Invited Paper

Verification methods for conic linear programming problems

Marko Lange\textsuperscript{1a)}

\textsuperscript{1} Institute of Reliable Computing, Hamburg University of Technology
Am Schwarzenberg-Campus 3, Hamburg 21073, Germany

\textsuperscript{a)} m.lange@tuhh.de

Received October 14, 2019; Revised March 23, 2020; Published July 1, 2020

Abstract: This paper gives an overview of verification methods for finite dimensional conic linear programming problems. Besides the computation of verified tight enclosures for unique non-degenerate solutions to well-posed conic linear programming instances, we discuss a rigorous treatment of problems with multiple or degenerate solutions. It will be further shown how a priori knowledge about certain boundedness qualifications can be exploited to efficiently compute verified bounds for the optimal objective value. The corresponding approach is applicable even to ill-posed programming problems. Examples from linear and semidefinite programming are used to illustrate the respective approaches and give further explanations. Another topic is the treatment of programming problems whose parameters are subject to uncertainties. Rough but inclusive estimates for the variability of the corresponding programming problems are given, and also a best and worst case analysis is taken into account. At the end of this paper, special consideration is given to typical issues when applying interval arithmetic in the context of conic linear programming. Different ways are shown on how to resolve these issues.

Key Words: verification, linear programming, cone programming, interval uncertainties

1. Introduction

In this note we give a survey of verification methods for conic linear programming problems in finite dimensional real space. Conic linear programming (CLP) problems are specified by a linear objective function, linear equality constraints and a convex cone restriction on the decision variables. Its most used special cases are linear programming, convex quadratic programming, second order cone programming, semidefinite programming, and exponential cone programming. All these classes have in common that there exist efficient methods to solve the respective problem instances. Simplex methods, sequential quadratic programming, interior-point methods, and augmented Lagrangian methods with variants such as the alternating direction method of multipliers are just some of the methods designed for solving constrained optimization problems efficiently.

The convexity of the conic linear programming problem, i.e., convexity of the objective function, the constraints and the cone condition, is the central property that makes CLP so attractive for research and in practice. By this property the optimal set is convex and every local minimum is also a global optimum. Any local optimization strategy can be used to tackle the CLP problem globally. This is
the reason why many classes of CLP admit polynomial-time algorithms. Nevertheless, there are also interesting and practically relevant cones, such as the cone of completely positive matrices, for which it is known that already the membership question is NP-hard [1].

The backbone of the verification methods presented in the following sections is an arithmetic with a set Ω of elementary operations over convex sets which satisfies the following fundamental property:

**Inclusion Principle**

Let a system of arithmetic expressions $F: \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a convex set $x \subset \mathbb{R}^n$ be given, and suppose that $F(x)$ is evaluated by replacing every operation with its counterpart in $\Omega$. Then, $F(x) \subset \mathbb{R}^m$ is a convex set and

$$\forall x \in x: \quad F(x) \in F(x).$$

A particular choice for such an arithmetic is *interval arithmetic* (IA) where the convex sets are restricted to be interval quantities and the elementary operations are interval operations. The use of interval arithmetic as a key tool in many verification methods is justified by its simple and fast implementation.

The set of real intervals may be defined as

$$\mathbb{I}\mathbb{R} := \{ [\bar{x}, \bar{x}] : \bar{x} \preceq_{\mathbb{R}^+} \bar{x} \},$$

where $\preceq_{\mathbb{R}^+}$ denotes the partial ordering induced by the cone of nonnegative real numbers, i.e.,

$$\bar{x} \preceq_{\mathbb{R}^+} \bar{x} \iff \bar{x} - \bar{x} \in \mathbb{R}^+.$$

By replacing $\mathbb{R}^+$ with some convex cone $K$, the interval arithmetic can be generalized to a *cone arithmetic*. In comparison to interval operations, depending on the choice of the cone $K$ the implementation of the respective operations in $\Omega$ may become arbitrarily difficult. On the other hand, it is typical beneficial to apply a cone arithmetic that exploits the conic structure of the corresponding programming problem. In semidefinite programming, for instance, it is preferable if uncertainties in the parameter matrices $A_i$ are modeled via

$$A_i := \{ A_i : \bar{A}_i \preceq_{S^+} A_i \preceq_{S^+} \bar{A}_i \},$$

where $S^+$ denotes the set of symmetric positive semidefinite matrices with appropriate size. If the operations in $\Omega$ exploit this property, significantly tighter inclusions than with pure interval arithmetic are possible.

In the following, quantities with uncertainties are in bold face. For various cones, it is straightforward to generalize interval uncertainties to adequate conic uncertainties as above. Nevertheless, for reasons of simplicity, all subsequent quantities in bold face are assumed to be interval quantities.

For a better understanding of the verification approaches in the following sections, some basic knowledge of interval arithmetic is beneficial. An element of $\mathbb{I}\mathbb{R}$ is typically represented either in infimum-supremum or midpoint-radius notation:

$$x = [x, \bar{x}] = [x_m - x_r, x_m + x_r] = (x_m)[x_r].$$

Any $x \in x$, i.e., any $x \in \mathbb{R}$ satisfying $x \leq x \leq \bar{x}$ or $|x - x_m| \leq x_r$, is called a realization of $x$. In particular this term is used in the context of interval programming problems. The notation for real intervals $x \in \mathbb{I}\mathbb{R}$ extends naturally to other interval quantities such as interval vectors $v \in \mathbb{I}\mathbb{R}^n$ or interval matrices $A \in \mathbb{I}\mathbb{R}^{m \times n}$. The most important property of interval arithmetic is that it satisfies the inclusion principle. This and many more aspects of interval arithmetic can be found in [2–4] and the references therein.

We try to follow the standardized notation as much as possible; however, to avoid confusion with inner products, we use a different notation for the midpoint-radius representation, see above. Whenever a condition or inequality involves quantities with interval (or conic) uncertainties, this is meant in regard to all possible realizations. For example,
\[ x^T y > 0 \iff \forall x \in \mathbf{x}, y \in \mathbf{y}: x^T y > 0. \]

Any other necessary notation will be introduced at the time of usage.

Before discussing the more general class of conic optimization, we will first introduce the basic ideas by means of verification approaches for linear programming (LP) problems. Later on special consideration is also given to the semidefinite programming problem and its interval version.

2. The linear programming problem

The primal standard form of a linear programming problem is

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \\
& \quad x \geq 0,
\end{align*}
\]

where \( x \in \mathbb{R}^n \) is the vector of nonnegative decision variables, \( A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m \) and \( c \in \mathbb{R}^n \). The inequality \( x \geq 0 \) is understood entrywise.

Its Lagrangian function has the form

\[ L(x, y) = c^T x + (b - Ax)^T y. \]  

And the Lagrangian dual problem is

\[ \sup_{y \in \mathbb{R}^m} \inf_{x \in \mathbb{R}^n_+} L(x, y), \]

by which we obtain the following standard form:

\[
\begin{align*}
\text{maximize} & \quad b^T y \\
\text{subject to} & \quad c - A^T y \geq 0, \quad y \in \mathbb{R}^m
\end{align*}
\]

where \( y \in \mathbb{R}^m \) denotes the vector of the free Lagrangian parameters. They are the dual decision variables.

Let \( x \in \mathbb{R}^n \) and \( y \in \mathbb{R}^m \) be primal and dual feasible vectors, respectively. Then

\[ c^T x = c^T x + (b - Ax)^T y = x^T (c - A^T y) + b^T y \geq b^T y. \]  

This relation is called weak duality. If, and only if, the complementary slackness condition

\[ \forall j \in \{1, \ldots, n\}: (c - A^T y)_j \cdot x_j = 0 \]  

is satisfied, \( c^T x = b^T y \) holds with equality. If this is true for a pair of primal and dual optimal vectors, we say that the problem satisfies strong duality. Indeed, it is well-known that strong duality is always satisfied for linear programming problems.

A possible approach to solve (LP) is to numerically solve the corresponding Karush-Kuhn-Tucker (KKT) system of equations and inequalities. The KKT conditions to (LP) are

\[
\begin{align*}
& b - Ax = 0 \\
& x \geq 0 \\
& c - A^T y \geq 0 \\
& (c - A^T y)_j \cdot x_j = 0 \quad \text{for} \quad j = 1, \ldots, n,
\end{align*}
\]

where the stationary condition of KKT is implied and not given separately because we have not introduced auxiliary variables for the nonnegative KKT multipliers. Due to the convexity of linear programming problems, any pair \((x^*, y^*)\) satisfying the KKT conditions above yields an optimal solution \(x^*\) to the corresponding (LP) instance.

The KKT conditions are rarely used directly for solving the respective optimization problem. On the other hand, many barrier methods, including the widely used interior-point methods, solve a sequence of problems via their KKT systems converging to the problem of interest.
2.1 Verified enclosures for unique solutions

Many early verification approaches for (LP) are built upon methods to verify solutions to the respective KKT system. Let \( \hat{x} \) and \( \hat{y} \) be approximate solutions to the primal and dual linear programming problem, respectively. Then (\( \hat{x}, \hat{y} \)) also approximates a point satisfying the KKT conditions and, in particular, a solution to the nonlinear system

\[
F(x, y) = \begin{pmatrix} b - Ax \\ \text{diag}(c - ATy) \end{pmatrix} x = 0,
\]

where \( \text{diag}(c - ATy) \) denotes a diagonal matrix with the elements of the vector \( c - ATy \) on its diagonal. The nonlinear system consists of the primal equality constraints in (LP) and the complementary slackness condition (4).

If the Jacobian matrix \( J(x, y) \) to the nonlinear system (5) is regular around the approximate point (\( \hat{x}, \hat{y} \)), then we may use the Krawczyk operator [5] to compute an interval inclusion (\( x, y \)) for an actual solution (\( \hat{x}, \hat{y} \)) to (5). For more information and references on interval solutions to nonlinear systems, see also [3, 6, 7]. By the complementary slackness condition, at least one of the entries \( (c - AT\hat{y})_j \) or \( \hat{x}_j \) has to be zero for every feasible index \( j \). Naturally, the corresponding interval quantities contain not only zeroes but also negative values. Thus, the inequality conditions of the KKT system are not satisfied for every realization of (\( x, y \)).

The good news is that this is actually not necessary; there is a weaker sufficient condition. If either \( (c - AT\hat{y})_j > 0 \) or \( x_j > 0 \), then the complementary entry has to be zero. To prove the existence of a solution (\( \hat{x}, \hat{y} \)) of the KKT conditions, it is therefore adequate to show the positivity of one factor in each complementary slackness equality.

A sufficient condition for the positivity of one of the complementary factors is non-degeneracy of primal and dual solution, i.e., if \( \hat{x} \) has exactly \( m \) positive entries and \( c - AT\hat{y} \) has exactly \( m \) zeros (thus \( n - m \) positive entries). In this particular case the primal and dual solution vectors are unique.

In the presence of a good primal approximate \( \hat{x} \) or dual approximate \( \hat{y} \), it is typically not necessary to compute interval inclusions for the nonlinear system (5). To show this, let an approximate primal solution \( \hat{x} \) be given and assume that \( \hat{x} \) has \( m \) significant entries and \( n - m \) entries equal to or close to zero. The index set that consists of the \( m \) indices corresponding to the significant entries of \( \hat{x} \) is called basis. We shall denote it by \( B \) and its complement consisting of the remaining \( n - m \) indices by \( N \). If the index set \( B \) derived from our approximate solution \( \hat{x} \) is also the basis to an actual solution, then we can compute tight interval enclosures for \( \hat{x} \) and \( \hat{y} \), respectively, using the following approach.

Let \( c_B \) denote the vector of length \( m \) that consists of the entries of \( c \) indexed by \( B \). Accordingly, \( A_B \) denotes the \( m \times m \) matrix containing the corresponding columns of \( A \). The entities \( c_F, A_N \) as well as \( x_B, x_N \) shall be defined similarly. The index set \( B \) is the basis to a linear programming problem with primal and dual non-degenerate solution, if, and only if,

- the primal solution \( \hat{x} \) satisfies
  \[
  A_B \hat{x}_B = b, \quad \hat{x}_B > 0, \quad \hat{x}_N = 0,
  \] (6)

- the dual solution \( \hat{y} \) satisfies
  \[
  A^T_B \hat{y} = c_B, \quad A^T_N \hat{y} < c_N,
  \] (7)

both hold true. The solutions are unique and the strict inequalities can be satisfied only in the non-degenerate case.

Assume that \( A_B \) is regular. It is then possible to compute an interval enclosure \( x \) for \( \hat{x} \) by solving the system of linear equations in (6). Similarly, we can compute a verified enclosure \( y \) for \( \hat{y} \) using the linear system in (7). Efficient verification methods for solving square linear systems are discussed for instance in [8–10]. More sophisticated verification methods are available if \( A_B \) has a specific structure which occurs in certain structural programming problems. For instance, if \( A_B \) is an \( H \)-matrix, we can apply the method discussed in [11]. Programming problems of even larger size may be tackled.
using an interval block Gaussian algorithm together with one of the referenced methods. Moreover, it is possible to apply residual iterations to increase the precision of the computed interval enclosures.

Let \( x_B \) and \( y \) be enclosures for solutions to the linear systems in (6) and (7), respectively, and set \( x_N = 0 \). If now
\[
\forall x \in x: \quad x_B > 0 \quad \text{and} \quad \forall y \in y: \quad A_N^T y < c_N,
\]
then we verified existence of a unique, non-degenerate primal-dual solution \((\hat{x}, \hat{y})\) within the interval pair \((x, y)\). The optimal objective value\(^1\) \( \hat{\rho} \) is bounded by
\[
\inf c^T x \leq \hat{\rho} \leq \sup c^T x
\]
as well as
\[
\inf b^T y \leq \hat{\rho} \leq \sup b^T y,
\]
where \( \inf c^T x \) abbreviates \( \inf \{ c^T x \mid x \in x \} \) and the related terms are understood likewise. These enclosures can be very tight.

2.2 Linear programming problems with degenerate solution sets

Although non-degeneracy is a generic property of linear programming problems, c.f. [12], in practice, degeneracy occurs frequently. A typical cause for degeneracy is the introduction of redundant constraints in the modeling process. The previous approach, which uses the KKT conditions or (6) and (7) together with a (guessed) basis \( B \), will not work for this set of problems.

