RIGOROUS ERROR BOUNDS FOR THE OPTIMAL VALUE IN SEMIDEFINITE PROGRAMMING

CHRISTIAN JANSSON[†], DENIS CHAYKIN[†], AND CHRISTIAN KEIL[†]

Abstract. A wide variety of problems in global optimization, combinatorial optimization as well as systems and control theory can be solved by using linear and semidefinite programming. Sometimes, due to the use of floating point arithmetic in combination with ill-conditioning and degeneracy, erroneous results may be produced. The purpose of this article is to show how rigorous error bounds for the optimal value can be computed by carefully postprocessing the output of a linear or semidefinite programming solver. It turns out that in many cases the computational costs for postprocessing are small compared to the effort required by the solver. Numerical results are presented including problems from the SDPLIB and the NETLIB LP library; these libraries contain many ill-conditioned and real life problems.

Key words. semidefinite programming, linear programming, interval arithmetic, rigorous error bounds, sensitivity analysis, SDPLIB, NETLIB LP library.

AMS subject classifications. 90C22, 65G30, 65N15

1. Introduction. We consider the *(primal) semidefinite program* in block diagonal form

$$p^* := \min \sum_{j=1}^{n} \langle C_j, X_j \rangle \quad \text{s.t.} \quad \sum_{j=1}^{n} \langle A_{ij}, X_j \rangle = b_i \quad \text{for } i = 1, \ldots, m, \qquad X_j \succeq 0 \quad \text{for } j = 1, \ldots, n, \tag{1.1}$$

where C_j, A_{ij}, and X_j are symmetric $s_j \times s_j$ matrices, $b \in \mathbb{R}^m$, and

$$\langle C, X \rangle = \text{ trace } (C^T X) \tag{1.2}$$

denotes the *inner product* for the set of symmetric matrices. Moreover, $\succeq$ is the *Löwner partial order*, that is $X \succeq Y$ iff $X - Y$ is positive semidefinite. In the case $n = 1$ we suppress the index j, and write shortly C, X, A_i, and s for the dimension.

If $s_j = 1$ for $j = 1, \ldots, n$ (i.e., C_j, A_{ij}, and X_j are real numbers), then (1.1) defines the standard linear programming problem. Hence, semidefinite programming is an extension of linear programming.

The *Lagrangian dual* of (1.1) is

$$d^* := \max b^T y \quad \text{s.t.} \quad \sum_{i=1}^{m} y_i A_{ij} \preceq C_j \quad \text{for } j = 1, \ldots, n, \tag{1.3}$$

where $y \in \mathbb{R}^m$. The constraints $\sum_{i=1}^{m} y_i A_{ij} \preceq C_j$ are called *linear matrix inequalities (LMI)*. We use the convention that $p^* = -\infty$ if (1.1) is unbounded and $p^* = \infty$ if (1.1) is infeasible. The analogous convention is used for (1.3).

The duality theory is similar to linear programming, but more subtle. The programs satisfy the *weak duality* condition

$$d^* \leq p^*, \tag{1.4}$$

[†]Institute for Reliable Computing, Hamburg University of Technology, Schwarzenbergstraße 95, 21071 Hamburg, Germany.

but strong duality requires in contrast to linear programming additional conditions (see for example Nemirovski [32], Ramana, Tunçel, and Wolkowicz [39] and Vandenberghe and Boyd [49]).

THEOREM 1.1 (Strong Duality Theorem).

a) If (1.1) is strictly feasible (i.e., there exist feasible positive definite matrices X_j for $j = 1, \ldots, n$) and p^* is finite, then $p^* = d^*$ and the dual supremum is attained.

b) If (1.3) is strictly feasible (i.e., there exists some $y \in \mathbb{R}^m$ such that $C_j - \sum_{i=1}^m y_i A_{ij}$ are positive definite for $j = 1, \ldots, n$) and d^* is finite, then $p^* = d^*$, and the primal infimum is attained.

In general, one of the problems (1.1) and (1.3) may have optimal solutions while its dual is infeasible, or the duality gap may be positive at optimality. The strict feasibility assumptions in Theorem 1.1 are called *Slater's constraint qualifications*.

Semidefinite programming and LMI-methods are documented by many applications and a number of survey papers (for example Skelton and Iwasaki [45], Balakrishnan and Feron [3], and Vandenberghe and Boyd [49]). Applications include global optimization problems, optimal state space realizations, robust controller design, integer programming problems, as well as eigenvalue problems in the form of minimizing the largest, or minimizing the sum of the first few largest eigenvalues of a symmetric matrix X subject to linear constraints on X.

Semidefinite programs can be solved in polynomial time if an a priori bound for the size of their solution is known (see M. Grötschel, L. Lovász, and A. Schrijver [12]). This is a consequence of the ellipsoid method for convex programming. The ellipsoid method has not proven practical, and interior point methods turned out to be the method of choice in semidefinite programming.

Conventionally, algorithms assume that the input data are given exactly, and they use floating point arithmetic for computing an approximate solution. Occasionally, wrong results may be produced, not solely but especially for ill-conditioned and ill-posed problems in the sense defined by Renegar [40]. He defines the condition number as the scale-invariant reciprocal of the smallest data perturbation that will render the perturbed data instance either primal or dual infeasible. It is set to ∞ if the distance to primal or dual infeasibility is 0, and in this case the problem is called ill-posed. Examples where commercial solvers fail to solve linear optimization problems can be found in Neumaier and Shcherbina [37], and in [17]. It cannot be answered how frequently such failures occur. Ill-conditioning is, however, frequently observed. In a paper by Ordóñez and Freund [38] it is stated that 71% of the LP-instances in the NETLIB Linear Programming Library [33] are ill-posed, i.e., the problems have an infinite condition number. Recently, Freund, Ordóñez and Toh [9] solved 85 out of the 92 problems of the SDPLIB [5] with SDPT3 [47] and investigated the interior-point iteration counts with respect to different measures for semidefinite programming problems. They omitted the four infeasible problems and three very large problems where SDPT3 ran out of memory. Of the remaining 85 problems they have shown 32 to be ill-posed.

As pointed out in Neumaier and Shcherbina [37], ill-conditioning is also likely to take place in combinatorial optimization when branch-and-cut procedures sequentially generate linear or semidefinite programming relaxations. Therefore, the computation of rigorous error bounds, which take account of all rounding errors and of small errors in the input data, is valuable in practice.

The primary purpose of this paper is to show that by properly postprocessing

the output of a semidefinite or linear solver, rigorous error bounds for the optimal value can be obtained. Moreover, existence of optimal solutions can be proved, or a certificate of infeasibility can be given. The input data are allowed to vary within small intervals. Our numerical experience with the NETLIB LP library and the SDPLIB demonstrates that, roughly speaking, rigorous lower and upper error bounds for the optimal value are computed even for ill-conditioned and degenerate problems. The quality of the error bounds depends on the quality of the computed approximations and the distances to dual and primal infeasibility. By comparing these bounds, one knows whether the computed results are good.

The presented results can be viewed as a further development of similar methods for linear programming (Neumaier and Shcherbina [37], and [17]) and convex programming [16].

The paper is organized as follows. Section 2 contains notation. In Section 3 an algorithm for computing a rigorous lower bound of the global minimum value is considered, and in Section 4 a rigorous upper bound of the optimal value together with a certificate of existence of optimal solutions is presented. In Section 5 we show how these rigorous bounds can be used for obtaining certificates of infeasibility. Section 6 contains numerical results and some remarks on other software packages. Finally, in Section 7 some conclusions are given.

2. Notation, interval arithmetic. Throughout this paper we use the following notation. We denote by $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^n_+$, and $\mathbb{R}^{m\times n}$ the sets of real numbers, real vectors, real nonnegative vectors, and real $m \times n$ matrices, respectively. Comparisons $\leq$, absolute value $|\cdot|$, min, max, inf, and sup are used entrywise for vectors and matrices. The identity matrix is denoted by I_d.

For a symmetric matrix A the eigenvalues are sorted non-increasingly, $\lambda_{\max}(A) = \lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_{\min}(A)$.

For $\mu \in \mathbb{R}$ the operator

$$\operatorname{svec}(A, \mu) := (A_{11}, \mu A_{21}, \ldots, \mu A_{n1}, A_{22}, \mu A_{32}, \ldots, \mu A_{nn-1}, A_{nn})^T, \tag{2.1}$$

transforms symmetric $n \times n$ matrices into $(n + 1)n/2$ dimensional vectors with the property that the inner product of two symmetric matrices A, B is

$$\langle A, B\rangle = \operatorname{svec}(A, 2)^T \operatorname{svec}(B, 1) = \operatorname{svec}(A, \sqrt{2})^T \operatorname{svec}(B, \sqrt{2}), \tag{2.2}$$

and $\operatorname{svec}(A, \sqrt{2})$ is the customary svec operator. We prefer the first representation of the inner product, since this avoids conversion errors of the input data of semidefinite programs in its vector representation form. The inverse operator of svec is denoted by $\operatorname{smat}(a, \mu)$ where a is the vector representation (2.1).

For block matrices with blocks A_j for $j = 1, \ldots, n$ we define the concatenated vector

$$\operatorname{svec}((A_j), \mu) := (\operatorname{svec}(A_1, \mu); \ldots; \operatorname{svec}(A_n, \mu)). \tag{2.3}$$

We require only some elementary facts about interval arithmetic, which are described here. There are a number of textbooks on interval arithmetic and self-validating methods that we highly recommend to readers. These include Alefeld and Herzberger [1], Moore [31], and Neumaier [34], [35].

