal{X}$ is a real normed vector space, $\mathcal{K} \subseteq \mathcal{X}$ is a convex cone, $c \in \mathcal{X}^*$, $\mathcal{Y}$ is a real normed vector space, A denotes a continuous linear operator from $\mathcal{X}$ to $\mathcal{Y}$, and $b \in \mathcal{Y}$. A vector x that satisfies the constraints $Ax = b$, $x \in \mathcal{K}$ is called *primal feasible*. With $\hat{f}_p$ we denote the primal optimal value, where $\hat{f}_p := +\infty$ if the problem is infeasible. Many interesting examples of optimization problems can be formulated in this framework. In the following some familiar facts are described. The *Lagrangian function* of problem (3.1) has the form

$$L(x, y) := \langle c, x\rangle + \langle y, b - Ax\rangle, \tag{3.2}$$

where $y \in \mathcal{Y}^*$. The optimization problem

$$\inf_{x\in\mathcal{K}} \sup_{y\in\mathcal{Y}^*} L(x, y) \tag{3.3}$$

is equivalent to (3.1). Indeed, if $b - Ax = 0$ then $\langle y, b - Ax\rangle = 0$ for each $y \in \mathcal{Y}^*$, and the supremum of $L(x, y)$ is equal to $\langle c, x\rangle$. In the case where $b - Ax \neq 0$ there is some y with $\langle y, b - Ax\rangle > 0$, and hence the supremum is infinite. The *adjoint operator* of A is the linear operator $A^*\colon \mathcal{Y}^* \to \mathcal{X}^*$ defined by the condition

$$\langle y, Ax\rangle = \langle A^*y, x\rangle \quad \text{for all } x \in \mathcal{X},\ y \in \mathcal{Y}^*. \tag{3.4}$$

Hence, the Lagrangian satisfies $L(x, y) = \langle y, b\rangle + \langle -A^*y + c, x\rangle$. By exchanging infimum and supremum in (3.3) we obtain the dual problem

$$\sup_{y\in\mathcal{X}^*} \inf_{x\in\mathcal{K}} L(x, y) \tag{3.5}$$

with optimal value $\hat{f}_d$. Since exchanging inf and sup always produces a lower bound, *weak duality* holds, that is $\hat{f}_d \leq \hat{f}_p$. Because $\inf_{x\in\mathcal{K}} L(x, y) = -\infty$ whenever $-A^*y + c \notin \mathcal{K}^*$, the dual problem can be written equivalently in the form

$$\text{maximize } \langle y, b\rangle \quad \text{s.t. } -A^*y + c \in \mathcal{K}^*,\ y \in \mathcal{Y}^*. \tag{3.6}$$

A vector y that satisfies the constraints $-A^*y + c \in \mathcal{K}^*$, $y \in \mathcal{Y}^*$ is called *dual feasible*. Hence, the set of dual feasible solutions is the inverse image of the cone $\mathcal{K}^*$ under an affine mapping. We set $\hat{f}_d := -\infty$, if the dual problem is infeasible.

Let x be primal feasible, and let y be dual feasible, then

$$\langle c, x\rangle = \langle c, x\rangle + \langle y, b - Ax\rangle = \langle -A^*y + c, x\rangle + \langle y, b\rangle \geq \langle y, b\rangle, \tag{3.7}$$

and using $s := -A^*y + c$ it follows that $\langle c, x\rangle - \langle y, b\rangle = \langle s, x\rangle \geq 0$. Hence, equality holds iff the *complementarity condition*

$$\langle -A^*y + c, x\rangle = \langle s, x\rangle = 0 \tag{3.8}$$

is fulfilled. This condition implies that *strong duality* holds, i.e., the duality gap $\hat{f}_p - \hat{f}_d$ is zero. Both problems have optimal solutions iff there exists a primal and a dual feasible solution fulfilling the complementarity condition. Obviously, a primal-dual optimal pair of solutions can be found by solving the following nonlinear system of equations

$$Ax = b, \quad A^*y + s = c, \quad \langle s, x\rangle = 0, \quad x \in \mathcal{K},\ s \in \mathcal{K}^* \tag{3.9}$$

which are called the *optimality conditions*.

In other cases, where such primal-dual optimal pairs do not exist, strong duality may be not fulfilled. But there are other sufficient conditions guaranteeing strong duality (see for example [7]):

THEOREM 3.1 (Duality theorem).

a) *If the primal problem is strictly feasible (i.e., there exists a primal feasible point x in the interior of $\mathcal{K}$) and $\hat{f}_p$ is finite, then $\hat{f}_p = \hat{f}_d$ and the dual supremum is attained.*

b) *If the dual problem is strictly feasible (i.e., there exists some $y \in \mathcal{Y}^*$ such that $-A^*y + c$ is in the interior of $\mathcal{K}^*$) and $\hat{f}_d$ is finite, then $\hat{f}_p = \hat{f}_d$ and the primal supremum is attained.*

An immediate consequence is that if both, the primal and the dual problem, admit strictly feasible points then the duality gap is zero and there exists a primal-dual pair of optimal solutions. In general, one problem may have optimal solutions and its dual is infeasible, or the duality gap may be positive at optimality. The strict feasibility assumptions in Theorem 3.1 are called *Slater's constraint qualifications*.

Conic duality is symmetric, and the dual of the dual is the primal. Moreover, conic duality is completely similar to LP duality, with the only exception that in the LP case strong duality is ensured by mere feasibility and not strict feasibility. Duality theory is central to the study of optimization. First, algorithms are frequently based on duality (like primal-dual interior point methods), secondly they enable one to check whether or not a given feasible point is optimal, and thirdly it allows one to compute verified results.

3.2. Three generic conic problems

The linear programming problem in standard form

$$\text{minimize } c^{\mathrm{T}}x \quad \text{s.t. } Ax = b,\ x \geq 0 \tag{3.10}$$

is the special case of the conic optimization problem where $\mathcal{X} = \mathcal{X}^* = \mathbf{R}^n$, $\mathcal{K} = \mathcal{K}^* = \mathbf{R}^n_+$ and $\mathcal{Y} = \mathcal{Y}^* = \mathbf{R}^m$. It follows that the dual problem is defined as

$$\text{maximize } b^{\mathrm{T}}y \quad \text{s.t. } -A^{\mathrm{T}}y + c \geq 0,\ y \in \mathbf{R}^m. \tag{3.11}$$

In SOCP the partial ordering is defined by the ice-cream cones (2.4). Let $\mathcal{K}$ be the Cartesian product of the cones $\mathcal{L}^{n_j} \subseteq \mathbf{R}^{n_j}$ for $j = 1, \ldots, n$. This is a convex, self-dual cone (see Alizadeh, Goldfarb [2]). The *standard SOCP problem* has the form

$$\text{minimize } \sum_{j=1}^{n} c_j^{\mathrm{T}}x_j \quad \text{s.t. } \sum_{j=1}^{n} A_j x_j = b,\ x_j \in \mathcal{L}^{n_j} \text{ for } j = 1, \ldots, n, \tag{3.12}$$

where $A_j \in \mathbf{R}^{m \times n_j}$, $c_j, x_j \in \mathbf{R}^{n_j}$ and $b \in \mathbf{R}^m$. If we merge these quantities

$$\begin{aligned} A &:= (A_1; \ldots; A_n), \\ c &:= (c_1; \ldots; c_n), \\ x &:= (x_1; \ldots; x_n), \end{aligned} \tag{3.13}$$

then the standard SOCP problem has the form (3.1), and it follows that the dual problem (3.6) can be written as

$$\text{maximize } b^{\mathrm{T}}y \quad \text{s.t. } -A_j^{\mathrm{T}}y + c_j \in \mathcal{L}^{n_j} \text{ for } j = 1, \ldots, n. \tag{3.14}$$

Here we have chosen the finite-dimensional spaces $\mathcal{X} := \mathbf{R}^{\bar{n}}$, $\bar{n} = \sum_j n_j$ and $\mathcal{Y} := \mathbf{R}^m$ equipped with the Euclidean inner products.

