VSDP: Verified SemiDefinite Programming
USER'S GUIDE,
Beta Version 0.1 for MATLAB 7.0

Christian Jansson

December 18, 2006

Abstract

VSDP is a MATLAB software package for rigorously solving semidefinite programming problems. It expresses these problems in a notation closely related to the form given in textbooks and scientific papers. Functions for computing verified forward error bounds of the true optimal value and verified certificates of feasibility and infeasibility are provided. All rounding errors due to floating point arithmetic are taken into account. Computational results are given, including results for the SDPLIB benchmark problems. This package supports interval input data and sparse format.

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

Contents

1 Introduction

> "*If error analysis could be automated, the mysteries of floating-point arithmetic could remain hidden; the computer would append to every displayed numerical result a modest over-estimate of its uncertainty due to imprecise data and approximate arithmetic, or else explain what went wrong, and all at a cost not much higher than if no error analysis had been performed. So far, every attempt to achieve this ideal has been thwarted.*"
>
> **William M. Kahan, The Regrettable Failure of Automated Error Analysis, 1989**.

Semidefinite Programming has emerged as a powerful tool in many different areas ranging from control engineering to structural design, combinatorial optimization and global optimization (see the Handbook of Semidefinite Programming [31]). One reason is that there exists a kind of calculus of conic quadratic and semidefinite representable sets and functions, which offers a systematic way to recognize and reformulate a convex program as a semidefinite program. This calculus is applied for example in CVX [5], an optimization modelling language which is designed to support the formulation and construction of optimization problems that the user intends from the outset to be convex. On the other hand non-convex problems are frequently solved by using convex relaxations, where consequently SDP-solvers can also be used.

Many algorithms for solving semidefinite programming problems require that appropriate rank conditions are fulfilled, and that strictly feasible solutions of the primal and the dual problem exist, i.e *Slater's constraint qualification* holds. All these solvers do not provide a guaranteed accuracy or prove existence of optimal solutions. Nevertheless, appropriate warranties for computed results and rigorous forward error bounds can be useful in many applications, especially for ill-conditioned problems with dependencies in the input data, or ill-posed problems. It is well-known that for such problems (but not solely) rounding errors may affect the computation, and even many state-of-the-art solvers may produce erroneous approximations (cf. Neumaier and Shcherbina [22]).

Ill-conditioned and ill-posed problems are not rare in practice. In the paper of Ordóñez and Freund [23] it is stated that 71% of the lp-instances in the NETLIB Linear Programming Library are ill-posed, and recently Freund, Ordóñez and Toh [3] have shown that 32 out of 85 problems of the SDPLIB are ill-posed. Likewise, several classes of ill-posed semidefinite programming problems arise from combinatorial problems (see for example Gruber and

Rendl [7] and Gruber et al. [6]). Moreover, there are SDP problems with zero duality gap but no strict complementary primal-dual solution pair. Such problems are also known as *hard instances*, and can result in both theoretical and numerical difficulties (see Wei and Wolkowicz [30]).

VSDP is a software package which computes verified forward error bounds. *Verified* or sometimes also called *rigorous error bounds* means that the computed results are claimed to be valid with mathematical certainty even in the presence of rounding errors due to floating point arithmetic; that is, all errors are estimated correctly. VSDP is based on a rigorous postprocessing applied to the output of semidefinite programming solvers. It is of particular importance that each solver can be used, and the solver need not produce any error bounds, neither in the forward nor in the backward error sense. This package implements techniques described in [11] and [12] and has several features:

- computes **verified lower and upper bounds** of the optimal value for semidefinite programs,
- proves **existence of feasible and optimal solutions**, also for LMI's,
- provides **rigorous certificates of infeasibility**,
- facilitates to **solve approximately** the problem by using well-known semidefinite programming solvers,
- can handle **several formats** (full, sparse, SDPA format, interval data, ...)

This guide is organized as follows. In the next section some hints are given how to install VSDP, and Section 3 contains some remarks about VSDP. Section 4 contains a short theoretical introduction together with step-by-step tutorial of VSDP. In Section 5 it is shown how problems from the SDPLIB can be solved, and moreover a short statistic of the numerical results is presented. The following section 6 is devoted to ill-conditioned and ill-posed problems. Especially, results about the ill-posed SDPLIB problems, some ill-posed combinatorial problems as well as ill-conditioned large scale SDP problems in Electronic Structure Calculations are presented. Some elementary facts about interval arithmetic and INTLAB are introduced in Section 7, and it is demonstrated how SDPLIB problems can be solved with interval input data. Section 8 contains some remarks about running times of VSDP. Last some conclusions are given, and in the Appendix the tables containing all numerical results are displayed.

2 Installation

VSDP is completely written in MATLAB [15] and requires version 7.0 or higher [15]. VSDP uses the MATLAB-toolbox INTLAB [26], (INTerval LABoratory), version 5.3 which is available from

`http://www.ti3.tu-harburg.de/rump/intlab`

The semidefinite solver SDPT3, version 3.02 [28]

`http://www.math.cmu.edu/~reha/sdpt3.html`

or alternatively the semidefinite solver SDPA [32]

`http://grid.r.dendai.ac.jp/sdpa/download.html`

is required. If in the VSDP M-file `SDP_GLOBALPARAMETER` the global variable `VSDP_CHOICE_SDP = 1`, then the SDPT3 solver is chosen, and if `VSDP_CHOICE_SDP = 2` then SDPA is chosen. The VSDP source files can be downloaded from

`http://www.ti3.tu-harburg.de/jansson/vsdp`

The installation was successful if the VSDP routine `DEMOVSDP` runs through.

3 About VSDP

Numerical errors fall into three classes: Intentional errors, like idealized models or discretization, unavoidable rounding errors, and unintentional bugs and blunders in software and hardware. VSDP controls rigorously rounding errors. Please contact me, if you discover unintentional bugs (`jansson@tu-harburg.de`). In the latter case the version numbers of the used software together with an appropriate M-file should be provided. Any suggestions, comments, and criticism are welcome.

The intention was not to write a very sophisticated code which minimizes the storage and the computing time. The major focus was to write the MATLAB files in a concise form, such that also students can easily understand the principles of verification used here. An unavoidable consequence is the interpretation overhead of the MATLAB and INTLAB environments. Part of this software was presented in some lectures on optimization with result verification during the winter semester 2005 for students of electrical engineering and computer science.

It is in the nature of verification methods that not every approximate solution can be verified, as well as approximate solvers cannot compute an approximate solution for each solvable problem. However, a good verification method should compute rigorous error bounds in almost all well-posed cases, whenever approximate solvers can compute a sufficiently good

approximation. The numerical experiments of VSDP with the SDPLIB suite exhibit that at least for problems of middle size (up to thousands of constraints and millions of variables) rigorous lower (upper bounds) of the optimal value can be computed, provided the distance to dual infeasibility (primal infeasibility) is greater than zero. Notice that feasible problems with a zero distance to infeasibility are infeasible for appropriate arbitrarily small perturbations of the data, and hence are ill-posed (see Renegar's condition measure [24] and [23]). Nevertheless, VSDP allows to treat ill-posed problems rigorously if an a priori assumption about the existence of an optimal solution and its magnitude is known.

4 Quick Start

> *"...the routine can produce a computed result that is nonsensical and the application proceeds as if the results were correct. ... What should you do? The first defense is to adopt a sceptical attitude toward numerical results until you can verify them by independent methods."*
> **When Good Computers Make Bad Calculations, R. Meyer, President NAG, 2001**, `www.nag.com/local/whitepapers/index.asp`

VSDP solves rigorously *semidefinite programming problems* in *block diagonal form*:

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

where $b \in \mathbf{R}^m$, and $C_j, A_{ij}, X_j \in S^{s_j}$, the linear space of real symmetric $s_j \times s_j$ matrices. By $\langle ., . \rangle$ we denote the usual inner product on the linear space of symmetric matrices, which is defined as the trace of the product of two matrices. $X \succeq 0$ means that X is positive semidefinite. Hence, $\succeq$ denotes the *Löwner partial order* on this linear space. If the set of feasible solutions is empty then we define $f_p^* := +\infty$, and $f_p^* := -\infty$ if the problem is unbounded.

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

The *Lagrangian dual* of (1) is

$$f_d^* := \max b^T y \quad \text{s.t. } \sum_{i=1}^{m} y_i A_{ij} \preceq C_j, \quad j = 1, \ldots, n, \tag{2}$$

where $y \in \mathbf{R}^m$. We assign $f_d^* := -\infty$, if the set of dual feasible solutions is empty, and $f_d^* := +\infty$ in the unbounded case. The constraints $\sum_{i=1}^m y_i A_{ij} \preceq C_j$ are called *linear matrix inequalities (LMI's).*

Both problems are connected by the *weak duality* condition

$$f_d^* \leq f_p^*. \tag{3}$$

Strong duality requires in contrast to linear programming additional conditions (see Vandenberghe and Boyd [29]).

Theorem 1 (Duality Theorem)

a) If (1) is strictly feasible (i.e. there exist feasible positive definite matrices X_j for $j = 1, \ldots, n$) and f_p^* is finite, then $f_p^* = f_d^*$, the dual supremum is attained, and the set of dual optimal solutions is bounded.

b) If (2) is strictly feasible (i.e. there exist some $y \in \mathbf{R}^m$ such that $C_j - \sum_{i=1}^m y_i A_{ij}$ are positive definite for $j = 1, \ldots, n$) and f_d^* is finite, then $f_p^* = f_d^*$, the primal infimum is attained, and the set of primal optimal solutions is bounded.

In general, one problem may have optimal solutions and its dual is infeasible, or the duality gap may be finite and positive at optimality. The strict feasibility assumptions in Theorem 1 are called *Slater's constraint qualifications.*

VSDP computes rigorous results by carefully postprocessing the output of semidefinite programming solvers. Basically, two theorems (see below) are used which allow to bound rigorously the primal and the dual optimal value. For further details, results and proofs we refer to [11] and [12]. Although this guide is written independently of these papers, a rough knowledge of them is beneficial. Rigorous error bounds in the special case of linear programming can be found in Neumaier and Shcherbina [22] and [10], and the general convex case is treated in [9].

VSDP exploits the block-diagonal structure by an $n \times 2$ cell-array `blk`, n cell-arrays `C, X,` and an $m \times n$ cell-array `A` as follows: The j-th blocks `C{j},X{j}` and the blocks `A{i,j}` for $i = 1, \ldots, m$ are real symmetric matrices of common size s_j which is expressed by

`blk{j,1} = 's', blk{j,2} =` s_j. [1]

[1] At the moment only real symmetric matrices are incorporated, which makes the first instruction redundant. But in future versions I want to treat also other types of matrices. I mention that this structure is closely related to an older version of SDPT3.

The block-matrices `C{j}` and `A{i,j}` may be symmetric floating-point or interval matrices, and can be stored in dense or sparse format.