If $\mathbb{V}$ is one of the spaces $\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, and $\underline{v}, \overline{v} \in \mathbb{V}$, then the box

$$\mathbf{v} := [\underline{v}, \overline{v}] := \{v \in \mathbb{V} : \ \underline{v} \leq v \leq \overline{v}\} \tag{2.4}$$

is called an *interval quantity* in $\mathbb{IV}$ with *lower bound* $\underline{v}$ and *upper bound* $\overline{v}$. In particular, $\mathbb{IR}$, $\mathbb{IR}^n$, and $\mathbb{IR}^{m\times n}$ denote the set of real intervals $\mathbf{a} = [\underline{a}, \overline{a}]$, the set of real interval vectors $\mathbf{x} = [\underline{x}, \overline{x}]$, and the set of real interval matrices $\mathbf{A} = [\underline{A}, \overline{A}]$, respectively. The real operations $A \circ B$ with $\circ \in \{+, -, \cdot, /\}$ between real numbers, real vectors, and real matrices can be generalized to *interval operations*. The result $\mathbf{A} \circ \mathbf{B}$ of an interval operation is defined as the interval hull of all possible real results, that is

$$\mathbf{A} \circ \mathbf{B} := \cap\{\mathbf{C} \in \mathbb{IV} : \ A \circ B \in \mathbf{C} \quad \text{for all} \quad A \in \mathbf{A}, B \in \mathbf{B}\}. \tag{2.5}$$

All interval operations can be easily executed by working appropriately with the lower and upper bounds of the interval quantities. For example, in the simple case of addition, we obtain

$$\mathbf{A} + \mathbf{B} = [\underline{A} + \underline{B}, \overline{A} + \overline{B}]. \tag{2.6}$$

Interval multiplications and divisions require a distinction of cases. Similarly all operations (2.5) between interval vectors and interval matrices can be executed. For example the i, j component of the product of two interval matrices $\mathbf{C}, \mathbf{X} \in \mathbb{IR}^{n\times n}$ is

$$(\mathbf{CX})_{ij} = \sum_{k=1}^{n} \mathbf{C}_{ik}\mathbf{X}_{kj}. \tag{2.7}$$

and the inner product

$$\langle \mathbf{C}, \mathbf{X} \rangle = \ \text{trace}\ (\mathbf{C}^T\mathbf{X}) = \sum_{i,j=1}^{n} \mathbf{C}_{ij}\mathbf{X}_{ij}. \tag{2.8}$$

For interval quantities $\mathbf{A}, \mathbf{B} \in \mathbb{IV}$ we define

$$\text{mid}\mathbf{A} := (\underline{A} + \overline{A})/2 \quad \text{as the } \textit{midpoint}, \tag{2.9}$$
$$\text{rad}\mathbf{A} := (\overline{A} - \underline{A})/2 \quad \text{as the } \textit{radius}, \tag{2.10}$$
$$|\mathbf{A}| := \max\{|A| : \ A \in \mathbf{A}\} \quad \text{as the } \textit{absolute value}, \tag{2.11}$$
$$\mathbf{A}^+ := \max\{0, \overline{A}\}, \tag{2.12}$$
$$\mathbf{A}^- := \min\{0, \underline{A}\}. \tag{2.13}$$

Moreover, the comparison in $\mathbb{IV}$ is defined by

$$\mathbf{A} \le \mathbf{B} \quad \text{iff} \quad \overline{A} \le \underline{B},$$

and other relations are defined analogously. Real quantities v are embedded in the interval quantities by identifying $v = \mathbf{v} = [v, v]$.

We call $\mathbf{A} \in \mathbb{IR}^{n\times n}$ *symmetric*, if $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ for all i, j, and $\mathbf{A}$ is called positive semidefinite if all $A \in \mathbf{A}$ have this property.

For linear systems of equations with inexact input data, the aim frequently is to compute an interval vector $\mathbf{x} \in \mathbb{IR}^n$ containing the *solution set*

$$\Sigma(\mathbf{A}, \mathbf{b}) := \{x \in \mathbb{R}^n : \ Ax = b \text{ for some } (A, b) \in (\mathbf{A}, \mathbf{b})\}, \tag{2.14}$$

where $\mathbf{A} \in \mathbb{IR}^{n\times n}$, and $\mathbf{b} \in \mathbb{IR}^n$. This is an NP-hard problem, but there are several methods that compute enclosures $\mathbf{x}$. A precise description of such methods, required

assumptions, and approximation properties can be found for example in Neumaier [34]. Roughly speaking, it turns out that for interval matrices with $\|I_d - R\mathbf{A}\| < 1$ (R is an approximate inverse of the midpoint mid$\mathbf{A}$) there are several methods which compute an enclosure $\mathbf{x}$ with $O(n^3)$ operations. The radius rad$\mathbf{x}$ decreases linearly with decreasing radii rad$\mathbf{A}$ and rad$\mathbf{b}$. For the computation of enclosures in the case of large-scale linear systems the reader is referred to Rump [42].

In interval arithmetic several methods for computing rigorous bounds for all or some eigenvalues of interval matrices were developed. Some important references are Floudas [7], Mayer [30], Neumaier [36], and Rump [42, 43].

3. Rigorous lower bound. In many applications some or all input data are uncertain. We model these uncertainties by intervals. In the case of semidefinite programming we assume that symmetric interval matrices $\mathbf{C}_j, \mathbf{A}_{ij} \in \mathbb{IR}^{s_j \times s_j}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, and an interval vector $\mathbf{b} \in \mathbb{IR}^m$ are given. This yields a family of semidefinite programs (1.1), where the input data $P = (A, b, C)$ are allowed to vary within interval bounds $\mathbf{P} := (\mathbf{A}, \mathbf{b}, \mathbf{C})$.

In order to indicate the dependency on the input data, we sometimes write $p^*(P)$, $d^*(P)$, $X^*(P)$, etc.

First, we state a lemma proving a lower bound for the inner product of two symmetric matrices.

LEMMA 3.1. *Let D, X be symmetric matrices of dimension s that satisfy*

$$\underline{d} \le \lambda_{\min}(D), \quad 0 \le \lambda_{\min}(X), \quad \textit{and} \quad \lambda_{\max}(X) \le \overline{x}. \tag{3.1}$$

Then

$$\langle D, X \rangle \ge s \cdot \underline{d}^- \cdot \overline{x}, \tag{3.2}$$

where $\underline{d}^- := \min\{0, \underline{d}\}$.

Proof. Let D have the eigenvalue decomposition

$$D = Q\Lambda(D)Q^T, \quad QQ^T = I_d,$$

where $\Lambda(D)$ is the diagonal matrix with eigenvalues of D on the diagonal. Then

$$\begin{aligned} \langle D, X \rangle &= \operatorname{trace}(Q\Lambda(D)Q^T X) \\ &= \operatorname{trace}(\Lambda(D)Q^T X Q) \\ &= \sum_{k=1}^{s} \lambda_k(D) Q(:,k)^T X Q(:,k). \end{aligned}$$

Because of (3.1), we have $0 \le Q(:,k)^T X Q(:,k) \le \overline{x}$ yielding

$$\langle D, X \rangle \ge \sum_{k=1}^{s} \lambda_k(D)^- \cdot \overline{x} \ge s \cdot \underline{d}^- \cdot \overline{x}. \qquad \square$$

We are now ready to prove a rigorous lower bound for the optimal value p^*.

THEOREM 3.2. *Let $\mathbf{P}$ define a family of semidefinite programs (1.1) with input data $P \in \mathbf{P}$, let $\tilde{y} \in \mathbb{R}^m$, set*

$$\mathbf{D}_j := \mathbf{C}_j - \sum_{i=1}^{m} \tilde{y}_i \mathbf{A}_{ij} \quad \textit{for } j = 1, \ldots, n, \tag{3.3}$$

and suppose that

$$\underline{d}_j \leq \lambda_{\min}(\mathbf{D}_j) \quad \textit{for } j = 1, \ldots, n. \tag{3.4}$$

Assume further that upper bounds for the maximal eigenvalues of the primal feasible solutions of (1.1)

$$\lambda_{\max}(X_j) \leq \overline{x}_j, \quad \textit{for } j = 1, \ldots, n \tag{3.5}$$

are known, where $\overline{x}_j$ may be infinite. If

$$\underline{d}_j \geq 0 \quad \textit{for } \overline{x}_j = +\infty, \tag{3.6}$$

then for every $P \in \mathbf{P}$ the inequality

$$p^*(P) \geq \inf\{\mathbf{b}^T\tilde{y} + \sum_{j=1}^{n} s_j \cdot \underline{d}_j^- \cdot \overline{x}_j\} \tag{3.7}$$

is satisfied, and the right hand side of (3.7) is finite[1]*. Moreover, for every $P \in \mathbf{P}$ and every j with $\underline{d}_j \geq 0$ the LMI*

$$\sum_{i=1}^{m} y_i A_{ij} - C_j \preceq 0$$

is feasible with $y := \tilde{y}$.

