A Generator of Nonregular Semidefinite Programming Problems

Preprint

Eloísa Macedo[*]
Tatiana Tchemisova[†]

Abstract

Regularity is an important property of optimization problems. Various notions of regularity are known from the literature, being defined for different classes of problems. Usually, optimization methods are based on the optimality conditions, that in turn, often suppose that the problem is regular. Absence of regularity leads to theoretical and numerical difficulties, and solvers may fail to provide a trustworthy result. Therefore, it is very important to verify if a given problem is regular in terms of certain regularity conditions and in the case of nonregularity, to apply specific methods. On the other hand, in order to test new stopping criteria and the computational behaviour of new methods, it is important to have an access to sets of reasonably-sized nonregular test problems. The paper presents a generator that constructs nonregular Semidefinite Programming (SDP) instances with prescribed irregularity degrees and a database of nonregular test problems created using this generator. Numerical experiments using popular SDP solvers on the problems of the database are carried out and permit to conclude that the most popular SDP solvers are not efficient when applied to nonregular problems.

Keywords: Semidefinite Programming, regularity, constraint qualification, good behaviour, generator of nonregular SDP problems

1 Introduction

Semidefinite programming is an active area of research due to its many applications in combinatorial, convex, and robust optimization, computational biology, systems and control theory, sensor network location, and data analysis, among others [1]. SDP refers to convex optimization problems where a linear function is minimized subject to constraints in the form of linear matrix inequalities (LMIs).

The most efficient methods for solving SDP problems are based on the first-order necessary optimality conditions, also called Karush-Kuhn-Tucker-type (KKT) conditions [23], which in turn are derived under some special assumptions on the feasible set of the problem, the regularity conditions [9, 11, 23]. Regularity plays an important role in characterizing optimality of feasible solutions, guaranteeing the efficiency of numerical methods and stability of solutions. There exist different notions of regularity, such as Constraint Qualification (CQ) [1, 18, 23], well-posedness [6, 10], or good behaviour in the sense of Pataki [16], which were recently proved to be closely related to each other [15, 16].

The Slater condition, which consists in existence of strictly feasible solutions, is a widely used CQ in SDP, and many authors assume in their works that this condition holds. However, in practice, there are many SDP problem instances that fail to satisfy the Slater condition, *i.e.*, are nonregular (*e.g.*, [3, 6, 8, 10, 21]). In the absence of regularity, theoretical and numerical difficulties may occur. Although in these cases some special regularization techniques (*e.g.*, preprocessing [3], presolving [7], self-dual

[*](macedo@ua.pt) TEMA, Department of Mechanical Engineering, University of Aveiro
[†](tatiana@ua.pt) CIDMA, Department of Mathematics, University of Aveiro

embedding [11]) can be applied, in practice, the SDP solvers may still run into numerical difficulties. In fact, the popular SDP solvers do not check the regularity of problems, consequently, trustworthiness of results is not guaranteed while solving nonregular problems. Therefore, it is very important to verify if a given SDP problem is regular in some sense before passing it to a solver. In [14] and [15], we presented a detailed description of two numerical procedures to check regularity of SDP problems in terms of the fulfilment of the Slater condition and conducted several numerical experiments, which show that many problems from the known SDPLIB database [2] are nonregular. Nevertheless, this database has been widely used for testing the performance and robustness of SDP software, which works under assumption of problem's regularity.

As it was pointed out in [6, 17], it would be important to have a library of problems with a particular structure, irregular or even infeasible instances, to develop and test new stopping criteria for SDP methods, and create more efficient solvers. In [13], an algorithm for generating infeasible SDP instances was presented. A generator of *hard* SDP instances, for which the strict complementary fails, was proposed in [22]. In this light, it is also important to create libraries of nonregular SDP problems, or develop procedures that permit to generate nonregular SDP problem instances to evaluate the computational behaviour of new methods, in particular those specially conceived for solving the nonregular problems.

The main purpose of the paper is to describe an algorithm for generating SDP problems failing the Slater condition, and present a generator of nonregular SDP problem instances that was implemented in MATLAB. We also present a collection of nonregular SDP instances with a particular structure, encoded in standard format.

The paper is organized as follows. Section 1 hosts the Introduction. In Section 2, some notation and definitions are introduced. The regularity notions, such as the Slater condition and good behaviour, as well as their relationships are discussed in Section 3. In Section 4, we present procedures to verify regularity of SDP problems in terms of the fulfilment of the Slater condition and determine the level of their (ir)regularity. Section 5 is devoted to a particular class of nonregular SDP problems. A generator of nonregular SDP problems with prescribed irregularity degree is presented. The collection of nonregular SDP test problems called NONREGSDP and numerical experiments are described in Section 6. The final Section 7 contains conclusions.

2 Linear Semidefinite Programming Problem

Given $s \in \mathbb{N}$, $\mathcal{S}(s)$ denotes the space of the $s \times s$ real symmetric matrices equipped with the trace inner product given by $tr(\mathbf{AB}) = \sum\limits_{i=1}^{s}\sum\limits_{j=1}^{s} a_{ij}b_{ji}$, for $\mathbf{A}, \mathbf{B} \in \mathcal{S}(s)$, and $\mathcal{P}(s) \subset \mathcal{S}(s)$ denotes the cone of $s \times s$ positive semidefinite symmetric matrices. Consider the following SDP problems:

$$\min_{x \in \mathbb{R}^n} \quad c^T x \quad \text{s.t.} \quad \mathcal{A}(x) \preceq 0, \tag{1}$$

$$\max_{\mathbf{Z} \in \mathcal{S}(s)} \quad \operatorname{tr}(\mathbf{A}_0 \mathbf{Z}) \quad \text{s.t.} \quad -\operatorname{tr}(\mathbf{A}_i \mathbf{Z}) = c_i, \forall i = 1, \dots, n, \quad \mathbf{Z} \succeq 0, \tag{2}$$

where x is the primal vector variable, $\mathbf{Z}$ is the dual matrix variable, $c \in \mathbb{R}^n$ and $\mathcal{A}(x)$ is a matrix-valued function defined as $\mathcal{A}(x) := \sum\limits_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{A}_0$, where $\mathbf{A}_i \in \mathcal{S}(s)$, $i = 0, 1, ..., n$.