For the purpose of illustration, we start with the following semidefinite programming problem of dimension $m = 4$, $n = 1$, and $s_1 = 3$, i.e. the matrices consist of only one block. The problem depends on a fixed parameter `DELTA`:

```
>> DELTA = 1e-4;

>> C{1}    = [ 0   1/2   0;
               1/2 DELTA  0;
                0    0   DELTA ];
>> A{1,1} =  [ 0 -1/2 0;
              -1/2 0  0;
                0   0  0];
>> A{2,1} =  [ 1 0 0 ;
                0 0 0;
                0 0 0];
>> A{3,1} =  [ 0 0 1;
                0 0 0;
                1 0 0];
>> A{4,1} =  [ 0 0 0;
                0 0 1;
                0 1 0];
>> b = [1; 2*DELTA; 0; 0];
>> blk{1,1} = 's'; blk{1,2} = 3;
```

The routine `VSDPCHECK` may be used to perform a check whether there are inconsistencies in the input data:

```
>> [m, n] = vsdpcheck(blk,A,C,b)
m =
  4
n =
  1
```

Sine no error messages occur, all checks w.r.t. the sizes and the data types of the input data are carried out correctly.

It is easy to prove that this problem has a zero duality gap with optimal value -0.5 for every `DELTA > 0`. The primal optimal nonzero coefficients are $X_{11}^* = 2$ `Delta`, $X_{12}^* = -1, X_{22}^* = 1/(2$ `Delta`$)$, and the dual optimal nonzero coefficient is $y_2^* = -1/(4$ `Delta`$)$.

For `DELTA = 0` the problem is ill-posed with nonzero duality gap, and for negative `DELTA` it is primal and dual infeasible. Especially, it follows that the optimal value is not continuous in `DELTA = 0`.

At the moment the two semidefinite solvers SDPT3 (version 3.02) [28] and SDPA [32] can be used in VSDP via the routine `MYSDPS`. Therefore, VSDP can be used for computing only approximations with different solvers. The user can integrate other solvers very easily. In the following, our intention is **not** to compare these solvers (for a comparison see Mittelmann [16]), but we want to illustrate that the proposed rigorous error bounds depend very much on the quality of the computed approximations. By default, the function `MYSDPS` calls the semidefinite programming solver SDPT3:

```
>> [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b);
```

The output consists of approximations of (i) the primal and dual optimal value both stored in `objt`, (ii) the primal and dual solutions `Xt, yt, Zt`, and (iii) information about termination and performance stored in `info`.

```
>> objt, termination = info(1),
objt =
   -5.000322560803474e-001   -5.000000062262615e-001
termination =
     0
```

The solver terminates without any warning if `termination = 0`. The first four decimal digits of the primal and dual optimal value are correct, but week duality is not satisfied since the approximate primal optimal value is smaller than the dual one. In other words, the algorithm is not backward stable for this example. The approximate solutions `Xt` and `yt` are:

```
>> celldisp(Xt), yt
Xt{1} =
    2.0001290435698e-004   -1.0000000000562e+000                         0
   -1.0000000000562e+000    4.9996774230724e+003                         0
                        0                       0    1.6686221506340e-005
yt =
```

```
   -9.0168234380828e-005
   -2.4995491899594e+003
                       0
                       0
```

Hence, for the nonzero coefficients about four decimal digits are correct. If we set in the file `SDP_GLOBALPARAMETER` the global variable `VSDP_CHOICE_SDP = 2`, then the SDPA solver is chosen via `MYSDPS`, and we obtain

```
>> [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b);
>> objt, termination = info(1)

objt =
   -8.472053923768221e-001    6.995412095264103e-001
termination =
     3
```

No decimal digit of the optimal value is correct, but a warning is given, which indicates that the problem is primal or dual infeasible. Also the approximate primal and dual optimal solutions are nonsensical.

To obtain more reliability we can use the function `VSDPLOW` which computes a verified lower bound of the primal optimal value by using a previously computed dual approximation `yt`. This function is based on the following theorem:

Theorem 2 *Let $(\overline{x}_j)$ be a nonnegative vector which may also have infinite components. Assume that either there are no primal feasible solutions, or there exists a primal optimal solution (X_j) where its maximal eigenvalues are bounded by $(\overline{x}_j)$. Let $\widetilde{y} \in \mathbf{R}^m$ (for example a computed dual approximation). Let*

$$D_j = C_j - \sum_{i=1}^{m} \widetilde{y}_i A_{ij} \text{ and } \underline{d}_j \leq \lambda_{\min}(D_j) \text{ for } j = 1, \ldots, n, \tag{4}$$

where $\lambda_{\min}$ denotes the smallest eigenvalue. Assume that D_j has at most l_j negative eigenvalues. Then:

(i) The primal optimal value is bounded from below by

$$f_p^* \geq b^T \widetilde{y} + \sum_{j=1}^{n} l_j \underline{d}_j^- \overline{x}_j =: \underline{f}_p^* \text{ where } \underline{d}_j^- := \min(0, \underline{d}_j). \tag{5}$$

(ii) Moreover, if

$$\underline{d}_j \geq 0 \text{ for } \overline{x}_j = +\infty, \tag{6}$$

then the right hand side $\underline{f}_p^*$ *is finite.*

(iii) If $\underline{d}_j \geq 0$ *for* $j = 1, \ldots, n$*, then* $\widetilde{y}$ *is dual feasible and* $f_d^* \geq \underline{f}_p^*$*, and if moreover* $\widetilde{y}$ *is optimal, then* $f_d^* = \underline{f}_p^*$*.*

(iv) If $\underline{d}_j > 0$ *for* $j = 1, \ldots, n$*, then* $f_d^* = f_p^*$ *and the set of primal optimal solutions is bounded or empty.*

There are no assumptions about the quality of $\widetilde{y}$, but assertion (iii) implies that an approximation close to optimality should produce a rigorous lower bound with modest overestimation. The lower bound $\underline{f}_p^*$ sums up the approximate dual value $b^T\widetilde{y}$ and the violations of dual feasibility by taking into account the signs and multiplying these violations with appropriate primal weights. Observe the subtle difference that $\underline{f}_p^*$ is a lower error bound of the primal optimal value, not of the dual optimal value. Hence, for problems with non-zero duality gap it can happen that the dual optimal value is smaller than this lower bound. We mention that all required quantities can be computed rigorously.

First, we treat the situation where all components $(\overline{x}_j)$ are infinite (i.e. finite bounds are not known or are not available). We describe briefly how `VSDPLOW` utilizes this theorem. If the computed bounds $\underline{d} = (\underline{d}_j)$ are nonnegative, then (ii) and (iii) imply that $\underline{f}_p^*$ is a finite lower bound of the primal and the dual optimal value and it is obtained via formula (5) (in the current implementation we use Weyl's Perturbation Theorem for computing $\underline{d}$). Moreover, $\widetilde{y}$ is a dual feasible solution, and provides a rigorous certificate of dual feasibility. If $\underline{d}_j$ is negative for some infinite $\overline{x}_j$, then it follows immediately that the lower bound $\underline{f}_p^*$ is infinite, and therefore of no value. In this case `VSDPLOW` tries to find iteratively new dual approximations $\widetilde{y}$ by solving appropriate perturbed semidefinite problems until either $\underline{d}$ is nonnegative, or dual infeasibility for the perturbed problem is indicated by the semidefinite solver. In the latter case `VSDPLOW` sets $\underline{f}_p^* = -\infty$, and the certificate of dual feasibility is defined as `NaN`.

`VSDPLOW` uses as starting point the already computed approximations `Xt,yt,Zt` (here previously computed by SDPA), and the call has the form

```
>> [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt)
```

The output `fL, Y`, and `dl` corresponds to the lower bound $\underline{f}_p^*$, the certificate of dual feasibility $\widetilde{y}$, and the corresponding vector of eigenvalue bounds $\underline{d}$, respectively. In the case where `Y = NaN`, all components of the vector `dl` are set NaN.

In our case this call yields

```
fL =
   -Inf
 Y =
    NaN
dl =
    NaN
```

Therefore, with the SDPA approximations the rigorous lower bound is infinite, and dual feasibility is not verified.

Secondly, we describe the case, where additional information about finite upper bounds ($\overline{x}_j$) of the maximal eigenvalues of a primal optimal solution (X_j) is available. Then condition (6) is satisfied, and `VSDPLOW` executes formula (5) yielding a finite lower bound of the optimal value.

For our example, let the corresponding upper bound be $\overline{x} = 10^5$:

```
>> xu = 1e5;
>> [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt,xu)
   fL =
      -7.932462706436944e+001
   Y =
      NaN
   dl =
      NaN
```

Hence, a lower bound far away from the optimal value -0.5 is computed. A certificate of dual feasibility is not found. This is not surprising since the computed SDPA approximations are far from optimality.

If we set in the file `SDP_GLOBALPARAMETER` the global variable `VSDP_CHOICE_SDP = 1` (default), then the solver SDPT3 is chosen via `MYSDPS`, and working with the SDPT3 approximations yields

```
>> [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b);
>> [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt)
   fL =
     -5.000000062262615e-001
   Y =
     -9.016817444180712e-005
     -2.499549190259066e+003
```

```
                          0
                          0
dl =
   4.319659051028185e-013
```

Hence, for the same problem, with SDPT3 a finite rigorous lower bound close to the optimal value together with a certificate of dual feasibility is computed. Since `dl` is positive, the dual problem contains the strictly dual feasible interior point `Y`, i.e. the dual Slater condition holds true. Therefore, strong duality is verified, and `fL` is also a lower bound of the dual optimal value.

If we use, as above, the upper bound `xu = 1e5`, then for this example we obtain the same output:

```
>> xu = 1e5;
>> [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt,xu)
fL =
   -5.000000062262615e-001
Y  =
   -9.016817444180712e-005
   -2.499549190259066e+003
                         0
                         0
dl =
    4.319659051028185e-013
```

We see that **the quality of the rigorous results depends strongly on the quality of the computed approximations**, which is an immediate consequence of the proposed error bound.

Similarly, with function `VSDPUP` we can compute a verified upper bound `fU` of the dual optimal value using the previously computed approximations. This function is based on the following theorem:

Theorem 3 *Let $\overline{y}$ be a nonnegative vector which may have also infinite components. Assume that either there exist no dual feasible solutions, or there exists a dual optimal solution where its absolute value is bounded by $\overline{y}$. Let $\widetilde{X}_j \in S^{s_j}$ for $j = 1, \ldots, n$, and assume that each $\widetilde{X}_j$ has at most k_j negative eigenvalues. For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, let*

$$r_i \geq |b_i - \sum_{j=1}^{n} \langle A_{ij}, \widetilde{X}_j \rangle| \tag{7}$$

$$\underline{\lambda}_j \le \lambda_{\min}(\widetilde{X}_j), \ \textit{and} \tag{8}$$

$$\varrho_j \ge \sup\{\lambda_{\max}(C_j - \sum_{i=1}^{m} y_i A_{ij}) : \ -\overline{y} \le y \le \overline{y}, \ C_j - \sum_{i=1}^{m} y_i A_{ij} \succeq 0\}. \tag{9}$$

Then:

(i) The dual optimal value is bounded from above by

$$f_d^* \le \sum_{j=1}^{n} \langle C_j, \widetilde{X}_j \rangle - \sum_{j=1}^{n} k_j \underline{\lambda}_j^- \varrho_j + \sum_{i=1}^{m} r_i \overline{y}_i =: \overline{f}_d^*, \tag{10}$$

where $\underline{\lambda}_j^- := \min(0, \underline{\lambda}_j)$.