The *standard primal semidefinite programming problem* is

$$\text{minimize } \langle C, X \rangle \quad \text{s.t. } \langle A_i, X \rangle = b_i,\ i = 1, \ldots, m,\ X \in S^s_+, \tag{3.15}$$

where C, X and A_i are real symmetric $s \times s$ matrices, $b \in \mathbf{R}^m$, and $\langle \cdot \rangle$ denotes the inner product in the linear space of symmetric matrices. It follows immediately that the dual problem has the form

$$\text{maximize } b^{\mathrm{T}}y \quad \text{s.t. } -\sum_{i=1}^{m} y_i A_i + C \in S^s_+. \tag{3.16}$$

3.3. Block structured variables

Frequently conic optimization problems have block structured variables, that is, the variables are in the Cartesian product of different cones. More precisely, there are n real normed vector spaces $\mathcal{X}_1, \ldots, \mathcal{X}_n$, convex cones $\mathcal{K}_1 \subseteq \mathcal{X}_1, \ldots, \mathcal{K}_n \subseteq \mathcal{X}_n$, a real normed vector space $\mathcal{Y}$, and n continuous linear operators $A_j \colon \mathcal{X}_j \to \mathcal{Y}$. Let $\mathcal{X}$ and $\mathcal{K}$ denote the Cartesian products of the spaces $\mathcal{X}_j$ and the cones $\mathcal{K}_j$, respectively. The vectors x and c and the linear operator A are partitioned as follows:

$$\begin{aligned} x &= (x_1; \ldots; x_n), \quad && \text{where } x_j \in \mathcal{X}_j, \\ c &= (c_1; \ldots; c_n), && \text{where } c_j \in \mathcal{X}_j^*, \\ A &= (A_1; \ldots; A_n). \end{aligned}$$

Defining

$$Ax := \sum_{j=1}^{n} A_j x_j \quad \text{and} \quad \langle c, x \rangle := \sum_{j=1}^{n} \langle c_j, x_j \rangle, \tag{3.17}$$

it follows that $A \colon \mathcal{X} \to \mathcal{Y}$ is a continuous linear operator, and $c \in \mathcal{X}^*$. The *primal conic optimization problem with block structured variables* has the form

$$\text{minimize } \sum_{j=1}^{n} \langle c_j, x_j \rangle \quad \text{s.t. } \sum_{j=1}^{n} A_j x_j = b,\ x_j \in \mathcal{K}_j \text{ for } j = 1, \ldots, n, \tag{3.18}$$

and hence the dual problem is

$$\text{minimize } \langle y, b \rangle \quad \text{s.t. } (-A_1^* y; \ldots; -A_n^* y) + (c_1; \ldots; c_n) \in \mathcal{K}_1^* \times \cdots \times \mathcal{K}_n^*,\ y \in \mathcal{Y}^*. \tag{3.19}$$

3.4. Interior point methods

Interior point methods are the most efficient algorithms for solving convex optimization problems, including linear, second order cone and semidefinite problems. The main idea of these methods is based on the common wisdom in optimization that the simplest convex program is to minimize a three times continuously differentiable convex function when the Hessian is positive definite and the level sets are compact. Then, in the unconstrained case, Newton's method is the method of choice. In order to obtain such a simple convex program, a so-called self-concordant barrier function is constructed and added to the objective of the conic problem. The bulk of the work in each iteration step lies in the evaluation of the first and second derivatives and the solution of the linear system of optimality conditions solved by Newton's method. The most efficient interior point methods use barrier functions for both the primal and the dual problem. For details we refer the reader to Nesterov [46], Nesterov and Nemirovski [47], Nesterov and Todd [48], and Tuncel [70].

3.5. The condition number

The purpose of sensitivity analysis is to determine how the solution of a problem changes when the data are perturbed. In the case of inverting a linear continuous regular operator $A\colon \mathcal{X} \to \mathcal{X}$, the *condition number* of A is (see also von Neumann and Goldstine [54] and Turing [71]) defined to be the quantity

$$\operatorname{cond}(A) := \lim_{\|\Delta A\|\to 0} \sup \frac{\|(A+\Delta A)^{-1} - A^{-1}\|/\|A^{-1}\|}{\|\Delta A\|/\|A\|} = \|A\|\,\|A^{-1}\|. \tag{3.20}$$

The condition number quantifies the sensitivity of A^{-1} to perturbations in A, and roughly spoken, for each additional significant digit of accuracy in the inverse, it is necessary to use $\log(\operatorname{cond}(A))$ additional significant digits of accuracy in computing A^{-1}. It is well-known that

$$\operatorname{cond}(A) = \frac{1}{\varrho}, \quad \text{where } \varrho = \inf\left\{ \frac{\|\Delta A\|}{\|A\|} : A + \Delta A \text{ is singular} \right\}. \tag{3.21}$$

The quantity ϱ is the relative distance to the next singular operator. Renegar [59] generalized this condition number to conic programs with data $d = (A, b, c)$. Similar to the quantity ϱ, the *distance to primal infeasibility* is defined as

$$\varrho_{\mathrm{P}}(d) := \inf\left\{ \frac{\|\Delta d\|}{\|d\|} : \text{problem } d + \Delta d \text{ is primal infeasible} \right\}, \tag{3.22}$$

and the *distance to dual infeasibility* is

$$\varrho_{\mathrm{D}}(d) := \inf\left\{ \frac{\|\Delta d\|}{\|d\|} : \text{problem } d + \Delta d \text{ is dual infeasible} \right\}. \tag{3.23}$$

Here, $\|\cdot\|$ denotes a suitable norm in the set of data. The *condition number* of the problem instance d is

$$\operatorname{cond}(d) := \frac{1}{\min\{\varrho_{\mathrm{P}}(d), \varrho_{\mathrm{D}}(d)\}}, \tag{3.24}$$

that is, the scale-invariant reciprocal of the smallest data perturbation that will render the perturbed data instance either primal or dual infeasible. The problem is called *ill-posed* if $\varrho_{\mathrm{P}}(d)$ or $\varrho_{\mathrm{D}}(d)$ is zero, or equivalently $\operatorname{cond}(d) = \infty$.

As in the case of linear operators the condition number (3.24) describes the sensitivity of a conic program, and in [59] it is proved that the sensitivity of the optimal solutions and the optimal value can be bounded by the condition number. Crucial is that the *bounds for the optimal value depend cubically on the inverses of the relative distances to primal and dual infeasibility.*

For a very ill-conditioned or even ill-posed problem it follows that there may be arbitrarily small perturbed data instances such that the difference between the optimal value of the original problem and the perturbed problem is almost

arbitrarily large, but the optimality conditions for the perturbed problem almost coincide with the optimality conditions for the original problem. Since conic solvers are terminated if the optimality conditions are satisfied approximately, it cannot be distinguished between the optimal values of the original and the perturbed problem in the case of ill-conditioned or ill-posed problems. A consequence is that the noise introduced by floating point arithmetic may occasionally yield to wrong termination and nonsensical computational results. Hence, for such problems reliable results cannot be obtained without further assumptions. In the next section we show that only rough bounds for the norm of an optimal solution are completely sufficient for computing efficiently rather sharp bounds for the exact optimal value.

The distance to primal and dual feasibility, and hence the condition number, can be computed (see Ordóñez and Freund [57], and Freund, Ordóñez and Toh 2006 [20]), but it is rather expensive. For example in semidefinite programming, computing $\varrho_{\mathrm{P}}(d)$ requires solving $2m$ SDP-instances of comparable size and structure as the original problem instance. There it is also shown that ill-conditioned and ill-posed problems are not rare in practice, they occur even in linear programming. In [57] it is stated that 71 % of the lp-instances in the NETLIB linear programming library [49] are ill-posed. This library contains many industrial problems. In [20] it is shown that 32 out of 85 problems of the SDPLIB are ill-posed.

3.6. Certificates of infeasibility

Identifying infeasibility in conic optimization is of key importance. An infeasible model usually does not yield valuable information. Either it will have to be corrected and resolved, or discarded in branch-and-bound algorithms. Thus, early identification of infeasibility saves time. Certificates of infeasibility are based on alternative theorems. A theorem of alternatives states that for two systems of equations or inequalities, one or the other system has a solution, but not both. A solution of one of the systems is called a *certificate of infeasibility* for the other which has no solution, since in principle this allows an easy check to prove infeasibility.

Proposition 3.1. *If there is a $\tilde{y} \in Y^*$ that satisfies $A^*\tilde{y} \in \mathcal{K}^*$, $\langle \tilde{y}, b\rangle < 0$, then the system of primal constraints $Ax = b$, $x \in \mathcal{K}$ has no solution.*

Proof. If the system $Ax = b$, $x \in \mathcal{K}$ has a solution x, then $0 \leq \langle A^*\tilde{y}, x\rangle = \langle \tilde{y}, Ax\rangle = \langle \tilde{y}, b\rangle$ contradicting our assumption $\langle \tilde{y}, b\rangle < 0$. □

The linear functional $\tilde{y}$ is called a *certificate* of *primal infeasibility*, and represents a dual unbounded ray.