Without loss of generality, we can assume that the matrices $\mathbf{A}_i$, $i = 1, ..., n$, are linearly independent. Problem (1) is called primal, and (2) is its dual. We denote the primal and dual feasible sets of (1) and (2), by $\mathcal{X} = \{x \in \mathbb{R}^n : \mathcal{A}(x) \preceq 0\}$ and $\mathcal{Z} = \{\mathbf{Z} \in \mathcal{P}(s) : -\operatorname{tr}(\mathbf{A}_i \mathbf{Z}) = c_i, i = 1, ..., n\}$, respectively.

3 Regularity in Semidefinite Programming

The most common regularity notions are given in terms of some special conditions on the feasible sets or on the constraint functions. Constraint qualifications are special conditions that guarantee that the first-order necessary optimality conditions – the KKT optimality conditions – are satisfied. An optimization problem is often called regular if certain CQ is satisfied [9], and nonregular, otherwise.

Duality results are fairly subtle in SDP, requiring regularity of the problem in some sense. It is well known that as well as in Linear Programming, in SDP the weak duality holds for any pair of primal and dual feasible solutions $x \in \mathcal{X}$ and $\mathbf{Z} \in \mathcal{Z}$ of problems (1) and (2), *i.e.*, $p = c^T x \geq \operatorname{tr}(\mathbf{A}_0 \mathbf{Z}) = d$. Let p^* and d^* denote the optimal values of the SDP problems (1) and (2), respectively. The difference $p^* - d^*$ is called duality gap. In SDP, to guarantee the vanishing of the duality gap some additional assumptions have to be made. An often used sufficient condition to ensure zero duality gap is the existence of a strictly feasible solution. This condition is called strict feasibility or the Slater regularity condition [11].

Definition 1 *The constraints of the problem* (1) *satisfy the Slater (regularity) condition if the interior of its feasible set $\mathcal{X}$ is nonempty, i.e., $\exists\ \bar{x} \in \mathbb{R}^n : \mathcal{A}(\bar{x}) \prec 0$.*

If assume that the primal optimal value is finite and the Slater condition holds, then strong duality holds, *i.e.*, the duality gap is zero, and the dual optimal value can be attained [11]. The strong duality plays an important role in the numerical solving of SDP problems. However, it can fail in the absence of the Slater condition and either a dual optimal solution may not exist or the duality cap may be not zero. Therefore, solvers may run into numerical difficulties and not be able to provide trustworthy solutions.

In the literatures, there are other notions of regularity for SDP problems, such as well-posedness and good behaviour. The well-posedness of a problem is related to its behaviour in terms of (in)feasibility under small perturbations [6, 10, 11]. The good behaviour of a SDP problem is related to the fulfilment of the strong duality property [16]. More specifically, assuming that a SDP problem is feasible, the following definition was introduced in [16].

Definition 2 *The SDP problem in the form* (1) *is said to be well-behaved, if strong duality holds for all objective functions. Otherwise, the problem is said to be badly-behaved.*

A SDP problem is well-behaved in the sense of Pataki [16] if strong duality holds, which can be ensured if a regularity condition, such as the Slater condition, holds. Therefore, the good behaviour of a SDP problem is closely related to the Slater condition. The following result was proved in [16] (Corollary 1).

Proposition 1 *If the constraints of the SDP problem* (1) *satisfy the Slater condition, then the problem is well-behaved.*

On the basis of this proposition, we can conclude that if the SDP problem (1) is badly-behaved, then it does not satisfy the Slater condition.

4 Testing and Measuring Regularity in SDP

Different approaches to verify regularity of SDP problems have been proposed in the literature. In terms of well-posedness, two characterizations are known, one based on the Renegar condition number [6], and another based on a rigorous upper bound on the primal optimal value [10]. In [16], a characterization of good behaviour in the sense of Pataki is described and we will briefly discuss it later. In what follows, we suggest our original approach to verify regularity in terms of the fulfilment of the Slater condition.

4.1 Subspace of immobile indices and irregularity degree of SDP problems

The following definition was given in [12].

Definition 3 *Given the linear SDP problem* (1)*, the subspace of $\mathbb{R}^s$ defined by*

$$\mathcal{M} := \left\{ l \in \mathbb{R}^s : l^T \mathcal{A}(x) l = 0, \forall x \in \mathcal{X} \right\} \tag{3}$$

is called the subspace of immobile indices.

On the basis of this definition, and the results in [12], we can prove the following theorem.

Theorem 1 *The SDP problem* (1) *satisfies the Slater condition if and only if the subspace of immobile indices $\mathcal{M}$ is null, i.e., $\mathcal{M} = \{0\}$.*

Proof First, let us reformulate the linear SDP problem (1) in the equivalent form:

$$\min \quad c^T x \quad \text{s.t.} \quad l^T \mathcal{A}(x) l \leq 0, \forall l \in L := \{l \in \mathbb{R}^s : \|l\|_2 = 1\}, \tag{4}$$

where the set L is an (infinite) index set. This problem has an infinite number of constraints, and thus, is a convex Semi-Infinite Programming (SIP) problem. Notice that the feasible sets of (4) and (1) coincide: $\left\{x \in \mathbb{R}^n : l^T \mathcal{A}(x) l \leq 0, \forall l \in L\right\} = \{x \in \mathbb{R}^n : \mathcal{A}(x) \preceq 0\} = \mathcal{X}$.

It was proved in [12] that the SIP problem (4) satisfies the Slater condition, *i.e.*, $\exists \bar{x} \in \mathcal{X} : l^T \mathcal{A}(\bar{x}) l < 0, \forall l \in L$, if and only if the *set of immobile indices* given by $L^* = \left\{l \in L : l^T \mathcal{A}(x) l = 0, \forall x \in \mathcal{X}\right\}$ is empty. Evidently, $L^* = L \cap \mathcal{M}$ and then, the subspace $\mathcal{M}$ of immobile indices is null if and only if L^* is empty.

Since the problems (1) and (4) are equivalent, then they satisfy or not the Slater condition, simultaneously. Therefore, (1) satisfies the Slater condition if and only if $\mathcal{M}$ is null. ■

The connection established between the subspace of immobile indices and the Slater condition permits us to introduce a measure of nonregularity (or irregularity) for SDP problems, which we will call here the *irregularity degree* of a SDP problem.