It is worth noticing that even in the presence of a degenerate solution, we may still find a primal-dual solution pair \((\hat{x}, \hat{y})\) so that one of the factors \((c - A^T \hat{y})_j\) or \( x_j \) is strictly positive for all complementary slackness conditions. For example, assume that the primal problem (LP) has exactly two optimal edges with bases \( B_1 \) and \( B_2 \), respectively. The convex combination of the corresponding optimal points \( \hat{x}_1 \) and \( \hat{x}_2 \), i.e., the line segment between these points, defines the primal solution set. The bases \( B_1 \) and \( B_2 \) are identical in all but one index. If \( \hat{x}_1 \) and \( \hat{x}_2 \) are non-degenerate solutions, then all other points in the solution set have \( m + 1 \) strictly positive entries with indices \( B_1 \cup B_2 \). Hence, a corresponding dual solution can have at most \( n - m - 1 \) nonzeros in \( c - A^T \hat{y} \). The dual problem has a degenerate solution, but we may still find a solution pair \((\hat{x}, \hat{y})\) with \( m + 1 \) nonzeros in \( \hat{x} \) and \( n - m + 1 \) nonzeros in \( c - A^T \hat{y} \). The main obstacle in this example is the computation of a vector \( \hat{y} \) satisfying \( A_{B_1 \cup B_2}^T \hat{y} = c_{B_1 \cup B_2} \). Without additional knowledge, either about the dimension of the solution space or more explicitly about the optimal bases \( B \), and therefore the consistency of \( A_{B_1 \cup B_2}^T \hat{y} = c_{B_1 \cup B_2} \), computing a solution to the respective overdetermined linear system is an ill-posed problem. It lies outside of the scope of verification methods that rely on approximate floating-point operations.

The ill-posedness just discussed above is not a shortcoming of the described verification method, it is inherited from the original problem of computing a tight inclusion for a solution to a linear programming problem with multiple or degenerate solutions. If a given instance of (LP) has multiple solutions, then infinitesimal small perturbations can change this circumstance and make one of the optimal points the unique solution to the perturbed problem. On the other hand, even if primal and dual solution are unique, as long as one of them is a degenerate solution, we still have to be able to solve the corresponding overdetermined linear system. Slightest perturbations may change the consistency of the problem. In both cases the optimal solution (set) changes discontinuously with the input data. The problem is ill-posed in the sense of Hadamard [13].

Deciding whether a given instance of (LP) has multiple or degenerate solutions is an ill-posed problem.

In light of this, also the problem of computing tight enclosures for an actual solution to a linear programming problem with multiple or degenerate solutions is ill-posed. Thus we cannot hope to

---

\(^1\) By strong duality of linear programming problems, primal and dual optimal objective value are identical.
derive the same nice inclusions as in the previous subsection. The good news is that, in contrast to the (set of) optimal solution vector(s), the optimal objective value typically changes continuously with the input data even if (LP) or its dual (DLP) have multiple solutions. In the following we therefore consider the problem of computing verified bounds for the optimal objective value.

In [14], Jansson tackles this problem simply by exploiting the duality of the problems (LP) and (DLP). Every primal feasible vector \( x \) yields an upper bound for the optimal objective value: \( \hat{\rho} \leq c^T x \). Every dual feasible vector \( y \), in turn, yields a lower bound: \( b^T y \leq \hat{\rho} \). The verification procedure reduces to finding suitable points and verifying their feasibility.

Any interior feasible point \( \tilde{y} \) of the dual problem (DLP) satisfies \( c - A^T \tilde{y} > 0 \). If such a point is given, the computation of a verified lower bound becomes particularly simple. We just have to verify that every element of \( c - A^T \tilde{y} \) is nonnegative, for instance, by using interval arithmetic or monotone rounding. The better \( \tilde{y} \) approximates an actual dual solution \( \hat{y} \), the tighter is the lower bound \( b^T \tilde{y} \).

Given an approximate interior feasible vector \( \tilde{x} \) of the primal problem (LP), we first have to find an actual vector satisfying the equality constraints \( Ax = b \). Some details on how this can be realized in a rigorous manner are discussed in the latter part of this subsection. For the moment assume that we have an oracle that returns a vector \( \tilde{x} > 0 \) with small residual \( A\tilde{x} - b \) and that we are able to compute a tight interval enclosure \( x \) for an actual solution to \( Ax = b \) and favorably small distance to \( \tilde{x} \). If now \( x \geq 0 \) for all \( x \in x \), then \( x \) contains an actual primal feasible point and \( \hat{\rho} \leq \sup c^T x \).

Summarizing, we have the following

Let \( x \in \mathbb{R}^n \) and \( \tilde{y} \in \mathbb{R}^m \) be given. If \( A\tilde{y} \leq c, \ b \in \{Ax \mid x \in x \} \) and \( x \geq 0 \), then

\[
b^T \tilde{y} \leq \hat{\rho} \leq \sup c^T x. \tag{9}
\]

Moreover, let \( \varepsilon := \sup \{c^T x - b^T \tilde{y} \} \). Then \( \tilde{y} \) is an \( \varepsilon \)-optimal dual feasible point in the sense that \( b^T (\tilde{y} - \tilde{y}) \leq \varepsilon \) for every dual solution \( \hat{y} \). Similarly, there exists \( x_0 \in x \) so that \( x_0 \) is an \( \varepsilon \)-optimal primal feasible point satisfying \( c^T (x_0 - \tilde{x}) \leq \varepsilon \) for every primal solution \( \tilde{x} \).

The remaining questions are how to obtain suitable approximates \( \tilde{x}, \tilde{y} \) and how to compute a tight enclosure \( x \). Let an oracle be available that returns sufficiently accurate approximates \( (\tilde{x}, \tilde{y}) \) for a primal-dual solution pair \( (\tilde{x}, \tilde{y}) \) of (LP) and its dual (DLP). If we are lucky, the oracle, possibly a numerically stable and accurate interior-point solver, returns good approximates \( \tilde{x}, \tilde{y} \) for interior feasible (nearly optimal) points. These are particularly suitable for the verification approach above. On the other hand, solvers that use simplex methods tend to produce boundary points and solvers based on penalty methods typically produce outer approximations. Jansson dealt with these cases by using an approximate solution to a perturbed problem instance. The general idea for computing a primal feasible point is as follows.

(a1) Ask the oracle for an approximate solution \( \tilde{x} \) to (LP). Return failure if the oracle is unable to do so.

(a2) Check if the vector \( \tilde{x} \) returned by the oracle is nonnegative in every component. If not, compute a measure for the violation of \( \tilde{x} \geq 0 \) and go to (a5).

(a3) Compute an interval vector \( x \) containing an actual solution to \( Ax = b \) using \( \tilde{x} \) as starting point. If the verification method fails, return failure.

(a4) If \( x \geq 0 \), return \( x \) as an enclosure for a primal feasible point. Otherwise, compute a measure for the violation of the condition \( x \geq 0 \) and go to (a5).

(a5) Take the measure for the violation of the inequality conditions in (a2) or (a4) and add it to previous violation measures if this is not the first time for step (a5). Compute a perturbation vector \( 0 \leq \xi_\rho \in \mathbb{R}^n \) parameterized with this violation measure.
(a6) Ask the oracle for an approximate solution to (LP) with perturbed right-hand side \( b' = b - A\xi_p \). If the oracle fails to do so, return failure. Otherwise take the vector \( x' \) returned by the oracle, set \( \hat{x} = x' + \xi_p \) and go to (a2).

If the oracle in (a6) works well, then \( A\hat{x} = A(x' + \xi_p) \approx b' + A\xi_p = b \). The returned vector \( x' \) may be an outer approximation with nonpositive entries. Nevertheless, if the perturbation \( \xi_p \) is chosen appropriately, then \( \hat{x} = x' + \xi_p \) is strictly positive and in step (a3) we should be able to compute an opportune enclosure that satisfies the condition \( \mathbf{x} \geq 0 \). One possibility to compute the enclosure \( \mathbf{x} \) is to look for a solution \( \hat{r} \) to the least square problem

\[
\min_r \|r\|_2 \quad Ar = b - A\hat{x}.
\]

Ways to compute an interval enclosure \( \mathbf{r} \) for \( \hat{r} \) are discussed for instance in [15] and references therein. The desired \( \mathbf{x} \) is then derived as the sum of \( \hat{x} \) and \( \mathbf{r} \).

Alternatively, we may choose an index set \( J \) with cardinality \( m \) such that the matrix \( A_J \) consisting of the columns of \( A \) with indices in \( J \) is regular. Such an index set can be found by applying an \( LU \)-decomposition with (partial) pivoting to \( A \). The complementary index set to \( J \) shall be denoted by \( \bar{J} \). We then may compute an inclusion \( \mathbf{z} \) for the solution to the linear system:

\[
A_J \hat{z} = b - A_J \hat{x}_J.
\]

By setting \( \mathbf{x}_J = \mathbf{z} \) and \( \mathbf{x}_J = \hat{x}_J \), we obtain an alternative inclusion \( \mathbf{x} \) whose computation is typically more efficient than with the previous method.

The violation measure in (a2) and (a4), respectively, can have many different forms. For example, one could use \( \min\{0, \min_i \hat{x}_i\} \) or \( \sqrt{\sum_i \min\{\hat{x}_i, 0\}^2} \) as single value measurements. On the other hand, it is also possible to use a vector quantity as violation measure such as the element-wise minimum of \( \hat{x} \) and zero. For the perturbation \( \xi_p \) in step (a5) one can simply take this violation vector, scale it with a factor (slightly) larger than 1, and add a small positive constant. Of course many other parameterizations can lead to sensible perturbation vectors \( \xi_p \). If this is done well and the problem is well conditioned, then after very few iterations\(^2\) the algorithm (a1)–(a6) stops successfully. Under the assumption of a predictable accuracy of the oracle, Keil [16] has shown convergence for his particular implementation in Lurupa [17, 18]\(^3\).

The general idea for computing a dual feasible point is similar.

(b1) Ask the oracle for an approximate solution \( \hat{y} \) to (DLP). Return failure if the oracle is unable to do so.

(b2) Try to verify the inequality condition \( A^T\hat{y} \leq c \), for instance, by using interval arithmetic or monotone rounding. If satisfied, return \( \hat{y} \) as actually feasible dual point. Otherwise, compute a measure for the violation of \( A^T\hat{y} \leq c \).

(b3) Take the violation measure of the condition in (b2) and add it to previous violation measures if this is not the first time for step (b3). Compute a perturbation vector \( 0 \leq \xi_d \in \mathbb{R}^n \) parameterized with this violation measure.

(b4) Ask the oracle for an approximate solution \( \hat{y} \) to (DLP) with perturbed objective coefficient vector \( c' = c - \xi_d \). Return failure if the oracle does not produce a sensible \( \hat{y} \). Otherwise go to (b2).

For computing rigorous bounds it is necessary to take any floating-point rounding errors into account, including possible rounding errors in the evaluation of the inner product \( b^T\hat{y} \). Once more this

\(^2\)The implementations in Lurupa and VSDP often require no more than a single iteration.

\(^3\)Keil considered a different standard form of linear programming problems. The algorithm that is taken from [14] is a little bit more complex than the scheme (a1)–(a6). Nevertheless, the application of his results to our case is straightforward.
can be done using an interval floating-point implementation or monotone rounding. Alternatively, we can use error estimates as in [19].

If we are able to compute upper and lower bounds for the optimal objective value, then we prove the consistency of (LP) and (DLP). Additionally, the algorithms may provide $\varepsilon$-optimal feasible points or at least tight inclusions of them, where $\varepsilon := \sup \{ c^T \bar{x} - b^T \bar{y} \}$ is defined as before.

It is also noteworthy that the algorithms for upper and lower bound computation can be combined to reduce the overall computational effort. The main effort of these algorithms lies in the steps (a6) and (b4), where the oracle has to guess (therefore a solver has to compute) an approximate solution to the perturbed problem. The computation of a primal-dual pair $(\bar{x}, \bar{y})$ is typically no more expensive than computing only the primal or the dual approximate. So if we use the same oracle in (a6) and (b4) which returns both $\bar{x}$ and $\bar{y}$, then the computational cost for the oracle iteration can be almost halved by combining both algorithms. In particular, this requires to bundle the perturbations in (a5) and (b3). However, this approach is only sensible if both, the primal and the dual problem, are well-posed.

The algorithms (a1)–(a6) and (b1)–(b4) have been implemented in Lurupa [17] and VSDP [20] for even more general cases. Due to the focus on accuracy and stability, in neither of the software packages a combined algorithm have been implemented.

### 2.3 Ill-posed problems

In the previous subsections we discussed the nice non-degenerate case as well as the cases with multiple or degenerate solutions but nonempty primal and dual interior feasible sets. The former enables us to compute (tight) enclosures for the unique optimal solution. The latter still allows for the computation of rigorous bounds for the optimal objective value. This subsection is devoted to the most difficult case: the family of consistent but ill-posed linear programming problems.

For a better insight into this topic, let us first recap the condition measure introduced by Renegar.

**Definition 1 (Renegar [21]).** The condition measure to an instance of (LP) specified via the data vector $d = (A, b, c)$ is defined by

$$
\text{cond}(d) := \sup \left\{ \frac{\|d\|_\infty}{\|d - d'\|_\infty} \mid d' \in \partial \mathcal{F} \right\},
$$

where $\partial \mathcal{F}$ is the boundary of the set of all consistent instances:

$$
\mathcal{F} := \{(A, b, c) \in \mathbb{R}_m \times \mathbb{R}^m \times \mathbb{R}^n \mid \exists x \in \mathbb{R}^n : Ax = b, x \geq 0\}. \tag{11}
$$

By means of the condition measure, Renegar discussed the complexity of solving the respective linear programming problem to a desired accuracy and developed bounds for the solution size as well as approximation errors. For reasons of clarity, we refrain from discussing more details here. The main point for our analysis is that the condition measure quantifies the difficulty of deciding whether a given instance of (LP) is consistent or not. This directly leads to the following definition of ill-posedness.

An instance of (LP) is called **ill-posed for the decision problem** if its condition measure tends to infinity.

In the following, whenever we characterize an (LP) instance as *ill-posed*, the term is meant in respect to its decision problem, i.e., regarding the problem to decide whether the (LP) is consistent or not. Complementary, we use the term **well-posed** for all other problem instances.

Clearly, the condition measure of a consistent (LP) is anti proportional to the distance to the closest inconsistent problem instance. The problem is ill-posed if $d \in \partial \mathcal{F}$. Then (LP) has redundant equality constraints, an empty interior feasible set, or both. In any case, smallest perturbations can change the consistency of the problem. Computing verified bounds for such problems can be very hard and is typically outside of the scope of verification methods relying on approximate floating-point arithmetic.
Well-posed and ill-posed instances of (DLP) are defined accordingly. It is important to notice that ill-posedness of the primal problem does not imply that the dual problem is ill-posed as well. Neither is the opposite implication true. Indeed, it is rather the other way around. If one of the problems, primal or dual, is ill-posed, then often the complementary problem is well-posed. Primal and dual ill-posed problems are comparable seldom.