Proposition 3.2. *If there is a $\tilde{x} \in X$ that satisfies $A\tilde{x} = 0$, $\tilde{x} \in \mathcal{K}$, $\langle c, \tilde{x}\rangle < 0$, then the system of dual constraints $-A^*y + c \in \mathcal{K}^*$, $y \in Y^*$ has no solution.*

Proof. If the system $-A^*y + c \in \mathcal{K}^*$, $y \in Y^*$ has a solution $y \in Y^*$, then $0 \leq \langle -A^*y + c, \tilde{x}\rangle = -\langle y, A\tilde{x}\rangle + \langle c, \tilde{x}\rangle = \langle c, \tilde{x}\rangle < 0$ contradicting our assumption. □

The vector $\tilde{x}$ is called a *certificate* of *dual infeasibility* and represents a primal unbounded ray.

Approximate certificates of infeasibility are frequently computed by optimization algorithms if no feasible solutions of the primal or dual constraints exist. When equality constraints are present, certificates cannot be represented exactly in floating point arithmetic, and approximate certificates can satisfy the constraints only within certain tolerances. This effect is amplified by roundoff errors during the calculations for computing the approximate certificate. However, it turns out (see the next section) that in order to prove infeasibility by using floating-point arithmetic it is sufficient if an interval of small diameter can be computed which guarantees to contain a certificate. We call such an interval a *rigorous* or *verified certificate of infeasibility*.

4. Verified results for conic programming

In this section we give an overview of concepts related to verified numerical computations in conic optimization. Many results of this section can be found in [31]. They can be viewed as an extension of results for linear programming (cf. [28], and Neumaier and Shcherbina [53]), and for smooth convex programming (see [27]) to conic problems using the framework of functional analysis. Particular attention is payed to ill-conditioned and ill-posed problems.

4.1. Bounds for the optimal value

This section is elementary but important for understanding both the basic ideas behind rigorous forward error bounds and implementations. It turns out that for computing error bounds for the optimal value only approximate primal and dual solutions $\tilde{x}$, $\tilde{y}$ as well as rough bounds for the norm of ε-optimal solutions are required. Further assumptions about the accuracy of the approximations are not necessary; they need to be neither primal nor dual feasible. If the accuracy is poor, however, then the error bounds cause overestimation.

The cones $\mathcal{K}$ and $\mathcal{K}^*$ create partial orderings for the vector spaces $\mathcal{X}$ and $\mathcal{X}^*$, respectively. We assume that these spaces are vector lattices. However, many of the following results remain valid if the underlying spaces are partially ordered topological vector spaces where only lower and upper bounds of $\{x, y\}$ exist. For rigorously bounding the optimal value, we assume that the conic optimization problem satisfies the following condition which we call *primal boundedness qualification* (PBQ):

(i) Either the primal problem is infeasible,

(ii) or $\hat{f}_p$ is finite, and there is a simple bound $\overline{x} \in \mathcal{K}$ such that for every $\varepsilon > 0$ there exists a primal feasible solution $x(\varepsilon)$ satisfying $x(\varepsilon) \leq \overline{x}$ and $\langle c, x(\varepsilon)\rangle - \hat{f}_p \leq \varepsilon$.

Observe that PBQ implies that the primal problem is bounded from below, but the existence of an optimal solution is not required, only simple bounds $\overline{x}$ for ε-optimal solutions are required. This qualification and even more restrictive assumptions, like certain smoothness properties, are fulfilled in allmost all applications, and they are customary when solving ill-posed or very ill-conditioned

problems with regularization methods. Notice that we do not assume that Slater's constraint qualifications are fulfilled.

The following theorem provides a finite lower bound $\underline{f}_p$ on the exact primal optimal value.

THEOREM 4.1. *Assume that PBQ holds. Let $\tilde{y} \in \mathcal{Y}^*$ and let $d := -A^*\tilde{y} + c$. Suppose further that $\underline{d}^- \leq \inf\{d, 0\}$. Then*

(a) *The primal optimal value is bounded from below by*

$$\hat{f}_p \geq \langle \tilde{y}, b\rangle + \langle \underline{d}^-, \overline{x}\rangle =: \underline{f}_p. \tag{4.1}$$

(b) *If $\underline{d}^- = 0$, then $\tilde{y}$ is dual feasible and $\hat{f}_d \geq \underline{f}_p = \langle \tilde{y}, b\rangle$. Moreover, if $\tilde{y}$ is optimal, then $\hat{f}_d = \underline{f}_p$.*

Sketch of Proof. (a) If the primal problem is infeasible, then $\hat{f}_p = +\infty$, and each finite value is a lower bound. Hence, assume that PBQ (ii) is satisfied with $x := x(\varepsilon)$ and $\varepsilon > 0$. Then

$$\begin{aligned}\langle c, x\rangle &= \langle d, x\rangle + \langle A^*\tilde{y}, x\rangle \\ &= \langle \tilde{y}, b\rangle + \langle \tilde{y}, Ax - b\rangle + \langle d, x\rangle.\end{aligned}$$

Since x is primal feasible, $Ax - b = 0$ and $\langle c, x\rangle = \langle \tilde{y}, b\rangle + \langle d, x\rangle$. It is easy to prove the inequality $\langle d, x\rangle \geq \langle \underline{d}^-, \overline{x}\rangle$. Hence, $\langle c, x\rangle \geq \langle \tilde{y}, b\rangle + \langle \underline{d}^-, \overline{x}\rangle$. Because of PBQ (ii)

$$\hat{f}_p \geq \langle c, x\rangle - \varepsilon \geq \langle \tilde{y}, b\rangle + \langle \underline{d}^-, \overline{x}\rangle - \varepsilon.$$

For $\varepsilon \to 0$ the assertion (a) follows.

(b) If $\underline{d}^- = 0$ then $d \in \mathcal{K}^*$, implying that $\tilde{y}$ is dual feasible, and the assertion follows. □

The lower bound uses the approximate optimal value $\langle \tilde{y}, b\rangle$, and a correction is added which takes into account the violation of dual feasibility $\underline{d}^-$ evaluated at the upper bound $\overline{x}$. Since an approximate solution $\tilde{y}$ which is close to an optimal solution is almost feasible, it follows that d is close to $\mathcal{K}^*$. Hence, each lower bound $\underline{d}^-$ sufficiently close to d^- is almost zero implying that $\langle \underline{d}^-, \overline{x}\rangle \approx 0$ provided $\overline{x}$ is not too large. In this case $\underline{f}_p \approx \langle \tilde{y}, b\rangle$ is reasonable; that is, the overestimation is not very much larger than necessary.

We illustrate the bound for linear programming problems in standard form (3.10). Linear programming problems have always zero duality gap implying a unique optimal value $\hat{f}$. Theorem 4.1 and (2.3) yield immediately the lower bound

$$\hat{f} \geq b^{\mathrm{T}}\tilde{y} + (\underline{d}^-)^{\mathrm{T}}\overline{x} =: \underline{f}_p, \tag{4.2}$$

where $\underline{d}_j^- \leq \min\{0, (-A^{\mathrm{T}}\tilde{y} + c)_j\}$ for $j = 1, \ldots, n$. It is straightforward to control all effects of rounding errors for computing $\underline{f}_p$ by using directed rounding or interval

arithmetic. The MATLAB toolbox INTLAB [65] provides the directed rounding modes, and the following short INTLAB program produces a rigorous lower bound:

```
setround(-1);
dlminus = min(0,A'*(-yt)+c);
flow = b'*yt + dlminus'*xup;
setround(0);
```

If interval arithmetic is used, then the input data A, b, c may be intervals, and we obtain a lower bound for each instance within the interval data. Verified error bounds for general linear programming problems also with free variables can be found in [28], and for formula (4.2) see Corollary 6.1 in [28].

For semidefinite programming problems in standard form it follows from Theorem 4.1 and (2.6) that the primal optimal value is bounded from below by

$$\hat{f}_p \geq b^{\mathrm{T}}\tilde{y} + l \cdot \underline{d}^{-} \cdot \overline{x} =: \underline{f}_p, \tag{4.3}$$

where $\underline{d}^{-} \leq \min\{\lambda_{\min}(D), 0\}$, D is the defect matrix $D = C - \sum_{i=1}^{m} \tilde{y}_i A_i$ describing the violations of dual feasibility, $\lambda_{\min}(D)$ is the smallest eigenvalue of D, and l is an upper bound of the number of negative eigenvalues of D. For controlling all rounding errors and computing a verified lower bound $\underline{f}_p$ see [31].