(ii) Moreover, if

$$r_i = 0 \textit{ for } \overline{y}_i = +\infty \textit{ and } \underline{\lambda}_j \ge 0 \textit{ for } \varrho_j = +\infty, \tag{11}$$

then the right hand side $\overline{f}_d^*$ *is finite.*

(iii) If $\underline{\lambda}_j \ge 0$ *and* $r_i = 0$ *for all* i, j*, then* $(\widetilde{X}_j)$ *is primal feasible and* $f_p^* \le \overline{f}_d^*$*, and if moreover* $(\widetilde{X}_j)$ *is optimal, then* $f_p^* = \overline{f}_d^*$.

(iv) If $\underline{\lambda}_j > 0$ *and* $r_i = 0$ *for all* i, j*, then* $f_d^* = f_p^*$ *and the set of dual optimal solutions is bounded or empty.*

There are no assumptions about the quality of the primal approximation $(\widetilde{X}_j)$, but assertion (iii) implies that an approximation close to optimality should produce a rigorous upper bound with modest overestimation.. The bound $\overline{f}_d^*$ sums up the primal objective value $\sum_{j=1}^{n} \langle C_j, \widetilde{X}_j \rangle$ and the violations of primal feasibility (r_i and $\underline{\lambda}_j^-$) by taking into account the signs and multiplying these violations with appropriate weights ϱ_j and $\overline{y}_i$. Notice that only the dual optimal value is bounded from above. Hence, for problems with non-zero duality gap the rigorous upper bound may be smaller than the lower bound. However, this cannot happen for problems where strong duality holds valid, for example linear programming problems, or if strictly feasible solutions are verified with `VSDPUP` or `VSDPLOW`. For well-posed as well as for ill-posed problems with zero duality gap $\overline{f}_d^*$ is also an upper bound of the primal optimal value. The quantities $r_i, \underline{\lambda}_j$ and ϱ_j can be computed rigorously by using interval arithmetic.

First, we treat the case where all components of $(\overline{y}_i)$ are infinite and describe shortly `VSDPUP`. The primal equations $\sum_{j=1}^{n} \langle A_{ij}, X_j \rangle = b_i, \ (i = 1, \ldots, m)$ are solved with a linear interval solver. If an enclosure `X` containing an exact solution of the primal equations can be computed, it is

automatically proved that the linear system of equations has full rank. Then, with Weyl's Perturbation Theorem (c.f. routine `VEIGSYM`) verified lower bounds $\underline{\lambda}_j$ of the eigenvalues for the blocks in `X` are computed and stored in a vector `lb`. If `lb` is nonnegative, then condition (11) is fulfilled, and `X` contains a primal feasible solution. If `lb` is positive, then `X` contains a strictly primal feasible solution; that is, the primal Slater condition holds true. If `lb` has negative components, then `VSDPUP` tries to find iteratively new primal approximations by solving appropriate perturbed semidefinite problems, until either `lb` is nonnegative, or primal infeasibility is indicated by the semidefinite solver for the perturbed problem. In the latter case `VSDPUP` sets $\overline{f}_d^* = +\infty$, and the certificate of primal feasibility `X` and the vector `lb` are defined as `NaN`.

The call of `VSDPUP` has the form

```
>> [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt);
```

The output `fU, X` and `lb` corresponds to the upper bound $\overline{f}_d^*$, the interval block-diagonal matrix (containing the rigorous certificate of primal feasibility), and the vector of eigenvalue bounds $\underline{\lambda}_j$, respectively. Because of clarity, we display only the first column of `X` in midpoint radius format; that is, the first component denotes the midpoint of the interval and the second one is the radius.

```
>> intvalinit('DisplayMidRad')
>> fU, Xout = X{1}; Xout(:,1), lb   % Only one block

fU =
   -4.999677693282600e-001
intval ans =
   <  2.000000000000000e-004, 2.710505431213762e-020>
   < -1.000000000000000e+000, 0.000000000000000e+000>
   <  0.000000000000000e+000, 0.000000000000000e+000>
lb =
    1.289076539560735e-008
```

Hence, we obtain a lower bound of the optimal value together with the interval matrix `X` containing a strictly feasible primal solution. The radius of the first component is of order 10^{-20}. Since, `lb > 0` Slater's condition is fulfilled and strong duality holds valid.

Summarizing, for the above problem we have verified strong duality by using the SDPT3 approximations. Moreover, the inequality

$$-5.000000062262615e-001 = \underline{f}_p^* \leq f_d^* = f_p^* \leq \overline{f}_d^* = -4.999677693282600e-001,$$

is fulfilled, and certificates of strictly primal and strictly dual feasible solutions are computed. The Strong Duality Theorem implies that the primal and the dual problem have a nonempty compact set of optimal solutions. The upper and lower bounds of the optimal value show a modest overestimation, mainly due to the accuracy of SDPT3. The bounds are rigorous, and thus satisfy weak duality.

In the case where finite upper bounds ($\overline{y}_i$) are available, the condition (11) is trivially satisfied, and `VSDPUP` computes rigorously the finite upper bound (10) without verifying primal feasibility. Hence, solving the primal equations with a linear interval solver is not necessary, yielding an acceleration in most cases (for details see [11]). The fact that primal feasibility is not verified is expressed in `VSDPUP` by `NaN` for `X` and `lb`. The call of `VSDPUP` with using the finite upper bounds $(\overline{y}_i) = 10^5$ has the form

```
>> yu = 1e5 * [1 1 1 1]';
>> [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt,yu)

fU =
   -4.987416925451506e-001
X =
   NaN
lb =
   NaN
```

For this example the upper bound is worse than the previous one.

For negative DELTA our problem is primal and dual infeasible.

```
>> DELTA = -1e-4;

>> C{1}   = [ 0   1/2   0 ;
              1/2 DELTA 0 ;
              0   0   DELTA ];

>> b = [1; 2*DELTA; 0; 0];
```

The routine SDPT3

```
>> [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b);
>> info(1)
ans =
   1
```

gives the termination code `1` with the warning `primal infeasibility has deteriorated too much`. In accordance the rigorous lower bound is

```
>> [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt)
fL =
   -Inf
Y  =
    NaN
dl =
    NaN
```

and for the upper bound we obtain

```
>> [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt)
fU =
    Inf
X  =
    NaN
lb =
    NaN
```

Notice that these are the exact optimal values, since the problem is primal and dual infeasible. However, `VSDLOW` and `VSDPUP` only indicate infeasibility, but do not proof this property (both certificates are `NaN`). This is the task of `VSDPINFEAS` which tries to verify primal or dual unbounded rays that deliver certificates of dual or primal infeasibility. For proving primal infeasibility we can use `VSDPINFEAS` as follows:

```
>> choose = 'p';
>> [isinfeas, X, Y] = vsdpinfeas(blk,A,C,b,choose,Xt,yt,Zt),
isinfeas =
   0
 X =
   NaN
 Y =
   NaN
```

The first line declares that we want to prove primal infeasibility. In the second line the input of `VSDPINFEAS` contains the data of the problem and the selection variable `choose` as well as the previously computed SDPT3 approximations `Xt, yt, Zt`. Since `isinfeas = 0` the

routine `VSDPINFEAS` has not proved primal infeasibility for this example, and the certificates `X, Y` are set to `NaN`. The reason is that a dual improving ray y must satisfy

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

and for this example a short calculation yields

$$-\sum_{i=1}^{m} y_i A_{ij} = \begin{pmatrix} -y_2 & \frac{y_1}{2} & -y_3 \\ \frac{y_1}{2} & 0 & -y_4 \\ -y_3 & -y_4 & 0 \end{pmatrix}.$$

Hence, each improving ray has a zero eigenvalue. Since, we cannot compute exact eigenvalues with enclosure methods, there is in general always a small overestimation. Hence, positive semidefiniteness cannot be verified for matrices with zero eigenvalues. This is out of scope of VSDP.

Proving dual infeasibility is as follows:

```
>> choose = 'd';
>> [isinfeas, X, Y] = vsdpinfeas(blk,A,C,b,choose,Xt,yt,Zt),
isinfeas =
   0
 X =
   NaN
 Y =
   NaN
```

Also dual infeasibility cannot be proved. An easy calculation shows that the primal improving ray must satisfy the inequality

$$X = \begin{pmatrix} 0 & 0 & 0 \\ 0 & X_{22} & 0 \\ 0 & 0 & x_{53} \end{pmatrix}$$

and with the same argument as above positive semidefiniteness cannot be verified.

The example

```
>> C{1} = [0 0; 0 0];
>> A{1,1} = [1 0; 0 0];
```

```
>> A{2,1} = [0 1; 1 0.005];
>> b = [-0.01 1]';
>> blk{1,1} = 's'; blk{1,2} = 2;
```

is primal infeasible, since the first primal equation implies `X(1,1) = -0.01`. Hence, X cannot be positive semidefinite.

```
>> choose = 'p';
>> [isinfeas, X, Y] = vsdpinfeas(blk,A,C,b,choose),
isinfeas =
   1
X =
   NaN
Y =
   -1.010835012952675e+002
   -1.083501295267454e-002
```

Now, `isinfeas = 1` shows that the problem is primal infeasible and a rigorous certificate is provided by the dual improving ray `Y`. Observe that `vsdpinfeas` can be called also without the optional input `Xt,yt,Zt`. Then appropriate approximations are generated in this routine, with the consequence of more computational effort. Since this example is dual feasible, proving dual infeasibility should not be successful:

```
>> choose = 'd';
>> [isinfeas, X, Y] = vsdpinfeas(blk,A,C,b,choose),
isinfeas =
   0
 X =
   NaN
 Y =
   NaN
```

An online demonstration of VSDP is contained in the M-file `DEMOVSDP`.

5 Rigorous Error Bounds for the SDPLIB

For computing verified results of the SDPLIB collection (Borchers [2]) it is required that these problems are encoded in SDPA sparse format and stored in an appropriate directory. Then this collection must be called:

```
>> sdplibfiles = dir('***\sdplib_vsdp/*.dat-s');
```

The routine SDPA_TO_VSDP reads such a problem in SDPA sparse format, and converts it to VSDP format; here we take the third SDPLIB problem which is from truss topology design:

```
>> i=3;
>> [blk,A,C,b] = sdpa_to_vsdp...
      (['***\sdplib_vsdp/' sdplibfiles(i).name]);
       Problem = sdplibfiles(i).name,

Problem =
arch4.dat-s
```

The size of the problem can be obtained with

```
>> blk, size(b)

blk =
    's'    [161]
    's'    [174]
ans =
    174     1
```

Hence, this is a block diagonal problem consisting of two symmetric blocks with dimensions 161 and 174 (yielding 28266 variables), and it has 174 constraints. At first, we solve this problem approximately with SDPT3, and then we compute the rigorous error bounds for the optimal value together with the required times in seconds.

```
>> tic; [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b); t = toc;
>> tic; [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt); tfL = toc;
>> tic; [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt); tfU = toc;
>> objt, fL, fU, t, tfL, tfU
objt =
   -9.726274106486620e-001   -9.726274188393249e-001
fL =
   -9.726274188393326e-001
fU =
   -9.726274053586463e-001
t =
    9.797000000000001e+000
```

```
tfL =
    6.879999999999988e-001
tfU =
    2.470300000000000e+001
```

The bounds for the optimal value provide a guaranteed accuracy of about eight decimal digits for this example. The time for computing the lower bound is small compared to the times for the approximations and for the upper bound. This behavior is very typical. There are two reasons. Firstly, solving the primal equations with a linear interval solver is time consuming. Secondly, solving perturbed semidefinite programming problems during the iteration is also time consuming.