Proof. Let $P = (A, b, C) \in \mathbf{P}$ be chosen fixed, and let $X_j = X_j(P)$ be primal feasible for P and $j = 1, \ldots, n$. Let

$$D_j = C_j - \sum_{i=1}^{n} \tilde{y}_i A_{ij} \quad \text{for } j = 1, \ldots, n,$$

then

$$\sum_{j=1}^{n} \langle C_j, X_j \rangle - b^T\tilde{y} = \sum_{j=1}^{n} \langle D_j, X_j \rangle.$$

Since $D_j \in \mathbf{D}_j$, Lemma 3.1 implies

$$\sum_{j=1}^{n} \langle D_j, X_j \rangle \geq \sum_{j=1}^{n} s_j \cdot \underline{d}_j^- \cdot \overline{x}_j,$$

which proves the inequality (3.7), and the assumption (3.6) yields a finite right hand side. The last statement is an immediate consequence of $D_j \in \mathbf{D}_j$ and $\lambda_{\min}(D_j) \geq \underline{d}_j \geq 0$. □

Observe that $\tilde{y}$ is dual feasible provided $\underline{d}_j \geq 0$ for $j = 1, \ldots, n$. Hence in this case, (3.7) yields the lower bound $\inf\{\mathbf{b}^T\tilde{y}\}$ for the dual optimal value $d^*(P)$ for every $P \in \mathbf{P}$.

In order to judge the quality of the lower bound (3.7), we assume that

[1]Notice that $\mathbf{b}^T y$ is an interval operation yielding an interval for the expression in the braces in (3.7). Hence, the infimum denotes the lower bound of this interval. This notation applies also for the supremum and subsequently.

i) exact input data $P = \mathbf{P}$ are given,
ii) $D = \mathbf{D}$ is computed exactly, and
iii) Slater's constraint qualifications (cf. Section 1) are fulfilled.

Moreover, let $\tilde{y}$ be the optimal solution of the dual problem (1.2), and let $\underline{d}_j = \lambda_{\min}(D)$ for $j = 1, \ldots, n$. Then $\underline{d}_j \geq 0$ for $j = 1, \ldots, n$, and

$$p^*(P) = d^*(P) = b^T \tilde{y}.$$

Hence, no overestimation occurs, and it follows that the quality of this lower bound mainly depends on the quality of the $\underline{d}_j$ and on the computed approximation $\tilde{y}$.

An immediate consequence of Theorem 3.2 is the following error bound for linear programming problems

$$p^* := \min c^T x \quad \text{s.t. } Ax = b,\ x \geq 0, \tag{3.8}$$

which is proved in [17], and in [37] for finite bounds $\overline{x}_j$. The input data are $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $P = (A, b, c) \in \mathbb{R}^{m \times n + m + n}$.

COROLLARY 3.1. *Let* $\mathbf{P} = (\mathbf{A}, \mathbf{b}, \mathbf{c}) \in \mathbb{IR}^{m \times n + m + n}$, $\tilde{y} \in \mathbb{R}^m$, *and let*

$$\mathbf{d} := \mathbf{c} - \mathbf{A}^T \tilde{y}. \tag{3.9}$$

Assume further that upper bounds of the primal feasible solutions

$$x_j \leq \overline{x}_j \quad \textit{for } j = 1, \ldots, n$$

are known for all $P \in \mathbf{P}$, *which may also be infinite. If*

$$\mathbf{d}_j \geq 0 \quad \textit{for} \quad \overline{x}_j = +\infty, \tag{3.10}$$

then for every $P \in \mathbf{P}$ *the optimal value* $p^*(P)$ *satisfies the inequality*

$$p^*(P) \geq \inf\{\mathbf{b}^T \tilde{y} + \sum_{j=1}^{n} \mathbf{d}_j^- \cdot \overline{x}_j\}. \tag{3.11}$$

Proof. Apply Theorem 3.2 to the semidefinite program where the symmetric matrices A_{ij}, C_j and X_j are one-dimensional. □

Next, we describe an algorithm for computing a lower bound of the optimal value, which is based on Theorem 3.2. We assume that an approximate dual optimal solution $\tilde{y} \in \mathbb{R}^m$ of the midpoint problem $\operatorname{mid} \mathbf{P}$ is known. If condition (3.6) is fulfilled, the only work is to compute the right hand side of (3.7). Otherwise, the idea is to perturb all constraints which violate condition (3.6); that is, we solve a perturbed midpoint problem $P = (\operatorname{mid} \mathbf{A}, \operatorname{mid} \mathbf{b}, C(\varepsilon))$ with

$$C_j(\varepsilon) = \operatorname{mid} \mathbf{C}_j - \varepsilon_j I_d,\ \varepsilon_j = \begin{cases} > 0 & \text{if } \underline{d}_j < 0 \text{ and } \overline{x}_j = +\infty \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

Then the dual optimal solution $y(\varepsilon)$ satisfies the constraints

$$\operatorname{mid} \mathbf{C}_j - \sum_{i=1}^{m} y_i(\varepsilon) \operatorname{mid} \mathbf{A}_{ij} \succeq \varepsilon_j I_d.$$

ALGORITHM 3.1. *Rigorous lower bound*

given: real or interval input data $\mathbf{P} = (\mathbf{A}, \mathbf{b}, \mathbf{c})$,
upper bounds $\overline{x}_j$ for $j = 1, \ldots, n$,
approximate dual optimal solution $\tilde{y}$ for $\operatorname{mid} \mathbf{P}$,
$\underline{p}^* := -\infty$,
ε, k are n-dimensional zero vectors,
maximal numbers of iterations $l_{\max}$,
$l := 0$.

while perturbed problem $P(\varepsilon)$ is dual feasible **and** $l \leq l_{\max}$

1. Compute $\mathbf{D}_j = \mathbf{C}_j - \sum_{i=1}^{m} \tilde{y}_i \mathbf{A}_{ij}$, $j = 1, \ldots, n$.
2. Compute rigorous lower bounds $\underline{d}_j \leq \lambda_{\min}(\mathbf{D}_j)$, for $j = 1, \ldots, n$.
3. **If** $\underline{d}_j \geq 0$ for every j with $\overline{x}_j = +\infty$ **then** compute

$$\underline{p}^* = \inf\{\mathbf{b}^T \tilde{y} + \sum_{j=1}^{n} s_j \cdot \underline{d}_j^- \cdot \overline{x}_j\},$$

STOP.

4. Compute for $j = 1, \ldots, n$

$$k_j := \begin{cases} k_j + 1 & \text{if } \underline{d}_j < 0 \text{ and } \overline{x}_j = +\infty \\ k_j & \text{otherwise,} \end{cases}$$

$$\varepsilon_j := \begin{cases} -2^{k_j} \underline{d}_j + \varepsilon_j & \text{if } \underline{d}_j < 0 \text{ and } \overline{x}_j = +\infty \\ \varepsilon_j & \text{otherwise.} \end{cases}$$

5. Solve the perturbed midpoint problem $P(\varepsilon) = (\operatorname{mid} \mathbf{A}, \operatorname{mid} \mathbf{b}, C(\varepsilon))$, where $C_j(\varepsilon) = \operatorname{mid} \mathbf{C}_j - \varepsilon_j I_d$ for $j = 1, \ldots, n$, and set $\tilde{y} := \tilde{y}(\varepsilon)$ (approximate dual optimal solution).
6. $l := l + 1$.

end

Hence, the minimal eigenvalues of the new defect

$$\mathbf{D}_j(\varepsilon) := \mathbf{C}_j - \sum_{i=1}^{m} y_i(\varepsilon) \mathbf{A}_{ij}$$

will increase. Choosing ε_j very large may imply dual infeasibility, choosing $\varepsilon_j > 0$ too small may not be sufficient for satisfying (3.6). Our current trade off is to solve repeatedly perturbed programs until either condition (3.6) is satisfied, or the dual is infeasible. The details are given in Algorithm 3.1. This algorithm requires interval arithmetic (or at least the monotonic rounding operations) for computing the defect matrices $\mathbf{D}$, an algorithm for computing rigorous lower bounds $\underline{d}_j$, and a semidefinite solver for computing approximate solutions of the perturbed problems.

The algorithm terminates during the first iteration in step 3 if all simple bounds $\overline{x}_j$ are finite or all $\underline{d}_j$ are nonnegative. In this case the computational costs are $O(m \cdot \sum_{j=1}^{n} s_j^2)$ for computing the $\mathbf{D}_j$'s, the lower bounds $\underline{d}_j$ require $O(\sum_{j=1}^{n} s_j^3)$

operations, and the bound $\underline{p}^*$ needs $O(m+n)$ operations. Hence the costs are negligible compared to the costs for approximately solving a semidefinite program.

In other cases, however, the computational costs may increase because perturbed semidefinite programs must be solved until either condition (3.6) is satisfied, the semidefinite programming solver indicates dual infeasibility of the perturbed problem, or the maximal number of iterations $l_{\max}$ is reached.

Several modifications of this algorithm are possible and may yield improvements. Here we have considered a simple choice of perturbations: In each step we add to ε_j the negative defects $-\underline{d}_j$ multiplied by a factor 2^{k_j}, where k_j counts the number of iterations that violated the inequality $\underline{d}_j \geq 0$.

In applications we recommend to use infinite bounds $\overline{x}_j$ instead of unreasonable large bounds, because otherwise the sum in (3.7) may yield an unnecessary overestimation.

If the upper bounds $\overline{x}_j = +\infty$ for $j = 1, \ldots, n$, and Algorithm 3.1 delivers a finite lower bound $\underline{p}^*$, then the lower eigenvalue bounds $\underline{d}_j$ must be nonnegative. This proves dual feasibility, and if $\underline{d}_j$ is positive for $j = 1, \ldots, n$ strict dual feasibility is verified.

4. Rigorous upper bound. In this section we investigate the computation of a rigorous upper bound for the optimal value of a semidefinite program together with a certificate of existence of primal feasible solutions. The basic idea is to compute interval matrices $\mathbf{X}_j$ for $j = 1, \ldots, n$ that contain a primal feasible solution for every semidefinite program $P \in \mathbf{P}$. The desirable characteristics of the matrices $\mathbf{X}_j$ are given in the next theorem.