Definition 4 *The dimension of a basis of the subspace $\mathcal{M}$ of immobile indices for the SDP problem* (1)*, denoted by s^*, is called irregularity degree of this problem.*

This definition permits to classify SDP problems in the form (1) taking into account the dimension s^* of the subspace $\mathcal{M}$ as follows:

- if $s^* = 0$, then the problem is regular, *i.e.*, the Slater condition holds;
- if $s^* = 1$, then the problem is nonregular, with minimal irregularity degree;
- if $s^* = s$, then the problem is nonregular, with maximal irregularity degree.

In fact, for a given SDP problem, the nonvanishing dimension of a basis of the subspace of immobile indices can be considered as a *certificate* of nonregularity, *i.e.*, it proves the failure of the Slater condition.

4.2 Testing regularity and determining the irregularity degree

In [12], an algorithm DIIS (Determination of the Immobile Index Subspace) was proposed to find a basis of the subspace $\mathcal{M}$. Here, we will show that the DIIS algorithm can be used to check weather the Slater condition is satisfied for a given SDP problem.

Given a feasible SDP problem in the form (1), the DIIS algorithm constructs a basis of the subspace $\mathcal{M}$ of immobile indices which is formed by s^* vectors $m_i \in \mathbb{R}^s$, $i = 1, ..., s^*$ obtained by (3). The vectors of this basis form a matrix $\mathbf{M}$. It can be shown that the rank of $\mathbf{M}$ is equal to the irregularity degree of the problem, that in turn permits to conclude about the regularity of the problem. For the sake of completeness, we present the algorithm here. At the k-th iteration, let I^k denote a set of indices and M^k a set of vectors which, at the end of the algorithm, will form the basis of $\mathcal{M}$.

Algorithm 1 Testing regularity and determining the irregularity degree of SDP problems

```
input:  $n$, number of variables in the SDP problem;
        $s$, dimension of the constraint matrices;
        $\mathbf{A}_j$, $j = 0, 1, ..., n$, $s \times s$ symmetric real constraint matrices of a SDP problem in the form (1).
output: status, classification of the problem as regular or nonregular;
        $s^*$, irregularity degree value.
 1: set $k := 1$, $I^1 := \emptyset$ and $M^1 := \emptyset$.
 2: repeat
 3:     given $k \geq 1$, $I^k$, $M^k = \{m_1, m_2, ..., m_{|I^k|}\}$ with $m_i \in \mathbb{R}^s$, $i \in I^k$:
 4:     set $p_k := s - |I^k|$ and solve the system
```

$$\begin{cases} \sum\limits_{i=1}^{p_k} l_i^T \mathbf{A}_j l_i + \sum_{i \in I^k} \gamma_i^T \mathbf{A}_j m_i = 0, \; j = 0, 1, \ldots, n, \\ \sum\limits_{i=1}^{p_k} \|l_i\|^2 = 1, \\ l_i^T m_j = 0 \;, \; j \in I^k, \; i = 1, \ldots, \; p_k, \end{cases} \tag{5}$$

```
        w.r.t. the variables $l_i \in \mathbb{R}^s, i = 1, ..., p_k$, $\gamma_i \in \mathbb{R}^s, i \in I^k$
 5:     if system (5) is inconsistent, then stop
 6:     else given the solution $\{l_i \in \mathbb{R}^s, i = 1, \ldots, p_k, \gamma_i \in \mathbb{R}^s, i \in I^k\}$ of (5):
 7:         construct the maximal subset of linearly independent vectors of the set $\{l_1, \ldots, l_{p_k}\}$; rename its vectors
    as $\{\xi_1, \ldots, \xi_{s_k}\}$, where $s_k$ is the number of linearly independent vectors in $\{l_1, \ldots, l_{p_k}\}$
 8:         given $\{\xi_1, \ldots, \xi_{s_k}\}$, update:
 9:                 $\triangle I^k := \{|I^k| + 1, \ldots, |I^k| + s_k\}$,
10:                 $m_{|I^k|+i} := \xi_i$, $i = 1, ..., s_k$,
11:                 $M^{k+1} := M^k \cup \{m_j, \; j \in \triangle I^k\}$,
12:                 $I^{k+1} := I^k \cup \triangle I^k$.
13:         set $k := k + 1$
14: until system (5) is inconsistent
15: given $M^k$:
16:     construct $\mathbf{M}$, whose columns are the vectors from $M^k$;
17:     compute $s^* := \text{rank}(\mathbf{M})$.
18: if $k = 1$ then set status: Regular
19: else set status: Nonregular
    return status, Irregularity degree $= s^*$.
```

The procedure of the Algorithm 1 is constructed so that:

- if the Slater condition holds, then the algorithm stops at the first iteration with $k = 1$, $\mathcal{M} = \{0\}$ and $s^* = 0$;
- if the Slater condition fails to hold, then the algorithm returns a basis $\mathbf{M}$ with $\text{rank}(\mathbf{M}) = s^* > 0$.

The main task on each iteration of this algorithm consists in solving the system of quadratic equations (5). At the k-iteration, this system has $p_k + |I^k|$ vector variables (and $s(p_k + |I^k|)$ scalar variables) and $n+2+p_k \times |I^k|$ equations. Notice that one iteration is enough to verify if a given SDP problem is regular in terms of the Slater condition and in this case, one has to solve a system with s vector variables and $n + 2$ equations.

In [15], we have developed two MATLAB numerical tools:

- `SDPreg`, verifies regularity by performing a single iteration of the Procedure 1;
- `DIISalg`, determines the irregularity degree of SDP problems, performing all iterations of the Procedure 1.

These tools are available from the authors upon request. These presolving tools should be run before solving any SDP problem, in order to foresee either a standard SDP solver may be applied for the numerical solving of the given problem. In the case the test indicates that the given SDP problem is irregular, to ensure trustworthiness of solution, some special procedure should be applied.

4.3 Testing regularity in terms of good behaviour

In [16], the following characterizations of the badly-behaved SDP problems were proposed.