The computed quantities `X` and `Y` are both unequal `NaN` with positive minimal eigenvalue bounds

```
>> min(lb),min(dl)
ans =
     4.491819549324221e-013
ans =
     1.459491057731760e-012
```

Hence, strictly primal and dual feasible solutions are verified, implying strong duality and compactness of the sets of optimal solutions.

Freund, Ordóñez and Toh [3] have solved 85 problems with SDPT3 out of the 92 problems of the SDPLIB. They have omitted the four infeasible problems and three very large problems where SDPT3 produced out of memory. In their paper interior-point iteration counts with respect to different measures for semidefinite programming problems are investigated, and it is pointed out that 32 are ill-posed, i.e. Renegar's condition number is infinite (cf. Renegar [24]). VSDP could compute (by using SDPT3 as approximate solver) for all 85 problems a rigorous lower bound of the optimal value and could 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 one exception; this is `hinf2`. For all 32 ill-posed problems VSDP has computed $\overline{f}_d^* = +\infty$, which reflects exactly 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 Tables 2 to 5 in the Appendix. There, 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 $a - b$, and a negative sign implies $a < b$. Therefore, negative signs in the column $\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$ show that the approximate solutions violate weak duality and are not backward stable. The computational times for computing the approximations, the rigorous lower bound and the rigorous upper bound are `t, tfL` and `tfU`, respectively. The termination code `tc = 0` means normal termination, otherwise a warning is displayed.

For the 85 test problems (not counting the 4 infeasible ones) SDPT3 (with default values) gave 7 warnings, and 2 warnings are given for well-posed problems. Hence, no warnings are 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.

Some major characteristics of the numerical results of VSDP for the well-posed SDPLIB-problems are as follows: The median of the time ratio for computing the rigorous lower (upper) bound and the approximation is $0.085, (1.99)$, respectively. The median of the guaranteed accuracy for the problems with finite condition number is $7.01 \cdot 10^{-7}$. We have used the median here because there are some outliers.

One of the largest problems which could be solved by VSDP is `thetaG51` where the number of constraints is $m = 6910$, and the dimension of the primal symmetric matrix `X` is $s = 1001$ (implying 501501 variables). For this problem SDPT3 gave the message out of memory, and we used SDPA as approximate solver. The rigorous lower and upper bounds computed by VSDP are $\underline{f}_p^* = -3.4900 \cdot 10^2$, $\overline{f}_d^* = -3.4406 \cdot 10^2$, respectively. This is an outlier because the guaranteed relative accuracy is only 0.014, which may be sufficient in several applications, but is insufficient from a numerical point of view. However, existence of optimal solutions and strong duality is proved. The times in seconds for computing the approximations, the lower and the upper bound of the optimal value are `t`= 3687.95, `tfL`= 45.17, and `tfU`= 6592.52, respectively.

6 Verification of Ill-posed problems with VSDP

> *"One might say that the majority of applied problems are, and always have been ill-posed, particularly when they require numerical answers. ..... Past experience suggests that the concepts and methods used in the discussion of ill-posed problems will in turn stimulate advances in pure mathematical analysis."*
>
> **Fritz John:Translation Editor's Preface in Tikhonov, Arsenin: Solutions of Ill-posed Problems, 1977**

Given very ill-conditioned or even ill-posed problems, frequently the guidance is that one should not trust the computed solution, because it is unstable and dominated by rounding errors due to floating point arithmetic. Occasionally, such problems are solved with a higher precision arithmetic yielding irrelevant solutions in many cases. However, in practice ill-conditioned and ill-posed systems are solved successfully by *numerical regularization methods where the solution is stabilized by including appropriate additional information* (see for example Hansen [8], Neumaier [20], and Tikhonov and Arsenin [27]). More precisely, in many cases it is implicitly presumed that *the problem has a solution and reasonable bounds of its norm and its smoothness are known a priori*. Regularization can be viewed as a technique which prefers approximate solutions with moderate norm over very large norm solutions, and solutions to nearby and stable problems are favored. To apply regularization methods requires the availability of appropriate software and moreover the knowledge that the problem is ill-posed. The sole method known to the author which is appropriate for ill-posed semidefinite programming problems with zero duality gap is described in Gruber and Rendl [7]. The surprising results of Ordóñez and Freund [23] and Freund, Ordóñez and Toh [3] about the number of ill-posed problems in the NETLIB LP collection and in the SDPLIB demonstrate that in many applications it is not known whether problems are well-posed or ill-posed. Therefore, an appropriate a posteriori verification depending upon the degree of information available about the problem is desirable.

In the following it is important to emphasize that our approach of verification has nearly nothing in common with regularization methods, which want to find appropriate solutions. In contrast, we want to verify rigorously strong duality, and we want to compute rigorous a posteriori error bounds by using already computed approximations, which is a completely different task but can improve the reliability of the approximations significantly. It turns out that for ill-posed problems the stage of rigor depends on additional information. Important is that each solver may be used for computing approximations, irrespectively of its regularization properties.

A well-known source of ill-posed problems arises in combinatorial optimization by trying to model tight relaxations. Then frequently Slater's constraint qualification is not fulfilled. As an example of use (see also [11]) we consider Graph Partitioning Problems. In the same line other convex relaxations of combinatorial problems may be treated. Graph Partitioning problems are known to be NP-hard, and finding an optimal solution is difficult. Graph Partitioning has many applications among those is VLSI design. Because of the nonlinearity introduced by the positive semidefinite cone, semidefinite relaxations provide tighter bounds for many combinatorial problems than linear programming relaxations. Here, we investigate semidefinite relaxations for the special case of Equicut Problems, which have turned out to deliver tight lower bounds (see also Gruber and Rendl [7]). The general case of Graph Partitioning Problems can be treated similarly.

Given an edge-weighted graph G with an even number of vertices, the problem is to find a partitioning of the vertices into two sets of equal cardinality which minimizes the weight of the edges joining the two sets. The algebraic formulation is obtained by representing the partitioning as an integer vector $x \in \{-1, 1\}^n$ satisfying the parity condition $\sum_i x_i = 0$. Then the Equicut Problem is equivalent to

$$\min \sum_{i<j} a_{ij} \frac{1 - x_i x_j}{2} \quad \text{subject to} \quad x \in \{-1, 1\}^n, \ \sum_{i=1}^{n} x_i = 0,$$

where $A = (a_{ij})$ is the symmetric matrix of edge weights. This follows immediately, since $1 - x_i x_j = 0$ iff the vertices i and j are in the same set. The objective can be written as

$$\frac{1}{2} \sum_{i<j} a_{ij}(1 - x_i x_j) = \frac{1}{4} x^T (\mathrm{Diag}(Ae) - A)x = \frac{1}{4} x^T L x,$$

where e is the vector of ones, and $L := \mathrm{Diag}(Ae) - A$ is the *Laplace matrix* of G. Using $x^T L x = \mathrm{trace}(L(xx^T))$ and $X = xx^T$, it can be shown that this problem is equivalent to

$$f_p^* = \min \frac{1}{4} \langle L, X \rangle \quad \text{subject to} \quad \mathrm{diag}(X) = e, \ e^T X e = 0, \ X \succeq 0, \ \mathrm{rank}(X) = 1.$$

Since $X \succeq 0$ and $e^T X e = 0$ implies X to be singular, the problem is ill-posed, and for arbitrarily small perturbations of the right hand side the problem becomes infeasible. By definition, the Equicut Problem has a finite optimal value f_p^*, and a rigorous upper bound of f_p^* is simply obtained by evaluating the objective function for a given partitioning integer vector x. Hence, it is left over to compute a rigorous lower bound. At first, the nonlinear rank one constraint is left out yielding an ill-posed semidefinite relaxation, where the Slater condition does not hold. The related constraints $\mathrm{diag}(X) = e$ and $e^T X e = 0$ can be written as

$$\langle A_i, X \rangle = b_i, \ b_i = 1, \ A_i = E_i \text{ for } i = 1, \ldots, n, \text{ and } A_{n+1} = ee^T, \ b_{n+1} = 0.$$

where E_i is the $n \times n$ matrix with a one on the ith diagonal position and zeros otherwise. Hence, the dual semidefinite problem has the form

$$\max \sum_{i=1}^{n} y_i \quad \text{s.t.} \quad \text{Diag}(y_{1:n}) + y_{n+1}(ee^T) \preceq \frac{1}{4}L, \quad y \in \mathbf{R}^{n+1}.$$

The constraints $\text{diag}(X) = e,\ X \succeq 0$ imply that the principle two-dimensional submatrices must be positive semidefinite, and hence $-1 \leq X_{ij} \leq 1$ for the primal feasible solutions. Therefore, we have from the model the *additional information of existence of an optimal solution with finite upper bounds* $\overline{x} = \lambda_{\max}(X) = n$. The application of Theorem 2 yields:

Corollary 1 *Let* $\widetilde{y} \in \mathbf{R}^{n+1}$*, and assume that the matrix*

$$D = \frac{1}{4}L - \text{Diag}(\widetilde{y}_{1:n}) - \widetilde{y}_{n+1}(ee^T).$$

has at most l *negative eigenvalues, and let* $\underline{d} \leq \lambda_{\min}(D)$*. Then*

$$f_p^* \geq \sum_{i=1}^{n} \widetilde{y}_i + l \cdot n \cdot \underline{d}^- =: \underline{f}_p^*.$$

In Table 1 some numerical results for problems given by Gruber and Rendl [7] are displayed. The number of nodes is denoted by n. For this suite of ill-posed problems with up to 601 constraints and 180300 variables SDPT3 has computed approximations of the dual optimal value $\widetilde{f}_d^*$, which are close to the approximate primal optimal value $\widetilde{f}_p^*$ (see the column $\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$). The negative signs in this column show that weak duality is violated in four cases. The small quantities $\mu(\overline{f}_d^*, \underline{f}_p^*)$ show that the overestimation of the rigorous lower bound $\underline{f}_p^*$ can be neglected. SDPT3 gave $tc = 0$ (normal termination) for the first five ill-posed examples. Only in the last case $n = 600$ the warning $tc = -5$ (that means *: Progress too slow*) was returned. We have computed the lower bound $\underline{f}_p^*$ first by using Corollary 1 and then with the general routine `VSDPLOW`. Both bounds are equal but the time differs drastically. In Table 1 the times for computing the approximations with SDPT3, for computing $\underline{f}_p^*$ by using Corollary 1, and for computing $\underline{f}_p^*$ by using Theorem 2 are denoted by t, t_1, and t_2, respectively. It follows that the additional time t_1 for computing the rigorous bound $\underline{f}_p^*$ is small compared to the time t needed for the approximations. The reason for the different times t_1 and t_2 is that in the first case the special structure of the problems is taken into account, which yields an acceleration due to properties of our current implementation (see also the section about computing times).

Summarizing, VSDP facilitates cheap and rigorous lower bounds with modest overestimation for the optimal value of these ill-posed semidefinite relaxations. These bounds can be viewed

n	$\underline{f}_p^*$	$\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$	$\mu(\widetilde{f}_d^*, \underline{f}_p^*)$	t	t_1	t_2	tc
100	-3.58065e+003	-7.117e-008	3.843e-011	4.2	0.5	0.4	0
200	-1.04285e+004	-7.018e-008	9.621e-010	7.9	0.2	2.3	0
300	-1.90966e+004	-2.573e-008	8.918e-009	21.1	0.9	8.6	0
400	-3.01393e+004	-1.633e-008	3.008e-008	39.0	2.0	20.5	0
500	-4.22850e+004	1.431e-008	2.584e-008	67.5	3.7	40.0	0
600	-5.57876e+004	5.418e-009	1.829e-008	124.7	6.0	73.9	-5

Table 1: Results for Graph Partitioning

as an a posteriori verification of the approximations computed by SDPT3, and they can be used in a branch-and-bound scheme to obtain verified results. We mention that for these examples the approximations computed by SDPT3 are meaningful, but in other cases an appropriate regularization may be necessary.