THEOREM 4.1. *Let $\mathbf{P}$ define a family of semidefinite programs (1.1), and suppose that there exist interval matrices $\mathbf{X}_j$ for $j = 1, \ldots, n$, such that*

$$\begin{aligned} &\forall\, b \in \mathbf{b},\ \forall\, A_{ij} \in \mathbf{A}_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n \\ &\quad \exists \text{ symmetric } X_j \in \mathbf{X}_j : \sum_{j=1}^{n} \langle A_{ij}, X_j \rangle = b_i, \end{aligned} \tag{4.1}$$

and for $j = 1, \ldots, n$

$$X_j \succeq 0 \text{ for all symmetric } X_j \in \mathbf{X}_j. \tag{4.2}$$

Then, the optimal value is bounded from above by

$$p^*(P) \leq \sup\{\sum_{j=1}^{n} \langle \mathbf{C}_j, \mathbf{X}_j \rangle\} \tag{4.3}$$

Moreover, if all symmetric $X_j \in \mathbf{X}_j$ are positive definite and $p^*(P)$ is bounded from below, then $p^*(P) = d^*(P)$ for every $P \in \mathbf{P}$ (no duality gap), and the dual supremum is attained.

Proof. Let $P \in \mathbf{P}$ be a fixed chosen problem. Then the conditions (4.1) and (4.2) imply that there exists a primal feasible solution $X_j = X_j(P)$ for $j = 1, \ldots, n$. Hence, $\sum_{j=1}^{n} \langle C_j, X_j \rangle \geq p^*(P)$, and the inclusion property (2.5) yields (4.3). If all $X_j \in \mathbf{X}_j$ are positive definite, then (4.1) and (4.2) imply the existence of strictly primal feasible solutions, and hence Theorem 1.1 shows that the dual optimal solution is attained and strong duality holds valid. □

By weak duality the upper bound in (4.3) is also an upper bound of the dual optimal value. Moreover, if all $X_j \in \mathbf{X}_j$ are positive definite, then the Strong Duality

Theorem 1.1 implies that the right hand side of (3.7) is also a lower bound of the dual optimal value for all $P \in \mathbf{P}$. Hence, in this case it is not necessary to assume $\underline{d}_j \geq 0$ for $j = 1, \ldots, n$.

In the following, we describe an algorithm for computing this rigorous upper bound. This algorithm must find appropriate interval matrices $\mathbf{X}_j$, and verify the conditions (4.1) and (4.2). We discuss these items below.

To make sure that the upper bound (4.3) is close to the optimal value, the interval matrices $\mathbf{X}_j$ must be close to optimality. In general the complementary slackness relations yield rank-deficient matrices, which are not positive definite. Therefore, we solve the slightly perturbed midpoint problem

$$\begin{aligned} \min \sum_{j=1}^{n} \langle C_j, X_j \rangle \quad \text{s.t.} \quad \sum_{j=1}^{n} \langle A_{ij}, X_j \rangle = b_i \quad \text{for } i = 1, \ldots, m, \\ X_j \succeq \varepsilon_j \cdot I_d, \quad \text{for } j = 1, \ldots, n, \end{aligned} \tag{4.4}$$

where ε_j is positive and the input data $(A, b, c) = \operatorname{mid} \mathbf{P}$. Then for small ε_j the optimal solution $(X_j(\varepsilon_j))$ is positive definite and close to the optimal solution of the midpoint problem.

In the following we show how we can construct an appropriate interval matrix $(\mathbf{X}_j)$ by using an approximate optimal solution $(X_j(\varepsilon_j))$ of (4.4).

The semidefinite program (1.1) can be written in the equivalent vector representation form

$$\min c^T x \quad \text{s.t.} \quad A^{\text{mat}} x = b, \; X_j \succeq 0, \; \text{for } j = 1, \ldots, n, \tag{4.5}$$

where

$$c := \operatorname{svec}((C_j), 2), \tag{4.6}$$

$$x := \operatorname{svec}((X_j), 1), \tag{4.7}$$

and the i-th row of the $m \times \sum_{j=1}^{n} \frac{s_j(s_j+1)}{2}$ matrix A^{mat} is defined by

$$A^{\text{mat}}(i, :) = \operatorname{svec}((A_{ij})_{j=1}^{n}, 2). \tag{4.8}$$

If interval input data $\mathbf{P}$ are given, then we denote by $\mathbf{A}^{\text{mat}}$, $\mathbf{b}$, and $\mathbf{c}$ the corresponding interval quantities. Thus condition (4.1) is equivalent to

$$\forall\, b \in \mathbf{b}, \; \forall\, A^{\text{mat}} \in \mathbf{A}^{\text{mat}} \; \exists\, x \in \mathbf{x} \;\text{ such that }\; A^{\text{mat}} x = b, \tag{4.9}$$

which is an underdetermined system of linear equations with interval input data. Given an approximate optimal solution $(X_j(\varepsilon_j))_{j=1}^{n}$, it is straight forward to solve such a system.

We start by assuming that the $m \times m$ submatrix $\operatorname{mid} \mathbf{A}_I^{\text{mat}}$ with the m columns $\operatorname{mid} \mathbf{A}^{\text{mat}}(:, \beta_i)$ is nonsingular, where the index set $I := \{\beta_1, \ldots, \beta_m\}$. Let N denote all indices of columns of $\operatorname{mid} \mathbf{A}^{\text{mat}}$ which are not in I, let $\mathbf{A}_N^{\text{mat}}$ be the matrix with columns corresponding to the indices of N, and let $\tilde{x} = \operatorname{svec}((X_j(\varepsilon_j)), 1)$. In our algorithm we choose the index set I by performing an LU-decomposition on $(\operatorname{mid} \mathbf{A}^{\text{mat}})^T$ and assembling the computed pivot columns to A_I^{mat}. Now we fix the variables $\tilde{x}_N$, and compute with some verification method for interval linear systems an enclosure $\mathbf{x}_I$ of the solution set

$$\Sigma_I := \{x_I \in \mathbb{R}^m : \; A_I^{\text{mat}} x_I = b - \sum_{\gamma \in N} A_N^{\text{mat}} \tilde{x}_N, \; A \in \mathbf{A}^{\text{mat}}, \; b \in \mathbf{b}\}. \tag{4.10}$$

Then $\mathbf{x} := (\mathbf{x}_I; \tilde{x}_N)$ fulfills (4.9), and therefore $(\mathbf{X}_j) := \mathrm{smat}(\mathbf{x}, 1)$ satisfies condition (4.1). Condition (4.2) must be verified by some method for computing a rigorous lower bound for the smallest eigenvalue of a symmetric interval matrix.

Algorithm 4.1 contains the details for computing a rigorous upper bound for the optimal value and for proving existence of primal feasible solutions. The algorithm needs verified solvers for interval linear systems and eigenvalue problems, and a semidefinite solver for computing approximations of the perturbed problems.

If Algorithm 4.1 delivers a finite upper bound $\overline{p}^*$, then the lower eigenvalue bounds $\underline{\lambda}_j$ must be nonnegative. If $\underline{\lambda}_j > 0$ for $j = 1, ..., n$, then strict primal feasibility is verified.

Krawczyk [27] was the first who solved non degenerate interval linear programming problems by using the technique of fixing appropriate variables (the nonbasic variables) and solving a remaining quadratic interval linear system for the basic variables. In [15] this technique was used to compute enclosures of all optimal vertices in the degenerate case. Hansen [13] used this technique in order to prove existence of a feasible point for nonlinear equations within a bounded box. It was further modified and investigated numerically by Kearfott [20], and is also described in his book [19].

5. Certificate of Infeasibility. In branch and bound algorithms a subproblem is discarded if the local nonlinear solver detects infeasibility. Occasionally local solvers do not find feasible solutions of a subproblem, although they exist (see for example the comments for use of SDPT3). A consequence is that the global minimum solutions may be cut off.

To avoid this disadvantage we can apply the algorithms for computing rigorous bounds described in the previous sections to a phase I problem in order to verify infeasibility for primal and dual semidefinite problems. In the literature there are several variations of the phase I method. It is common, however, that the auxiliary objective function describes the infeasibility in the sense that the problem has no feasible solutions, provided the optimal value is greater than zero. The latter property can be verified by the algorithms of the previous sections.

Another approach is based on certificates of infeasibility. For linear programs with bounded variables rigorous certificates of infeasibility are described in Neumaier and Shcherbina [37]. For infeasible semidefinite problems often (but not every time) certificates of infeasibility exposed by improving rays can be obtained (see also the discussion in Todd [46]).