Any ill-posed instance of (LP) whose equality constraints are linearly independent is a problem with degenerate solution(s). It is clear in this context, that well-posedness is a generic property of (LP). In [12], Pataki and Tunçeli proved this for conic linear programming problems. Recently, Dür, Jargalsaikhan, and Still [22] surveyed different genericity results for this kind of programming problems. They showed that strong duality holds generically in a stronger sense.

Despite these beneficial characteristics, degeneracy is a problem occurring frequently in linear programming. The cause for this seemingly contrary observation is easily found: the modeling process of LP problems is not haphazard. The introduction of redundancies easily happens in this process and this issue affects a wide range of problems occurring in practice. The results in [12, 22, 23] hold for programming problems with specific conic structures but without consideration of sparsity pattern or typical modeling characteristics. They are therefore not applicable to the limited set of problems that actually occur in practice. A good demonstration for the coherence of this statement is due to Ordóñez and Freund [24] who found that 71% of the linear programming problem instances from the NETLIB test suite [25] have infinite condition measure. Numerical difficulties associated with the ill-posedness of these problems were also investigated by Keil and Jansson [26].

Without additional knowledge about the problem, there is little hope to compute verified inclusions for the optimal objective value of an ill-posed (LP) problem. But what can be done about ill-posed linear programming problems for which we have additional information? Can we compute rigorous bounds if we know a priori that the primal and the dual problem are consistent?

To signify the purpose of these questions, consider a consistent linear system with rank deficient coefficient matrix. Deciding whether the system is consistent or not, let alone computing a solution to this system are ill-posed problems. However, if the rank of the matrix and the consistency of the system are known a priori, we can compute a rigorous enclosure for an actual solution. On the other hand, even if we know that the system is solvable but have no information about the rank of the matrix, the problem to compute an enclosure for a solution remains ill-posed.

Similarly, the knowledge about consistency of (LP) and (DLP) is not sufficient for computing rigorous upper and lower bounds for the optimal objective value. Further information is required. As we will see in a moment, one possibility to tackle ill-posedness in linear programming problems is to exploit specific boundedness qualifications if known a priori for the corresponding solution vector.

Jansson [14] and Neumaier [27] both tackled this problem in very similar ways at almost the same time without prior knowledge of the work of each other. The general idea is the same in both works. If the primal variables are bounded, we can exploit duality for computing rigorous lower bounds for the optimal objective value. Similarly, if the dual variables are bounded, we are typically able to compute verified upper bounds. In this context, it is not really accurate to speak about ill-posedness of the decision problem because the knowledge about consistency of the problem resolves the decision problem altogether. Nevertheless, the problem to compute rigorous enclosures for the optimal objective values remains to be ill-posed. Some further explanation for this circumstance is given at the end of this subsection.

For our linear programming problems in the standard form (LP), Jansson’s and Neumaier’s result breaks down to:

**Theorem 1.** Consider an instance of (LP) with the dual problem (DLP). Let \( \tilde{y} \in \mathbb{R}^m \) be given and let \( \tilde{d} \in \mathbb{R}^n \) be a nonpositive lower bound for \( c - A^T \tilde{y} \) so that \( \tilde{d}_j \leq \min \{0, c_j - (A^T \tilde{y})_j\} \) for all indices \( j \). If the primal problem is consistent and \( \bar{x} \) is an upper bound for an optimal solution \( \hat{x} \) to (LP), i.e., \( \bar{x} \geq \hat{x} \), then

\[
\hat{\rho} \geq b^T \tilde{y} + \tilde{d}^T \bar{x}. \tag{12}
\]

Moreover, to a given vector \( \tilde{x} \in \mathbb{R}^n_+ \) let \( \tilde{r} \) be an upper bound for the element-wise absolute of the
residual $A\tilde{x} - b$. If the dual problem is consistent and $\bar{y}$ is an upper bound for the absolute of an optimal solution $\hat{y}$ to (DLP), i.e., $\bar{y} \geq |\hat{y}|$, then

$$\hat{y} \leq c^T \tilde{x} + \tilde{r}^T \bar{y}. \quad (13)$$

The bounds in (12) and (13) easily follow by the various estimates given in [27]. Neumaier developed his bounds with regard to the particular application in a branch-\&-cut approach for mixed-integer linear programming problems. In [14], Jansson went a little bit further and considered a more general formulation for an interval family of linear programming problems. Moreover, he developed his postprocessing routine together with the algorithms presented in the last subsection. This means that Jansson also tackled problems with solutions, for which a priori bounds are only partially known.

As an example for the applicability of Theorem 1, consider the (LP) formulation for a linear assignment problem. The linear assignment problem is about assigning a number of agents to a number of tasks such that every agent takes care of at most one task. The costs vary depending on the agent-task assignment and the goal is the minimization of the overall cost. Its trace formulation has the form

$$\min_{X \in \Pi_n} \langle C, X \rangle, \quad (LAP)$$

where $C$ is the parameter matrix containing the assignment costs, $\Pi_n$ denotes the set of $n \times n$ permutation matrices, and $\langle \cdot, \cdot \rangle$ terms the trace inner product. The convex hull of $\Pi_n$ is the set of doubly stochastic matrices which can be easily formulated via linear inequalities and equality constraints. In this regard, the convex combination of the solutions to (LAP) is identical to the solution set of

$$\min_{X \in \mathbb{R}^{n \times n}} \langle C, X \rangle \quad X e = X^T e = e \quad (LAP-LP)$$

where $e$ denotes the vector of all ones. For obtaining a linear programming problem in primal standard form, we simply have to vectorize the decision variable $X$.

This instance of (LP) has a degenerate solution or multiple of them, but it also has interior feasible points. The ill-posedness of (LAP-LP) is introduced in a different way: by a redundant equality constraint. The coefficient matrix $A$ realizing the equality constraints in (LAP-LP) has the size $2n \times n^2$, its rank however is $2n - 1$.

Without prior knowledge of this redundancy, most verification methods will fail to compute a rigorous enclosure for the optimal objective value of (LAP-LP). Of course the reason for ill-posedness of (LAP-LP) is not difficult to find and the issue is easily resolved by removing one of the equality constraints. Nevertheless, in many other practical problems, the detection and removal of ill-posedness can be much more complicated. Since we know that (LAP-LP) is always consistent and that every element of the decision variable is bounded from above by 1, Theorem 1 is applicable. A lower bound can be computed very efficiently by using (12).

Although the explanation above sounds somewhat promising, it is time for a reality check. The example above is actually not very well suited to demonstrate how additional boundedness qualifications can be used to circumvent ill-posedness. Strictly speaking the ill-posedness has not been resolved, not even been tackled.

The dual problem to (LAP-LP) has multiple solutions and its feasible set is unbounded. It is thus not immediately clear how to derive an upper bound $\bar{y}$ for any of the solutions, and inequality (13) of Theorem 1 cannot be exploited. Furthermore, since the feasible set of the dual problem has a nonempty interior set, the algorithm (b1)\-(b4) is as well applicable for the computation of a lower bound for the optimal objective value. The benefit of Theorem 1 is rather the computationally much more efficient way to compute lower bounds than the actual ability to do so.

It is noteworthy that there are a few cases where ill-posedness is introduced not only in the primal but also in the dual problem but still bounds for all primal variables are known. In this rare case it
is actually possible to circumvent ill-posedness by applying Theorem 1. A possible situation where this might happen is discussed in the next subsection. Nevertheless, if we assume that boundedness qualifications can be given only for variables which are actually bounded over the whole feasible set, then the following evident but fundamental connection remains true even in the presence of these boundedness qualifications.

If the primal problem (LP) is ill-posed for the decision problem, also the computation of a rigorous upper bound for the primal objective value is an ill-posed problem.

Similarly, if (DLP) is ill-posed for the decision problem, then so is the computation of a rigorous lower bound for the dual objective value.

Apparently, these statements also hold true the other way around. Although ill-posedness can affect drastically the speed and accuracy of the numerical solver, in the context of verification methods, ill-posedness often affects the computation of only one bound, the upper or the lower bound for the optimal objective value.

Moreover, the bound that can be computed relatively easy is often enough the bound that we are interested in. A good example are relaxation programs for combinatorial problems such as (LAP-LP). The variables in the relaxation program are typically bounded and the dual problem is most often well-posed so that we are able to compute just the bound that the relaxation is designed for. The complementary bound is often computed by using some meta heuristic or some other approach.

2.4 Free variables

Every linear programming problem can be formulated in the standard form (LP). Nevertheless, for numerical reasons and better performance, alternative formulations are often more beneficial. In [14] and based on this work also in [16], the authors considered the formulation:

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
Ax & \leq a \\
Bx &= b \\
l_b & \leq x \leq u_b,
\end{align*}
\]

where entries of \( l_b \) and \( u_b \) can be also \(-\infty\) and \(+\infty\), respectively, to formulate unbounded/free variables as well as variables that are bounded on one side only. Moreover, by setting \((l_b)_j = (u_b)_j\) we can fix the variable \( x_i \) and use it as a constant.

Whereas it is straightforward to formulate (LP) or (DLP) as (G-LP), the other way around requires slightly more effort but is not that difficult either. A major reason for choosing (G-LP) over (LP) is that the bounds on the variables can be exploited directly for the computation of rigorous lower bounds as in (12). Another good reason to use a more general linear programming problem formulation is the absence of free variables in (LP). If the underlying problem model requires free variables and does not provide sensible a priori bounds on them, then we need to transform them into the difference of two nonnegative variables. This transformation is problematic because it introduces ill-posedness in the dual problem. If a free variable is transformed into the difference of two nonnegative variables, slightest perturbations of \( c \) can lead to the existence of a primal improving ray and therefore the inconsistency of the dual problem.

For reasons of simplicity, until now we considered the standard formulation (LP). Nevertheless, in the context of necessary transformation of free variables, this formulation is not ideal for the use with verification methods. We therefore introduce free variables in (LP) yielding

\[
\begin{align*}
\text{minimize} & \quad c_f^T x_f + c_l^T x_l \\
A_f x_f + A_l x_l &= b \\
x_l & \geq 0,
\end{align*}
\]

where \( x_f \) denotes the vector of free and \( x_l \) the vector of nonnegative decision variables, respectively. The dual problem changes accordingly:
maximize $b^T y$

$A^T y = c_f$

$A^T y \leq c_l.$

The results and algorithms in the previous subsections can be generalized nicely for this new formulation. The biggest adaption is required for the algorithm (b1)–(b4). Between step (b1) and (b2) it is necessary to compute an enclosure $y$ for an actual solution to $A^T y = c_f$. In (b2) one would then check that $A^T y \leq c_l$ is satisfied for all $y \in y$. Ways to do this have been discussed already for the algorithm (a1)–(a6). Moreover, the linear programming formulation above is a special case of the conic linear programming formulation that is discussed thoroughly in the next section.

2.5 Verifying inconsistency and unboundness

Until now we considered the situation that the linear programming problem is feasible with bounded optimal objective value. We complete the section with verification methods for linear programming problems that are inconsistent or whose objective values are unbounded. The fundamental result for checking inconsistency of (LP) is Farka’s lemma [28, 29]:

**Lemma 1.** Exactly one of the following statements is true:

- The primal problem (LP) is consistent.
- The set \{ $d \in \mathbb{R}^m$ | $A^T d \leq 0, b^T d > 0$ \} is nonempty.

A nonzero vector $d \in \mathbb{R}^m$ that satisfies $A^T d \leq 0$ is called dual recession direction or dual ray. If its dual objective value $b^T d$ is positive, then $d$ is specified as dual improving ray. Let the dual problem be consistent and let $y_0$ be a feasible point of (DLP). For any $\alpha \geq 0$ the point $y_0 + \alpha d$ is dual feasible with increasing objective value. Then the dual objective value is unbounded and naturally the primal problem has to be inconsistent. Nevertheless, even if the dual problem is inconsistent, by Farka’s lemma the existence of a dual improving ray implies the inconsistency of the primal problem.

Many LP solvers are able to compute a dual improving ray if existent. Assume that we have an oracle that returns such a ray $d$ in case of an inconsistent primal problem. Verifying that $d$ is actually a dual improving ray is straightforward. We may compute $A^T d$ and $b^T d$ using monotone rounding to verify the corresponding inequalities with mathematical rigor. However, this typically requires that all inequalities are satisfied strictly. If the dual problem is ill-posed, for instance due to the transformation of free variables in the primal problem, this also affects the computation and verification of dual improving rays.

The complementary case is treated similarly. Let $x_0$ be a feasible point of (LP). If there exists another feasible primal point $x_1$ that is element-wise larger or equal to $x_0$ but with strictly smaller objective value, i.e., $x_1 \geq x_0$ and $c^T x_1 < c^T x_0$, then $p = x_1 - x_0$ is a primal improving ray. For any $\alpha \geq 0$ the point $x_0 + \alpha p$ is primal feasible with decreasing objective value. Thus the primal objective value is unbounded and the dual problem is inconsistent. The variant of Farka’s lemma that applies in this case is as follows.

**Lemma 2.** Exactly one of the following statements is true:

- The dual problem (DLP) is consistent.
- The set \{ $p \in \mathbb{R}^n_+$ | $A p = 0, c^T p < 0$ \} is nonempty.

Similar as before, it is not necessary that the primal problem is consistent; just the existence of a primal improving ray implies that the dual problem is inconsistent.

In contrast to a dual improving ray, a primal improving is typically not exactly representable in floating-point arithmetic due to the presence of equality constraints. Since the computation of an exact result can be extremely difficult, it is preferable to use some verification method to compute an enclosure $p$ for a solution to the underdetermined linear system $A p = 0$. As a starting point we
use the approximate improving ray $\tilde{p}$ returned by the oracle and then update the vector by a (least square) solution to $A p_{\text{update}} = -A \tilde{p}$. If the computed enclosure satisfies the nonnegativity condition $p \geq 0$ as well as $c^T p < 0$, then (DLP) is necessarily inconsistent due to the existence of a primal improving ray. For this approach to work, it is usually necessary that at least $m$ entries of $\tilde{p}$ are strictly positive. Ill-posedness of the primal problem can be a major issue also for the verification of dual inconsistency.