Another very important class of ill-posed problems are discrete inverse problems, which arise in many applications like tomography, image restoration, identification in dynamical systems and many other areas. In these applications, there is a discrete version of a compact operator (i.e. a very ill-conditioned matrix A), a noisy right hand side b obtained from measurements, and the problem is to find a meaningful solution x which minimizes the residual $\|Ax - b\|^2$ and satisfies additionally constraints like non-negativity of the variables. Since the matrix A is very ill-conditioned, the computation of an accurate solution of the equation $Ax = b$ in general yields a nonsensical solution of large norm. This solution causes for small perturbations of A large variations Ax, and hence should be avoided. A common technique is to use Tikhonov regularization or a similar variant. Regularization can be viewed as adding norm constraints for the solution and its smoothness, and thus yields a convex quadratically constrained optimization problem which can be represented as a semidefinite program, and verified a posteriori with VSDP.

A further source of ill-posed problems that we want to mention here are originally well-posed problems which are transformed to ill-posed ones due to simple details of modelling: For example because of redundant constraints, identically zero variables, or converting free variables x to the difference of nonnegative variables $x = x_+ - x_-$. The latter transformation is never a good idea although often mentioned in many textbooks. Free variables are frequently used to eliminate a row, but more wicked is the fact that this transformation causes an unbounded set of optimal solutions. Whatever the value of an optimal variable x^*, the transformed variables x_+^* and x_-^* have infinitely many values with the same difference equal to x^*. Using interior-point methods causes sometimes divergence because of divergence of the values x_+^* and x_-^*, while preserving a constant difference. However, this transformation has the property that the transformed problem has an optimal solution with finite upper bounds

of its variables equal to the upper bounds of the optimal variables for the original solution. These finite bounds can be used in Theorems 2 and 3 for computing rigorous bounds of the optimal value. Similar arguments can be applied to other transformations.

But even if the modelling, the transformations or established bounds for the magnitude of the optimal solution are not available, the numerical reliability of the computed approximations can be improved. Computer codes for optimization algorithms terminate if the optimality conditions are approximately fulfilled, whereas the optimality conditions are evaluated by using floating point arithmetic. For semidefinite programming the optimality conditions require the computation of residuals for linear equations and of some minimal eigenvalues. At the best the computed approximations are backward stable, i.e they are exact solutions of slightly perturbed problems. However, this is questionable if the problem is ill-conditioned or the input data cannot vary independently. Moreover, especially because of the complementarity condition the minimal eigenvalues are close to zero, causing numerical difficulties for several eigenvalue solvers. In our experiments especially the MATLAB routine `EIGS`, which is a Krylov subspace method, has produced erroneous approximations in many cases.

The routines `VSDPLOW` and `VSDPUP` may be used for improving the reliability as follows: It is evident that trusting only the magnitude of a computed solution is much weaker than to accept the computed digits. This is in the sense of Kahan [13] who writes:

> "Nobody can know even roughly how wrong a computation is without knowing at least roughly what would have been right."

Therefore, similar to the case of regularization methods for ill-posed problems, we assume the existence of an optimal solution and accept bounds of the form

$$\begin{array}{l} \overline{x}_j = \mu \cdot \lambda_{\max}(\widetilde{X}_j) \text{ for } j = 1, \ldots, n, \\ \text{and } \ \overline{y}_i = \mu \cdot |\widetilde{y}_i| \text{ for } i = 1, \ldots, m, \end{array}$$

where $(\widetilde{X}_j), \widetilde{y}$ are the computed approximations, and μ is a positive factor of order one. A more conservative estimation would be factors like $\mu = 10$, an optimistic one (trusting at least one decimal digit) is $\mu = 0.1$. Since this boundedness assumption is not verified, the results are not fully rigorous. Nevertheless, this stage of rigor is completely with rounding error control, and we may speak of a *rounding error controlled weak verification*.

In the following we illustrate the use of `VSDPLOW` and `VSDPUP` for the ill-posed problem `gpp124-1` of the SDPLIB. We call this problem:

```
>> sdplibfiles = dir('***\sdplib_vsdp/*.dat-s');
>> i=19;
```

```
>> [blk,A,C,b] = sdpa_to_vsdp...
      (['***\sdplib_vsdp/' sdplibfiles(i).name]);
       Problem = sdplibfiles(i).name,

Problem =
gpp124-1.dat-s
```

The size of the problem can be obtained with

```
>> blk, size(b)
blk =
    's'    [124]
ans =
    125     1
```

Hence, this is a block diagonal problem consisting of one symmetric block with dimension 124 (yielding 7750 variables), and it has 125 constraints. At first, we solve this problem without bounds for the solution.

```
>> tic; [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b); t = toc;
>> tic; [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt); tfL = toc;
>> tic; [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt); tfU = toc;
>> objt, fL, fU, t, tfL, tfU
objt =
     7.343071180003589e+000    7.343073711914145e+000
fL =
   7.343073667562324e+000
fU =
   Inf
t =
  3.483999999999998e+000
tfL =
    4.827999999999999e+000
tfU =
    8.015999999999998e+000
```

We obtain a finite rigorous lower bound `fL`, and looking at `Y` and `dl` it can be seen that the dual problem contains strictly interior points. Hence, strong duality is fulfilled. The upper bound `fU = Inf` reflects that the problem is ill-posed and the distance to primal infeasibility is zero. If we accept bounds for the optimal solution with the conservative factor $\mu = 10$

```
>> mu = 10;
>> xu = mu * max(eig(Xt{1}));
>> yu = mu * abs(yt);
```

and solve the problem rigorously using the previous approximations, then:

```
>> tic; [fL, Y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt,xu); tfL = toc;
>> tic; [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt,yu); tfU = toc;
>> objt, fL, fU, t, tfL, tfUfL =
objt =
     7.343071180003589e+000    7.343073711914145e+000
fL =
   7.343073234661038e+000
fU =
   7.343096928246631e+000
t =
  3.483999999999998e+000
tfL =
    5.160000000000000e-001
tfU =
 9.529999999999999e-001
```

The lower and the upper bound of the optimal value are close together. Therefore, if one accepts the existence of an optimal solution within the above bounds `xu` and `yu`, then an accuracy of about 6 decimal digits is guaranteed for this ill-posed problem. Because here no iterations and verified solutions of linear systems are necessary, the required computational times are small compared with the previous times.

The other numerical results for the SDPLIB can be found in Tables 6 and 7. There, in all cases we have chosen the bounds with the conservative factor $\mu = 10$. The major characteristics of these results are as follows: The median of the time ratio for computing the rigorous lower(upper) bound and the approximation is $0.078, (0.21)$, respectively. The median of the guaranteed accuracy for these problems is 2.5210^{-6}, and the medium of the approximate accuracy is -4.3810^{-8}. It is noteworthy to observe that *the accuracy is independent of the condition number of the problem.*

We mention that accepting the order of magnitude of an approximate large norm solution produces large bounds for the variables, which is reflected by more distant values $\underline{f}_p^*$ and $\overline{f}_d^*$. Therefore, a semidefinite solver producing regularized small norm solutions yield better rigorous bounds for the optimal value. Notice that semidefinite solvers usually terminate if

the optimality conditions are satisfied approximately. But these conditions do not distinguish between regularized and non-regularized approximations.

It is well-known that the dual variables describe the sensitivity and stability of the problem, in the sense that large dual variables indicate instability of the primal optimal solution if the input data are slightly perturbed. This means that small nonnegative bounds $\overline{y}_i$ for the dual variables characterize stable problems. On the other hand there may also be hard constraints, i.e. constraints where even small violations are not allowed from the modelling point of view. Such constraints can be modelled by setting the bound $\overline{y}_i = \infty$. Then in Theorem 3, property (ii) the condition $r_i = 0$ requires that the i-th equation must be satisfied, which causes to solve this equation rigorously. Unfortunately, in the current version of VSDP either $\overline{y}$ is handled as a finite vector or an infinite vector. The case where some components are finite and some are infinite will be implemented in the next version.

Last, we look at verified error bounds for large-scale semidefinite programs in electronic structure calculations. The problem of computing a close lower bound of the ground state energy of atomic and molecular systems can be described as a large semidefinite program by using a variational approach in which the two-body reduced density matrix is the unknown quantity (see for example Fukuda et al. [4] and Nakata et al. [18]). In [4] the ground stage energy of a large variety of molecules is determined, and the input data of the semidefinite programs can be downloaded in SDPA format from `http://www.is.titech.ac.jp/ mituhiro/`. In [4] it is mentioned that the SDP problems must be solved to high accuracy, typically 7 digits. Morever, they used SDPARA (a parallel version of SDPA), and since this code cannot handle equality constraints in the dual SDP they replaced the equations by perturbed inequalities. Hence, these problems are ill-conditioned.

As an illustrative example we take the molecule NH- using $r = 12$ basis states. This yields a semidefinite program with $m = 948$, and the size of the block matrices is 6, 6, 6, 6, 15, 15, 36, 15, 15, 36, 72, 36, 36, 20, 90, 90, 20, 306, 306, 90, 90. If we solve this problem with SDPA then we obtain

```
% Generating VSDP input
>> [blk,A,C,b] = sdpa_to_vsdp('NH_3SIGMA-_STO-6GN8r12g1T2.dat-s');
% Approximately solving with SDPA
>> [objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b);
% Verified lower bound
>> [fL, y, dl] = vsdplow(blk,A,C,b,Xt,yt,Zt);
% Verified upper bound
>> [fU, X, lb] = vsdpup(blk,A,C,b,Xt,yt,Zt);
>> objt, fL, fU,
objt =
```

```
      5.843807529542937e+001      5.833070686520880e+001
fL =
      5.833075312544655e+001
fU =
      5.843803999358170e+001
```

Hence, the approximate as well as the guaranteed accuracy is about 3 decimal digits, far away from the required accuracy of 7 digits. The times for computing the approximations, the rigorous lower bound and the rigorous upper bound are 431, 1344 and 49 seconds, respectively.

An insufficient accuracy for some of these problems was already observed in [18]. They write that the numerical accuracy becomes worse for some molecules. Comparing their results with the full-CI method they noticed that some computed SDP energies have values **higher** than the full-CI ones, though from the theory it follows that these values must be **lower** that the full-CI ones. We mention that our verified lower bound always gives values lower than the ground state energy. If we use for the above problem SDPT3 then we obtain the rigorous lower and upper bound `fL = 5.839100045713466e+001` and `fU = 5.839111213365607e+001`, yielding 6 correct decimal digits.

VSDP could not verify the extremely large-scale molecule problems ($m > 7000$ and more than 2.5 million variables), since SDPA as well as SDPT3 did not compute an approximation after 3 days, where then the programs were stopped. Thus the largest verified molecule problems are those with $r = 16$ basis states. For example Ammonia (NH3) yields $m = 2964$ with size of the block matrices 8, 8, 8, 8, 28, 28, 64, 28, 28, 64, 128, 64, 64, 56, 224, 224, 56, 736, 736, 224, 224. Using SDPA we obtain the rigorous lower and upper bound `fL = 6.792487010727453e+001` and `fU = 6.792487859935491e+001`, yielding 7 correct decimal digits.