Theorem 2 *([16], Theorem 2) The SDP problem (1) is badly-behaved if and only if there exists a matrix $\mathbf{V}$, which is a linear combination of the matrices $\mathbf{A}_i$, for $i = 0, ..., n$, of the form*

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} \\ \mathbf{V}_{12}^T & \mathbf{V}_{22} \end{bmatrix}, \tag{6}$$

where $\mathbf{V}_{11}$ is a $(r \times r)$ symmetric matrix, $\mathbf{V}_{22}$ is a $((s-r) \times (s-r))$ positive semidefinite matrix and $\mathbf{V}_{12}$ is a $((s-r) \times r)$ matrix such that the range space of $\mathbf{V}_{12}^T$, denoted here by $\mathcal{C}(\mathbf{V}_{12}^T)$, is not contained in $\mathcal{C}(\mathbf{V}_{22})$.

Theorem 3 *([16], Theorem 4) The SDP problem (1) is badly-behaved if and only if it has a reformulation in the form*

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \sum\limits_{i=1}^{k} x_i \begin{bmatrix} \mathbf{F}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \sum\limits_{i=k+1}^{n} x_i \begin{bmatrix} \mathbf{F}_i & \mathbf{G}_i \\ \mathbf{G}_i^T & \mathbf{H}_i \end{bmatrix} \preceq \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{S}, \end{array} \tag{7}$$

where

1. *$\mathbf{S}$ is the maximum rank slack matrix, $\mathbf{S} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, where r is an integer taking values between 1 and $s-1$, $\mathbf{I}_r$ is the identity matrix of order r and $\mathbf{0}$ is the null matrix of suitable dimensions;*
2. *the matrices $\begin{bmatrix} \mathbf{G}_i \\ \mathbf{H}_i \end{bmatrix}$, for $i = k+1, ..., n$, are linearly independent;*
3. *$\mathbf{H}_n \succeq 0$.*

According to [16], the matrices $\mathbf{S}$ and $\mathbf{V}$ provide a *certificate* of the bad behaviour of a SDP problem in the form (1).

Since it can be shown that a badly-behaved problem does not satisfy the Slater condition, then we can use the developed numerical tool `SDPreg` to verify the good behaviour of a given SDP problem.

5 A Generator of Nonregular SDP instances

As it was remarked in [6, 15, 17], to develop and test new numerical SDP methods, it is important to have libraries of nonregular SDP problems, as well as problems with a particular structure or infeasible SDP instances. In this section, we propose a generator of nonregular SDP problem instances with certain predefined properties.

Based on the Theorems 2 and 3 formulated in the previous section, we can describe a class of nonregular SDP problems with a desired irregularity degree s^*, $1 \leq s^* \leq s-1$, and the optimal value $p^* = 0$ as a class of problems in the form (1) that satisfy the following conditions:

1. the integers s and n are as follows: $s \geq 2$, $1 \leq n \leq \frac{s(s+1)}{2}$;
2. c is a n-dimensional vector: $c = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T$;
3. $\mathbf{A}_0 := -\left[\begin{array}{c:c} \mathbf{D}_r & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} \end{array}\right]_{s \times s}$, where $r = 1, ..., s-1$, and $\mathbf{D}_r = \operatorname{diag}(\beta_1, \dots, \beta_\mathrm{r})$ with $\beta_i \in \mathbb{R}^+$, $i = 1, ..., r$,
4. the matrices $\mathbf{A}_i$, $i = 1, ..., n$, have the form

$$\mathbf{A}_i = \left[\begin{array}{c:c} \mathbf{F}_i & \mathbf{G}_i \\ \hdashline \mathbf{G}_i^T & \mathbf{H}_i \end{array}\right]_{s \times s}, \quad i = 1, ..., n. \tag{8}$$

Here for $i = 1, ..., n$, matrices $\mathbf{F}_i$ are symmetric: $\mathbf{F}_i \in \mathcal{S}(r)$; matrices $\mathbf{H}_i \in \mathcal{S}(s-r)$, have null diagonal, $\mathbf{H}_1$ being a null matrix: $\mathbf{H}_1 := \mathbf{0} \in \mathcal{S}(s-r)$. A non vanishing matrix $\mathbf{G}_1$ has the form $\mathbf{G}_1 = \sum_{j=1}^{r(s-r)} \alpha_j \mathbf{T}_j$, where $\alpha_j \in \mathbb{R}$, and $\mathbf{T}_j \in T$, $j = 1, ..., r(s-r)$, T being the canonical basis of $\mathbb{R}^{r\times(s-r)}$. Matrices $\mathbf{G}_i \in \mathbb{R}^{r\times(s-r)}$, $i = 2, ..., s-1$, are linearly independent and are chosen in the form of multiples of the matrices from T. For $i \geq s$, we set $\mathbf{G}_i := \mathbf{0} \in \mathbb{R}^{r\times(s-r)}$.

We can then outline an algorithm for generating nonregular SDP instances as follows.

Algorithm 2 Generating SDP instances with pre-specified irregularity degree s^*

```
input: $n$, number of variables in the SDP problem;
       $s$, dimension of the constraint matrices;
       $s^*$, desired irregularity degree.
output: $\mathbf{A}_i$, $i = 0, ..., n$, constraint matrices;
        $c$, vector of coefficients of an objective function.
 1: compute $r = s - s^*$
 2: choose an arbitrary $(r \times r)$ diagonal matrix $\mathbf{D}_r$ with $r$ positive entries
 3: set the $(s \times s)$ matrix $\mathbf{A}_0$ to $\mathbf{A}_0 = -\left[\begin{array}{c|c} \mathbf{D}_r & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right]$
 4: generate random symmetric $(r \times r)$ matrices $\mathbf{F}_i$, $i = 1, ..., n$
 5: obtain the canonical basis of $\mathbb{R}^{r\times s^*}$, $T = \{\mathbf{T}_j, j = 1, ..., rs^*\}$
 6: choose the matrix $\mathbf{G}_1 \neq \mathbf{0} \in \mathbb{R}^{r\times s^*}$ such that $\mathbf{G}_1 = \sum_{j=1}^{rs^*} \alpha_j \mathbf{T}_j$, for $\mathbf{T}_j \in T$ and arbitrary coefficients $\alpha_j \in \mathbb{R}$, $j = 1, ..., rs^*$
 7: for $i = 2, ..., s$ do
 8:     choose matrices $\mathbf{G}_i \in \mathbb{R}^{r\times s^*}$ such that $\mathbf{G}_i = \alpha\mathbf{T}$, for some $\mathbf{T} \in T$, $\alpha \in \mathbb{R}$, and matrices $\mathbf{G}_i$, $i = 1, ..., s$, are linearly independent
 9: for $i > s$ do
10:     set $\mathbf{G}_i := \mathbf{0}$
11: set $\mathbf{H}_1 := \mathbf{0}$
12: choose arbitrary $\mathbf{H}_i \in \mathcal{S}(s^*)$, $i = 2, ..., n$, having a null diagonal
13: for $i = 1, ..., n$ do $\mathbf{A}_i = \left[\begin{array}{c|c} \mathbf{F}_i & \mathbf{G}_i \\ \hline \mathbf{G}_i^T & \mathbf{H}_i \end{array}\right]$
14: set $c_1 := 1$ and $c_i := 0$, for $i = 2, ..., n$
15: return $\mathbf{A}_i$, $i = 0, 1, ..., n$, and $c$.
```

The following theorem states the main properties of the algorithm.