The primal problem (1.1) has a primal improving ray if there exists a block-diagonal matrix (X_j) such that for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$

$$X_j \succeq 0, \quad \sum_{j=1}^{n} \langle A_{ij}, X_j \rangle = 0, \quad \text{and} \quad \sum_{j=1}^{n} \langle C_j, X_j \rangle < 0. \tag{5.1}$$

It is well-known and straightforward to show that the existence of a primal improving ray implies dual infeasibility. If interval input data $\mathbf{P}$ are given, and for the midpoint problem of $\mathbf{P}$ an approximate primal improving ray is known, then we can try to verify dual infeasibility for all problems with $P \in \mathbf{P}$ by using a similar approach as in the previous section. We assume that the semidefinite solver has computed an approximate primal improving ray $(\tilde{X}_j)$ for the midpoint problem. Let $\beta \approx \sum_{j=1}^{n} \langle \mathrm{mid}\, \mathbf{C}_j, \tilde{X}_j \rangle$ be approximately calculated and assume that $\beta < 0$. Notice that for positive β the conditions (5.1) are in general not satisfied. Now, analogously to the previous section, we can compute enclosures $(\mathbf{X}_j)$ such that for every $A_{ij} \in \mathbf{A}_{ij}$ and for every $C_j \in \mathbf{C}_j$

ALGORITHM 4.1. *Rigorous upper bound, certificate of feasibility*

given: real or interval input data $\mathbf{P} = (\mathbf{A}, \mathbf{b}, \mathbf{c})$,
approximate primal optimal solution $(\tilde{X}_j)_{j=1}^n$ of the midpoint problem,
$\overline{p}^* := \infty$,
ε, k are n-dimensional zero vectors,
maximal number of iterations $l_{\max}$,
$l := 0$.

Choose an index set I such that the submatrix $\operatorname{mid} \mathbf{A}_I^{\mathrm{mat}}$ is (at least numerically) nonsingular (for example, by performing an LU-decomposition on $(\operatorname{mid} \mathbf{A}^{\mathrm{mat}})^T$).

if there is no nonsingular submatrix **then STOP**.

while perturbed problem $P(\varepsilon)$ is primal feasible **and** $l \leq l_{\max}$

1. Compute an enclosure $\mathbf{x}_I$ of the solution set Σ_I (4.10), and set $\mathbf{x} := (\mathbf{x}_I; \tilde{x}_N)$.
2. Set $(\mathbf{X}_j) = \operatorname{smat}(\mathbf{x}, 1)$, and compute rigorous bounds
$$\underline{\lambda}_j \leq \lambda_{\min}(\mathbf{X}_j) \quad \text{for} \quad j = 1, \ldots, n.$$
3. **if** $\underline{\lambda}_j \geq 0$ for $j = 1, \ldots, n$ **then** compute
$$\overline{p}^* = \sup\{\mathbf{c}^T \mathbf{x}\},$$
STOP.
4. Compute for $j = 1, \ldots, n$
$$k_j := \begin{cases} k_j + 1 & \text{if } \underline{\lambda}_j < 0 \\ k_j & \text{otherwise,} \end{cases}$$
$$\varepsilon_j := \begin{cases} -2^{k_j} \underline{\lambda}_j + \varepsilon_j & \text{if } \underline{\lambda}_j < 0 \\ \varepsilon_j & \text{otherwise.} \end{cases}$$
5. Solve the perturbed problem (4.4), set $\tilde{X}_j := \tilde{X}_j(\varepsilon)$ for $j = 1, \ldots, n$ (approximate primal optimal solution), and set $\tilde{x} := \operatorname{svec}((\tilde{X}_j), 1)$.
6. $l := l + 1$.

end

there exist solutions $X_j \in \mathbf{X}_j$ of the underdetermined linear system

$$\sum_{j=1}^n \langle A_{ij}, X_j \rangle = 0 \quad \text{for} \quad i = 1, \ldots, m, \qquad \sum_{j=1}^n \langle C_j, X_j \rangle = \beta, \tag{5.2}$$

If an enclosure $(\mathbf{X}_j)$ is computed, and if all minimal eigenvalues $\lambda_{min}(\mathbf{X}_j)$ are nonnegative, then, because $\beta < 0$, it follows that for every $P \in \mathbf{P}$ there exists $X_j \in \mathbf{X}_j$ for $j = 1, \ldots, n$ such that (5.1) is satisfied. Therefore, all problems with $P \in \mathbf{P}$ are dual infeasible, and the block-matrices $(\mathbf{X}_j)$ contain the primal improving rays.

The dual problem (1.3) has a dual improving ray if there is a vector $y \in \mathbb{R}^m$ such

that

$$\sum_{i=1}^{m} y_i A_{ij} \preceq 0 \quad \text{for } j = 1, \ldots, n, \quad \text{and} \quad b^T y > 0. \tag{5.3}$$

The existence of a dual improving ray implies primal infeasibility. If interval input data $\mathbf{P}$ are given and for some problem $P \in \mathbf{P}$ an approximate dual improving ray $\tilde{y}$ is known, then we can try to verify primal infeasibility for all problems with $P \in \mathbf{P}$ as follows: First we compute by using interval arithmetic upper bounds of the maximal eigenvalues of $\sum_{i=1}^{m} \tilde{y}_i \mathbf{A}_{ij}$ for $j = 1, \ldots, n$. If these upper bounds are nonpositive and if $\mathbf{b}^T \tilde{y} > 0$, then $\tilde{y}$ is a dual improving ray for all $P \in \mathbf{P}$. Hence, $\tilde{y}$ is a rigorous certificate of primal infeasibility for all $P \in \mathbf{P}$.

6. Numerical results. In this section, we present some numerical experiments for semidefinite and linear programming problems. The results for the semidefinite programming problems were obtained by using VSDP [14], a MATLAB software package for verified semidefinite programming. This package uses the toolbox INTLAB [44] and allows interval input data. For all experiments we used the default values of the solver.

First we consider a semidefinite program of small size

$$\begin{aligned}
\min \quad & \left\langle \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & \delta & 0 \\ 0 & 0 & \delta \end{pmatrix}, X \right\rangle \\
\text{s.t.} \quad & \left\langle \begin{pmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, X \right\rangle = 1, \\
& \left\langle \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, X \right\rangle = \varepsilon, \\
& \left\langle \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, X \right\rangle = 0, \\
& \left\langle \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, X \right\rangle = 0, \\
& X \succeq 0.
\end{aligned}$$

TABLE 6.1
Approximations $\tilde{p}^*$, $\tilde{d}^*$ and rigorous bounds $\overline{p}^*$, $\underline{p}^*$

ε	δ	$\tilde{p}^*$	$\tilde{d}^*$	tc	$\overline{p}^*$	$\underline{p}^*$
0	0	−1.0004	−0.99355	0	∞	$-\infty$
10^{-8}	10^{-8}	−0.99184	−0.98372	0	∞	−0.98373
10^{-6}	10^{-10}	−1.0007	−1.0027	0	−0.99965	−1.0061
10^{-4}	10^{-3}	8.9004	8.9990	0	9.3353	8.9990
-10^{-4}	10^{-3}	28.228	142.86	0	∞	142.86
10^{-4}	-10^{-4}	−5.9323	−1.0361	−7	−5.9324	$-\infty$

The Lagrangian dual is

$$d^* = \max y_1 + \varepsilon y_2 \quad \text{s.t.} \quad Y := C - \sum_{i=1}^{4} A_{i1} y_i$$
$$= \begin{pmatrix} -y_2 & \frac{1+y_1}{2} & -y_3 \\ \frac{1+y_1}{2} & \delta & -y_4 \\ -y_3 & -y_4 & \delta \end{pmatrix} \succeq 0.$$

The linear constraints of the primal problem imply

$$X = \begin{pmatrix} \varepsilon & -1 & 0 \\ -1 & X_{22} & 0 \\ 0 & 0 & X_{33} \end{pmatrix},$$

and X is positive semidefinite iff $X_{22} \geq 0$, $X_{33} \geq 0$, and $\varepsilon \cdot X_{22} - (-1)^2 \geq 0$. Hence, for $\varepsilon \leq 0$, the problem is primal infeasible and $p^* = +\infty$. The dual problem is infeasible for $\delta < 0$ with $d^* = -\infty$.

For $\varepsilon = 0$ and $\delta = 0$ we obtain a duality gap with $p^* = +\infty$ and $d^* = -1$, and the problem is ill-posed. For $\varepsilon > 0$ and $\delta > 0$ Slater's constraint qualifications are satisfied and the optimal value $p^* = d^* = -1 + \delta/\varepsilon$.

Numerical results for different values ε and δ are summarized in Table 6.1. The termination code $tc = 0$ in SDPT3 means normal termination without warning, whereas $tc = -7$ indicates primal infeasibility. In the following, $\tilde{p}^*$ and $\tilde{d}^*$ denote the computed approximations of the primal and dual optimal value, respectively.

We see that SDPT3 is not backward stable, since in five cases $\tilde{p}^* < \tilde{d}^*$, violating the weak duality. This example demonstrates that the measures for termination and accepting an approximation are not appropriate for ill-conditioned or ill-posed problems. The rigorous bounds $\overline{p}^*$ and $\underline{p}^*$ recognize difficult problems much better and overestimate the optimal value only slightly, and this overestimation depends on the quality of the computed approximations. The bounds are infinite if the problem is infeasible or very ill-conditioned. For larger values $\varepsilon > 0$ and $\delta > 0$ the approximations and the rigorous bounds are almost identical, and are not displayed here.

In the following, we describe the numerical results on problems from the SDPLIB collection of Borchers [5]. In summary, VSDP could compute (by using SDPT3 as approximate solver) for all 85 problems discussed in [9] a rigorous lower bound of the optimal value and verify the existence of strictly dual feasible solutions which implies a zero duality gap. A finite rigorous upper bound could be computed for all well-posed problems with the exception of `hinf8`, which is ill-conditioned. For the 32 ill-posed problems VSDP has computed $\overline{p}^* = +\infty$, which reflects that the distance to the next primal infeasible problem is zero as well as the infinite condition number. Detailed

numerical results can be found in the tables contained in the appendix. We measure the accuracy by the quantity

$$\mu(a,b) := \frac{a-b}{\max\{1.0, (|a|+|b|)/2\}}.$$

Notice that we do not use the absolute value of $a-b$. Hence, a negative sign implies that $a < b$. Table A.1 contains the rigorous upper bound $\overline{p}^*$, the rigorous lower bound $\underline{p}^*$, the rigorous error $\mu(\overline{p}^*, \underline{p}^*)$, and the approximate duality gap $\mu(\tilde{p}^*, \tilde{d}^*)$. We have set $\mu(\overline{p}^*, \underline{p}^*) = NaN$ if the upper bound $\overline{p}^*$ is infinite. Table A.2 contains the termination code tc given by SDPT3 and the time t in seconds for computing the approximations. The times for computing the rigorous lower and upper bound are $\underline{t}$ and $\overline{t}$, respectively.