7 Interval Arithmetic and INTLAB

In VSDP the input data can also be intervals. In the following, only some elementary definitions about interval arithmetic are required. For interested users we recommend to look at the two INTLAB [26] demonstrations `DEMOINTVAL` and `DEMOINTLAB`. There, input, output, operations, and functions for interval quantities are described in a very compact and clear form. For mathematical details see also Rump [25]. There are a number of textbooks on interval arithmetic and self-validating methods which can be highly recommended to readers. These include Alefeld and Herzberger [1], Kearfott [14], Moore [17], and Neumaier [19], [21].

Let $\mathbf{V}$ be one of the spaces $\mathbf{R}$ (real numbers), $\mathbf{R}^n$ (real vectors), $\mathbf{R}^{m\times n}$(real matrices) with partial ordering $\leq$. If $\underline{v}, \overline{v} \in \mathbf{V}$, then the box

$$\mathbf{v} := [\underline{v}, \overline{v}] := \{v \in \mathbf{IV} : \ \underline{v} \leq v \leq \overline{v}\} \tag{12}$$

is called an *interval quantity* in $\mathbf{V}$ with *lower bound* $\underline{v}$ and *upper bound* $\overline{v}$. In particular, $\mathbf{IR}$, $\mathbf{IR}^n$ and $\mathbf{IR}^{m\times n}$ 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 $\mathbf{A} \circ \mathbf{B}$ is defined as the interval hull of all possible real results, that is

$$\mathbf{A} \circ \mathbf{B} := \bigcap\{\mathbf{C} \in \mathbf{IV} : \ A \circ B \in \mathbf{C} \quad \text{for all} \quad A \in \mathbf{A}, B \in \mathbf{B}\}. \tag{13}$$

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

$$\mathbf{A} + \mathbf{B} = [\underline{A} + \underline{B}, \overline{A} + \overline{B}]. \tag{14}$$

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

$$\begin{aligned} \text{mid}(\mathbf{A}) &:= (\underline{A} + \overline{A})/2 \quad \text{as the } \textit{midpoint}, & (15)\\ \text{rad}(\mathbf{A}) &:= (\overline{A} - \underline{A})/2 \quad \text{as the } \textit{radius}. & (16)\\ & & (17) \end{aligned}$$

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

INTLAB allows several possibilities to generate intervals. Firstly, by specification of midpoint and radius

```
>> x = midrad( [ 3 2 ], [ 2 5 ])

intval x =
[    1.0000,    5.0000] [   -3.0000,    7.0000]
```

or by the representation with lower and upper bounds

```
>> x = infsup( [ 1 -3 ] , [ 5 7 ] )
```

```
intval x =
[    1.0000,    5.0000] [   -3.0000,    7.0000]
```

Lower and upper bounds, midpoint and radius can be obtained as follows:

```
>> inf_(x)
ans =
     1    -3
>> sup(x)
ans =
     5     7
>> rad(x)
ans =
     2     5
>> mid(x)
ans =
     3     2
```

An operation uses interval arithmetic if one of the operands is of type `intval`, for example

```
>> x/2
intval ans =
[    0.5000,    2.5000] [   -1.5000,    3.5000]
```

If the input data of a semidefinite programming problem are intervals then the rigorous bounds computed by VSDP hold valid for all problems with real input data inside the intervals. Following, we solve the Truss Topology Design Problem `arch4` with interval input data:

```
% Generating the problem
[blk,A,C,b] = sdpa_to_vsdp...
   (['*** \sdplib_vsdp/' sdplibfiles(3).name]);
% Dimension
m = length(b);
n = length(C);
% Generating interval input data with relative radius r
r=1e-8;
for j = 1 : n
   CI{j} = midrad(C{j},r*abs(C{j}));
```

```
   for i = 1 : m
      AI{i,j} = midrad(A{i,j},r*abs(A{i,j}));
   end
end
bI = midrad(b,r*abs(b));

% Call of SDPT3 for the midpoint problem:
[objt,Xt,yt,Zt,info] = mysdps(blk,A,C,b);

% Lower and upper bounds for the interval problem:

[fL, Y, dl] = vsdplow(blk,AI,CI,bI,Xt,yt,Zt);
[fU, X, lb] = vsdpup(blk,AI,CI,bI,Xt,yt,Zt);
[fL; fU]
ans =
   -9.726469030815481e-001
   -9.725514124620581e-001
```

For `arch4` the optimal value for all problems with real input data inside the intervals is between `fL` and `fU`. Hence, a relative perturbation radius of 10^{-8} yields a relative radius of $4.9 \cdot 10^{-5}$ for the set of optimal values. Looking at the other output data it follows that `Y` is a strictly dual feasible solution for all real problems, and the interval quantity `X` contains for each real problem a strictly primal feasible solution. Especially, strong duality holds valid for all real problems.

8 Computing Times

The aim in designing VSDP was to write the programs in a convenient form, appropriate also for students. 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. This causes a loss of efficiency.

The difference in the running time can be very dramatic if additionally intervals are used. The reason is that in INTLAB the new interval data types and the interval operations are added to MATLAB by classes and objects. For the purpose of illustration let us generate randomly $n = 100$ blocks of dimension $s = 20$, and let $m = 100$:

```
n = 100; m = n; s = 20;
```

```
y = rand(m,1);
yint = intval(y);
for j = 1 : n
   for i = 1 : m
       A{i,j} = rand(s);
   end
end
```

Computing the defect matrices (cf. (4)) by using the usual rounding to nearest yields:

```
tic;
for j = 1 : n
   D{j} = zeros(s);
   for i = 1 : m
      D{j} = D{j} + y(i) * A{i,j};
   end
end;
toc;
Elapsed time is 0.235000 seconds.
```

INTLAB makes the monotonic rounding modes available. Computing an enclosure of the defect matrices by using monotonic rounding yields

```
tic
for j = 1 : n
   setround(-1);
   Dlow{j} = zeros(s);
   for i = 1 : m
      Dlow{j} = Dlow{j} + y(i) * A{i,j};
   end
   setround(1);
   Dup{j} = zeros(s);
   for i = 1 : m
      Dup{j} = Dup{j} + y(i) * A{i,j};
   end
   D{j} = infsup(Dlow{j},Dup{j});
end;
setround(0);
toc;
Elapsed time is 0.656000 seconds.
```

If interval quantities are involved, then we obtain

```
tic;
for j = 1 : n
   D{j} = zeros(s);
   for i = 1 : m
      D{j} = D{j} - yint(i) * A{i,j};
    end
 end
 toc;
 Elapsed time is 9.953000 seconds.
```

which is about fifteen times slower than using monotonic rounding.

Using the tool PROFILE it can be seen that for the SDPLIB problem `gpp250-3` about 90% of the computing time for `VSDLOW` is required for computing the defect matrices. An efficient C implementation should be able to save a similar amount of time.

9 Conclusion

Although VSDP is not the most efficient implementation of rigorous error bounds, the numerical results for well-posed and ill-posed problems demonstrate that VSDP provides safety to the user in a reasonable amount of computational effort. This is especially important if the user cannot judge the computed approximations. For example in branch-bound-and-cut procedures semidefinite subproblems are likely to be ill-conditioned or ill-posed. Erroneous approximations may lead to incorrect decisions and the loss of global optimal solutions. This can be avoided if rigorous error bounds are used.

A Appendix

Table 2: Rigorous lower bounds for the SDPLIB

Problem	$\overline{f}_d^*$	$\underline{f}_d^*$	$\mu(\overline{f}_d^*, \underline{f}_d^*)$	$\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$
arch0	-5.6651e-001	-5.6652e-001	5.12e-006	6.53e-007
arch2	-6.7151e-001	-6.7152e-001	4.46e-006	-2.70e-007
arch4	-9.7263e-001	-9.7263e-001	1.07e-008	8.19e-009
arch8	-7.0570e+000	-7.0570e+000	3.56e-007	1.85e-008
control1	-1.7776e+001	-1.7785e+001	4.91e-004	7.28e-008
control2	-8.2958e+000	-8.3000e+000	5.05e-004	-1.56e-007
control3	-1.3605e+001	-1.3633e+001	2.08e-003	-3.96e-007
control4	-1.9760e+001	-1.9794e+001	1.75e-003	-7.32e-007
control5	-1.6698e+001	-1.6884e+001	1.10e-002	-4.80e-006
control6	-3.6621e+001	-3.7304e+001	1.85e-002	-8.03e-007
control7	-2.0326e+001	-2.0625e+001	1.46e-002	-9.99e-007
control8	-1.9927e+001	-2.0286e+001	1.79e-002	-1.26e-006
control9	-1.3993e+001	-1.4675e+001	4.76e-002	-3.03e-006
control10	-3.5891e+001	-3.8533e+001	7.10e-002	-2.78e-006
control11	-2.6994e+001	-3.1959e+001	1.68e-001	-6.48e-006
equalG11	-6.2766e+002	-6.2916e+002	2.38e-003	2.92e-009
equalG51	-4.0056e+003	-4.0056e+003	1.05e-005	5.30e-010
gpp100	Inf	4.4944e+001	NaN	-7.00e-008
gpp124-1	Inf	7.3431e+000	NaN	-3.45e-007
gpp124-2	Inf	4.6862e+001	NaN	-7.70e-008
gpp124-3	Inf	1.5301e+002	NaN	-4.51e-008
gpp124-4	Inf	4.1899e+002	NaN	-7.31e-008
gpp250-1	Inf	1.5445e+001	NaN	-7.12e-008
gpp250-2	Inf	8.1869e+001	NaN	-4.26e-008
gpp250-3	Inf	3.0354e+002	NaN	-3.06e-008
gpp250-4	Inf	7.4733e+002	NaN	-1.82e-008
gpp500-1	Inf	2.5321e+001	NaN	-1.09e-008
gpp500-2	Inf	1.5606e+002	NaN	-2.91e-008
gpp500-3	Inf	5.1302e+002	NaN	-1.14e-009
gpp500-4	Inf	1.5670e+003	NaN	4.66e-009
hinf1	Inf	-2.0328e+000	NaN	-1.01e-004
hinf2	Inf	-1.0967e+001	NaN	-3.37e-005
hinf3	Inf	-5.6955e+001	NaN	-2.26e-004
hinf4	Inf	-2.7477e+002	NaN	-1.36e-005
hinf5	Inf	-3.6241e+002	NaN	-5.51e-004
hinf6	Inf	-4.4911e+002	NaN	-7.04e-004
hinf7	Inf	-3.9083e+002	NaN	4.40e-005
hinf8	Inf	-1.1618e+002	NaN	-3.62e-004
hinf9	-7.5500e+001	-2.3739e+002	1.03e+000	-3.57e-002
hinf10	Inf	-1.0886e+002	NaN	-1.39e-003
hinf11	Inf	-6.5938e+001	NaN	-1.15e-003
hinf12	Inf	-7.5031e-001	NaN	-6.66e-001
hinf13	Inf	-4.5987e+001	NaN	-3.55e-002
hinf14	Inf	-1.3000e+001	NaN	2.22e-004
hinf15	Inf	-2.6085e+001	NaN	-8.33e-002