3. Conic linear programming

The linear programming problem belongs to the class of convex and, more specifically, conic optimization problems. The availability of efficient methods for solving a linear programming problem and its applicability to a wide range of practical problems were and still are the primary reasons for the strong interest in linear programming. The efficiency is explained by the convexity of the problem and the simplicity of the equality and inequality conditions. And there are still interesting open questions about this problem class. For instance: Is there a variant of the simplex algorithm that has no worst case exponential runtime?

Many of these interesting and beneficial properties are taken over to a wider class of optimization problems: conic linear programming (CLP), often also referred to as cone-LP. A conic linear programming problem can be written in the following form

$$\begin{align*}
\min_x & \quad c^T x \\
\text{subject to} & \quad Ax = b, \\
x & \in K,
\end{align*}$$

(CLP)

where $K$ is a closed convex cone in $\mathbb{R}^n$. For reasons of simplicity, here we refrain from discussing cones in more general vector spaces.

The Lagrangian function has the same form as in (1). For deriving the Lagrangian dual problem, we have to replace $\mathbb{R}^n_+$ in (2) with $K$ which yields

$$\begin{align*}
\max_y & \quad b^T y \\
\text{subject to} & \quad c - A^T y \in K^*,
\end{align*}$$

(DCLP)

where $K^*$ denotes the dual cone to $K$, i.e.,

$$K^* := \{ z \mid z^T x \geq 0 \text{ for all } x \in K \}.$$

It is clear from the formulation above that conic linear programming is a natural extension of linear programming. In conic linear programming, we replace the nonnegative orthant $\mathbb{R}^n_+$ as a special closed convex cone with other cones that are suitable to model practical problems. Alongside the positive orthant, the two most well studied cones for (CLP) are the second order cone

$$Q^n := \{ x \in \mathbb{R}^n \mid x_1 \geq \|(x_2, \ldots, x_n)^T\|_2 \}$$

and the cone of symmetric semidefinite matrices

$$S^n_+ := \{ X \in S^n \mid \forall v \in \mathbb{R}^n : v^T X v \geq 0 \}.$$

All three cones are self-dual, meaning $K^* = K^4$. But there are also practical relevant non-self-dual cones such as the power and the exponential cone [30].

3.1 Cone-LPs with non-degenerate unique solutions

Much of the verification theory for LP problems can be generalized straightforward for CLP problems. An important restriction in conic linear programming is that strong duality is not necessarily satisfied and requires additional qualifications. There are various constraint qualifications under which strong duality holds. One of the simplest and most well known of them is

\[\text{Here we skipped formal details regarding dual spaces and the comparability of primal and dual cones.}\]
Slater’s condition
If the primal or dual problem are strictly feasible, i.e., there exist interior feasible points, then strong duality holds.

Another difficulty of (CLP), that is not present in linear programming, is the possibility of weak inconsistency. This term is referring to inconsistent problems whose decision problem is ill-posed. For more details, see [31–33] and the references therein. For reasons of simplicity, in the following we assume the problems to be either consistent or strictly inconsistent.

Degeneracy in mathematical programming can take different forms and is used for different characteristics. In linear programming, degeneracy typically refers to degenerate solutions, i.e., solutions $\hat{x}$ to (LP) with less non-zero entries than there are linear independent constraints on the decision variable $x$. In nonlinear programming, one typically uses the term degenerate constraints for constraints that violate the linear independent constraint qualification (LICQ), therefore leading to a singular KKT system. Sometimes, degenerate conic linear programs, in particular semidefinite programming problems, are even affiliated with the absence of interior feasible points. The related terminology is more widespread. For instance, the non-singularity of Clarke’s generalized Jacobian of the KKT system is called BD-regularity [34] and the notion of weak primal and dual non-degeneracy was introduced by Yildirim [35]. Although the term degenerate solution is not factoring in possible redundancies in the equality constraints $Ax = b$, we stick to this notion of degeneracy used in linear programming. The generalization for conic linear programming problems was given by Pataki [36] in terms of faces.

The KKT conditions to (CLP) are

$$\begin{align*}
b - Ax &= 0 \quad \text{(primal feasibility)} \\
x &\in \mathcal{K} \\
c - A^T y &\in \mathcal{K}^* \quad \text{(dual feasibility)} \\
(c - A^T y) \bullet x &= 0. \quad \text{(complementary slackness)}
\end{align*}$$

These conditions are very similar to the linear programming case. The inequalities ‘$\geq 0$’ (alternatively written as ‘$\in \mathbb{R}^n_+$’) have been replaced with ‘$\in \mathcal{K}$’ and ‘$\in \mathcal{K}^*$’, respectively. Moreover, the operation $\bullet$ in the complementary slackness condition is not necessarily understood as an element-wise product. In the general case, it may just be interpreted as inner product. Nevertheless, by exploiting the cone conditions on the factors of the inner product, the single equality is typically lifted to a vector identity that leads to a square Jacobian of the KKT system. For instance, in semidefinite programming, the primal variable $x$ and the dual slack $c - A^T y$ form symmetric matrices and we use $\bullet$ to denote a matrix product.

To compute a verified inclusion of a solution to the KKT system, we follow the same approach as before. Assuming that (LICQ) is satisfied, we can compute an interval inclusion $(x, y)$ for a solution to the nonlinear system

$$F(x, y) = \left( \begin{array}{c} b - Ax \\ (c - A^T y) \bullet x \end{array} \right) = 0. \quad (14)$$

Since the solution $\hat{x}$ is typically found at the boundary of $\mathcal{K}$, it is extremely unlikely that $x \in \mathcal{K}$ is satisfied for all $x \in \mathcal{K}$. Nevertheless, it is often possible to apply a similar method as for linear programming where we exploit strict complementary. Pataki generalized the notion of strict complementary for CLPs in term of faces in [36]. Alternatively, we may use the following definition.

**Definition 2.** A primal-dual solution pair $(\hat{x}, \hat{y})$ is strictly complementary if $\hat{x}$ lies in the relative interior of the cone \( \{ z \mid (c - A^T \hat{y})^T z = 0, z \in \mathcal{K} \} \).

If (LICQ) is satisfied for the KKT system at a strictly complementary solution pair $(\hat{x}, \hat{y})$, then this solution is unique and, assuming high enough accuracy, we are able to compute rigorous enclosures for $(\hat{x}, \hat{y})$. A more detailed explanation is given for the semidefinite programming problem in Subsection 3.4.
3.2 Treatment of cone-LPs with degenerate solutions

The verification idea in the previous section works for (CLP) instances with non-degenerate strictly complementary solutions. If the considered cone-LP has degenerate solution(s), the issues are of similar nature as for LP.

Computing exact solutions to conic linear programming problems with degenerate solution sets is an ill-posed problem. Naturally this entails the computation of tight inclusions for these solutions. The discrepancy between various genericity results and observations for practical problem instances have already been discussed for the class of linear programming problems, and the situation is not different for cone-LPs. We can circumvent this issue in a similar way as before. To be precise, we derive rigorous bounds for the optimal objective value by computing enclosures of $\varepsilon$-optimal feasible primal/dual points.

The general idea for computing such enclosures is the same as for linear programming problems. If the oracle returns a point for which we are not able to prove feasibility, we ask for the solution to a slightly perturbed problem in order to generate a more suitable vector for the verification method. The following method is based on Jansson’s algorithm for semidefinite programming problems [20, 37].

\begin{enumerate}
\item[(c1)] Ask the oracle for an approximate solution $\tilde{x}$ to (CLP). Return failure if the oracle is unable to do so.
\item[(c2)] Compute an interval enclosure $x$ for an actual solution to the underdetermined linear system $Ax = b$ using $\tilde{x}$ as starting point. If the verification method fails, return failure.
\item[(c3)] If the condition $x \in \mathcal{K}$ can be verified for all $x \in x$, return $x$. Otherwise, compute a measure for the violation of $x \subseteq \mathcal{K}$.
\item[(c4)] Take the measure for the violation of the cone condition in (c3) and add it to previous violation measures if this is not the first time for step (c4). Compute a perturbation vector $\xi_p \in \mathcal{K}$ parameterized with this violation measure.
\item[(c5)] Ask the oracle for an approximate solution to (CLP) with perturbed right-hand side $b' = b - A\xi_p$. If the oracle fails to do so, return failure. Otherwise take the vector $x'$ returned by the oracle, set $\tilde{x} = x' + \xi_p$, and go to (c2).
\end{enumerate}

This algorithm reminds strongly on the algorithm (a1)–(a6) for the linear programming problem. The principle is the same. Here we have no equivalent for step (a2) in order to avoid the additional computational expense for checking $\tilde{x} \in \mathcal{K}$. The verification method used in step (c2) follows the explanation for (a3) in Subsection 2.2. Also the other remarks thereafter can be easily generalized for the family of conic linear programming problems. Nevertheless, the definition and computation of a suitable violation measure may be comparatively complicated.

Although there are no convergence results for this algorithm as there are in the case of linear programming, in practical tests the algorithms stops after no more than a handful of iterations. This statement is supported by the implementation in VSDP [20] and the various numerical results presented in [26, 38, 39].

The general idea for lower bound computation is similar as in the algorithm (b1)–(b4). However, whereas the presupposition for algorithm (c1)–(c5) that $\mathcal{K}$ has a nonempty interior is no fundamental restriction for the formulation of primal-dual well-posed problem instances, the same is not necessarily true for $\mathcal{K}^*$. Addressing the issue discussed in Subsection 2.4 regarding free variables, we assume that $\mathcal{K}$ can be flat, i.e., $\mathcal{K}$ may contain a nontrivial linear subspace by which the interior of $\mathcal{K}^*$ is empty. For reasons of simplicity and without loss of generality, assume that such a subspace exists with dimension $k > 0$. Let $M \in \mathbb{R}^{n \times k}$ be a matrix whose columns form a basis of the linear subspace contained in $\mathcal{K}$ and let $M_\mathcal{C} \in \mathbb{R}^{n \times (n-k)}$ be a matrix to a complementary basis. The generalized verification approach is as follows.
(d1) Ask the oracle for an approximate solution \( \tilde{y} \) to (DCLP). Return \texttt{failure} if the oracle is unable to do so.

(d2) Compute an enclosure \( y \) for a solution to the (underdetermined) linear system \( M^T(c - A^T y) = 0 \) with \( \tilde{y} \) as starting point. If no sensible enclosure could be computed, return \texttt{failure}.

(d3) If the condition \( M_0^T(c - A^T y) \subseteq M_0^T K \) can be verified, return \( y \). Otherwise, compute a measure for the violation of this condition.

(d4) Take the violation measure of the conic membership condition in (d3) and add it to previous violation measures if this is not the first time for step (d4). Compute a perturbation vector \( \xi_d \in K^* \) parameterized with this violation measure.

(d5) Ask the oracle for an approximate solution \( \tilde{y} \) to (DCLP) with perturbed objective coefficient vector \( c' = c - \xi_d \). Return \texttt{failure} if the oracle does not produce a sensible \( \tilde{y} \). Otherwise go to (d2).

In most practical applications, the linear subspace of \( K \) has the form of free variables so that \( K \) is a Cartesian product of a set of free variables \( R^{n_f} \) and some salient cone \( K_s \subseteq R^{n - n_f} \). The matrices \( M \) and \( M_0 \) can then be chosen in such a way that \([M \ M_0]\) is a permutation matrix and

\[
\begin{align*}
  c - A^T y \in K^* & \iff M^T(c - A^T y) = 0 \land M_0^T(c - A^T y) \in K_s^*.
\end{align*}
\]

Thus \( M, M_0 \) realize a partitioning of the cone constraint, which simplifies the verified computations in (d2) and (d3).

If both algorithms (c1)–(c5) and (d1)–(d5) run to completion and return sensible enclosures \( x \) and \( y \), respectively, (CLP) and (DCLP) are both consistent with bounded optimal objective values. Let \( \hat{\varrho}_p \) and \( \hat{\varrho}_d \) denote the primal and the dual optimal objective value, respectively. The following result follows immediately from duality.

Let \( x \in \mathbb{R}^n \) and \( y \in \mathbb{R}^m \) be given. If \((x, y)\) contains a primal-dual feasible pair, i.e.,

\[
\exists x \in x: \quad Ax = b, \ x \in K, \quad \exists y \in y: \quad c - A^T y \in K^*,
\]

then

\[
\inf b^T y \leq \hat{\varrho}_d \leq \hat{\varrho}_p \leq \sup c^T x. \tag{15}
\]

Unlike for linear programming, the inequality \( \hat{\varrho}_d \leq \hat{\varrho}_p \) can be strict because strong duality is not always satisfied. Nevertheless, it is typically not necessary to check if Slater’s condition or other constraint qualifications are satisfied. If the problem is well-conditioned, the gap \( \sup c^T x - b^T y \) is usually very small so that a tight enclosure for both the primal and dual optimal objective is computed.

Similarly as for the class of linear programming problems, it is possible to exploit a priori known bounds for some optimal point to compute rigorous bounds for the optimal objective function much more efficiently. Shortly after introducing his verification method for linear programming problems in [14], Jansson generalized his approach for lower bounds of convex optimization problems [40] and in particular semidefinite programming problems [37, 41]. In 2007, he unified the theory for conic linear programming problems using the notation of vector lattices [42].

Denote by \( \succeq_K \) a partial ordering induced by the cone \( K \), i.e.,

\[
x \succeq_K z \iff x - z \in K.
\]

The fundamental characteristic of a vector lattice for a partial ordering \( \succeq_K \) is that for any subset of this vector lattice there exists an infimum and supremum in terms of the corresponding partial ordering. In Jansson’s main result [42, Theorem 4.1 and Theorem 4.2] the existence of lower and upper bounds is assumed. In the setting of vector lattices this is always satisfied. Nevertheless, the understanding of his result does not require the notation of vector lattices.
Theorem 2. Consider an instance of (CLP) with the dual problem (DCLP). Let \( \tilde{y} \in \mathbb{R}^m \) be given and let \( d \in \mathbb{R}^n \) be such that \( \{0, c - A^T \tilde{y}\} \succeq K^* \cdot d \). If the primal problem is consistent and \( \bar{x} \) is an upper bound for an optimal solution \( \hat{x} \) to (CLP), i.e., \( \bar{x} \succeq_k \hat{x} \), then

\[
\hat{\rho}_p \geq b^T \tilde{y} + d^T \bar{x}.
\]

(16)

Moreover, let \( \bar{x} \in K \) be given and, assuming consistency, denote by \( \hat{y} \) an optimal solution to (DCLP). If \( \bar{r}, \bar{y} \in \mathbb{R}^m \) are such that \( \bar{r} \geq A\bar{x} - b \geq -\bar{r} \) and \( \bar{y} \geq \hat{y} \geq -\bar{y} \), then

\[
\hat{\rho}_d \leq c^T \tilde{x} + \bar{r}^T \bar{y}.
\]

(17)

The inequalities on \( A\bar{x} - b \) and \( \hat{y} \) can be replaced with any compatible cone inequality \( \succeq C \) and its dual \( \succeq C^* \), respectively.