Theorem 4 *Given positive integers s, $n \leq \frac{s(s+1)}{2}$ and s^* with $1 \leq s^* \leq s-1$ as input in the Algorithm 2, the following properties hold for any problem of the form (1) generated by the Algorithm 2:*

1. *the generated problem is feasible;*
2. *any feasible solution is optimal with $x_1 = 0$ and the corresponding optimal value is $p^* = 0$;*
3. *the Slater condition is not satisfied.*

Proof A problem generated by the Algorithm 2 is a SDP problem of the form (1). It is feasible, since it admits the trivial solution. By construction, the constraint matrices $\mathbf{A}_i$, $i = 1, ..., n$, have the form (8) and have at least s^* zeros on the same entries of the main diagonal, while $\mathbf{A}_0$ has exactly s^* zeros. Additionally, for $i = 1, ..., n-s$, the matrices $\mathbf{A}_i$ are linearly independent. Thus, the constraint matrix of the problem will have s^* zeros on the diagonal. Since the matrices $\mathbf{G}_i$, $i = 2, ..., n-s$, and $\mathbf{G}_1$ form a linearly independent set, using the property that if any diagonal entry is zero, then the corresponding row and column are also full of zeros, it follows that any feasible solution has $x_1 = 0$. Hence, it is easy to see that all feasible solutions are optimal and the optimal value is $p^* = 0$.

Since any problem generated by the Algorithm 2 is badly-behaved ([16]), it follows from Proposition 1 that it does not satisfy the Slater condition. ■

5.1 Implementation details of the nonregular SDP instance generator `nonregSDPgen`

We have implemented the Algorithm 2 in MATLAB programming language, since many SDP solvers are either coded in MATLAB, or have interface with MATLAB. The resulting function is called `nonregSDPgen` and generates nonregular SDP instances with a pre-specified irregularity degree, s^*, from 1 up to $s-1$. In the steps of the Algorithm 2, one has to generate random symmetric $(r \times r)$ matrices $\mathbf{F}_i$, $i = 1, ..., n$. We have implemented a procedure to obtain such matrices as linear combinations of elements of the canonical basis of $\mathcal{S}(r)$. The generated instances have a specific structure and have integer entries in their constraint matrices. The `nonregSDPgen` function returns a nonregular SDP instance written in dat-s format in a new file, whose name should be pre-specified by users. In the MATLAB environment, the user starts by choosing the parameters `n`, `s` and `d`, which correspond to the number of variables of the SDP problem, dimension of the constraint matrices and desired irregularity degree, respectively. The name for the new file that will be created to store the generated data in sparse SDPA format [24], *e.g.*, `examplename.dat-s`, should be specified as well. The basic calling statement structure of the `nonregSDPgen` function is

```
> nonregSDPgen(n,s,d,'examplename.dat-s')
```

The `nonregSDPgen` will create a new dat-s file with a nonregular SDP instance of a pre-specified irregularity degree, which can be used by any SDP solver that requires this input format.

6 NONREGSDP: a nonregular SDP database

For numerical testing, it is important to have access to collections of test problems “for comparing the performance and robustness of software for solving these optimization problems. Such comparisons have led to significant improvements in the speed and robustness of optimization software” [2]. The SDPLIB [2] is a library of linear SDP test problems with a wide range of sizes, which is usually used to test the performance of solvers. In [6], it is mentioned that it would be interesting to have “a reasonably-sized set of SDP problem instances that might be better suited to empirically examine issues related to the computational behaviour of algorithms for SDP”. Since the performance of SDP solvers may be compromised when the Slater condition fails to hold, thus, it makes sense to have a collection of moderate-sized nonregular SDP instances, that is, failing the Slater condition. In this light, we have created a new SDP database and conducted computational experiments.

6.1 NONREGSDP

We have generated 100 nonregular SDP instances using the routine `nonregSDPgen` and we have called this collection of test problems NONREGSDP. The current version of this new database is available from the author upon request. The NONREGSDP database is a moderate-sized set of SDP problem instances that can be used for testing the behaviour of SDP algorithms and new stopping criteria. The SDP problems from NONREGSDP were obtained for different values of n and s, with n varying from 1 to 12, s from 2 to 30, and with irregularity degree d varying from 1 up to 29. We have tested the instances from NONREGSDP with our MATLAB function `DIISalg` in order to confirm the irregularity degree of the SDP instances. Table 1 provides detailed information on the new SDP library. The column “*Problem*” contains the instance's name, and the parameters n, s, and d refer to the number of variables, the dimension of the constraint matrices and the irregularity degree value, respectively.

6.2 Numerical results and discussion

In this section, we used 54 instances from the NOREGSDP database to test the computational behaviour of the popular SDP solvers SDPT3 [20] and SeDuMi [19]. All computations were performed on a computer with an Intel Core i7-2630QM processor CPU@2.0GHz, with Windows 7 (64 bits) and 12 GB RAM, using MATLAB (v.7.12 R2013a). We tried to solve some generated instances using two different solvers available on the package CVX [4], SDPT3 and SeDuMi, and the default precision or tolerance values.

Table 1: SDP instances from the NONREGSDP database.