In the following we consider the computation of a rigorous upper bound for the optimal value. We assume that the conic optimization problem satisfies the following condition, which we call the *dual boundedness qualification* (DBQ):

(i) either the dual problem is infeasible,

(ii) or $\hat{f}_d$ is finite, and there is a simple bound $\overline{y}$ such that for every $\varepsilon > 0$ there exists a dual feasible solution $y(\varepsilon)$ satisfying $|y(\varepsilon)| \leq \overline{y}$ and $\hat{f}_d - \langle y(\varepsilon), b\rangle \leq \varepsilon$.

THEOREM 4.2. *Assume that DBQ holds. Let $\tilde{x} \in \mathcal{K}$, and suppose further that $|A\tilde{x} - b| \leq \overline{r}$. Then*

(a) *The dual optimal value is bounded from above by*

$$\hat{f}_d \leq \langle c, \tilde{x}\rangle + \langle \overline{y}, \overline{r}\rangle =: \overline{f}_d. \tag{4.4}$$

(b) *If $\overline{r} = 0$, then $\tilde{x}$ is primal feasible and $\hat{f}_p \leq \overline{f}_d = \langle c, \tilde{x}\rangle$, and if moreover $\tilde{x}$ is optimal, then $\hat{f}_p = \overline{f}_d$.*

For a proof see [31].

Conic solvers compute in general only approximations $\tilde{x} \notin \mathcal{K}$. But a reasonable approximation must be close to $\mathcal{K}$. In order to satisfy the assumption that the approximation is in $\mathcal{K}$, we replace $\tilde{x}$ by the supremum $\tilde{x}^+ = \sup\{\tilde{x}, 0\}$ or a close upper bound of $\tilde{x}^+$.

In the special case of linear programming we can take the exact supremum $\tilde{x}^+ \in \mathbf{R}^n_+$ defined by (2.3), obtaining the upper bound

$$\hat{f} \leq c^{\mathrm{T}}\tilde{x}^+ + \overline{y}^{\mathrm{T}}\overline{r} = \overline{f}_d, \tag{4.5}$$

where $\overline{r} := |A\tilde{x}^+ - b|$. The following short INTLAB program produces this upper bound:

```
xtplus = max(0,xt)
setround(-1);
rn = abs(A*xtplus -b);
setround(+1);
rp = abs(A*xtplus -b);
r = max(rn,rp);
fu = c'*xtplus + yup'*r;
setround(0);
```

As well as for the lower bound the input data A, b, c may be intervals, and using interval arithmetic we obtain a lower bound for each instance within the interval data. The computational costs for computing the lower or the upper bound in the case of linear programming are $O(mn)$, and thus negligible compared with the costs for computing the approximate solutions. In other words, the boundedness qualifications imply safety almost for free, and this is also true in the case of SOCP and SDP. Similar formulas for lower and upper bounds on the optimal value in the case of SOCP as well as for problems with structured variables are given in [31]. *In our experience, if the conic solver produces suitable approximations, then the bounds are quite accurate, while in other cases, the lower and upper bound differ so much that it warns the user that something went wrong or needs special attention. This observation applies also to ill-conditioned and ill-posed problems.*

4.2. Bounds for the optimal solution

A more difficult task is the computation of rigorous error bounds for optimal solutions. Krawzcyk [40] was the first who solved rigorously non-degenerate linear programming problems in standard form. In his approach first an approximate primal and dual basic solution together with the corresponding set of basic indices B are computed by any LP solver. The basic indices split the matrix A into two parts $A = (A_B, A_N)$, where A_B is a square matrix and N denotes the set of indices that are not in B. Similarly, the variables $x = (x_B; x_N)$ and $s = (s_B; s_N)$ are split. Then the optimality conditions (3.9) for the standard LP, are

$$\begin{aligned} &Ax = b, && x \geq 0, \\ &A^{\mathrm{T}}y + s = c, && s \geq 0, \\ &s^{\mathrm{T}}x = 0. \end{aligned} \tag{4.6}$$

These conditions are split as follows:

$$\begin{aligned} &A_B x_B = b, \quad x_B \geq 0, \quad x_N := 0, \\ &A_B^{\mathrm{T}} y = c_B, \quad s_B := 0, \quad s_N = c_N - A_N^{\mathrm{T}} y \geq 0, \\ &s^{\mathrm{T}}x = 0. \end{aligned} \tag{4.7}$$

In other words, the nonbasic variables x_N and s_B are fixed to zero, and the other variables x_B and y are the solutions of two quadratic linear systems of equations.

Krawzcyk computes rigorous error bounds (i.e., enclosures) $[x_B] = [\underline{x}_B, \overline{x}_B]$ and $[y] = [\underline{y}, \overline{y}]$ for the solutions of these square $m \times m$ linear systems by using a verification method for linear systems of equations. He sets

$$[x] := ([x_B], x_N), \quad [s_N] := c_N - A_N^{\mathrm{T}}[y], \quad [s] := (s_B, [s_N]), \quad x_N := 0, \quad s_B := 0, \tag{4.8}$$

where $[s_N]$ is computed with interval arithmetic. Since $x_N = 0$ and $s_B = 0$, the nonlinear equation $s^{\mathrm{T}}x = 0$ is fulfilled for all vectors $x \in [x]$ and $s \in [s]$. Hence, if additionally the sign conditions $[x_B] \geq 0$ and $[s_N] \geq 0$ are fulfilled, then existence of an optimal solution within the error bounds $[x]$, $[y]$ and $[s]$ is verified. This method can also be applied to linear programming problems with interval data yielding enclosures of the optimal solutions for all point problems within the interval data. For modifications see also Beeck [4] and Rump [61].

There are several verification methods for computing enclosures of solutions of square linear and nonlinear systems in the finite dimensional case. A precise description of such methods, required assumptions and further properties can be found, for example, in Neumaier [50]. The complexity is $O(m^3)$ operations for full $m \times m$ matrices. Hence, the above error bounds require in addition to the computational work of the linear programming solver $\max\{O(m^3), O(mn)\}$ operations. For recent developments and fast algorithms for linear systems of equations see Oishi and Rump [55], and Rump and Ogita [66]. The computation of rigorous error bounds for the exact solution of linear systems with arbitrarily large ill-conditioned matrices is treated in Oishi et al. [56] and Rump [64]. For the computation of enclosures in the case of large sparse linear systems the reader is referred to Rump [62].

One disadvantage of Krawzcyk's approach is that only non-degenerate problems can be treated in this way, because the error bounds introduce a slight overestimation yielding in general a violation of the sign conditions in the degenerate case. The degenerate case, however, occurs rather frequently in practice. In [26] a method is described where degenerate problems and violations of the sign conditions are allowed. There, it is shown that the graph corresponding to the basic index sets of the optimal vertices is connected, and error bounds for all optimal vertices can be computed by using a graph search method. In particular, for each basic index set the two square linear systems are solved rigorously, yielding the computational work $k \cdot \max\{O(m^3), O(mn)\}$, where k is greater than or equal to the number of (degenerate) optimal basic index sets. Of course, for larger dimensions this method is applicable only for small degree k of degeneracy.

We mention that the technique of fixing appropriate variables (the nonbasic variables) and solving a quadratic system of equations for the remaining basic variables was later applied by Hansen [21] in order to prove existence of a feasible point for nonlinear equations within a bounded box. It was further modified and investigated numerically by Kearfott [34], [35], and is also described in his book [36].

For other conic problems, especially those with non-polyhedral cones, the computation of rigorous error bounds for optimal solutions is an open problem.

4.3. Bounds for ε-optimal solutions

For conic problems it is less complicated to compute rigorous error bounds for primal and dual feasible solutions close to optimality, instead of computing error bounds for optimal ones. The reason is that the nonlinear complementarity condition $\langle s, x\rangle = 0$ must not be fulfilled, and hence must not be verified. We describe briefly this approach. The basic algorithm for computing error bounds of a primal feasible solution that is close to optimality consists of the following steps:

(i) Perturb the primal constraints slightly such that the optimal solution of the perturbed problem is an interior feasible solution of the original problem.
(ii) Solve the perturbed problem with any conic solver, yielding an approximate optimal solution $\tilde{x}$.
(iii) Use this approximation to compute an enclosure containing a primal feasible solution.
(iv) Evaluate the objective function for the enclosure.