Jansson’s original result can be applied in more general settings including infinite dimensional problems. Theorem 2 is the broken down version for our specific problem formulation. It is noteworthy that most convex cones which are used for modeling practical problems, for example, the cone of symmetric semidefinite matrices, the second order cone, or the exponential cone, are compatible with the notation of vector lattices.

In contrast to Theorem 1 for linear programming problems, the result for cone-LPs has to be used with more care. The bounds in (16) and (17) do not necessarily define an interval containing the primal and the dual optimal objective value. This issue can be tackled by combining the residual bounds with the iteration procedure described in (c1)–(c5) and (d1)–(d5), respectively. The general idea is the same as for linear programming problems [14, 16]. On the other hand, for many practical applications, where Theorem 2 can be applied, this is not necessary.

3.3 Certificates of inconsistency

The general principle for verifying the inconsistency of a conic linear programming problem is the same as in the case of linear programming. However, there is one fundamental theoretical difference. Farka’s lemma cannot be generalized for arbitrary convex cones. Nevertheless, the related theorems of alternatives still serve our purpose. When concerned with the consistency of the primal problem, we use the following lemma.

Lemma 3. Only one of the following statements can be true:

- The primal problem (CLP) is consistent.
- The set \( \{d \in \mathbb{R}^m \mid A^T d \in -K^*, b^T d > 0\} \) is nonempty.

This result is very similar to Lemma 1. Besides replacing the nonnegative orthant with a more general convex cone, the major difference to Farka’s lemma is that both statements can be false. The primal problem can be inconsistent even in the absence of a dual improving ray. This phenomenon is called weak inconsistency or weak infeasibility.

In the context of verification methods, there is no significant difference to the linear programming case. If a (CLP) problem is weakly inconsistent, then the decision problem is ill-posed and verification methods that rely on approximate floating-point arithmetic are typically not applicable. Moreover, if a given instance of (LP) is inconsistent and its dual is ill-posed, then verification methods are not able to prove the existence of a dual improving ray even if this is the case.

Assume that primal and dual problem are well-posed and that our cone-LP oracle is able to return an approximate dual improving ray \( \tilde{d} \) if existent. Since \( K \) can be flat, it may be necessary to first compute an enclosure \( d \) for a nearby vector \( d \) satisfying the implicit equality constraints. The approach is the same as for the algorithm (d1)–(d5). To be precise, if \( M \) is a matrix whose columns form a basis of the linear subspace contained in \( K \), any dual improving direction \( d \) has to satisfy \( M^T A^T d = 0 \). For an interval enclosure \( d \) of a solution to this underdetermined system, we then need to verify

\[
M^T A^T d \subseteq -M^T K^* \quad \text{and} \quad b^T d > 0,
\]
where $M\mathcal{C}$ is a matrix whose columns form a complementary basis to $M$. If this was successful, then $d$ contains an actual improving ray and is called verified certificate of primal inconsistency. If additionally (DCLP) is consistent, then the dual objective value is unbounded.

The theorem of alternative for verifying dual inconsistency is similar.

**Lemma 4.** Only one of the following statements can be true:

- The dual problem (DCLP) is consistent.
- The set $\{p \in \mathcal{K} \mid Ap = 0, c^T p < 0\}$ is nonempty.

This lemma is applicable with the same restriction as above: the primal and the dual problem have to be well-posed. Starting with an approximate primal improving ray $\tilde{p}$ returned by the oracle, we compute an enclosure $p$ for a solution to $Ap = 0$ and try to verify $p \subseteq \mathcal{K}$ as well as $c^T p < 0$. In the case of success this yields a verified certificate of dual inconsistency.

### 3.4 Semidefinite programming

As a special class of cone-LPs we consider semidefinite programming problems. This class entails many of the problematic cases that cannot occur in linear programming, such as a nonzero duality gap or weak inconsistency.

For reasons of simplicity we consider the semidefinite programming problem in standard form without taking into account a specific block structure or introducing free variables:

\[
\begin{align*}
\text{minimize} \quad & \langle C, X \rangle \\
\text{subject to} \quad & \langle A_i, X \rangle = b_i \quad \text{for} \quad i = 1, \ldots, m, \\
& X \in S^r_+, \\
\end{align*}
\]  

(SDP)

where $S^r_+$ is the cone of positive semidefinite matrices in the space of $r \times r$ symmetric matrices $S^r$, $\langle \cdot, \cdot \rangle$ denotes the trace inner product, $C, A_1, \ldots, A_m \in S^r$ and $b \in \mathbb{R}^m$.

The Lagrangian dual problem to (SDP) is

\[
\begin{align*}
\text{maximize} \quad & b^T y \\
\text{subject to} \quad & C - \sum_{i=1}^m y_i A_i \in S^r_+. \\
\end{align*}
\]  

(D-SDP)

The expression that specifies the set of feasible dual solutions is called linear matrix inequality (LMI).

The symmetric matrices in the primal and dual problem are representable by vectors of $\frac{r(r+1)}{2}$ independent variables. A typical choice for this vector representation is

\[
x = \text{svect}(X) = [X_{11}, \sqrt{2}X_{12}, X_{22}, \sqrt{2}X_{13}, \sqrt{2}X_{23}, X_{33}, \ldots, X_{rr}],
\]

which is exploited, for instance, in SDPT3 [43]. The biggest benefit of this representation is that $\langle A_i, X \rangle = \text{svect}(A_i)^T \text{svect}(X)$. It is thereby straightforward to transform (SDP) to (CLP). On the other hand, in the context of verification methods, it is preferable to use representations that avoid scaling with the number $\sqrt{2}$ which is not a floating-point number.

The corresponding KKT system contains the $m$ primal equality constraints and $\frac{r(r+1)}{2}$ independent constraints from the complementary slackness condition $(C - \sum_{i=1}^m y_i A_i)X = 0$. We assume that (LICQ) is satisfied so that the Jacobian to this nonlinear system is regular. It is then possible to compute verified inclusions for solutions $\hat{X}$ and $\hat{y}$ by using the Krawczyk operator or some other related method. If the optimal solution satisfies the complementary slackness condition strictly, this can be exploited as follows.

Let $X \in \mathbb{R}^{r \times r} \cap S^r$, $y \in \mathbb{R}^m$ be given and assume

\[
\exists \hat{X} \in X, \hat{y} \in y: \quad (C - \sum_{i=1}^m y_i A_i)\hat{X} = 0, \quad \langle A_i, \hat{X} \rangle = b_i \quad \text{for} \quad i = 1, \ldots, m. \quad (18)
\]
Denote by $\lambda_k(X)$ the $k$th largest eigenvalue of a symmetric matrix $X$. If there is a nonnegative integer $k \leq r$ so that

$$\forall X \in \mathbf{X}: \lambda_k(X) > 0 \quad \text{and} \quad \forall y \in \mathbf{y}: \lambda_{r-k}(C - \sum_{i=1}^{m} y_i A_i) > 0,$$

then $\tilde{X} \in S^+_r$ with rank($\tilde{X}$) = $k$ and $C - \sum_{i=1}^{m} \hat{y}_i A_i \in S^+_r$ with rank equal to $r - k$. The interval matrix $X$ comprises an optimal solution $\hat{X}$ to (SDP).

Alternatively, also if (18) and

$$\forall X \in \mathbf{X}, y \in \mathbf{y}: \quad X + C - \sum_{i=1}^{m} y_i A_i \in \text{int} S^+_r$$

are both satisfied, $X$ contains a primal optimal solution.

As an example, consider the instance of (SDP) specified by

$$r = 3, \quad m = 1, \quad A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad b_1 = 1, \quad C = \begin{bmatrix} 3 & 0 & -6 \\ 0 & -3 & -6 \\ -6 & -6 & 0 \end{bmatrix}.$$ 

Hence we are looking for symmetric positive semidefinite $3 \times 3$ matrix $X$ that has a minimal trace inner product with $C$. The parameter matrix $C$ has the unique smallest eigenvalue $\lambda_3 = -9$ and corresponding eigenvector $v_3 = (\frac{1}{4}, \frac{3}{4}, \frac{3}{4})^T$. It is not difficult to see that $\hat{X} = v_3 v_3^T$ and $\hat{y} = \lambda_3$ are the unique primal and dual solution, respectively.

If everything works well, the verification method returns a tight enclosure $(X, y)$ for the solution $(\hat{X}, \hat{y})$ to the corresponding KKT system (in vectorized form). The interval matrix $X$ typically has indefinite realizations as well. Nevertheless, assuming a tight enough enclosure, already the positivity of the trace $\text{tr}(X)$ implies $\lambda_1(X) > 0$ for all $X \in \mathbf{X}$. The second inequality in (19) can be proved for $k = 1$ by using for example Weyl’s eigenvalue perturbation theorem [44] or Kahan’s residual bound on clustered eigenvalues [45]. Alternatively, it is possible to prove (20) using an efficient verification method based on the Cholesky decomposition [46] in combination with Weyl’s theorem [44]. More details on this are discussed in Subsection 4.4.

The uniqueness of the solution is due to the choice of $C$. If we replace the parameter matrix of the primal objective with

$$C = \begin{bmatrix} -1 & -8 & 4 \\ -8 & -1 & -4 \\ 4 & -4 & -7 \end{bmatrix},$$

then $\hat{X} = v_3 v_3^T$ is still a primal optimal solution but no more unique. The multiplicity of the solution is caused by the multiplicity of the smallest eigenvalue of $C$. Now the Jacobian matrix to the corresponding KKT system is not regular which impedes the computation of verified enclosures for solutions to the KKT system. On the other hand, the algorithms (c1)–(c5) and (d1)–(d5) are still applicable. A specific setting of these algorithms for semidefinite programming with a suitable choice for the violation measure was discussed in [37].

Interestingly, since the trace of every primal feasible point $X$ is equal to 1 in both examples above, this immediately yields an upper bound for any primal optimal point $\hat{X}$:

$$\hat{X} = I \succeq s_+ \hat{X}.$$ 

Let $\hat{y} = -\frac{23}{8}$ be an approximate dual solution returned by the oracle. The matrix $C - \hat{y}_1 A_1$ is indefinite with $\lambda_3(C - \hat{y}_1 A_1) = -\frac{1}{6}$. Hence, a simple negative semidefinite lower bound is $\hat{D} = -\frac{1}{8}I$ which then enables us to apply (16) for a lower bound of the primal optimal objective value:

$$\hat{b}_p \geq b^T \hat{y} + \langle D, \hat{X} \rangle = -\frac{26}{8}.$$ 

345
The approximate solution \( \tilde{y} \) is rather inaccurate and thus yields an inaccurate lower bound. In practice we can expect much better approximations and tight bounds.

As another, less hypothetical example for the applicability of Theorem 2, let us once again have a look on an assignment problem. This time, however, we consider a relaxation for the quadratic assignment problem [47]:

\[
\min_{X \in \Pi^r} \langle F, X G X^T \rangle + \langle H, X \rangle,
\]

(QAP)

where \( F, G \in S^r \) and \( H \in \mathbb{R}^{r \times r} \) are the parameter matrices, and \( \Pi^r \) denotes the set of \( r \times r \) permutation matrices. To be precise, we consider the low-dimensional semidefinite programming relaxation framework introduced by Ding and Wolkowicz [48]. Their relaxation is based on a matrix lifting approach and its base framework can be formulated as follows:

\[
\begin{align*}
\min_{X \in \mathbb{R}^{r \times r}, Y, Z \in S^r} & \langle F, Y \rangle + \langle H, X \rangle \\
& \begin{bmatrix} I & X^T & G X^T \\ X & I & Y \\ X G & Y & Z \end{bmatrix} \in S^3_{++}, \quad X \in \mathbb{R}^{r \times r} \\
& \text{diag}(Y) = X \text{diag}(G), \quad \text{diag}(Z) = X \text{diag}(G^2), \\
& Ye = XGe, \quad Ze = XG^2e, \quad Xe = XTe = e,
\end{align*}
\]

(21)

where \( \text{diag}(G) \) denotes the vector consisting of the diagonal elements of \( G \) and \( e \) is the vector of all ones.

By \( \text{tr}(Z) = e^T X \text{diag}(G^2) = \text{tr}(G^2) \), the trace of the matrix that realizes the semidefiniteness condition is equal to \( 2n + \text{tr}(G^2) \). This immediately yields an upper bound for the maximal eigenvalue of this matrix together with the upper bound

\[
(2n + \text{tr}(G^2)) \cdot \begin{bmatrix} I \\ I \\ I \end{bmatrix} \succeq S^r_{++}, \quad \begin{bmatrix} I & X^T & G X^T \\ X & I & Y \\ X G & Y & Z \end{bmatrix}.
\]

This semidefiniteness inequality holds valid not only for an optimal point of the relaxation but for every feasible point, and it can be improved slightly by exploiting the boundedness of each variable together with the Gershgorin circle theorem [49]. For applying the inequality in (16) of Theorem 2, as before we just need a negative semidefinite matrix that is also a lower bound for the complementary part of an approximate dual solution. Many numerical results and further explanations regarding the computation of rigorous lower bounds can be found in [38]. Many more examples are given in [37, 39, 42].

4. Verified computations for interval cone-LPs

In the previous sections we already encountered intervals several times. They served as enclosures for solutions to (underdetermined) linear systems which might not be representable in floating-point arithmetic. And they also occurred naturally when we applied the Krawczyk operator to the nonlinear KKT systems. The intervals are used to model inaccuracies in the iterative solution procedure and rounding errors due to floating-point arithmetic.

Another possible use of interval quantities is to model inaccuracies in the optimization parameters. These inaccuracies can be caused by rounding errors in the computations of the input parameters or by approximation errors of the optimization model itself. They also occur in form of accuracy tolerances for physical measurements. Typically, a combination of possible causes has to be factored in when modeling inaccuracies.

Let \( A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n \) with \( n > m \). The interval cone-LP of the form

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b \\
& \quad x \in K.
\end{align*}
\]

(I-CLP)

346
denotes the set of all realizations of (CLP) specified by a data vector \((A, b, c) \in (\mathbf{A}, \mathbf{b}, \mathbf{c})\).