Problem	*n*	*s*	*d*	*Problem*	*n*	*s*	*d*	*Problem*	*n*	*s*	*d*	*Problem*	*n*	*s*	*d*
nonreg1	2	2	1	*nonreg26*	4	2	1	*nonreg51*	3	5	4	*nonreg76*	12	25	12
nonreg2	3	3	1	*nonreg27*	1	3	1	*nonreg52*	7	4	1	*nonreg77*	12	25	24
nonreg3	3	3	2	*nonreg28*	1	3	2	*nonreg53*	7	4	2	*nonreg78*	2	18	4
nonreg4	4	4	1	*nonreg29*	2	3	1	*nonreg54*	7	4	3	*nonreg79*	9	23	1
nonreg5	4	4	2	*nonreg30*	2	4	1	*nonreg55*	2	20	2	*nonreg80*	9	23	11
nonreg6	4	4	3	*nonreg31*	1	4	1	*nonreg56*	4	21	1	*nonreg81*	9	23	22
nonreg7	5	4	3	*nonreg32*	1	4	3	*nonreg57*	2	30	29	*nonreg82*	4	17	2
nonreg8	3	4	2	*nonreg33*	2	4	1	*nonreg58*	3	11	9	*nonreg83*	4	17	12
nonreg9	6	2	2	*nonreg34*	2	4	2	*nonreg59*	10	15	10	*nonreg84*	10	30	1
nonreg10	1	4	2	*nonreg35*	2	4	3	*nonreg60*	1	27	25	*nonreg85*	10	30	5
nonreg11	5	10	1	*nonreg36*	3	4	1	*nonreg61*	10	30	29	*nonreg86*	10	30	10
nonreg12	5	10	2	*nonreg37*	3	4	3	*nonreg62*	6	24	11	*nonreg87*	10	30	15
nonreg13	5	10	3	*nonreg38*	5	4	1	*nonreg63*	5	13	10	*nonreg88*	10	30	20
nonreg14	5	10	4	*nonreg39*	5	4	2	*nonreg64*	5	13	1	*nonreg89*	10	30	25
nonreg15	5	10	5	*nonreg40*	1	5	1	*nonreg65*	12	30	29	*nonreg90*	1	30	1
nonreg16	5	10	6	*nonreg41*	1	5	2	*nonreg66*	12	30	1	*nonreg91*	1	30	10
nonreg17	5	10	7	*nonreg42*	1	5	3	*nonreg67*	2	25	5	*nonreg92*	1	30	20
nonreg18	5	10	8	*nonreg43*	1	5	4	*nonreg68*	7	28	2	*nonreg93*	1	30	29
nonreg19	5	10	9	*nonreg44*	2	5	1	*nonreg69*	7	28	7	*nonreg94*	8	21	1
nonreg20	2	10	1	*nonreg45*	2	5	2	*nonreg70*	7	28	12	*nonreg95*	8	21	8
nonreg21	12	10	1	*nonreg46*	2	5	3	*nonreg71*	7	28	19	*nonreg96*	8	21	15
nonreg22	6	4	1	*nonreg47*	2	5	4	*nonreg72*	7	28	27	*nonreg97*	8	21	20
nonreg23	6	4	3	*nonreg48*	3	5	1	*nonreg73*	12	11	10	*nonreg98*	12	30	22
nonreg24	1	2	1	*nonreg49*	3	5	2	*nonreg74*	12	11	2	*nonreg99*	12	30	8
nonreg25	3	2	1	*nonreg50*	3	5	3	*nonreg75*	12	25	1	*nonreg100*	12	30	17

The numerical results of the tests are displayed in the Tables 2 and 3. In these tables, the first column contains the NONREGSDP instance's name. The next three columns contain the number of variables, n, the dimension of the constraint matrices, s, and the desired irregularity degree, d, respectively. The fifth column presents the computed irregularity degree, s^*, obtained using the `DIISalg` function. The last columns of the Tables 2 and 3 contain the outputs of the SDP solvers SDPT3 and SeDuMi, respectively, where *iter* is the number of iterations, *time* is the computational time, *val* is the returned optimal value, p^* and d^* are the primal and dual optimal values, respectively, *gap* is the actual duality gap, and *Solver's Report* stands for observations which are (warning) output messages returned by solvers. The symbol $*$ in the last column of the tables means that the solver solved the dual problem to get the solution of the given (primal) SDP problem. The lack of results in the tables correspond to the cases when the solvers were not able to provide such results.

While solving the generated nonregular SDP problems, one of the first observations we can make from the experiments is that the number of warning messages delivered by the SDPT3 solver is quite higher than that by SeDuMi. Another observation is that for these nonregular instances the solvers chose to solve the dual problem instead of the given primal one for almost all tested SDP instances.

Observing the Table 2, we can see that for 7 generated instances the returned value p^* was quite far from the true one, which is zero. In terms of the returned optimal value *val*, we can see that SDPT3 provided wrong values for 13 instances (*i.e.*, NaN - not a number; $-$Inf - unbounded; or values far from the true optimal ones). We can also see that the most accurate optimal value p^* was computed for the problem *nonreg29* with $p^* = 7.0321e{-}14$. However, since the solver has chosen to solve the dual problem, the returned optimal value *val* was $-3.7952e - 7$.

As can be seen from this table, in 19 out of 54 instances the solver SDPT3 returned warning messages related to numerical issues. For all the 18 nonregular SDP instances with $n \geq s$, the solver ran into numerical difficulties and returned wrong solutions or values far from the true optimal values. The exceptions are the problems *nonreg4*, *nonreg6*, *nonreg21* and *nonreg38*, whose computed values can be considered roughly close to (the optimal) zero.

No general assertion about correlation between the level of nonregularity and the number of iterations used by SDPT3 can be made. It may be due to the use of the dual to solve the given problem. However, there are some examples supporting that large values of the irregularity degree correlate well with large number of iterations of the solver (*e.g.*, *nonreg40*$-$*nonreg43*, *nonreg44*$-$*nonreg47*, *nonreg48*$-$*nonreg51*).

From Table 3, it can be observed that SeDuMi reported 5 warning messages about numerical problems on solving the given SDP instances.

Table 2: Numerical results using `DIISalg` and SDPT3 on SDP instances from NONREGSDP (computation time is in seconds).