Some major characteristics of our numerical results for the SDPLIB are as follows: The median of the time ratio for computing the rigorous lower bound and the approximations is $\mathrm{med}(\underline{t}/t) = 0.045$, and $\mathrm{med}(\overline{t}/t) = 2.4$. The median of the guaranteed accuracy $\mu(\overline{p}^*, \underline{p}^*)$ for the problems with finite condition number is $4.9 \cdot 10^{-7}$. We have used the median here because there are some outliers.

For the SDPLIB problems, SDPT3 (with default values) gave 7 warnings, and 2 warnings were given for well-posed problems. Hence, no warnings were given for 27 ill-posed problems with zero distance to primal infeasibility. In other words, there is no correlation between warnings and the difficulty of the problem. At least for this test set our rigorous bounds reflect the difficulty of the problems much better, and they provide safety, especially in the case where algorithms subsequently call other algorithms, as is done for example in branch-and-bound methods. In our opinion the usefulness of a warning is dubious if for ill-posed problems normal termination occurs. In contrast, VSDP shows a strong correlation between the rigorous bounds and the difficulty of the problem. Moreover, the negative signs of $\mu(\tilde{p}^*, \tilde{d}^*)$ show that the approximations do not satisfy weak duality. Therefore they are not backward stable, i.e., they are not exact solutions of slightly perturbed problems.

One of the largest problems which could be solved by VSDP is `thetaG51` where the number of constraints is $m = 6910$, $n = 1$, and the dimension of the primal Solution X_1 is $s_1 = 1001$ (implying 501501 variables). For this problem SDPT3 gave the message out of memory, and we used SDPA [10] as approximate solver. The rigorous lower and upper bound computed by VSDP are $\underline{p}^* = -3.4900 \cdot 10^2$, $\overline{p}^* = -3.4406 \cdot 10^2$, respectively. The guaranteed relative accuracy is only 0.014, which may be sufficient in several applications, but is insufficient from a numerical point of view. The times in seconds for computing the approximations, the lower and the upper bound of the optimal value are $t = 3687.95$, $\underline{t} = 45.17$, and $\overline{t} = 6592.52$, respectively. Existence of optimal solutions and strong duality is proved. For the 11 instances, `control1` to `control11`, the guaranteed relative accuracies are also low, ranging from $1.21 \cdot 10^{-4}$ to $5.93 \cdot 10^{-2}$.

The aim in designing VSDP was to write the programs in a convenient form, appropriate also for students. For example, the user can specify the block-diagonal structure in the more natural way of cell arrays. Operations such as extracting selected elements of a matrix or the operations `vsvec` (rounding error free vectorization operator) and `vsmat` (corresponding rounding error free inverse operator) are written in MATLAB [29]. This causes a loss of efficiency. At present a C++ implementation of these error bounds using the interval library PROFIL/BIAS [25] is in process [6].

We already have a C++ implementation of the rigorous bounds for the special case of linear programming called Lurupa [21]. Here we present a summary of our

numerical results for the NETLIB suite of linear programming problems [33]. For details refer to [23]. The NETLIB collection contains problems with up to 15695 variables and 16675 constraints. They originate from various applications, for example forestry, flap settings on aircraft, and staff scheduling. In order to compare the results, we initially chose the set of problems that Ordóñez and Freund [38] have computed condition numbers for. Removing problems *DEGEN3* and *PILOT* because the approximate solver failed to solve these within 24 hours, the final test set included 87 problems. We applied our algorithm to the original problems without preprocessing. As Fourer and Gay [8] observed, preprocessing can change the state of an LP from feasible to infeasible and vice versa.

Roughly speaking, a finite bound can be computed iff the corresponding distance to infeasibility is greater than 0. For 76 problems a finite lower bound could be computed with a median accuracy of $\text{med}(\mu(\underline{p}^*, \tilde{p}^*)) = 2.2 \cdot 10^{-8}$ and a median time ratio of $\text{med}(\underline{t}/t) = 0.5$. For 35 problems the algorithm yielded a finite upper bound with $\text{med}(\mu(\overline{p}^*, \tilde{p}^*)) = 8.0 \cdot 10^{-9}$ and $\text{med}(\overline{t}/t) = 5.3$. For 32 problems finite rigorous lower and upper bounds could be computed with $\text{med}(\mu(\overline{p}^*, \underline{p}^*)) = 5.6 \cdot 10^{-8}$. Taking into account the approximate solver's default stopping tolerance of 10^{-9}, the guaranteed accuracy for the NETLIB LP suite is reasonable. The upper bound is more expensive, since linear systems have to be solved rigorously.

To our knowledge no other software packages compute rigorous results for semidefinite programs. There are several packages that compute verified results for optimization problems where the objective and the constraints are defined by smooth algebraic expressions. The most widely known solvers for global optimization that use interval arithmetic are probably GlobSol [18], Numerica [48], and COSY [4] (on inquiry we did not receive a copy of COSY). They do not permit the nonsmooth semidefinite constraints. But they can be used to solve linear programming problems (a special, smooth case of semidefinite programming) rigorously. Packages for verified constraint programming, like ICOS [28] and RealPaver [11], can also be used to compute rigorous results for an LP by adding bounds on the objective function to the set of constraints. Specifically dedicated to verified linear programming are solvers using a rational arithmetic. Examples are QSopxt_ex [2], exlp [24], and, to a limited extent, perPlex [26]. The latter one only verifying the optimality of a given basis.

We want to see how these packages compare for the special case of linear programming. For this purpose we generate dense test problems with a method similar to the one described by Rosen and Suzuki [41]. Setting a timeout of one hour on each solver run, we then determine the largest problem size that the packages can solve. First we have a look at the packages that do not exploit the linear structure in the test problems. GlobSol, Numerica, and RealPaver fail to solve the test problems if the number of variables and constraints exceeds 10. ICOS solves the problems within one hour with up to 100 variables and constraints. Now looking at the packages taking account of the linear structure, we see that these succeed for larger problems. QSopt_ex, exlp, and perPlex solve problems with 200 variables and constraints. Lurupa does not fail until the dimension exceeds 1500. For larger dimensions it turns out that it already takes more than one hour to compute the initial approximate solution. The verification part only needs a fraction of this time. Our algorithm, however, produces rigorous bounds for the optimal value and for feasible solutions close to optimality; enclosures of optimal solutions are not computed. Elaborate numerical results can be found in [22].

7. Conclusions. The computation of rigorous error bounds for linear or semidefinite optimization problems can be viewed as a carefully postprocessing tool that uses only approximate solutions computed by an LP or SDP solver. The numerical results show that such rigorous error bounds can be computed even for problems of large size.

8. Acknowledgment. We wish to thank Arnold Neumaier and Siegfried M. Rump for their stimulating input and two anonymous referees for many suggestions to improve this paper.

Appendix. Results from the SDPLIB.

Table A.1: Rigorous bounds for the SDPLIB

Notice that all problems with $\overline{p}^* = \infty$ are ill-posed with the exception of `hinf8`.