In some cases we might be interested in a ‘best case’ analysis for the set of problems in (I-CLP), i.e., we are looking for a solution to a realization \(d = (A, b, c)\) with minimal primal objective value:

\[
\hat{\omega} := \inf_{d \in \mathbf{d}} \inf_{x \in \mathcal{K}} \{ c^T x \mid Ax = b \}, \tag{22}
\]

where \(\mathbf{d} = (\mathbf{A}, \mathbf{b}, \mathbf{c})\) denotes the interval input data vector.

In the context of linear programming, where \(\mathcal{K} = \mathbb{R}^n_+\), the modeling of this case is particularly straightforward:

\[
\begin{align*}
\text{minimize} & \quad \bar{c}^T x \\
\text{subject to} & \quad \bar{A}x \leq \bar{b} \\
& \quad \bar{A}x \geq \bar{b} \quad (23) \\
& \quad x \geq 0,
\end{align*}
\]

where \((\mathbf{A}, \mathbf{b}, \mathbf{c})\) and \((\bar{\mathbf{A}}, \bar{\mathbf{b}}, \bar{\mathbf{c}})\) denote the interval lower and upper bounds of \((\mathbf{A}, \mathbf{b}, \mathbf{c})\), respectively. The actual parameters for a ‘best case’ realization, i.e., a problem instance for which this optimal objective value is attained, can be derived easily from a solution to (23). Of course, this program can also be transformed to the standard form (LP) by introducing 2m slack variables.

When dealing with semidefinite programming problems, interval perturbations are not very suitable for modeling inaccuracies. An efficient transformation as above is typically not possible. A better choice is to model the inaccuracies by a quantity that is compatible with the semidefinite cone, such as

\[
\{ C \in \mathbf{S}^n \mid \bar{C} \succeq S_s, C \succeq S_s, C \}
\]

for the parameter matrix \(C\) of the objective function. Inaccuracies in \(A_i\) can be described in the same way and the right-hand sides of the equality constraints remain to be modeled using intervals. Similar adaptations are advisable for other cones \(K\).

If the uncertainty model is compatible with the cone structure, this allows an efficient computation of an optimal solution to a ‘best case’ realization.

4.1 Basis stability and ‘worst case’ realizations

Let \(\mathcal{K}\) be the nonnegative orthant of dimension \(n\), i.e., \(\mathcal{K} = \mathbb{R}^n_+\), so that (I-CLP) describes an interval linear system. If all realizations of this interval family have primal and dual unique non-degenerate solutions, then all these solutions have necessarily the same basis \(B\). This is called a basis stable interval linear programming problem.

The verification method based on the conditions (6) and (7) that was reviewed in Subsection 2.1 can be generalized nicely for interval families of linear programming problems. The procedure is as before. Given an approximate solution to one of the realizations of (I-CLP), typically the midpoint realization, we determine the basis \(B\). Then we solve the linear systems in (6) and (7), but now for interval quantities, i.e.,

\[
\mathbf{A}_B^T \mathbf{x}_B = \mathbf{b} \quad \text{and} \quad \mathbf{A}_B^T \mathbf{y} = \mathbf{c}_B.
\]

In practice, this requires the regularity of all \(A_B \in \mathbf{A}_B\). Solutions to the interval systems above can be computed by the methods introduced in [8–10]. The computed enclosures \(\mathbf{x}_B\) and \(\mathbf{y}\) will not be as tight as before, but they contain the solutions to all linear systems in the corresponding interval family. Finally, we have to check the condition (8) to verify basis stability and the enclosure of all optimal points in the interval family of linear programming problems.

If uncertainties in the input data are relatively small and the problem instances in the interval linear programming problem are well-conditioned, then also the diameter of \(\mathbf{x}_B\) and \(\mathbf{y}\) will be relatively small. Hence we obtain narrow enclosures for the actual set of optimal solutions. Nevertheless, due to the

\[\footnote{Actually there are different variants of basis stability. For example, in [50], the author uses the notion of B-stability and unique non-degenerate B-stability.}\]

347
wrapping effect\textsuperscript{6}, the enclosure of the actual set of optimal objective values via $c^T_B x_B$ or $b^T y$ can be comparably inaccurate.

Let $[\hat{\varrho}, \tilde{\varrho}]$ denote the interval consisting of the optimal objective values of all realizations of (I-CLP), where the lower bound is defined by (22) and the upper bound is defined complementarily:

$$\begin{align*}
\tilde{\varrho} &:= \sup_{d \in d} \inf_{x \in K} \{ c^T x \mid Ax = b \}.
\end{align*}$$

A tight enclosure for the lower bound $\hat{\varrho}$ can be computed by applying the previously discussed verification methods to the model (23) for the ‘best case’ analysis. This works for every primal standard linear programming problem without free variables and independent of basis stability. On the other hand, in the presence of free variables and without basis stability, the computation of $\hat{\varrho}$ is NP-hard [51]. The same is true for the computation of the upper bound $\tilde{\varrho}$.

The good news is that basis stability typically allows to compute tight enclosures $y$ for the set of all dual optimal solutions. This enables us to set the signs of many or all dual variables. If there is no entry $y_i$ of $y$ such that $0 \in \text{int} y_i$, then we know the signs of all free variables for all possible solutions. Let $A_+$ and $A_-$ denote the matrices consisting of the rows $i$ of $A$ for which $y_i \geq 0$ and $y_i < 0$, respectively. The right-hand side $b$ shall be partitioned accordingly into $b_+$ and $b_-$. Then,

$$\begin{align*}
\minimize_x \quad & c^T x \\
A_+ x & = b_+ \\
A_- x & = b_- \\
x & \geq 0
\end{align*}$$

is a ‘best case’ realization with optimal objective value $\hat{\varrho}$, and

$$\begin{align*}
\minimize_x \quad & c^T x \\
A_+ x & = b_- \\
A_- x & = b_+ \\
x & \geq 0
\end{align*}$$

is a ‘worst case’ realization with optimal objective value $\tilde{\varrho}$. This connection can be easily seen in the dual problem formulation where ‘best case’ and ‘worst case’ are naturally swapping places. Moreover, since we already know the optimal basis, we can compute the solutions to these two problem instances directly, by which we also obtain a tight enclosure for $[\hat{\varrho}, \tilde{\varrho}]$.

The number of (LP) candidates, that need to be checked to find one with largest optimal objective value, grows exponentially with the number of variables with uncertain sign. If there are only a few entries $y_i$ with uncertain sign, we may still tackle the problem of computing $\tilde{\varrho}$ using a branch-and-bound approach.

Free variables in the primal problem instance can be handled in a very similar way. Many further information on interval linear programming and a lot more references are given in [50, 51].

The described procedure can be carried over for the treatment of more general cone-LP problems. There are however several difficulties. One of these is the question about basis stability. It is not immediately clear how this term can be generalized for arbitrary convex cones. One sufficient condition for the applicability of the verification method described in Subsection 3.1 is that all instances in the interval (CLP) family have unique strictly complementary solutions. If the interval Jacobian matrix to the corresponding KKT system is regular, we can apply the Krawczyk operator to compute a verified enclosure for a set of solutions for all realizations of (I-CLP).

For this enclosure it is still necessary to prove the strict complementary condition. We have already dealt with interval enclosures of solutions also in the absence of parameter uncertainties. The generalization for interval programming problems is therefore straightforward. As an example, consider the

\textsuperscript{6}The *wrapping effect* is a term that is often used in the context of interval ODEs. Beyond that, it generally refers to the overestimation that results from embedding non-interval objects in interval enclosures.
condition (20) for verified solution enclosures in semidefinite programs. To verify that this condition is satisfied for the whole interval family of problems, we simply replace the parameters with their interval quantities and check

$$X + C - \sum_{i=1}^{m} y_i A_i \subseteq \text{int} S_r.$$ 

Other conditions such as (18) or (19) are adapted similarly. The efficient construction of ‘best case’ and ‘worst case’ realizations as shown above, however, requires the uncertainties to be modeled compatible with the cone structure.

4.2 Rigorous bounds for interval problems

Regularity of the interval Jacobian matrix to the KKT system is a very beneficial property that enables us to compute narrow verified enclosures for the actual solution set. However, there are many problems occurring in practice without this property. In particular, basis stability is not a generic property of interval linear programming problems. Let \( \hat{x} \) be a (possibly unique) solution to a given instance of (LP) with basis \( B \). If the corresponding optimal objective value \( \hat{\rho} = c^T \hat{x} \) is not significantly smaller than for any other feasible basis, then even slight perturbations of the problem may change the basis of its solution. Now assume that small inaccuracies in the input data are modeled using intervals yielding an interval family of (LP) realizations. It is then very likely that the interval linear programming problem is not basis stable. In these cases the interval Jacobian matrix to the KKT system contains singular realizations and the problem to compute narrow enclosures for the solution set becomes ill-posed.

Moreover, even if the computation of a narrow interval enclosure for the solution set was possible, it would typically be too large to be of practical use. In particular, bounds for the objective values \( \bar{\hat{\rho}} \) and \( \bar{\hat{\rho}} \) obtained from these interval enclosures tend to be strongly overestimated. This statement traces back to the wrapping effect.

For the purpose of illustration, consider the interval linear programming problem specified by

\[
A = \begin{bmatrix} -1 & 8 & 1 & 4 \\ 1 & 1.9 & 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 8 \\ [1.97, 2.02] \end{bmatrix}, \quad c = \begin{bmatrix} 8 \\ 4 \\ 8 \\ 1.9 \end{bmatrix}, \quad K = \mathbb{R}^4_+.
\]

(24)

Only the second component of \( b \) is a proper interval. The primal solution set consists of the two line segments between the vertices

\[
\hat{x}^{(1)} = \begin{bmatrix} 0 \\ 0.3 \\ 0 \\ 1.4 \end{bmatrix}, \quad \hat{x}^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix}, \quad \hat{x}^{(3)} = \begin{bmatrix} 0.016 \\ 0 \\ 0 \\ 2.004 \end{bmatrix}.
\]

To be more precise, to every point

\[
\hat{x} \in \{ \alpha \hat{x}^{(1)} + (1 - \alpha)\hat{x}^{(2)} \mid \alpha \in [0, 1] \} \cup \{ \alpha \hat{x}^{(2)} + (1 - \alpha)\hat{x}^{(3)} \mid \alpha \in [0, 1] \},
\]

there exists a realization of the interval linear programming problem above such that \( \hat{x} \) is the unique primal optimal solution. The dual solution set is just the line segment between the vertices

\[
\hat{g}^{(1)} = \begin{bmatrix} 0.975 \\ -2 \end{bmatrix}, \quad \hat{g}^{(2)} = \begin{bmatrix} -1.22 \\ 6.78 \end{bmatrix},
\]

whereby, for all but one primal degenerate realization, the dual solution is unique and either of the two vertices.

The best possible interval enclosures for the primal and dual optimal set yield the following enclosures for the range of possible objective values:
respectively. Particularly the enclosure from the dual solution set overestimates the actual range $[3.8, 3.9356]$ drastically.

An early approach to tackle this problem is due to Jansson [52]. In his thesis, he considers interval linear programming problems that are not basis stable. He proved that the graph to the set of optimal bases is connected and introduced an efficient algorithm that walks through this graph to check every candidate basis to compute lower and upper bounds for the optimal objective value. For the example above, there are two candidate bases: $B^{(1)} = \{1, 4\}$ and $B^{(2)} = \{2, 4\}$. By using the method in [52], we compute the enclosure $[3.76, 3.9356]$, which approximates the actual range of possible objective values very well.

The major issue of this approach is that the number of candidate bases grows exponentially fast with increasing relative diameter of the parameter intervals and the degree of degeneracy. Hence, the approach is typically only practical for small inaccuracies modeled by the interval input data and a small degree of degeneracy introduced by the model itself.

In [14], the same author described a new verification approach that is the base for the algorithms in Subsection 3.2. As long as the interval conic linear programming problem contains no primal nor dual ill-posed realizations, we can apply these algorithms to compute enclosures for $[\tilde{\varrho}, \hat{\varrho}]$.

To efficiently compute an approximate solution, we choose a realization $(A, b, c)$ in the interval family of conic linear programming problems and ask the oracle for an approximate solution. A usual choice for $(A, b, c)$ are the midpoints of the corresponding interval quantities. This realization is then used in the algorithms (c1)–(c5) and (d1)–(d5). The only adaptions that are necessary for handling interval input data is to replace the underdetermined linear system in (c2), (d2) as well as the condition in (d3) with their interval equivalents, respectively.

This method is implemented in VSDP [20]. By using the mid-point realization for an approximate solution together with the iteration scheme implemented in VSDP, we verify that a floating-point approximation of the interval vector $x = \frac{1}{99} \begin{pmatrix} 2, 6 \\ 69.7, 70.2 \\ 2 \\ 58.6 \end{pmatrix}$ contains primal feasible points for all realizations of the interval LP specified in (24). Even though the returned interval vector $x$ is far from optimal, the derived upper bound $\sup c^T x = \frac{22807}{4950} \approx 4.6075$ is significantly better than in the initial enclosure (25). For the lower bound, the situation is even better. VSDP verifies a floating-point approximation of $\hat{y}^{(1)}$ as dual feasible point (for all realizations), which yields $\inf b^T \hat{y}^{(1)} = 3.76$ and thereby the range inclusion

$[3.76, 4.6075] \supseteq [3.8, 3.9356]$.

The reason why the upper bound of the method described in [52] is so much better than the upper bound returned by VSDP lies in the specific structure of our example. On the other hand, the enclosures for the corresponding dual optimal solutions are very tight because there is a single proper interval in the dual objective function. It is possible to construct other interval problems for which the implementation in VSDP leads to tighter bounds, and often by magnitudes faster so.

### 4.3 Ill-posed and inconsistent interval cone-LPs

The generalization of Farka’s lemma for interval families of conic linear programming problems is very much straightforward. We may simply apply the results given in Subsection 2.5 but now with interval parameters. An efficient application of this method requires that all realizations of the interval family have nearby improving rays. If the verification procedure is successful, then (primal or dual) inconsistency is a stable property of the corresponding interval system.
A different case occurs when uncertainties in the input data meet an optimization model that introduces ill-posedness. In some special cases all realizations of the interval family will be ill-posed but consistent. An example for this is the formulation (LAP-LP) for the linear assignment problem. There, uncertainties in the input data only affect the coefficients of the objective function and do not interact with the cause of ill-posedness. In general, however, the interval programming problem will have well-posed inconsistent, ill-posed and well-posed consistent realizations.