Problem	n	s	d	`DIISalg`						SDPT3	
				s^*	it	time	val	p^*	d^*	gap	Solver's Report
nonreg1	2	2	1	1	16	0.19	$-1.1775e-3$	$-1.1775e-3$	0.0000	$-1.18e-3$	Solved
nonreg2	3	3	1	1	24	0.24	NaN	$-2.4329e-2$	$8.7691e-9$	$-2.38e-2$	progress is bad; Failed
nonreg3	3	3	2	2	58	0.61	$-$Inf				* primal problem is suspected of being infeasible; Unbounded
nonreg4	4	4	1	1	24	0.26	$-1.7315e-7$	$1.0218e-7$	$1.7315e-7$	$-7.10e-8$	* Solved
nonreg5	4	4	2	2	31	0.35	$-8.7322e-7$	$4.9006e-7$	$8.7322e-7$	$-3.83e-7$	* Inaccurate; Solved
nonreg6	4	4	3	3	34	0.36	$-1.2851e-7$	$7.0894e-8$	$1.2851e-7$	$-5.76e-8$	* Solved
nonreg7	5	4	3	3	26	0.28	$-3.8859e-5$	$1.9393e-5$	$3.8859e-5$	$-1.95e-5$	* Inaccurate; Solved
nonreg8	3	4	2	2	24	0.26	$-1.1372e-6$	$6.4885e-7$	$1.1372e-6$	$-4.88e-7$	* Inaccurate; Solved
nonreg9	6	2	2	2	19	0.31	NaN	$7.3161e-6$	$8.0461e-4$	$-7.97e-4$	* Failed
nonreg10	1	4	2	2	22	0.19	$-1.1464e-7$	$6.6920e-8$	$1.1464e-7$	$-4.77e-8$	* Solved
nonreg11	5	10	1	1	28	0.62	$-2.5503e-8$	$2.3136e-8$	$2.5503e-8$	$-2.37e-9$	* Solved
nonreg12	5	10	2	2	22	0.26	$-7.1756e-7$	$6.4886e-7$	$7.1756e-7$	$-6.87e-8$	* Inaccurate; Solved
nonreg13	5	10	3	3	21	0.26	$-8.7671e-7$	$8.1750e-7$	$8.7671e-7$	$-5.92e-8$	* lack of progress in infeas; Inaccurate; Solved
nonreg14	5	10	4	4	28	0.33	$-2.2118e-8$	$1.5133e-8$	$2.2118e-8$	$-6.98e-9$	* Solved
nonreg15	5	10	5	5	27	0.30	$-2.0518e-8$	$1.5921e-8$	$2.0518e-8$	$-4.60e-9$	* Solved
nonreg16	5	10	6	6	28	0.31	$-2.0014e-8$	$1.4456e-8$	$2.0014e-8$	$-5.56e-9$	* Solved
nonreg17	5	10	7	7	27	0.30	$-1.9767e-8$	$1.3181e-8$	$1.9767e-8$	$-6.59e-9$	* Solved
nonreg18	5	10	8	8	30	0.30	$-3.6391e-8$	$2.1580e-8$	$3.6391e-8$	$-1.48e-8$	* Solved
nonreg19	5	10	9	9	32	0.34	$-2.7487e-8$	$1.4789e-8$	$2.7487e-8$	$-1.27e-8$	* Solved
nonreg20	2	10	1	1	24	0.23	$-2.5293e-8$	$2.3063e-8$	$2.5293e-8$	$-2.23e-9$	* Solved
nonreg21	12	10	1	1	23	0.26	$-3.4148e-8$	$2.9786e-8$	$3.4148e-8$	$-4.36e-9$	* Solved
nonreg22	6	4	1	1	29	0.31	NaN	$-1.2664e-9$	$1.6762e-2$	$-1.65e-2$	* progress is bad; Failed
nonreg23	6	4	3	3	21	0.20	$-6.9621e-5$	$1.8884e-6$	$6.9621e-5$	$-6.77e-5$	* Inaccurate; Solved
nonreg24	1	2	1	1	17	0.25	$-4.6124e-6$	$2.4798e-6$	$4.6124e-6$	$-2.13e-6$	* progress in duality gap has deteriorated; Inaccurate; Solved
nonreg25	3	2	1	1	21	0.25	$-1.4449e-3$	$-1.4449e-3$	0.0000	$-1.44e-3$	progress is bad; Inaccurate; Solved
nonreg26	4	2	1	1	22	0.26	$-$Inf				progress is bad; dual problem is suspected of being infeasible; Unbounded
nonreg27	1	3	1	1	32	0.29	$-2.2120e-7$	$1.2689e-7$	$2.2120e-7$	$-9.43e-8$	* Solved
nonreg28	1	3	2	2	31	0.29	$-5.2713e-7$	$2.8473e-7$	$5.2713e-7$	$-2.42e-7$	* Solved
nonreg29	2	3	1	1	51	0.41	$-3.7952e-7$	$7.0321e-14$	$3.7952e-7$	$-3.80e-7$	* Solved
nonreg30	2	4	1	2	30	0.29	$-3.3806e-7$	$1.8231e-7$	$3.3806e-7$	$-1.56e-7$	* Solved
nonreg31	1	4	1	1	19	0.22	$-5.9690e-8$	$4.1990e-8$	$5.9690e-8$	$-1.77e-8$	* Solved
nonreg32	1	4	3	3	26	0.19	$-4.3577e-7$	$2.4237e-7$	$4.3577e-7$	$-1.93e-7$	* Solved
nonreg33	2	4	1	1	36	0.31	$-2.2030e-7$	$1.2831e-7$	$2.2030e-7$	$-9.20e-8$	* Solved
nonreg34	2	4	2	2	29	0.25	$-1.6806e-7$	$9.5636e-8$	$1.6806e-7$	$-7.24e-8$	* Solved
nonreg35	2	4	3	3	35	0.34	$-1.4537e-7$	$8.9635e-8$	$1.4537e-7$	$-5.57e-8$	* Solved
nonreg36	3	4	1	1	19	0.22	$-3.2653e-8$	$2.6820e-8$	$3.2653e-8$	$-5.83e-9$	* Solved
nonreg37	3	4	3	3	32	0.30	$-3.4000e-7$	$1.8926e-7$	$3.4000e-7$	$-1.51e-7$	* progress is bad; Solved
nonreg38	5	4	1	1	30	0.34	$-2.6551e-7$	$1.5231e-7$	$2.6550e-7$	$-1.13e-7$	* Solved
nonreg39	5	4	2	2	20	0.39	$-1.6548e-5$	$1.1077e-5$	$1.6548e-5$	$-5.47e-6$	* lack of progress in infeas; Solved
nonreg40	1	5	1	1	21	0.23	$-2.0888e-7$	$6.0000e-1$	$6.0000e-1$	$-3.31e-8$	* Solved
nonreg41	1	5	2	2	21	0.18	$-5.4662e-8$	$3.5468e-8$	$5.4662e-8$	$-1.92e-8$	* Solved
nonreg42	1	5	3	3	25	0.30	$-1.1985e-7$	$6.8445e-8$	$1.1985e-7$	$-5.14e-8$	* Solved
nonreg43	1	5	4	4	29	0.23	$-2.8877e-7$	$1.5750e-7$	$2.8876e-7$	$-1.31e-7$	* Solved
nonreg44	2	5	1	1	21	0.21	$-5.5529e-8$	$3.7617e-8$	$5.5529e-8$	$-1.79e-8$	* Solved
nonreg45	2	5	2	2	28	0.29	$-4.9874e-8$	$3.3628e-8$	$4.9874e-8$	$-1.62e-8$	* Solved
nonreg46	2	5	3	3	32	0.34	$-1.3129e-7$	$7.4843e-8$	$1.3129e-7$	$-5.65e-8$	* Solved
nonreg47	2	5	4	4	30	0.20	$-1.7913e-7$	$9.8066e-8$	$1.7913e-7$	$-8.11e-8$	* Solved
nonreg48	3	5	1	1	24	0.26	$-1.4151e-7$	$8.6130e-8$	$1.4151e-7$	$-5.54e-8$	* Solved
nonreg49	3	5	2	2	27	0.27	$-7.2936e-8$	$4.5615e-8$	$7.2936e-8$	$-2.73e-8$	* Solved
nonreg50	3	5	3	3	30	0.32	$-1.1485e-7$	$5.9662e-8$	$1.1485e-7$	$-5.52e-8$	* Solved
nonreg51	3	5	4	4	30	0.32	$-2.4319e-7$	$1.3273e-7$	$2.4319e-7$	$-1.10e-7$	* Solved
nonreg52	7	4	1	1	56	0.56	NaN	$8.0698e-4$	$6.8436e-2$	$-6.33e-2$	* lack of progress in dual infeas; Failed
nonreg53	7	4	2	2	21	0.27	$-1.0291e-3$	$6.4224e-6$	$1.0291e-3$	$-1.02e-3$	* Inaccurate; Solved
nonreg54	7	4	3	3	28	0.42	NaN	$5.6348e-7$	$1.9355e-3$	$-1.93e-3$	* Failed