Table 3: Rigorous lower bounds for the SDPLIB

Problem	$\overline{f}_d^*$	$\underline{f}_d^*$	$\mu(\overline{f}_d^*, \underline{f}_d^*)$	$\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$
infd1	Inf	-Inf	NaN	-2.00e+000
infd2	Inf	-Inf	NaN	-2.00e+000
infp1	-3.9776e+009	-Inf	NaN	-2.00e+000
infp2	-3.5022e+001	-Inf	NaN	-2.00e+000
maxG11	-6.2916e+002	-6.2916e+002	1.03e-008	1.01e-008
maxG32	-1.5676e+003	-1.5676e+003	1.66e-007	-1.19e-008
maxG51	-4.0063e+003	-4.0063e+003	9.68e-009	1.22e-008
mcp100	-2.2616e+002	-2.2616e+002	1.39e-008	5.76e-009
mcp124-1	-1.4199e+002	-1.4199e+002	7.93e-009	1.61e-008
mcp124-2	-2.6988e+002	-2.6988e+002	3.69e-008	1.65e-008
mcp124-3	-4.6775e+002	-4.6775e+002	1.43e-008	-2.83e-009
mcp124-4	-8.6441e+002	-8.6441e+002	8.21e-009	-8.22e-010
mcp250-1	-3.1726e+002	-3.1726e+002	8.53e-009	4.77e-010
mcp250-2	-5.3193e+002	-5.3193e+002	7.15e-009	2.01e-009
mcp250-3	-9.8117e+002	-9.8117e+002	4.62e-009	7.35e-009
mcp250-4	-1.6820e+003	-1.6820e+003	6.78e-009	-4.63e-009
mcp500-1	-5.9815e+002	-5.9815e+002	1.03e-008	5.94e-009
mcp500-2	-1.0701e+003	-1.0701e+003	1.90e-008	3.19e-009
mcp500-3	-1.8480e+003	-1.8480e+003	1.58e-008	2.05e-008
mcp500-4	-3.5667e+003	-3.5667e+003	2.18e-008	-1.49e-008
qap10	Inf	1.0925e+003	NaN	-1.16e-004
qap5	Inf	4.3600e+002	NaN	1.18e-009
qap6	Inf	3.8140e+002	NaN	-9.41e-005
qap7	Inf	4.2479e+002	NaN	-7.25e-005
qap8	Inf	7.5687e+002	NaN	-1.22e-004
qap9	Inf	1.4099e+003	NaN	-4.54e-005
qpG11	-2.4487e+003	-2.4487e+003	3.85e-010	3.85e-010
qpG51	-1.1818e+004	-1.1818e+004	5.13e-009	5.13e-009
ss30	-2.0239e+001	-2.0240e+001	3.98e-005	2.85e-008
theta1	-2.3000e+001	-2.3000e+001	3.55e-008	3.45e-009
theta2	-3.2879e+001	-3.2879e+001	7.02e-007	-1.45e-008
theta3	-4.2167e+001	-4.2167e+001	2.41e-006	-7.93e-010
theta4	-5.0321e+001	-5.0321e+001	1.13e-006	-1.08e-008
theta5	-5.7232e+001	-5.7232e+001	3.80e-006	-5.00e-008
thetaG11	-4.0000e+002	-4.0000e+002	3.40e-008	3.41e-008
truss1	9.0000e+000	9.0000e+000	2.24e-007	1.12e-009
truss2	1.2338e+002	1.2338e+002	9.78e-007	-1.56e-007
truss3	9.1100e+000	9.1100e+000	8.01e-008	2.96e-009
truss4	9.0100e+000	9.0100e+000	1.61e-007	2.34e-009
truss5	1.3264e+002	1.3264e+002	3.61e-006	-5.10e-010
truss6	9.0160e+002	9.0100e+002	6.69e-004	-2.38e-005
truss7	9.0021e+002	9.0000e+002	2.33e-004	-8.89e-006
truss8	1.3313e+002	1.3311e+002	8.39e-005	-5.36e-006

Table 4: Rigorous lower bounds for the SDPLIB

Problem	*tc*	*t*	*tfL*	*tfU*
arch0	0	11.28	0.34	24.14
arch2	0	11.02	0.28	13.13
arch4	0	10.11	0.30	13.78
arch8	0	10.67	10.05	13.50
control1	0	0.84	0.02	0.84
control2	0	1.86	0.03	2.09
control3	0	4.09	0.08	4.94
control4	0	7.41	0.16	10.45
control5	0	15.86	0.23	22.38
control6	0	30.64	0.41	46.17
control7	0	55.05	0.73	95.23
control8	0	93.22	1.06	308.75
control9	0	148.50	211.31	273.89
control10	0	238.11	352.33	475.95
control11	0	347.75	543.20	746.98
equalG11	0	197.34	220.73	608.09
equalG51	0	431.97	492.52	723.81
gpp100	0	2.11	2.59	2.99
gpp124-1	0	3.19	4.56	7.66
gpp124-2	0	3.22	4.33	9.83
gpp124-3	0	3.03	4.14	4.69
gpp124-4	0	3.28	4.41	8.31
gpp250-1	0	10.16	34.97	29.66
gpp250-2	0	10.03	19.78	29.25
gpp250-3	0	9.73	19.44	20.44
gpp250-4	0	9.27	19.55	34.08
gpp500-1	0	62.91	254.47	199.11
gpp500-2	0	54.27	143.75	132.39
gpp500-3	0	56.88	143.50	333.91
gpp500-4	0	50.39	136.84	192.42
hinf1	0	0.94	0.06	7.23
hinf2	0	0.72	0.06	10.14
hinf3	0	0.88	0.06	9.26
hinf4	0	0.86	0.06	3.02
hinf5	-4	0.84	0.06	5.56
hinf6	0	1.02	0.05	6.77
hinf7	-4	0.77	0.05	7.17
hinf8	-4	0.86	0.05	4.30
hinf9	2	0.94	0.05	4.97
hinf10	0	1.44	0.03	1.39
hinf11	0	1.44	0.06	4.70
hinf12	0	2.38	2.42	4.41
hinf13	0	1.39	0.08	3.08
hinf14	-4	1.11	0.06	5.50
hinf15	0	1.50	0.09	4.78

Table 5: Rigorous lower bounds for the SDPLIB

Problem	*tc*	*t*	*tfL*	*tfU*
infd1	1	0.27	4.78	0.27
infd2	1	0.27	4.70	0.25
infp1	2	0.55	0.50	0.72
infp2	2	0.61	0.47	0.92
maxG11	0	62.31	12.05	278.94
maxG32	0	750.95	162.66	3452.05
maxG51	0	166.48	24.84	895.02
mcp100	0	1.33	0.09	3.39
mcp124-1	0	1.84	0.11	0.55
mcp124-2	0	2.08	0.13	5.27
mcp124-3	0	2.16	0.14	2.95
mcp124-4	0	2.28	0.17	5.77
mcp250-1	0	4.75	0.41	9.86
mcp250-2	0	4.97	0.47	10.11
mcp250-3	0	5.88	0.58	4.33
mcp250-4	0	5.89	0.66	11.20
mcp500-1	0	20.56	2.86	60.66
mcp500-2	0	25.99	3.44	64.83
mcp500-3	0	29.03	3.56	68.23
mcp500-4	0	28.98	4.34	69.09
qap10	0	10.20	2.31	43.97
qap5	0	0.63	0.06	3.17
qap6	0	1.06	0.13	4.05
qap7	0	1.69	0.23	5.33
qap8	0	2.89	0.52	9.23
qap9	0	5.86	1.09	17.81
qpG11	0	329.76	54.27	928.08
qpG51	0	675.34	104.74	1252.06
ss30	0	25.39	0.75	31.16
theta1	0	0.70	0.09	0.89
theta2	0	2.80	1.19	6.05
theta3	0	12.02	6.81	50.50
theta4	0	38.86	21.56	196.55
theta5	0	123.86	54.92	408.39
thetaG11	-7	212.23	12.03	455.83
truss1	0	0.72	0.06	0.84
truss2	0	4.41	0.27	9.75
truss3	0	0.98	0.08	1.13
truss4	0	0.75	0.06	0.88
truss5	0	11.03	0.61	13.28
truss6	0	25.77	2.05	31.09
truss7	0	22.77	1.30	49.91
truss8	0	12.97	1.48	19.86

Table 6: Rigorous lower bounds for the SDPLIB with boundedness qualifications

Problem	$\overline{f}_d^*$	$\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$	$\mu(\overline{f}_d^*, \underline{f}_d^*)$	t	tfL	tfU	tc
arch0	-5.66517e-001	6.52664e-007	3.89755e-006	11.14	0.48	2.22	0
arch2	-6.71516e-001	-2.69574e-007	3.88491e-006	10.44	0.36	2.31	0
arch4	-9.72627e-001	8.19066e-009	1.45549e-008	10.55	0.30	2.31	0
arch8	-7.05698e+000	1.84683e-008	1.67410e-007	10.05	0.31	2.11	0
control1	-1.77846e+001	7.27895e-008	2.57191e-005	1.08	0.03	0.08	0
control2	-8.30000e+000	-1.56463e-007	6.78045e-005	2.69	0.08	0.09	0
control3	-1.36333e+001	-3.95628e-007	1.21512e-005	5.33	0.11	0.23	0
control4	-1.97942e+001	-7.31969e-007	2.28246e-005	10.81	0.14	0.61	0
control5	-1.68837e+001	-4.80154e-006	7.46780e-005	18.14	0.27	1.44	0
control6	-3.73045e+001	-8.02555e-007	7.67001e-005	36.69	0.47	4.14	0
control7	-2.06251e+001	-9.98899e-007	3.50385e-005	63.03	0.81	7.44	0
control8	-2.02864e+001	-1.25749e-006	3.86513e-005	84.89	1.42	14.53	0
control9	-1.46755e+001	-3.02920e-006	1.67502e-004	140.91	1.84	30.06	0
control10	-3.85332e+001	-2.77620e-006	2.02277e-004	224.97	2.97	53.06	0
control11	-3.19589e+001	-6.47652e-006	1.94653e-004	334.70	3.84	87.22	0
equalG11	-6.29155e+002	2.91690e-009	4.85907e-007	197.94	13.41	134.33	0
equalG51	-4.00560e+003	5.30399e-010	8.59726e-007	432.20	26.92	263.13	0
gpp100	4.49435e+001	-7.00054e-008	6.64166e-007	2.22	0.33	0.58	0
gpp124-1	7.34307e+000	-3.44803e-007	3.22665e-006	3.41	0.61	1.14	0
gpp124-2	4.68623e+001	-7.70226e-008	7.25990e-007	3.42	0.63	1.11	0
gpp124-3	1.53014e+002	-4.50687e-008	4.17787e-007	3.17	0.55	1.13	0
gpp124-4	4.18988e+002	-7.30945e-008	7.74977e-007	3.42	0.58	1.08	0
gpp250-1	1.54449e+001	-7.12482e-008	8.99341e-007	10.00	4.66	8.98	0
gpp250-2	8.18690e+001	-4.26302e-008	5.56033e-007	9.98	4.78	9.03	0
gpp250-3	3.03539e+002	-3.06479e-008	4.27528e-007	9.45	4.83	9.03	0
gpp250-4	7.47328e+002	-1.82234e-008	3.03032e-007	9.34	4.72	9.14	0
gpp500-1	2.53205e+001	-1.09285e-008	4.28074e-006	63.19	42.47	74.89	0
gpp500-2	1.56060e+002	-2.91009e-008	5.17852e-007	54.94	42.63	74.83	0
gpp500-3	5.13018e+002	-1.13886e-009	1.06758e-007	57.36	42.77	74.83	0
gpp500-4	1.56702e+003	4.65840e-009	6.36747e-008	51.17	42.73	74.86	0
hinf1	-2.03301e+000	-1.00564e-004	9.34650e-004	1.20	0.05	0.06	0
hinf2	-1.09678e+001	-3.36629e-005	1.03539e-003	0.98	0.06	0.05	0
hinf3	-5.69677e+001	-2.26145e-004	2.41706e-003	1.11	0.08	0.05	0
hinf4	-2.74771e+002	-1.35848e-005	1.39322e-004	1.11	0.08	0.05	0
hinf5	-3.62613e+002	-5.50585e-004	6.25571e-002	1.11	0.06	0.05	-4
hinf6	-4.49426e+002	-7.03519e-004	6.37027e-003	1.27	0.08	0.05	0
hinf7	-3.90810e+002	4.39663e-005	8.30217e-004	1.02	0.06	0.05	-4
hinf8	-1.16222e+002	-3.61756e-004	5.08181e-003	1.13	0.06	0.05	-4
hinf9	-2.46011e+002	-3.56709e-002	2.00000e+000	1.22	0.06	0.06	2
hinf10	-1.09014e+002	-1.38839e-003	1.32661e-002	1.77	0.05	0.05	0
hinf11	-6.60146e+001	-1.15421e-003	1.18289e-002	1.78	0.08	0.06	0
hinf12	-1.56488e+000	-6.65981e-001	2.00000e+000	2.88	0.06	0.08	0
hinf13	-4.76508e+001	-3.55343e-002	4.74862e-001	1.75	0.06	0.09	0
hinf14	-1.29971e+001	2.21506e-004	2.66706e-002	1.31	0.08	0.11	-4
hinf15	-2.83516e+001	-8.32685e-002	1.77700e+000	1.80	0.09	0.14	0