The relaxation program (21) is an example that shows this characteristic whenever there are uncertainties in the parameter matrix $G$. On the other hand, the interval version of (21) is also a good example to demonstrate the applicability of Theorem 2. Let $(F, G, H)$ denote the parameter matrices of the quadratic assignment problem with an uncertainty model using intervals. The different realizations of $G \in G$ are accompanied by differently perturbed constraints in the relaxation program (21). The beneficial property of all these realizations is that in all of them the decision variables are bounded. In particular, the decision variable $Z$, that is used in (21) to relax the term $XG^2X^T$, satisfies

$$\text{tr}(Z) \leq \sup \text{tr}(G^2)$$

for all realizations of the interval semidefinite programming problem. This estimate is sharp, i.e., there exists an instance of (21) with input data $(F, G, H) \in (F, G, H)$ such that the estimate is satisfied with equality, and the upper bound $\sup \text{tr}(G^2)$ can be computed exactly\(^7\) with $O(r^2)$ operations.

The program (21) is a relaxation for a combinatorial problem of which we know that it has a solution. Hence the consistency of (21) is known a priori and independent of any parameter uncertainties. We can therefore simply ignore all inconsistent realizations. An efficient use of Theorem 2 requires the following three quantities:

1. an approximate dual solution vector $\tilde{y}$ for a consistent realization of (21),
2. an upper bound $\bar{x}$ for the optimal primal solutions of all consistent realizations, which can be derived by exploiting (26), and
3. a lower bound $\bar{d}$ for \(\{0, c - A^T \tilde{y}\}\) where the parameters are replaced with their interval quantities.

Without a priori knowledge about its consistency and the satisfaction of some boundedness criteria, we have no means to attack such problems, but in special cases like these, even the presence of inconsistent and ill-posed programming realizations have no affect on the possibility to compute tight verified lower bounds.

### 4.4 Interval Computations in Semidefinite Programming

In the world of reliable computing, interval arithmetic is an extremely beneficial tool. The following are just some of the benefits of and reasons to use interval arithmetic:

- It is comparable easy to implement and there are various efficient implementations available,
- There is a lot of ongoing research on this topic,
- Intervals can be used to model parameter uncertainties,
- In combination with monotone rounding, interval arithmetic is a suitable tool to trap any rounding errors in the computation, and
- The inclusion principle of interval arithmetic is a fundamental requirement for most verification methods.

On the other hand, the biggest drawback of interval arithmetic is possible overestimation caused by error propagation, unresolved data dependencies and the wrapping effect.

In the last part of this section we give special consideration to some of the difficulties and pitfalls when working with interval arithmetic. We do so in the context of verification methods for

\(^7\)Here ‘exactly’ is meant in the sense of a best possible floating-point approximation.
semidefinite programming problems. There are three essential ingredients to the methods discussed in Subsection 3.4:

(t1) the proof of positive (semi)definiteness in the steps (c3), (d3) and the condition (20),

(t2) positivity of the \( k \) and \( r - k \) largest eigenvalues in (19), and

(t3) the computation of a lower negative semidefinite bound as requirement for (16) in Theorem 2.

In the regard of an interval semidefinite programming problem, these tasks have to be implemented not only for a single instance but for every realization of a symmetric interval matrix. The good news is that all three tasks are very similar regarding applicable verification approaches. This means that they are correlated in such a way that, if we find a method to solve one of these tasks, then this method can be easily adapted for the others. This correlation will become more clear later on in this subsection. For reasons of clarity, we concentrate on the task of verifying \( \lambda_k(X) > 0 \) for all \( X \in \mathbf{X} \) that is listed under (t2).

An obvious strategy to tackle this problem is to compute verified bounds for the eigenvalues of the considered matrices. A good base for this purpose is Kahan's residual bound\(^\text{8}\).

**Theorem 3** (Cao et al. [45]). Let the \( r \times r \) symmetric matrix \( X \) have eigenvalues \( \lambda_1, \ldots, \lambda_r \), let \( H \) be a \( k \times k \) symmetric matrix with the eigenvalues \( \mu_1, \ldots, \mu_k \), and let \( V \) denote an \( r \times k \) matrix with full column rank. Then there is a subset of \( k \) eigenvalues \( \lambda_{i_1}, \ldots, \lambda_{i_k} \) of \( X \) satisfying

\[
\max_{1 \leq j \leq k} |\lambda_{i_j} - \mu_j| \leq \frac{\| XV - VH \|_2}{\sigma_{\text{min}}(V)}, \tag{27}
\]

where \( \sigma_{\text{min}}(V) \) is the smallest singular value of \( V \), and \( \| \cdot \|_2 \) denotes the spectral norm.

In our application, \( V \) is the matrix consisting of approximations for the eigenvectors corresponding to the \( k \) largest eigenvalues of some \( X \in \mathbf{X} \) and \( H \) is a diagonal matrix that contains the corresponding eigenvalue approximations. Since \( V \) can be assumed to be numerically orthogonal, we may use \( \sqrt{1 - \| I - VTV \|_2} \) as a lower bound for \( \sigma_{\text{min}}(V) \), see for instance [9]. If we replace \( X \) in (27) with the interval set \( \mathbf{X} \), the bounds on the left-hand side hold true for \( k \) eigenvalues of any \( X \in \mathbf{X} \). To be precise:

**Corollary 1.** Let \( \mathbf{X} \in \mathbb{IR}^{r \times r} \cap \mathcal{S}^r \), let \( H \in \mathcal{S}^k \) with eigenvalues \( \mu_1, \ldots, \mu_k \), and let \( V \in \mathbb{IR}^{r \times k} \) satisfy \( \| I - VT \|_2 < 1 \). Then there is an index set \( i_1, \ldots, i_k \) such that

\[
\forall X \in \mathbf{X} \colon \max_{1 \leq j \leq k} |\lambda_{i_j}(X) - \mu_j| \leq \frac{\sup \| XV - VH \|_2}{\sqrt{1 - \| I - VT \|_2}}. \tag{28}
\]

An upper bound for the supremum of \( \| XV - VH \|_2 \) over all \( X \in \mathbf{X} \) is easily computed, for instance by evaluating \( \sqrt{\| XV - VH \|_1 \cdot \| XV - VH \|_\infty} \) using interval arithmetic. More sophisticated approaches, such as the implementation in INTLAB [54], are based on an approximate singular value decomposition. They are more accurate but also computationally more expensive. On the other hand, it is often possible to significantly improve the efficiency and accuracy of this approach by separating the set of eigenvalues into clusters. For more details we refer to [55].

The result above enables us to compute verified lower bounds for \( k \) largest eigenvalues of all \( X \in \mathbf{X} \) by which we can check the first condition in (19). The second inequality in (19) can be verified similarly. Hence, if we can compute enclosures \((\mathbf{X}, \mathbf{y})\) satisfying (18) and are able to verify (19) in the described way, then \( \mathbf{X} \) and more specifically \( \mathbf{X} \cap \mathcal{S}_k^r \cap \{ M \mid \text{rank}(M) = k \} \) includes a solution \( \hat{X} \) to (SDP).

The main issue of the described approach is not so much that we choose the same eigenvector approximation for all \( X \in \mathbf{X} \) but the possible overestimation when evaluating \( XV \) using interval arithmetic. To illustrate this fact we use the midpoint-radius formulation of an interval matrix:

\(^\text{8}\)Kahan proved this bound with an additional factor \( \sqrt{2} \) on the right-hand side [53]. The final version of this result was proved by Cao, Xie, and Lie [45].
to derive by exploiting the Gershgorin circle theorem [49]:

\[(X_m)[X_r] := X_m + [-X_r, X_r] = [X_m - X_r, X_m + X_r] = [X, \bar{X}] = \mathbf{X}.
\]

The interval enclosure for the residual $\mathbf{X}V - VH$ in (28) is

\[(X_mV - VH)[X_r|V].\]

Let us assume that the eigenvalue and eigenvector approximates for the midpoint matrix $X_m$ are comparably accurate. Then $\|X_mV - VH\|_2 < \|X_r\|_2$ and

\[\|(X_mV - VH)[X_r|V]\|_2 \approx \|X_r|V\|_2 \leq \|X_r\|_2 \cdot \|V\|_2 \approx \|X_r\|_2 \cdot \sqrt{k},\]

where $k$ is the number of columns of $V$. By using interval arithmetic in a straightforward manner, the residual norm in (28) is roughly estimated by

\[\|X_mV - VH\|_2 + \sqrt{k} \cdot \|X_r\|_2.\]

The actual situation is slightly better and for $k = 1$ the estimate using interval arithmetic is actually sharp, but the overestimation increases proportional to $\sqrt{k}$.

This issue can be circumvented by combining Kahan’s residual bound [53] with Weyl’s perturbation bound [44]. Together with the weak monotonicity property of the spectral bound [56], the latter yields

\[\forall X \in \mathbf{X}, i \in \{1, \ldots, r\}: \quad |\lambda_i(X) - \lambda_i(X_m)| \leq \|X - X_m\|_2 \leq \|X_r\|_2,
\]

so that (28) can be replaced with

\[
\forall X \in \mathbf{X}: \quad \max_{1 \leq j \leq k} |\lambda_j(X) - \mu_j| \leq \frac{\|X_mV - VH\|_2}{\sqrt{1 - \|I - VH\|_2}} + \|X_r\|_2. \tag{29}
\]

The data dependency problem is even more immanent when we apply interval arithmetic carelessly to verify the second condition in (19). Although the generalization of the following statements for interval input data is straightforward, for the sake of clarity, here we assume point parameter matrices $C$ and $A_i$ for $i = 1, \ldots, m$. By using interval arithmetic to first evaluate $\mathbf{Z} = C - \sum_{i=1}^{m} y_i A_i$ and then the residual $\mathbf{Z}V - VH$, we derive the enclosure

\[
((C - \sum_{i=1}^{m} (y_m) A_i)V - VH)[\sum_{i=1}^{m} (y_r) A_i|V]. \tag{30}
\]

The overestimation of the residual norm is evident. If $r - k$ is small or the sparsity structure of the parameter matrices is such that the $A_i$ have only few overlapping nonzero entries, it can be beneficial to change the order of evaluation of the residual to $CV - \sum_{i=1}^{m} y_i (A_i V) - VH$. This reduces the large radius in (30) significantly to $[\sum_{i=1}^{m} (y_r) A_i|V]$. If the computational overhead for this approach is too large, then we still can use a similar estimate as in (29).

Another possibility to tackle the data dependency problem in the example above is to avoid the computation of residual eigenvalue bounds altogether. In [46], Rump introduced an efficient method to prove positive definiteness of a symmetric point matrix. His approach uses the Cholesky decomposition in combination with some a priori error bounds. No costly eigenvalue decomposition is necessary. Since we work with interval symmetric matrices $\mathbf{X}$, the utilization of Rump’s result requires a lower bound for $X$ satisfying $X \geq_s X$ for all $X \in \mathbf{X}$.

For this purpose, we may once more exploit the weak monotonicity property of the spectral bound, to derive

\[\lambda_{\min}(X - X_m) \geq -\|X - X_m\|_2 \geq -\|X_r\|_2\]

for all $X \in \mathbf{X} = (X_m)[X_r] \cap \mathcal{S}^r$. This immediately yields the lower bound:

\[\mathbf{X} \geq_{s+} X_m - \|X_r\|_2 \cdot I,
\]

where $I$ denotes the identity matrix of appropriate size. Another simple lower bound can be derived by exploiting the Gershgorin circle theorem [49]:

353
where \( e \) is the vector of all ones and \( \text{diag}(X, e) \) denotes the diagonal matrix with elements \( X, e \) on its diagonal. To prove that \( X \subseteq S^r_m \), we then apply Rump's algorithm to the right-hand side of (31). This method is more efficient and typically also more accurate than computing verified bounds for the eigenvalues using Kahan's residual bound as in (28).

The same method can be easily extended for the other two verification tasks in (t1) and (t3). For instance, to verify that \( \lambda_k(X) > 0 \) for all \( X \in X \), one can approximate the eigenspace to the \( r - k \) smallest eigenvalues of \( X_m \), construct a matrix \( W \) whose columns specify an approximate basis to this eigenspace, and then verify \( X + WW^T \subseteq \text{int} S^r_m \) using the method above. The procedure to compute a negative semidefinite lower bound \( \bar{D} \) for \( C - \sum_{i=1}^m y_i A_i \) is similar. We would first compute the lower bound \( D \) approximately and then verify the semidefiniteness condition using the same method as before. If not successful in the first step, we perturb the matrix accordingly and try again.

When dealing with other cones such as the second order cone or the exponential cone, similar side effects have to be taken into account.

Another possible improvement of the accuracy can be achieved by reducing the overestimation in the evaluation of the objective value caused by the interval inclusion of a non-interval object. As an example, consider the algorithm (c1)–(c5) with interval input data. In step (c2) we are looking for an enclosure \( x \) for the set of solutions to all realizations of \( Ax = b \) using \( \tilde{x} \) as initial approximation. If the condition \( x \subseteq K \) can be verified in step (c3), the algorithm returns \( x \) and an upper bound for \( \bar{\rho} \) is computed via sup \( c^T x \). However, the actual solution set to the interval linear system \( Ax = b \) is rarely an interval vector, which results in overestimation due to the wrapping effect.

This effect can be reduced significantly by using the approach previously exploited by Rump in [57]. In his habilitation, he was concerned with the basis stable case only. Nevertheless, a very similar idea can be used here as well. Let us now extend the linear system in step (c2) as follows:

\[
\begin{pmatrix}
A & 0 \\
-c^T & -1
\end{pmatrix}
\begin{pmatrix}
x \\
t
\end{pmatrix}
= \begin{pmatrix}
b \\
0
\end{pmatrix}.
\]

Using the same verification method as before, the computed enclosure \( x \) will be very similar to the enclosure for the original system \( Ax = b \). If the condition in (c3) is satisfied, then not only \( x \) contains primal feasible points of all realizations of (I-CLP) but also \( t \) encloses the set of the corresponding objective values. The upper bound \( t \) for the objective value \( \bar{\rho} \) is usually noticeably better than sup \( c^T x \). A similar modification is also applicable for computing a lower bound of \( \bar{\rho} \) via algorithm (d1)–(d5). Beyond that, if the considered problem is an interval linear program, we may use a similar method to check each inequality individually with higher precision. The idea here is again very similar to the method described in [57] that was applied to basis stable interval linear programming problems.