The first and the second step deliver an interior approximation close to optimality, provided that the conic solver produces reasonable results. Step (iii) is especially nontrivial, since the existence of feasible solutions must be proved rigorously. This can be done in a manner similar to the method as in the previous section. In the finite dimensional case, we have to find an exact solution of the linear equation $Ax = b$ for $x \in \mathcal{K}$. As before, we fix nonbasic variables of the approximate solution; that is, $x_N := \tilde{x}_N$. Then the other basic variables x_B are the solution to the corresponding square linear system of equations $A_B x_B = b - A_N x_N$, which is solved by any verification method yielding the enclosure $[x] := ([x_B], x_N)$. If $[x] = [\underline{x}, \overline{x}] \subseteq \mathcal{K}$, then there exists an $\tilde{\tilde{x}} \in [x]$ which is primal feasible and close to the approximation $\tilde{x}$. The check $[x] \subseteq \mathcal{K}$ depends on the underlying cone. For LP we immediately obtain the equivalent condition

$$\underline{x} \geq 0. \tag{4.9}$$

The algorithm for computing error bounds of dual feasible solutions is very similar. If in the primal and in the dual case error bounds for feasible solutions are available, then we have obtained enclosures for ε-optimal solutions, where ε results as the difference of the bounds for the primal and the dual optimal value.

Here, the boundedness qualifications PBQ and DBQ are not assumed. This necessitates that the above method also has to prove existence of feasible solutions and this is more expensive than the bounds of Section 4.1. Moreover, in general an upper bound on the optimal value can be obtained only if $\varrho_{\mathrm{P}}(d) > 0$, and a lower bound of the optimal value can be computed only if $\varrho_{\mathrm{D}}(d) > 0$.

A detailed description of the previous algorithms can be found in the case of linear programming in [28], for convex programming problems with smooth constraint functions in [27], and for semidefinite programming problems and linear matrix inequalities in [32].

4.4. Verified certificates of infeasibility

As already mentioned, many conic solvers expose infeasibility by computing approximate unbounded rays. It is easy to see that verified certificates of infeasibility can be computed with slight modification of the method for computing enclosures of ε-optimal solutions. The only difference is to verify the conditions in the propositions 3.1 and 3.2 instead of primal and dual feasibility. For details, see [31].

5. Applications

The interior point framework is especially well suited for solving efficiently and rigorously conic optimization problems. The conic representation has tremendous expressive abilities implying an extremely wide range of applications, including those in combinatorial optimization, polynomial optimization, control and system theory, design of statistical experiments, planning under uncertainty including robust optimization, Physics and Operations Research. Applications of conic optimization techniques in engineering are very exciting, for example in signal processing and communication, circuit design, channel equalization, filter design, digital beam forming, antennae design, truss topology design and many others.

Over the last decade, the spectrum of applications has been constantly growing, and a similar development may continue in the future. SDP is the most important class with a large variety of applications. The books by Ben-Tal and Nemirovskii [7] and Boyd and Vandenberghe [14] discuss several applications in science and engineering. Other references include the SDP handbook edited by Wolkowicz et al. [75], the monograph by De Klerk [18], the habilitation thesis of Helmberg [23], and the bibliography [74]. A recent survey of SDP based approaches in combinatorial optimization can be found in Krishnan and Terlaky [32]. Next, we present a few applications in more detail.

5.1. Conic representable sets and functions

We start with a brief description of a calculus which is suited for modelling conic programming problems. For a detailed description see Nemirovski [45]. A consequence of this calculus is that a large variety of problems and applications can be modelled as conic programs without any overestimation or wrapping effects, the latter being very likely in interval arithmetic. Specifically, we investigate how an optimization problem can be reformulated as a LP, SOCP or SDP problem. It turns out that there are some simple symbolic manipulations and transformations with so-called conic representable sets and functions that allow identification of convex conic programs. The conic form is favorable compared to the customary original form, because (i) it is much better suited to algorithmic processing with interior point methods, and (ii) the computed results can be verified efficiently, as shown before.

Usually, the original formulation of an optimization problem is in (or can be immediately converted to) the form that a linear objective function is minimized over a set X of feasible solutions. The feasible set X is defined as the intersection of

some sets $X_i = \{x\colon g_i(x) \le 0\}$, where g_i denotes the i-th constraint function. In this formulation we have in mind that always linearity of the objective can be assumed, since nonlinear objectives can be attached to the constraints by introducing one auxiliary variable and taking its level sets.

The idea for finding the desired conic reformulation is to describe the set X as a projection of the set of feasible solutions of a convex conic problem. Then, minimizing a linear objective $\langle c, x\rangle$ over the set X is equivalent to minimizing the same objective over the corresponding projection. More precisely, given a convex cone $\mathcal{K}$, and having in mind the dual formulation (3.6) of the conic problem, a set X is called *$\mathcal{K}$-representable* iff there exist additional variables u and an affine mapping A such that

$$x \in X \iff \exists u \text{ such that } A(x,u) \in \mathcal{K}. \tag{5.1}$$

In other words, X is the projection of the inverse image of $\mathcal{K}$ under an appropriate affine mapping A. It follows immediately that

$$\inf\{\langle c, x\rangle\colon x \in X\} = \inf\{\langle c, x\rangle\colon \exists u \text{ such that } A(x,u) \in \mathcal{K}\}, \tag{5.2}$$

where the optimization problem on the right hand side is the convex conic problem in dual form.

For the purpose of illustration, let B be a $m \times n$ matrix, and $U = [\underline{u}, \overline{u}]$ be a n-dimensional interval vector. Then the linear image $X = BU$ can be described as

$$X = \left\{ x \in \mathbf{R}^n\colon \begin{pmatrix} u - \underline{u} \ge 0 \\ -u + \overline{u} \ge 0 \\ x - Bu \ge 0 \\ -x + Bu \ge 0 \end{pmatrix} \right\}.$$

Obviously, this set is $\mathcal{K}$-representable with the affine mapping

$$A(x,u) := \begin{pmatrix} u - \underline{u} \\ -u + \overline{u} \\ x - Bu \\ -x + Bu \end{pmatrix},$$

that maps into the positive orthant $\mathcal{K} = \mathbf{R}^n_+$. Notice that this conic representation avoids the wrapping effect and the resulting overestimation which would be introduced if interval arithmetic is used for computing BU. Obviously, closed half spaces $X = \{x\colon a^{\mathrm{T}} x \le b\}$ are $\mathcal{K}$-representable, where $\mathcal{K} = \mathbf{R}^n_+$. Moreover, it is easy to see that closed half spaces are also Lorentz cone representable, since

$$x \in X \iff A(x) \in \mathcal{L}, \text{ where } A(x) := (0; b - a^{\mathrm{T}} x).$$

Hence, the linear image $X = BU$ is also $\mathcal{L}$-representable.

Sets are frequently described as level sets of functions. Therefore, a function f is called $\mathcal{K}$-*representable* if its epigraph is $\mathcal{K}$-representable; that is, there exists an affine mapping A and additional variables u such that

$$f(x) \le t \iff \exists u \text{ such that } A(x,t,u) \in \mathcal{K}. \tag{5.3}$$

For example, affine functions $f(x) = a^{\mathrm{T}}x + b$ are $\mathcal{L}$-representable, since

$$a^{\mathrm{T}}x + b \le t \iff A(x,t) \in \mathcal{L}, \text{ where } A(x,t) := (0, t - a^{\mathrm{T}}x - b).$$

Also the squared Euclidean norm $f(x) = x^{\mathrm{T}}x$ is $\mathcal{L}$-representable. Indeed, since

$$t = \frac{(t+1)^2}{4} - \frac{(t-1)^2}{4}$$

if follows that

$$x^{\mathrm{T}}x \le t \iff x^{\mathrm{T}}x + \frac{(t-1)^2}{4} \le \frac{(t+1)^2}{4} \iff A(x,t) \in \mathcal{L},$$

where $A(x,t) := \left(x; \frac{t-1}{2}; \frac{t+1}{2}\right)$.

In order to recognize $\mathcal{K}$-representable sets and functions we proceed similarly to the case of proving continuity or differentiability: We know a number of simple continuous functions and a number of basic continuity preserving operations, and if we see that the function can be obtained from the simple functions by appropriate operations, then continuity is proved. In the following we describe briefly a simple calculus consisting of a list of operations preserving $\mathcal{K}$-representability and a list of simple $\mathcal{K}$-representable sets and functions for some cones.