Table 7: Rigorous lower bounds for the SDPLIB with boundedness qualifications

Problem	$\overline{f}_d^*$	$\mu(\widetilde{f}_p^*, \widetilde{f}_d^*)$	$\mu(\overline{f}_d^*, \underline{f}_d^*)$	t	tfL	tfU	tc
infd1	-5.19432e+000	-2.00000e+000	2.00000e+000	0.33	0.06	0.05	1
infd2	5.42061e+000	-2.00000e+000	2.00000e+000	0.36	0.05	0.05	1
infp1	-5.94121e+009	-2.00000e+000	2.00000e+000	0.70	0.05	0.05	2
infp2	-2.57524e+009	-2.00000e+000	2.00000e+000	0.77	0.06	0.05	2
maxG11	-6.29165e+002	1.00858e-008	3.76123e-008	62.16	12.42	188.06	0
maxG32	-1.56764e+003	-1.19489e-008	2.56489e-007	748.03	161.17	2329.02	0
maxG51	-4.00626e+003	1.21822e-008	1.30882e-007	166.20	24.25	311.64	0
mcp100	-2.26157e+002	5.76094e-009	6.01171e-008	1.44	0.11	0.27	0
mcp124-1	-1.41990e+002	1.61076e-008	1.72602e-007	1.94	0.09	0.45	0
mcp124-2	-2.69880e+002	1.64754e-008	1.80790e-007	2.22	0.11	0.47	0
mcp124-3	-4.67750e+002	-2.83436e-009	8.83177e-008	2.30	0.16	0.52	0
mcp124-4	-8.64412e+002	-8.21767e-010	4.77618e-008	2.45	0.19	0.50	0
mcp250-1	-3.17264e+002	4.76695e-010	3.23009e-008	4.66	0.42	3.63	0
mcp250-2	-5.31930e+002	2.01287e-009	3.98574e-008	4.95	0.47	3.61	0
mcp250-3	-9.81173e+002	7.34522e-009	5.46713e-008	6.03	0.53	3.47	0
mcp250-4	-1.68196e+003	-4.62904e-009	5.97408e-008	5.84	0.64	3.44	0
mcp500-1	-5.98149e+002	5.93604e-009	7.21147e-008	20.88	2.86	29.73	0
mcp500-2	-1.07006e+003	3.19174e-009	1.24832e-007	26.20	3.30	29.95	0
mcp500-3	-1.84797e+003	2.05128e-008	2.62117e-007	29.33	3.53	27.80	0
mcp500-4	-3.56674e+003	-1.49140e-008	1.60369e-007	29.27	4.16	30.19	0
qap10	1.09236e+003	-1.16219e-004	1.05064e-003	10.45	2.56	5.69	0
qap5	4.36000e+002	1.17714e-009	2.66595e-007	0.75	0.08	0.11	0
qap6	3.81368e+002	-9.41390e-005	8.50372e-004	1.20	0.13	0.25	0
qap7	4.24759e+002	-7.24794e-005	7.13475e-004	1.89	0.23	0.55	0
qap8	7.56773e+002	-1.21964e-004	1.12744e-003	3.19	0.52	1.11	0
qap9	1.40982e+003	-4.54030e-005	5.83654e-004	6.08	0.97	2.73	0
qpG11	-2.44866e+003	3.84682e-010	3.96618e-010	329.47	54.09	891.63	0
qpG51	-1.18180e+004	5.13070e-009	5.29349e-009	675.59	104.88	1200.11	0
ss30	-2.02395e+001	2.84736e-008	1.80610e-006	25.70	0.73	3.33	0
theta1	-2.30000e+001	3.44570e-009	2.71595e-008	0.81	0.11	0.13	0
theta2	-3.28792e+001	-1.45377e-008	1.41623e-007	3.00	1.23	1.80	0
theta3	-4.21670e+001	-7.93078e-010	8.77306e-008	11.58	6.42	8.95	0
theta4	-5.03212e+001	-1.07772e-008	1.91404e-007	38.19	21.75	27.84	0
theta5	-5.72323e+001	-5.00122e-008	5.00145e-007	123.92	51.55	69.55	0
thetaG11	-4.00000e+002	3.40521e-008	3.47616e-008	212.95	12.25	381.63	-7
truss1	9.00000e+000	1.12011e-009	1.13109e-006	0.95	0.13	0.08	0
truss2	1.23380e+002	-1.56295e-007	4.70668e-006	5.95	0.38	0.64	0
truss3	9.11000e+000	2.95941e-009	5.40589e-007	1.27	0.09	0.11	0
truss4	9.01000e+000	2.33560e-009	1.25934e-006	0.97	0.09	0.08	0
truss5	1.32636e+002	-5.09856e-010	7.71633e-006	12.73	0.81	2.05	0
truss6	9.00979e+002	-2.37959e-005	2.34220e-004	34.64	2.80	6.64	0
truss7	8.99993e+002	-8.88501e-006	1.08335e-004	31.44	1.83	3.67	0
truss8	1.33114e+002	-5.36216e-006	5.97081e-005	15.38	1.84	5.89	0

References

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

[2] B. Borchers. SDPLIB 1.2, A Library of Semidefinite Programming Test Problems. *Optimization Methods and Software*, 11(1):683–690, 1999.

[3] R.M. Freund, F. Ordóñez, and K. Toh. Behavioral Measures and their Correlation with IPM Iteration on Semi-Definite Programming Problems.

[4] M. Fukuda, B.J. Braams, M. Nakata, M.L. Overton, J.K. Percus, M. Yamashita, and Z. Zhao. Large-scale semidefinte programs in electronic structure calculation. Mathematical Programming, to appear.

[5] M. Grant, S. Boyd, and Y. Ye. *cvx User's Guide*, 2006. `http://www.stanford.edu/~boyd/cvx`.

[6] G. Gruber, S. Kruk, F. Rendl, and H. Wolkowicz. Presolving for Semidefinite Programs Without Constraint Qualifications. In G. Gruber et al., editor, *Proceedings of HPOPT97, Second Workshop on High Performance Optimization Techniques, Rotterdam, Netherlands*, 1997.

[7] G. Gruber and F. Rendl. Computational experience with ill-posed problems in semidefinite programming. *Computational Optimization and Applications*, 21(2):201–212, 2002.

[8] C.P. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, 1998.

[9] C. Jansson. A rigorous lower bound for the optimal value of convex optimization problems. *J. Global Optimization*, 28:121–137, 2004.

[10] C. Jansson. Rigorous Lower and Upper Bounds in Linear Programming. *SIAM J. Optimization (SIOPT)*, 14(3):914–935, 2004.

[11] C. Jansson. Termination and Verification for Ill-posed Semidefinite Programming Problems, 2005.

[12] C. Jansson, D. Chaykin, and C. Keil. Rigorous Error Bounds for the Optimal Value in Semidefinite Programming. 2005. To appear in SIAM Journal on Numerical Analysis (SINUM).

[13] W.M. Kahan. The Improbability of Probabilistic Error Analysis for Numerical Computations. talk given at the UCB Statistics Colloquium, 1996. `http://http.cs.berkeley.edu/~wkahan/improber.ps`.

[14] R.B. Kearfott. *Rigorous Global Search: Continuous Problems.* Kluwer Academic Publisher, Dordrecht, 1996.

[15] MATLAB User's Guide, Version 7. The MathWorks Inc., 2004.

[16] H.D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Math. Programming Ser. B*, 95:407–430, 2003.

[17] R.E. Moore. *Methods and Applications of Interval Analysis.* SIAM, Philadelphia, 1979.

[18] M. Nakata, M. Fukuda, K. Nakata, and K. Fujisawa. Variational calculus of fermion second-order reduced density matrices by semidefinite programming algorithm. *J. of Chemical Physics*, 114(19):8282–8292, 2001.

[19] A. Neumaier. *Interval Methods for Systems of Equations.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1990.

[20] A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review (SIREV)*, 40:636–666, 1998.

[21] A. Neumaier. *Introduction to Numerical Analysis.* Cambridge University Press, 2001.

[22] A. Neumaier and O. Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming, Ser. A*, 99:283–296, 2004.

[23] F. Ordóñez and R.M. Freund. Computational experience and the explanatory value of condition measures for linear optimization. *SIAM J. Optimization (SIOPT)*, 14(2):307–333, 2003.

[24] J. Renegar. Linear Programming, complexity theory, and elementary functional analysis. *Mathematical Programming*, 70(3):279–351, 1995.

[25] S.M. Rump. Fast and parallel interval arithmetic. *BIT Numerical Mathematics*, 39(3):539–560, 1999.

[26] S.M. Rump. INTLAB - Interval Laboratory, the Matlab toolbox for verified computations, Version 5.3, 2006.

[27] A.N. Tikhonov and V.L. Arsenin. *Solutions of Ill-posed Problems.* John Wiley, New York, 1977.

[28] R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Math. Program.*, 95B(2):189–217, 2003.

[29] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review (SIREV)*, 38(1):49–95, 1996.

[30] H. Wei and H. Wolkowicz. Generating and Measuring Instances of Hard Semidefinite Programs. `www.optimization-online.org/DB_HTML/2006/01/1291.html`, 2006.

[31] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*, volume 27 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, Boston, MA, 2000.

[32] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of SDPA 6.0. *Optimization Methods and Software*, 18(4):491–505, 2003.