Problem	$\overline{p}^*$	$\underline{p}^*$	$\mu(\overline{p}^*, \underline{p}^*)$	$\mu(\tilde{p}^*, \tilde{d}^*)$
arch0	$-5.6651e-01$	$-5.6652e-01$	$5.05e-06$	$-2.77e-07$
arch2	$-6.7151e-01$	$-6.7152e-01$	$5.16e-06$	$1.01e-07$
arch4	$-9.7263e-01$	$-9.7263e-01$	$3.66e-08$	$3.66e-08$
arch8	$-7.0570e+00$	$-7.0570e+00$	$1.84e-07$	$1.61e-07$
control1	$-1.7782e+01$	$-1.7785e+01$	$1.21e-04$	$-6.13e-08$
control2	$-8.2909e+00$	$-8.3000e+00$	$1.10e-03$	$3.74e-08$
control3	$-1.3615e+01$	$-1.3633e+01$	$1.36e-03$	$-1.89e-07$
control4	$-1.9671e+01$	$-1.9794e+01$	$6.24e-03$	$-7.71e-07$
control5	$-1.6795e+01$	$-1.6884e+01$	$5.25e-03$	$-1.72e-06$
control6	$-3.6736e+01$	$-3.7304e+01$	$1.53e-02$	$2.15e-06$
control7	$-2.0434e+01$	$-2.0625e+01$	$9.29e-03$	$1.36e-06$
control8	$-2.0019e+01$	$-2.0286e+01$	$1.33e-02$	$-5.36e-08$
control9	$-1.4036e+01$	$-1.4675e+01$	$4.45e-02$	$-2.57e-06$
control10	$-3.6890e+01$	$-3.8533e+01$	$4.36e-02$	$2.79e-06$
control11	$-3.0119e+01$	$-3.1959e+01$	$5.93e-02$	$3.40e-06$
equalG11	$-6.2915e+02$	$-6.2916e+02$	$1.08e-05$	$2.62e-05$
equalG51	$-4.0055e+03$	$-4.0056e+03$	$3.58e-05$	$4.17e-08$
gpp100	∞	$4.4943e+01$	NaN	$1.24e-04$
gpp124-1	∞	$7.3431e+00$	NaN	$6.94e-07$
gpp124-2	∞	$4.6862e+01$	NaN	$1.23e-07$
gpp124-3	∞	$1.5301e+02$	NaN	$1.28e-04$
gpp124-4	∞	$4.1899e+02$	NaN	$9.83e-05$
gpp250-1	∞	$1.5443e+01$	NaN	$6.85e-03$
gpp250-2	∞	$8.1869e+01$	NaN	$1.44e-06$
gpp250-3	∞	$3.0354e+02$	NaN	$4.30e-07$
gpp250-4	∞	$7.4733e+02$	NaN	$2.81e-07$
gpp500-1	∞	$2.5320e+01$	NaN	$2.04e-06$
gpp500-2	∞	$1.5606e+02$	NaN	$1.30e-03$
gpp500-3	∞	$5.1302e+02$	NaN	$1.21e-05$
gpp500-4	∞	$1.5670e+03$	NaN	$8.81e-08$
hinf1	∞	$-2.0328e+00$	NaN	$-1.12e-04$
hinf2	$-7.1598e+00$	$-1.0967e+01$	$4.20e-01$	$-3.38e-05$
hinf3	∞	$-5.6954e+01$	NaN	$-2.30e-04$
hinf4	∞	$-2.7477e+02$	NaN	$-1.43e-05$
hinf5	∞	$-3.6259e+02$	NaN	$2.20e-03$
hinf6	∞	$-4.4914e+02$	NaN	$-1.21e-03$
hinf7	∞	$-3.9082e+02$	NaN	$-8.98e-06$
hinf8	∞	$-1.1618e+02$	NaN	$-3.93e-04$
hinf9	∞	$-2.3709e+02$	NaN	$-3.43e-02$
hinf10	∞	$-1.0887e+02$	NaN	$-1.42e-03$
hinf11	∞	$-6.5944e+01$	NaN	$-1.24e-03$
hinf12	∞	$-7.8714e-01$	NaN	$-6.66e-01$
hinf13	∞	$-4.6009e+01$	NaN	$-3.62e-02$

continued...

Problem	$\overline{p}^*$	$\underline{p}^*$	$\mu(\overline{p}^*, \underline{p}^*)$	$\mu(\tilde{p}^*, \tilde{d}^*)$
hinf14	∞	$-1.3001e+01$	NaN	$1.70e-04$
hinf15	∞	$-2.6324e+01$	NaN	$-9.20e-02$
maxG11	$-6.2916e+02$	$-6.2916e+02$	$5.13e-08$	$3.01e-08$
maxG32	$-1.5676e+03$	$-1.5676e+03$	$3.16e-07$	$-2.99e-08$
maxG51	$-4.0063e+03$	$-4.0063e+03$	$5.49e-08$	$-2.23e-08$
mcp100	$-2.2616e+02$	$-2.2616e+02$	$1.85e-08$	$1.11e-08$
mcp124-1	$-1.4199e+02$	$-1.4199e+02$	$8.92e-09$	$8.79e-09$
mcp124-2	$-2.6988e+02$	$-2.6988e+02$	$3.31e-08$	$9.67e-10$
mcp124-3	$-4.6775e+02$	$-4.6775e+02$	$8.57e-09$	$-1.45e-09$
mcp124-4	$-8.6441e+02$	$-8.6441e+02$	$1.46e-08$	$-7.77e-09$
mcp250-1	$-3.1726e+02$	$-3.1726e+02$	$1.00e-08$	$3.43e-08$
mcp250-2	$-5.3193e+02$	$-5.3193e+02$	$6.85e-09$	$-1.14e-10$
mcp250-3	$-9.8117e+02$	$-9.8117e+02$	$2.04e-08$	$1.86e-08$
mcp250-4	$-1.6820e+03$	$-1.6820e+03$	$1.59e-08$	$3.43e-08$
mcp500-1	$-5.9815e+02$	$-5.9815e+02$	$4.91e-08$	$-1.16e-08$
mcp500-2	$-1.0701e+03$	$-1.0701e+03$	$1.28e-07$	$-5.93e-08$
mcp500-3	$-1.8480e+03$	$-1.8480e+03$	$1.31e-08$	$7.66e-09$
mcp500-4	$-3.5667e+03$	$-3.5667e+03$	$1.19e-08$	$-3.91e-09$
qap10	∞	$1.0925e+03$	NaN	$-1.10e-04$
qap5	∞	$4.3600e+02$	NaN	$7.22e-10$
qap6	∞	$3.8141e+02$	NaN	$-9.00e-05$
qap7	∞	$4.2479e+02$	NaN	$-7.27e-05$
qap8	∞	$7.5686e+02$	NaN	$-1.29e-04$
qap9	∞	$1.4099e+03$	NaN	$-5.35e-05$
qpG11	$-2.4487e+03$	$-2.4487e+03$	$6.42e-09$	$6.42e-09$
qpG51	$-1.1818e+04$	$-1.1818e+04$	$7.44e-09$	$7.44e-09$
ss30	$-2.0239e+01$	$-2.0240e+01$	$1.55e-06$	$1.55e-06$
theta1	$-2.3000e+01$	$-2.3000e+01$	$1.55e-08$	$9.89e-09$
theta2	$-3.2879e+01$	$-3.2879e+01$	$1.18e-06$	$1.17e-08$
theta3	$-4.2167e+01$	$-4.2167e+01$	$9.64e-07$	$-1.73e-08$
theta4	$-5.0321e+01$	$-5.0321e+01$	$1.09e-06$	$9.41e-09$
theta5	$-5.7232e+01$	$-5.7232e+01$	$4.92e-07$	$-6.68e-09$
thetaG11	$-4.0000e+02$	$-4.0000e+02$	$4.23e-07$	$2.02e-10$
truss1	$9.0000e+00$	$9.0000e+00$	$1.61e-07$	$3.08e-09$
truss2	$1.2338e+02$	$1.2338e+02$	$1.60e-06$	$-5.86e-09$
truss3	$9.1100e+00$	$9.1100e+00$	$1.11e-07$	$1.11e-07$
truss4	$9.0100e+00$	$9.0100e+00$	$1.19e-08$	$4.15e-09$
truss5	$1.3264e+02$	$1.3264e+02$	$9.09e-06$	$-2.28e-07$
truss6	$9.0119e+02$	$9.0100e+02$	$2.16e-04$	$-3.26e-07$
truss7	$9.0015e+02$	$9.0000e+02$	$1.70e-04$	$-4.67e-06$
truss8	$1.3312e+02$	$1.3311e+02$	$7.79e-05$	$-4.98e-06$

Table A.2: Computational effort for the SDPLIB

Problem	*tc*	t	$\underline{t}$	$\overline{t}$
arch0	0	19.95	0.81	24.05
arch2	0	18.06	0.36	43.13
arch4	0	19.20	0.34	3.39
arch8	0	19.81	0.33	46.34
control1	0	1.28	0.05	1.66
control2	0	2.64	0.03	3.14
control3	0	6.20	0.09	6.56
control4	0	11.33	0.14	13.83
control5	0	23.11	0.28	34.75
control6	0	60.72	0.47	155.11
control7	0	87.67	0.72	138.31
control8	0	152.98	1.23	232.28
control9	0	238.88	635.77	390.06
control10	0	432.48	3.05	743.75
control11	0	661.59	3.95	1129.63

continued...

Problem	*tc*	t	$\underline{t}$	$\bar{t}$
equalG11	0	590.89	844.22	896.25
equalG51	0	1677.77	1369.69	1638.83
gpp100	0	2.34	0.30	35.88
gpp124-1	0	5.88	6.25	38.55
gpp124-2	0	6.11	0.56	41.75
gpp124-3	0	3.95	0.63	36.80
gpp124-4	0	4.16	0.59	43.89
gpp250-1	0	18.64	5.58	122.00
gpp250-2	0	31.14	5.53	112.63
gpp250-3	0	24.77	5.08	160.81
gpp250-4	0	24.66	5.56	130.88
gpp500-1	0	190.39	236.81	642.58
gpp500-2	0	145.53	46.69	809.75
gpp500-3	0	156.52	46.73	961.78
gpp500-4	0	205.06	46.42	815.34
hinf1	0	1.28	0.05	16.95
hinf2	0	1.03	0.06	12.23
hinf3	0	1.11	0.02	16.73
hinf4	0	1.09	0.06	5.98
hinf5	-4	1.23	0.05	6.81
hinf6	0	1.34	0.06	8.69
hinf7	-4	1.22	1.31	13.63
hinf8	-4	1.19	0.06	5.83
hinf9	2	1.38	0.06	7.98
hinf10	0	1.80	0.05	6.02
hinf11	0	1.94	0.05	11.25
hinf12	0	3.00	6.28	6.64
hinf13	0	1.94	0.08	7.39
hinf14	-4	1.44	0.06	6.91
hinf15	0	2.34	0.08	10.47
maxG11	0	449.28	13.14	678.13
maxG32	0	6910.72	176.53	9838.19
maxG51	0	990.22	26.47	1352.08
mcp100	0	2.44	0.09	5.53
mcp124-1	0	3.13	0.13	0.55
mcp124-2	0	3.69	0.13	9.16
mcp124-3	0	3.91	0.14	4.59
mcp124-4	0	4.09	0.17	4.91
mcp250-1	0	15.33	0.47	20.83
mcp250-2	0	16.41	0.52	22.16
mcp250-3	0	18.08	0.59	43.17
mcp250-4	0	18.80	0.70	23.08
mcp500-1	0	104.31	3.03	146.30
mcp500-2	0	125.19	3.58	167.39
mcp500-3	0	127.05	3.94	301.72
mcp500-4	0	134.67	4.58	176.47
qap10	0	12.03	2.50	42.36
qap5	0	0.84	0.06	6.08
qap6	0	1.42	0.13	4.27
qap7	0	2.28	0.25	6.19
qap8	0	3.84	0.58	11.22
qap9	0	7.08	1.06	21.95
qpG11	0	1081.17	58.72	998.61
qpG51	0	3520.34	113.42	1380.81
ss30	0	49.09	0.81	5.22
theta1	0	0.92	0.09	1.13
theta2	0	3.75	1.33	11.28
theta3	0	14.67	8.02	35.41
theta4	0	45.06	22.16	217.53
theta5	0	141.27	59.50	446.95
thetaG11	-7	725.06	768.33	2142.47

continued...