It can be proved that all basic convexity preserving operations with sets and functions also preserve $\mathcal{K}$-representability for the cones used in LP, SOCP and SDP. For example polyhedral sets, finite intersections, arithmetic sums, image and inverse image of affine mappings, and direct products of $\mathcal{K}$-representable sets are $\mathcal{K}$-representable. For $\mathcal{K}$-representable functions the maximum, nonnegative linear combinations, the direct sums, the infimum, under certain assumptions the composition, support function and the conjugate function are $\mathcal{K}$-representable.

Equipped with this calculus, we have now to understand what can be expressed by special cones like $\mathbf{R}^n_+$, $\mathcal{L}$, or S^n_+. In other words, we need to know the raw material. The situation with $\mathcal{K} = \mathbf{R}^n_+$ is simple: $\mathbf{R}^n_+$-representable sets are exactly the polyhedral sets, and the $\mathbf{R}^n_+$-representable functions are finite maxima of affine functions. Elementary $\mathcal{L}$-representable functions are convex quadratic functions, fractional-quadratic functions, projective transformations of $\mathcal{L}$-representable functions, the convex increasing power function of rational degree, the even power function, concave monomials, convex monomials, p-norms and several other functions. Their level sets provide a variety of $\mathcal{L}$-representable sets.

Very important is the case $\mathcal{K} = S^n_+$. It turns out that there is a large variety of S^n_+-representable functions: every $\mathcal{L}$-representable function, the largest eigenvalue

$\lambda_{\max}(X)$ as a function of a symmetric matrix X, the spectral norm of a symmetric matrix X, the sum of the k largest eigenvalues of a symmetric matrix X, the determinant of a symmetric positive semidefinite matrix X, negative powers of the determinant $(\det(X))^{-q}$, and the sum of the k largest singular values of a rectangular matrix X. Besides the level sets of these functions there are some further interesting S_+^n-representable sets. For example the set of all nonnegative (on the entire axis, or on a given ray, or on a given segment) polynomials of a given degree is S_+^n-representable. This is also true for trigonometric polynomials which are nonnegative on a segment.

5.2. Robust optimization

Frequently, a part of the data of optimization problems, or even all data, are uncertain; that is, they are not known exactly when the problem is solved. Uncertainties are mainly due to factors like prediction errors (using future data that do not exist and hence are replaced with their forecasts), measurement errors, or approximation errors (complex phenomena are described approximately by simple models). Typically, a specific "uncertainty set" $\mathcal{U}$ in the space of data is known or can be modeled. The data uncertainty can heavily affect the quality of the nominal solution, and in these cases it is of particular importance to generate a solution, which is immunized against uncertainty. Then we have to satisfy the actual constraints, and if all we know about the data is $\mathcal{U}$, then the only way is to restrict to *robust feasible solutions*: these are solutions which satisfy all possible realizations of the uncertain constraints. Hence, robustly feasible solutions remain feasible at the expense of conservatism. The *robust counterpart* of an optimization problem is to optimize the worst-case value of the objective among all robust solutions.

Although the notion of robustness is rather new in mathematical programming and should not be confused with sensitivity analysis, it is quite classical in control theory. Robust optimization models in mathematical programming have received much attention, see [5], [6] and [19]. In the case of linear programming

$$\text{minimize } c^{\mathrm{T}}x \quad \text{s.t. } Ax \le b \tag{5.4}$$

the robust counterpart has the form

$$\text{minimize } t \quad \text{s.t. } c^{\mathrm{T}}x \le t,\ Ax \le b,\ \forall (c, A, b) \in \mathcal{U}. \tag{5.5}$$

While improving significantly the reliability of the decision, the disadvantage is that the robust counterpart is a semi-infinite problem, i.e., a problem with infinitely many linear constraints. Fortunately, for several uncertainty sets this robust version is computational tractable. If, for instance, we have uncertainty sets for the rows of A and the objective in the form of ellipsoids, then the robust counterpart is

$$\begin{aligned}\text{minimize } t \quad \text{s.t. } & c^{\mathrm{T}}x \le t,\ a_i^{\mathrm{T}}x \le b_i,\ i = 1, \dots, m,\\ & \forall a_i \in \mathcal{U}_i = \{\hat{a}_i + P_i w\colon \|w\|_2 \le 1 \text{ and } P_i \succeq 0\},\\ & c \in \mathcal{U}_c = \{\hat{c}_i + P_c w\colon \|w\|_2 \le 1 \text{ and } P_c \succeq 0\},\end{aligned}$$

where $\mathcal{U}$ is the Cartesian product of the uncertainty set $\mathcal{U}_i$ and $\mathcal{U}_c$. Note, that the inequality

$$\max_{a_i \in \mathcal{U}_i} a_i^{\mathrm{T}} x = \max_{\|w\|_2 \leq 1} \hat{a}_i^{\mathrm{T}} x + w^{\mathrm{T}} P_i x = \hat{a}_i^{\mathrm{T}} x + \|P_i x\|_2 \leq b_i,$$

yields the robust formulation as the tractable SOCP problem

$$\min_{x \in \mathbf{R}^n} c^{\mathrm{T}} x \quad \text{s.t. } \bar{a}_i^{\mathrm{T}} x + \|P_i x\|_2 \leq b_i,\ i = 1, \ldots, m.$$

The robust counterpart depends on the structure of the uncertainty set and may be much harder to solve than the original problem. In Table 5.1, the robust counterparts of LP and SOCP are displayed for some uncertainty sets. Hence, in several cases the robust form of a conic problem is a conic problem that can be solved rigorously with the previous methods.

Table 5.1. Robust counterparts

Uncertainty	Problem	Robust optimization
polytopic	LP	LP
ellipsoid		SOCP
LMI		SDP
polytopic	SOCP	SOCP
ellipsoid		SDP
LMI		NP-hard

5.3. Combinatorial optimization

Linear and semidefinite programs play a very useful role in global and combinatorial optimization. Several methods are known for constructing linear or semidefinite relaxations, which are used in branch-bound-and-cut algorithms to eliminate regions that do not contain global minimizers. Neumaier and Shcherbina [53] have pointed out that backward error analysis has no relevance for combinatorial programs, since slightly perturbed coefficients no longer produce problems of the same class. There, one can also find an innocent-looking linear integer problem for which the commercial high quality solver CPLEX [25] and several other state-of-the-art solvers fail. The reason is that the relaxations are not solved with sufficient accuracy and global minimizers are truncated. Hence, in order to obtain safe results, it is important to have reliable, good and cheaply computable lower bounds on the optimal value for these relaxations.

Various problems like max-cut, partitioning, coloring and many others can be formulated as linear integer problems, where the vector of decision variables $x \in \{-1, 1\}^n$. Sometimes, these relaxations are even ill-posed. An example is graph partitioning problems with many applications such as VLSI design. Relaxing the integer conditions to $x \in [-1, 1]^n$ yields a linear relaxation, which can be solved rigorously with the previous methods. Improved tight semidefinite relaxations are

obtained by lifting the vector x into the space of semidefinite matrices with the operation

$$X = xx^{\mathrm{T}}. \tag{5.6}$$

It follows immediately that

$$X \succeq 0, \quad \mathrm{diag}(X) = e, \quad \text{and} \quad \mathrm{rank}(X) = 1, \tag{5.7}$$

where e is the vector of ones. Dropping the non-convex condition $\mathrm{rank}(X) = 1$ we obtain a semidefinite relaxation. Laurent and Poljak [41] have shown that for this type of relaxation $-1 \leq X_{ij} \leq 1$, and if $X_{ij} \in \{-1, 1\}$ then $X = xx^{\mathrm{T}}$ where $x \in \{-1, 1\}^n$. This property establishes the tightness of these relaxations. Moreover, it follows that the primal boundedness qualification is fulfilled in the way that an optimal solution exists with $\lambda_{\max}(X) \leq n$, and thus a rigorous lower bound for the optimal value can be computed.

6. Software and numerical results

In this section some software packages for computing approximate optimal solutions and for computing verified results are listed.