5. Conclusion

In this note we surveyed verification methods for conic linear programming problems in finite dimensional real space. Special consideration has been given to linear and semidefinite programming as well as interval versions of these. It was shown that verification methods are not only applicable for well-posed problems with unique non-degenerate solutions but also for many degenerate instances. Under certain boundedness qualifications or other a priori knowledge, it is even possible to circumvent ill-posedness to some extend.

There are certain limitations in how much sparsity in the corresponding underdetermined linear systems can be exploited by a verification method. Hence the size of the problems that can be treated by these methods is more restricted than the size of problems that are tackled by approximate methods. Nonetheless, the discussed verification methods can indeed be applied to problems of very high dimension. The largest problem reported in [20] has about 24 million variables and 7000 equality constraints. Moreover, in [39] the same verification methods have been applied to LMI relaxations with sizes ranging between 2 598 370 and 19 814 462 variables, and between 7230 and 27 888 constraints.
If the parameters of the conic linear programming problem are subject to uncertainties which can be reliably bounded, then these may be modeled using intervals. As long as the diameter of these intervals are not too large, the adapted verification methods from the previous section will still lead to meaningful results, giving qualitative feedback on the ‘variability’ of the optimal objective values. On the other hand, as mentioned previously, uncertainty models which are compatible with the structure of $K$ are typically preferable to interval uncertainties. They enable us to efficiently compute much tighter bounds. This type of uncertainties have not been studied so well yet, but a lot of the theoretical background can be borrowed from robust optimization. Robust optimization is also the field of choice in the presence of significant uncertainties whose modeling would lead to relatively large diameters when using intervals.

It was not a topic of this note, but the presented methods can be also exploited for verified results in global optimization. There exists a wide range of verified and approximate methods for various global optimization problems. These include a wide range of verified and approximate methods for various global optimization problems. These include meta heuristics [58], backtracking search algorithms [59], trust region methods [60], divide-and-conquer approaches [61], constraint propagation [62], contractor programming [63], and polyhedral branch-and-cut approaches [64]. As an introduction to the different techniques for constrained global optimization, we recommend Neumaier’s review article in Acta Numerica [65]. Of particular interest is also the polyhedral branch-and-cut approach by Tawarmalani and Sahinidis [64]. Their implementation performs well for a wide range of global optimization problems and brought the authors the Beale-Orchard-Hays Prize. Due to the nature of their algorithm, it can be combined with the verification methods discussed above in order to compute rigorous results. However, there is no adequate implementation for this yet.

Acknowledgments
The author wish to thanks the anonymous referee for pointing out various inconsistencies in a previous version of this paper. The feedback helped a lot to improve the readability.

References
[1] P.J.C. Dickinson and L. Gijben, “On the computational complexity of membership problems for the completely positive cone and its dual,” Comput. Optim. Appl., vol. 57, no. 2, pp. 403–415, 2014.
[2] R.B. Kearfott, M.T. Nakao, A. Neumaier, S.M. Rump, S.P. Shary, and P. van Hentenryck, “Standardized notation in interval analysis,” Comput. Technol., vol. 15, no. 1, pp. 7–13, 2010.
[3] S.M. Rump, “Verification methods: Rigorous results using floating-point arithmetic,” Acta Numer., vol. 19, pp. 287–449, 2010.
[4] S. Oishi, K. Ichihara, M. Kashiwagi, T. Kimura, X. Liu, H. Masai, Y. Morikura, T. Ogita, K. Ozaki, S.M. Rump, K. Sekine, A. Takayasu, and N. Yamanaka, Principle of Verified Numerical Computations, Tokyo, Japan: Corona publisher, 2018. [in Japanese].
[5] R. Krawczyk, “Fehlerabh¨atzung bei linearer Optimierung,” Interval Mathematics (K. Nickel, ed.), pp. 215–222, Springer Berlin Heidelberg, 1975.
[6] A. Neumaier, Interval Methods for Systems of Equations, Encyclopedia of Mathematics and its Applications, Cambridge: Cambridge University Press, 1991.
[7] R.B. Kearfott, Verified Solution of Nonlinear Systems, pp. 145–167, Boston, MA: Springer US, 1996.
[8] T. Ogita and S. Oishi, “Fast verified solutions of linear systems,” Japan J. Indust. Appl. Math., vol. 26, no. 2-3, pp. 169–190, 2009.
[9] S.M. Rump, “Verified bounds for singular values, in particular for the spectral norm of a matrix and its inverse,” BIT Numer. Math., vol. 51, no. 2, pp. 367–384, 2011.
[10] S.M. Rump, “Accurate solution of dense linear systems, part ii: Algorithms using directed rounding,” J. Comput. Appl. Math., vol. 355.
[12] G. Pataki and L. Tunçel, “On the generic properties of convex optimization problems in conic form,” Math. Program., vol. 89, no. 3, pp. 449–457, 2001.
[13] J. Hadamard, “Sur les problèmes aux dérivées partielles et leur signification physique,” Princeton University Bulletin, vol. 13, pp. 49–52, 1902.
[14] C. Jansson, “Rigorous lower and upper bounds in linear programming,” SIAM J. Optim., vol. 14, no. 3, pp. 914–935, 2004.
[15] S.M. Rump, “Improved componentwise verified error bounds for least squares problems and underdetermined linear systems,” Numer. Algorithms, vol. 66, no. 2, pp. 309–322, 2014.
[16] C. Keil, Rigorous Error Bounds for Finite Dimensional Linear Programming Problems, Books on Demand, 2009.
[17] C. Keil, “Lurupa documentation.” http://www.ti3.tu-harburg.de/~keil/pub/LRFILP.ps.gz
[18] 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.), no. 05391 in Dagstuhl Seminar Proceedings, (Dagstuhl, Germany), Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.
[19] M. Lange and S.M. Rump, “Sharp estimates for perturbation errors in summations,” Math. Comp., vol. 88, pp. 349–368, 2019.
[20] C. Jansson, M. Lange, and K.T. Ohlhus, “VSDP: Verified SemiDefinite-quadratic-linear programming.” http://www.ti3.tuhh.de/jansson/vsdp/, 2018.
[21] J. Renegar, “Incorporating condition measures into the complexity theory of linear programming,” SIAM J. Optim., vol. 5, no. 3, pp. 506–524, 1995.
[22] M. Dür, B. Jargalsaikhan, and G. Still, “Genericity results in linear conic programming – a tour d’horizon,” Math. Oper. Res., vol. 42, no. 1, 2016.
[23] F. Alizadeh, J-P.A. Haeberly, and M.L. Overton, “Complementarity and nondegeneracy in semidefinite programming,” Math. Program., vol. 77, no. 1, pp. 111–128, 1997.
[24] F. Ordóñez and R.M. Freund, “Computational experience and the explanatory value of condition measures for linear optimization,” SIAM J. Optim., vol. 14, no. 2, pp. 307–33, 2003.
[25] “NETLIB linear programming library.” A collection of linear programming problems, http://www.netlib.org/lp/
[26] C. Keil and C. Jansson, “Computational experience with rigorous error bounds for the NETLIB linear programming library,” Reliab. Comput., vol. 12, no. 4, pp. 303–321, 2006.
[27] A. Neumaier and O. Shcherbina, “Safe bounds in linear and mixed-integer linear programming,” Math. Program., vol. 99, no. 2, pp. 283–296, 2004.
[28] J. Farlás, “Theorie der einfachen Ungleichungen,” J. Reine Angew. Math., pp. 1–27, 1902.
[29] J. Matousek and B. Gärtner, A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Universitext, Springer-Verlag Berlin Heidelberg, 2007.
[30] P.R. Chares, Cones and Interior-Point Algorithms for Structured Convex Optimization involving Powers and Exponentials, PhD dissertation, Université Catholique de Louvain, 2007.
[31] M. Liu and G. Pataki, “Exact duality in semidefinite programming based on elementary reformulations,” SIAM J. Optim., vol. 25, no. 3, pp. 1441–1454, 2015.
[32] B.F. Lourenço, M. Muramatsu, and T. Tsuchiya, “A structural geometrical analysis of weakly infeasible SDPs,” ArXiv e-prints, 2015.
[33] B.F. Lourenço, M. Muramatsu, and T. Tsuchiya, “Solving SDP completely with an interior point oracle,” ArXiv e-prints, 2015.
[34] J.-S. Pang and L. Qi, “Nonsmooth equations: Motivation and algorithms,” SIAM J. Optim., vol. 3, no. 3, pp. 443–465, 1993.
[35] E.A. Yildirim, “An interior-point perspective on sensitivity analysis in semidefinite programming,” Math. Oper. Res., vol. 28, pp. 649–676, 2003.
[36] G. Pataki, “The geometry of semidefinite programming,” Handbook of Semidefinite Programming: Theory, Algorithms, and Applications (H. Wolkowicz, R. Saigal, and L. Vandenberghe, eds.), pp. 29–65, Boston, MA: Springer US, 2000.
finite programming,” *SIAM J. Numer. Anal.*, vol. 46, no. 1, pp. 180–200, 2008.

[38] M. Lange, *Semidefinite relaxation approaches for the quadratic assignment problem*, PhD dissertation, Hamburg University of Technology, 2016.

[39] D. Chaykin, C. Jansson, F. Keil, M. Lange, K. Ohlhus, and S.M. Rump, “Rigorous results in electronic structure calculations,” tech. rep., 2016.

[40] C. Jansson, “A rigorous lower bound for the optimal value of convex optimization problems,” *J. Global Optim.*, vol. 28, no. 1, pp. 121–137, 2004.

[41] C. Jansson, “VSDP: Verified SemiDefinite Programming.” http://www.ti3.tuhh.de/jansson/vsdp2006/, 2006.

[42] C. Jansson, “On verified numerical computations in convex programming,” *Japan J. Indust. Appl. Math.*, vol. 26, no. 2–3, pp. 337–363, 2009.

[43] K.-C. Toh, M.J. Todd, and R.H. Tütüncü, “On the implementation and usage of SDPT3 – a Matlab software package for semidefinite-quadratic-linear programming, version 4.0,” *Handbook on Semidefinite, Conic and Polynomial Optimization* (M.F. Anjos and J.B. Lasserre, eds.), vol. 166 of *International Series in Operations Research and Management Science*, pp. 715–754, Springer US, 2011.

[44] H. Weyl, “Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlräumstrahlung),” *Math. Ann.*, vol. 71, no. 4, pp. 441–479, 1912.

[45] Z. Hao Cao, J. Jun Xie, and R.-C. Li, “A sharp version of Kahan’s theorem on clustered eigenvalues,” *Linear Algebra Appl.*, vol. 245, pp. 147–155, 1996.

[46] S.M. Rump, “Verification of positive definiteness,” *BIT Numer. Math.*, vol. 46, no. 2, pp. 433–452, 2006.

[47] C.S. Edwards, “A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem,” *Combinatorial Optimization II*, vol. 13 of *Mathematical Programming*, pp. 35–52, Springer Berlin Heidelberg, 1980.

[48] Y. Ding and H. Wolkowicz, “A low-dimensional semidefinite relaxation for the quadratic assignment problem,” *Math. Oper. Res.*, vol. 34, no. 4, pp. 1008–1022, 2009.

[49] S.A. Gershgorin, “Über die Abgrenzung der Eigenwerte einer Matrix,” *Izv. Akad. Nauk SSSR, Ser. Mat.*, vol. 7, no. 6, pp. 749–754, 1931.

[50] M. Hladík, “Interval linear programming: A survey,” *Linear Programming – New Frontiers in Theory and Applications* (Z. Ádám Mann, ed.), ch. 2, pp. 85–120, New York: Nova Science Publishers, 2012.

[51] J. Rohn, *Interval linear programming*, pp. 79–100, Boston, MA: Springer US, 2006.

[52] C. Jansson, *Zur linearen Optimierung mit unscharfen Daten*, PhD dissertation, Universität Kaiserslautern, 1985.

[53] W. Kahan, “Inclusion theorems for clusters of eigenvalues of Hermitian matrices,” tech. rep., Institute of Computer Science, University of Toronto, 1967.

[54] S.M. Rump, “INTLAB - INTerval LABoratory,” *Developments in Reliable Computing* (T. Csendes, ed.), pp. 77–104, Kluwer Academic Publishers, 1999.

[55] S. Miyajima, T. Ogita, and S. Oishi, “Fast verification for respective eigenvalues of symmetric matrix,” *Computer Algebra in Scientific Computing* (V.G. Ganzha, E.W. Mayr, and E.V. Vorozhtsov, eds.), pp. 306–317, Springer Berlin Heidelberg, 2005.

[56] R.A. Horn and C.R. Johnson, *Matrix Analysis*, New York, NY, USA: Cambridge University Press, 2nd ed., 2012.

[57] S.M. Rump, “Solving algebraic problems with high accuracy,” *A New Approach to Scientific Computation* (U.W. Kulisch and W.L. Miranker, eds.), pp. 51–120, New York: Academic Press, 1983.

[58] D.E. Kvasov and M.S. Mukhametzhanov, “Metaheuristic vs. deterministic global optimization algorithms: The univariate case,” *Appl. Math. Comput.*, vol. 318, pp. 245–259, 2018.

[59] P. van Beek, “Chapter 4 - Backtracking search algorithms,” *Handbook of Constraint Programming* (F. Rossi, P. van Beek, and T. Walsh, eds.), vol. 2 of *Foundations of Artificial Intelligence*,
[60] T. Montanher, A. Neumaier, and F. Domes, “A computational study of global optimization solvers on two trust region subproblems,” J. Global Optim., vol. 71, no. 4, pp. 915–934, 2018.

[61] S.M. Rump, “Mathematically rigorous global optimization in floating-point arithmetic,” Optim. Methods Softw., vol. 33, no. 4–6, pp. 771–798, 2018.

[62] C. Bessiere, “Chapter 3 - Constraint propagation,” Handbook of Constraint Programming (F. Rossi, P. van Beek, and T. Walsh, eds.), vol. 2 of Foundations of Artificial Intelligence, pp. 29–83, Elsevier, 2006.

[63] G. Chabert and L. Jaulin, “Contractor programming,” Artif. Intell., vol. 173, no. 11, pp. 1079–1100, 2009.

[64] M. Tawarmalani and N.V. Sahinidis, “A polyhedral branch-and-cut approach to global optimization,” Math. Program., vol. 103, no. 2, pp. 225–249, 2005.

[65] A. Neumaier, “Complete search in continuous global optimization and constraint satisfaction,” Acta Numer., vol. 13, pp. 271–369, 2004.