Problem	*tc*	t	$\underline{t}$	$\overline{t}$
truss1	0	1.09	0.06	3.42
truss2	0	6.95	0.28	29.55
truss3	0	1.36	0.05	0.09
truss4	0	1.05	0.06	1.19
truss5	0	14.44	0.66	45.80
truss6	0	41.75	2.20	183.61
truss7	0	33.86	1.39	72.20
truss8	0	18.69	1.63	23.88

REFERENCES

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] D. L. Applegate, W. Cook, S. Dash, and D. G. Espinoza, *Qsopt_ex*. World Wide Web. `http://www.dii.uchile.cl/~daespino/QSoptExact_doc/main.html`.

[3] V. Balakrishnan and E. Feron, eds., *Linear Matrix Inequalities in Control Theory and Applications,* special edition of *International Journal of Robust and Nonlinear Control*, vol. 6, no. 9/10, 1996.

[4] M. Berz et al., *COSY Infinity*. World Wide Web. `http://www.bt.pa.msu.edu/index_files/cosy.htm`.

[5] B. Borchers, *SDPLIB 1.2, A Library of Semidefinite Programming Test Problems*, Optimization Methods and Software, 11 (1999), pp. 683–690.

[6] D. Chaykin, *A C++ implementation of rigorous error bounds in semidefinite programming.* In progress.

[7] C. A. Floudas, *Deterministic Global Optimization - Theory, Methods and Applications*, vol. 37 of Nonconvex Optimization and Its Applications, Kluwer Academic Publishers, Dordrecht, Boston, London, 2000.

[8] R. Fourer and D. M. Gay, *Experience with a primal presolve algorithm*, in Large Scale Optimization: State of the Art, W. W. Hager, D. W. Hearn, and P. M. Pardalos, eds., Kluwer Academic Publishers Group, Norwell, MA, USA, and Dordrecht, The Netherlands, 1994, pp. 135–154.

[9] R. M. Freund, F. Ordóñez, and K. C. Toh, *Behavioral measures and their correlation with ipm iteration on semi-definite programming problems*, Mathematical Programming. To appear.

[10] K. Fujisawa, Y. Futakata, M. Kojima, K. Nakata, and M. Yamashita, *SDPA-M (SemiDefinite Programming Algorithm in MATLAB) User's Manual — Version 2.00*, Research Report B-359, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-0033, Japan, 2000. Revised: July 2003.

[11] L. Granvilliers and F. Benhamou, *Algorithm 852: RealPaver: An Interval Solver using Constraint Satisfaction Techniques*, ACM Transactions on Mathematical Software, 32 (2006), pp. 138–156. `http://doi.acm.org/10.1145/1132973.1132980`.

[12] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1 (1981), pp. 169–197.

[13] E. R. Hansen, *Global Optimization using Interval Analysis*, Marcel Dekker, New York, 1992.

[14] C. Jansson, *VSDP: Verified SemiDefinite Programming, User's Guide*. Beta Version 0.1, to appear.

[15] ———, *A Self-Validating Method for Solving Linear Programming Problems with Interval Input Data*, Computing, Suppl. 6 (1988), pp. 33–45.

[16] ———, *A rigorous lower bound for the optimal value of convex optimization problems*, J. Global Optimization, 28 (2004), pp. 121–137.

[17] ———, *Rigorous Lower and Upper Bounds in Linear Programming*, SIAM J. Optimization (SIOPT), 14 (2004), pp. 914–935.

[18] R. B. Kearfott, *GlobSol*. World Wide Web. `http://interval.louisiana.edu`.

[19] ———, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publisher, Dordrecht, 1996.

[20] ———, *On proving existence of feasible points in equality constrained optimization problems*, Mathematical Programming, 83 (1998), pp. 89–100.

[21] C. Keil, *Lurupa – Rigorous Error Bounds in Linear Programming*, in Algebraic and Numerical Algorithms and Computer-assisted Proofs, B. Buchberger, S. Oishi, M. Plum, and S. Rump, eds., no. 05391 in Dagstuhl Seminar Proceedings, Internationales Begegnungs-

und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. http://drops.dagstuhl.de/opus/volltexte/2006/445.
[22] ———, *Verified Linear Programming – a Comparison*. Submitted, 2007.
[23] C. Keil and C. Jansson, *Computational Experience with Rigorous Error Bounds for the Netlib Linear Programming Library*, Reliable Computing, 12 (2006), pp. 303–321.
[24] M. Kiyomi, *exlp*. World Wide Web. http://members.jcom.home.ne.jp/masashi777/exlp.html.
[25] O. Knüppel, *PROFIL/BIAS and extensions, Version 2.0*, tech. rep., Inst. f. Informatik III, Technische Universität Hamburg-Harburg, 1998.
[26] T. Koch, *perPlex*. World Wide Web. http://www.zib.de/koch/perplex.
[27] R. Krawczyk, *Fehlerabschätzung bei linearer Optimierung*, in Interval Mathematics, K. Nickel, ed., vol. 29 of Lecture Notes in Computer Science, Springer Verlag, Berlin, 1975, pp. 215–222.
[28] Y. Lebbah, *ICOS (Interval COnstraints Solver)*. World Wide Web. http://www.essi.fr/~lebbah/icos/index.html.
[29] MATLAB User's Guide, Version 6, *The MathWorks Inc.*, 2000.
[30] G. Mayer, *Result verification for eigenvectors and eigenvalues*, in Topics in validated computations. Proceedings of the IMACS-GAMM international workshop, Oldenburg, Germany, 30 August - 3 September 1993, J. Herzberger, ed., Stud. Comput. Math. 5, Amsterdam, 1994, Elsevier, pp. 209–276.
[31] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.
[32] A. Nemirovski, *Lectures on Modern Convex Optimization*, 2003.
[33] Netlib, *Netlib linear programming library*. http://www.netlib.org/lp.
[34] A. Neumaier, *Interval Methods for Systems of Equations*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1990.
[35] ———, *Introduction to Numerical Analysis*, Cambridge University Press, 2001.
[36] ———, *Complete Search in Continuous Global Optimization and Constraint Satisfaction*, Acta Numerica, 13 (2004), pp. 271–369.
[37] A. Neumaier and O. Shcherbina, *Safe bounds in linear and mixed-integer programming*, Mathematical Programming, Ser. A, 99 (2004), pp. 283–296.
[38] F. Ordóñez and R. M. Freund, *Computational experience and the explanatory value of condition measures for linear optimization*, SIAM J. Optimization, 14 (2003), pp. 307–333.
[39] M. V. Ramana, L. Tunçel, and H. Wolkowicz, *Strong duality for semidefinite programming*, SIAM J. on Optimization, 7 (1997), pp. 641–662.
[40] J. Renegar, *Linear programming, complexity theory and elementary functional analysis*, Math. Programming, 70 (1995), pp. 279–351.
[41] J. B. Rosen and S. Suzuki, *Construction of Nonlinear Programming Test Problems*, Communication of ACM, 8 (1965), p. 113.
[42] S. M. Rump, *Validated Solution of Large Linear Systems*, in Validation numerics: theory and applications, R. Albrecht, G. Alefeld, and H. Stetter, eds., vol. 9 of Computing Supplementum, Springer, 1993, pp. 191–212.
[43] ———, *Verification Methods for Dense and Sparse Systems of Equations*, in Topics in Validated Computations — Studies in Computational Mathematics, J. Herzberger, ed., Elsevier, Amsterdam, 1994, pp. 63–136.
[44] ———, *INTLAB - Interval Laboratory, a Matlab toolbox for verified computations, Version 5.1*, 2005.
[45] R. E. Skelton and T. Iwasaki, *Increased roles of linear algebra in control education*, IEEE Control Syst. Mag., 15 (1995), pp. 76–89.
[46] M. J. Todd, *Detecting Infeasibility in Infeasible-Interior-Point Methods for Optimization*, in Foundations of Computational Mathematics, Minneapolis 2002, F. Cucker, R. DeVore, P. Olver, and E. Süli, eds., no. 312 in London Mathematical Society Lecture Note Series, Cambridge University Press, 2004, pp. 157–192.
[47] R. H. Tütüncü, K. C. Toh, and M. J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Math. Program., 95B (2003), pp. 189–217.
[48] P. Van Hentenryck, P. Michel, and Y. Deville, *Numerica: A Modelling Language for Global Optimization*, MIT Press Cambridge, 1997.
[49] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review, 38 (1996), pp. 49–95.