6.1. Approximate conic solvers

There are several software packages solving approximately special conic problems. The first group is that of primal-dual interior point methods which use second order derivative information. The codes SeDuMi [68], [69] and SDPT3 [72] can handle handle all 3 types LP, SOCP and SDP, whereas the codes SDPA [76], CSDP [11] and DSDP [8] are limited to SDP. MOSEK [3] is a code suited only for SOCP problems. The codes BMPR [15], BMZ [16], BUNDLE [22], [24] are suited for very large-scale SDP problems which do not make use of second order derivative information. All these computer codes were submitted to the Seventh DIMACS Implementation Challenge on semidefinite and related optimization problems. The codes were run on a standard platform and on all the benchmark problems provided by the organizers of the challenge. Mittelmann [43] has described the benchmarking results. In summary it can be said that all these codes proved to be valuable in their own right.

6.2. Rigorous conic solvers

There are three software packages, Lurupa [38], VSDP and verifiedSDP [17], for rigorously solving special conic problems. Lurupa and verifiedSDP are C++ implementations of the presented rigorous bounds for the special case of linear programming and semidefinite programming, respectively. They will be available soon. Detailed numerical results of Lurupa for the NETLIB suite of linear programming problems [49], a well-known collection of difficult to solve problems with up to 15695 variables and 16675 constraints originating from various applications, are presented in [39]. In summary, with exception of two problems a rigorous finite lower bound

(upper bound) of the optimal value could be computed iff the distance to dual infeasibility (primal infeasibility) is greater than zero. The median guaranteed accuracy was about 10^{-8}, which is almost equal to the approximate accuracy of the LP solver used. In other words, the overestimation for this test set is negligible.

VSDP [30] is a MATLAB software package for computing verified results of semidefinite programming problems. VSDP is completely written in MATLAB and uses the MATLAB-toolbox INTLAB [63]. This package computes verified lower and upper bounds on the optimal value for semidefinite programs, proves existence of feasible and optimal solutions, also for LMI's, provides rigorous certificates of infeasibility, facilitates solving the problem approximately by using the solvers SDPT3 and SDPA, and can handle full and sparse formats, as well as interval data. Now, some routines of VSDP can also be used under YALMIP [42], a toolbox for modelling and optimization in MATLAB.

Detailed numerical results of VSDP for the SDPLIB benchmark problems of Borchers [12] (a library of problems up to thousands of constraints and millions of variables) can be found in [29] and [32]. Freund, Ordóñez and Toh [20] pointed out that 32 problems in the SDPLIB are ill-posed. VSDP could compute for all problems a rigorous lower bound of the optimal value and could verify the existence of strictly dual feasible solutions, which proves that all problems have a zero duality gap. A finite rigorous upper bound could be computed for all well-posed problems with one exception; this is `hinf2`. For all 32 ill-posed problems VSDP has computed the upper bound $\overline{f}_d = +\infty$, which expresses exactly the zero distance to primal infeasibility. SDPT3 (with default values) has given 7 warnings, and 2 warnings were given for well-posed problems. Hence, no warnings were given for 27 ill-posed problems with zero distance to primal infeasibility. In other words, there is no correlation between warnings and the difficulty of the problem. At least for this test set our rigorous bounds reflect the difficulty of the problems much better, and they provide safety, especially in the case where algorithms subsequently call other algorithms, as is done for example in branch-and-bound methods.

6.3. Other rigorous solvers

There are some other software packages for computing rigorous results of global optimization problems. COSY [9], GlobSol [33], and Numerica [73] are probably the most widely known ones. These solvers can handle problems where the objective and the constraints are defined by smooth nonlinear algebraic expressions. These solvers have in common that the computations are done by enclosing all numbers in intervals and working with interval arithmetic. The current versions seem not make use of convex relaxations. A consequence is that these solvers are time-consuming and can be applied only to problems of small size. Elaborate comparisons with these packages and some others can be found in the forthcoming paper of Keil [37].

7. Conclusions

The computation of rigorous error bounds for conic optimization problems can be viewed as a careful postprocessing tool that uses only approximate solutions computed by any conic solver. The numerical results show that such rigorous error bounds can be computed even for problems of large size.

The computational costs for computing verified lower and upper bounds on the optimal value are negligible compared with the costs for computing the approximate solutions, provided certain boundedness qualifications are fulfilled. Then safety comes almost for free. If the conic solver produces suitable approximations, then usually the bounds for the optimal value are quite accurate, while in other cases, lower and upper bounds differ and the user receives a warning that something went wrong or needs special attention. This warning (see the numerical experiments with the SDPLIB) seems to be very reliable in comparison to the warnings given by conic solvers.

We end this survey by mentioning at least two challenges for the near future that would improve the state of the art significantly. Firstly, it is not known and it is not straightforward how to compute efficiently verified error bounds for the optimal solution in the case of nonpolyhedral cones. The techniques used by Krawzcyk essentially require that the cone is defined as a finite set of inequalities, like the positive orthant. A second challenge are rigorous bounds for the large variety of special infinite dimensional cones and infinite dimensional linear systems that appear in applications.

References

[1] G. Alefeld and J. Herzberger, Introduction to Interval Computations. Academic Press, New York, 1983.
[2] F. Alizadeh and D. Glodfarb, Second-order cone programming. Math. Program., **95** (2003), 3–51.
[3] E.D. Andersen, C. Roos and T. Terlaky, A primal-dual interior-point method for conic quadratic optimization. Math. Programming, **95** (2003), 249–277.
[4] H. Beeck, Linear programming with inexact data. Technical Report 7830, Abteilung Mathematik, TU München, 1978.
[5] A. Ben-Tal, L. El Ghaoui and A. Nemirovski, Robust semidefinite programming. Handbook of Semidefinite Programming, H. Wolkowicz, R. Saigal and L. Vandenberghe (eds.), Kluwer Academic Publishers, 2000.
[6] A. Ben-Tal and A. Nemirovski, Robust convex optimization. Math. Operations Res., **23** (1998), 769–805.
[7] A. Ben-Tal and A. Nemirovski, Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications. MPS-SIAM Series on Optimization, SIAM, Philadelphia, PA, 2001.
[8] S.J. Benson and Y. Ye, DSDP3: Dual scaling algorithm for general positive semidefinite programming. Technical Report Preprint ANL/MCS-P851-1000, Argonne National Labs, 2001.
[9] M. Berz et al., COSY Infinity. `http://www.bt.pa.msu.edu/index_files/cosy.htm`.
[10] G.D. Birkhoff, Lattice Theory, revised edition. Am. Math. Soc. Colloquium Publications, Vol. 25, Am. Math. Soc., New York, 1948.
[11] B. Borchers, CSDP, A C library for semidefinite programming. Optimization Methods and Software, **11** (1999), 613–623.

[12] B. Borchers, SDPLIB 1.2, a library of semidefinite programming test problems. Optimization Methods and Software, **11** (1999), 683–690.
[13] N. Bourbaki, Éléments de mathématique. XIII. 1 part: Les structures fondamentales de l'analyse, Livre VI: Intégration, Actualités scientifique et industrielles, 1952.
[14] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2004.
[15] S. Burer and R.D.C. Monteiro, A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Programming, **95** (2003), 329–357.
[16] S. Burer, R.D.C. Monteiro and Y. Zhang, Solving a class of semidefinite programs via nonlinear programming. Math. Programming, **93** (2002), 97–122.
[17] D. Chaykin, Verified Semidefinite Programming: Applications and the Software Package verifiedSDP. Ph.D. thesis, Technische Universität Hamburg-Harburg, 2009.
[18] E. De Klerk, Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications. Dordrecht: Kluwer Academic Publishers, 2002.
[19] L. El Ghaoui, F. Oustry and H. Lebret, Robust Solutions to Uncertain Semidefinite Programs. SIAM J. Optim., **9** (1998), 33–52.
[20] R.M. Freund, F. Ordóñez and K. Toh, Behavioral measures and their correlation with IPM iteration counts on semi-definite programming problems. Math. Programming, **109** (2007), 445–475.
[21] E.R. Hansen, Global Optimization Using Interval Analysis. Marcel Dekker, New York, 1992.
[22] C. Helmberg, SBmethoda C++ implementation of the spectral bundle method. Technical Report, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000, Manual to Version 1.1, ZIB-Report ZR 00-35, `http://www.mathematik.uni-kl.de/helmberg/SBmethod/`.
[23] C. Helmberg, Semidefinite programming for combinatorial optimization (Habilitationsschrift). Technical Report ZIB ZR-00-34, Konrad-Zuse-Zentrum Berlin, TU Berlin, 2000.
[24] C. Helmberg and K.C. Kiwiel, A spectral bundle method with bounds. Math. Programming, **93** (2002), 173–194.
[25] ILOG CPLEX 7.1, User's Manual. ILOG, France, 2001.
[26] C. Jansson, A self-validating method for solving linear programming problems with interval input data, Computing Suppl., **6** (1988), 33–45.
[27] C. Jansson, A rigorous lower bound for the optimal value of convex optimization problems. J. Global Optimization, **28** (2004), 121–137.
[28] C. Jansson, Rigorous lower and upper bounds in linear programming. SIAM J. Optimization (SIOPT), **14** (2004), 914–935.
[29] C. Jansson, VSDP: A MATLAB software package for verified semidefinite programming. NOLTA, 2006, 327–330.
[30] C. Jansson, VSDP: Verified Semidefinite Programming, User's Guide. 2006, `http://www.BetaVersion0.1.optimization-online.org/DB_HTML/2006/12/1547.html`.
[31] C. Jansson, Guaranteed accuracy for conic programming problems in vector lattices. 2007, arXiv:0707.4366v1, `http://arxiv.org/abs/0707.4366v1`.
[32] C. Jansson, D. Chaykin and C. Keil, Rigorous error bounds for the optimal value in semidefinite programming. SIAM Journal on Numerical Analysis, **46** (2007), 180–200, `http://link.aip.org/link/?SNA/46/180/1`.
[33] R.B. Kearfott, GlobSol. `http://interval.louisiana.edu`.
[34] R.B. Kearfott, On proving existence of feasible points in equality constrained optimization problems. Math. Program., **83** (1998), 89–100.
[35] R.B. Kearfott, On proving existence of feasible points in equality constrained optimization problems. Preprint, Department of Mathematics, Univ. of Southwestern Louisiana, U.S.L. Box 4-1010, Lafayette, La 70504, 1994.
[36] R.B. Kearfott, Rigorous Global Search: Continuous Problems. Kluwer Academic Publisher, Dordrecht, 1996.
[37] C. Keil, Verified linear programming—a comparison. Submitted, 2008, `http://www.optimization-online.org/DB_HTML/2008/06/2007.html`.
[38] C. Keil, Lurupa—rigorous error bounds in linear programming. Algebraic and Numerical Algorithms and Computer-Assisted Proofs, B. Buchberger, S. Oishi, M. Plum and S.M. Rump (eds.), Dagstuhl Seminar Proceedings, No. 05391, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006, `http://drops.dagstuhl.de/opus/volltexte/2006/445`.

[39] C. Keil and C. Jansson, Computational experience with rigorous error bounds for the Netlib linear programming library. Reliable Computing, **12** (2006), 303–321, http://www.optimization-online.org/DB_HTML/2004/12/1018.html.
[40] R. Krawczyk, Fehlerabschätzung bei linearer Optimierung, Interval Mathematics, K. Nickel (ed.), Lecture Notes in Computer Science, Vol. 29, Springer-Verlag, Berlin, 1975, 215–222.
[41] M. Laurent and S. Poljak, On a positive semidefinite relaxation of the cut polytope. Linear Algebra and Its Applications (LAA), **223/224** (1995), 439–461.
[42] J. Löfberg, YALMIP: A toolbox for modeling and optimization in MATLAB. Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.
[43] H.D. Mittelmann, An independent benchmarking of SDP and SOCP solvers. Math. Programming Ser. B, **95** (2003), 407–430.
[44] R.E. Moore, Methods and Applications of Interval Analysis. SIAM, Philadelphia, 1979.
[45] A. Nemirovskii, Lectures on Modern Convex Optimization. 2003.
[46] Y. Nesterov, Long-step strategies in interior-point primal-dual methods. Math. Programming, **76** (1997), 47–94.
[47] Y. Nesterov and A. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming. SIAM, Philadelphia, 1994.
[48] Y.E. Nesterov and M.J. Todd, Self-scaled barriers and interior-point methods for convex programming. Math. Oper. Res., **22** (1997), 1–42.
[49] NETLIB Linear Programming Library. http://www.netlib.org/lp.
[50] A. Neumaier, Interval Methods for Systems of Equations. Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 1990.
[51] A. Neumaier, Introduction to Numerical Analysis. Cambridge University Press, 2001.
[52] A. Neumaier, Complete search in continuous global optimization and constraint satisfaction. Acta Numerica, Vol. 13, A. Iserles (eds.), Cambridge University Press, 2004, 271–369.
[53] A. Neumaier and O. Shcherbina, Safe bounds in linear and mixed-integer programming. Mathematical Programming, Ser. A, **99** (2004), 283–296.
[54] J. von Neumann and H.H. Goldstine, Numerical inverting of matrices of high order. Bull. Amer. Math. Soc., **53** (1947), 1021–1099.
[55] S. Oishi and S.M. Rump, Fast verification of solutions of matrix equations. Numer. Math., **90** (2002), 755–773.
[56] S. Oishi, K. Tanabe, T. Ogita and S.M. Rump, Convergence of Rump's method for inverting arbitrarily ill-conditioned matrices. J. Comput. Appl. Math., **205** (2007), 533–544.
[57] F. Ordóñez and R.M. Freund, Computational experience and the explanatory value of condition measures for linear optimization. SIAM J. Optimization (SIOPT), **14** (2003), 307–333.
[58] A.L. Peressini, Ordered Topological Vector Spaces. Harper and Row, 1967.
[59] J. Renegar, Some perturbation theory for linear programming. Mathematical Programming, **65** (1994), 79–91.
[60] J. Renegar, Linear programming, complexity theory, and elementary functional analysis. Mathematical Programming, **70** (1995), 279–351, citeseer.ist.psu.edu/renegar95linear.html.
[61] S.M. Rump, Solving algebraic problems with high accuracy (Habilitationsschrift), A New Approach to Scientific Computation, U.W. Kulisch and W.L. Miranker (eds.), Academic Press, New York, 1983, 51–120.
[62] S.M. Rump, Validated solution of large linear systems. Validation Numerics: Theory and Applications, R. Albrecht, G. Alefeld and H.J. Stetter (eds.), Computing Supplementum, Vol. 9, Springer, 1993, 191–212.
[63] S.M. Rump, INTLAB—interval laboratory, a Matlab toolbox for verified computations, Version 5.1, 2005.
[64] S.M. Rump, Error bounds for extremely ill-conditioned problems. Proceedings of 2006 International Symposium on Nonlinear Theory and Its Applications, Bologna, Italy, September 11–14, 2006.
[65] S.M. Rump, INTLAB—interval laboratory, the Matlab toolbox for verified computations, Version 5.3, 2006.
[66] S.M. Rump and T. Ogita, Super-fast validated solution of linear systems. Special issue on scientific computing, computer arithmetic, and validated numerics (SCAN 2004), Journal of Computational and Applied Mathematics (JCAM), **199** (2006), 199–206.
[67] H.H. Schaefer, Banach lattices and positive operators. Springer, 1974.

[68] J.F. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software, **11** (1999), 625–653.
[69] J.F. Sturm, Central region method. High Performance Optimization, J.B.G. Frenk, C. Roos, T. Terlaky and S. Zhang (eds.), Kluwer Academic Publishers, 2000, 157–194.
[70] L. Tuncel, Generalization of primaldual interior-point methods to convex optimization problems in conic form. Found. Comput. Math., **1** (2001), 229–254.
[71] A.M. Turing, Rounding-off errors in matrix processes. Quarterly J. of Mechanics & App. Maths., **1** (1948), 287–308.
[72] R.H. Tütüncü, K.C. Toh and M.J. Todd, Solving semidefinite-quadratic-linear programs using SDPT3. Math. Program., **95** (2003), 189–217.
[73] P. Van Hentenryck, P. Michel and Y. Deville, Numerica: A Modelling Language for Global Optimization. MIT Press Cambridge, 1997.
[74] H. Wolkowicz, Semidefinite and Cone Programming Bibliography, Comments. `http://orion.uwaterloo.ca/~hwolkowi/henry/book/fronthandbk.d/sdpbibliog.pdf`.
[75] H. Wolkowicz, R. Saigal and L. Vandenberghe (eds.), Handbook of Semidefinite Programming. International Series in Operations Research and Management Science, Vol. 27, Kluwer Academic Publishers, Boston, MA, 2000.
[76] M. Yamashita, K. Fujisawa and M. Kojima, Implementation and evaluation of SDPA 6.0. Optimization Methods and Software, **18** (2003), 491–505.