Table 3: Numerical results using `DIISalg` and SeDuMi on SDP instances from NONREGSDP (computation time is in seconds).

Problem	n	s	d	`DIISalg`			SeDuMi				
				s^*	iter	time	val	p^*	d^*	gap	Solver's Report
nonreg1	2	2	1	1	5	0.30	−1.6672e − 1	−1.6672e − 1	0.0000	−1.67e − 1	Run into numerical problems; Solved
nonreg2	3	3	1	1	25	0.30	−7.1391e − 2	−7.1391e − 2	−1.1324e − 9	−7.14e − 2	Solved
nonreg3	3	3	2	2	25	0.50	−4.5511e − 2	0.0000	4.5511e − 2	−4.55e − 2	∗ Solved
nonreg4	4	4	1	1	23	0.30	−9.1231e − 5	7.4726e − 5	9.1231e − 5	−1.65e − 5	∗ Solved
nonreg5	4	4	2	2	18	0.20	−9.4062e − 5	6.3427e − 5	9.4062e − 5	−3.06e − 5	∗ Solved
nonreg6	4	4	3	3	16	0.20	−5.1708e − 5	3.1247e − 5	5.1708e − 5	−2.05e − 5	∗ Solved
nonreg7	5	4	3	3	20	0.30	−2.6713e − 4	1.6394e − 4	2.6713e − 4	−1.03e − 4	∗ Solved
nonreg8	3	4	2	2	19	0.20	−2.2489e − 5	1.7561e − 5	2.2489e − 5	−4.93e − 6	∗ Solved
nonreg9	6	2	2	2	16	0.40	−1.1971e − 1	1.8696e − 6	1.1971e − 1	−1.20e − 1	∗ Run into numerical problems; Inaccurate; Solved
nonreg10	1	4	2	2	19	0.30	−4.0186e − 5	3.4643e − 5	4.0186e − 5	−5.54e − 6	∗ Solved
nonreg11	5	10	1	1	23	0.50	−2.8726e − 5	1.8988e − 5	2.8726e − 5	−9.74e − 6	∗ Solved
nonreg12	5	10	2	2	22	0.50	−1.4633e − 5	9.3842e − 6	1.4633e − 5	−5.25e − 6	∗ Solved
nonreg13	5	10	3	3	23	0.40	−1.0469e − 5	6.8688e − 6	1.0469e − 5	−3.60e − 6	∗ Solved
nonreg14	5	10	4	4	22	0.30	−4.3341e − 5	2.8588e − 5	4.3341e − 5	−1.47e − 5	∗ Solved
nonreg15	5	10	5	5	22	0.30	−8.1019e − 6	5.0566e − 6	8.1019e − 6	−3.04e − 6	∗ Solved
nonreg16	5	10	6	6	21	0.30	−8.6966e − 6	7.1936e − 6	8.6966e − 6	−1.50e − 6	∗ Solved
nonreg17	5	10	7	7	19	0.20	−9.4683e − 6	8.2214e − 6	9.4683e − 6	−1.25e − 6	∗ Solved
nonreg18	5	10	8	8	20	0.20	−1.0959e − 5	8.8137e − 6	1.0959e − 5	−2.14e − 6	∗ Solved
nonreg19	5	10	9	9	18	0.20	−8.6502e − 6	5.5011e − 6	8.6502e − 6	−3.15e − 6	∗ Solved
nonreg20	2	10	1	1	25	0.40	−1.7112e − 5	1.5546e − 5	1.7112e − 5	−1.57e − 6	∗ Solved
nonreg21	12	10	1	1	24	0.30	−5.0062e − 5	3.6993e − 5	5.0062e − 5	−1.31e − 5	∗ Solved
nonreg22	6	4	1	1	27	0.40	−7.9842e − 2	−1.5632e − 9	7.9842e − 2	−7.98e − 2	∗ Solved
nonreg23	6	4	3	3	19	0.40	−6.6666e − 2	2.0036e − 7	6.6666e − 2	−6.67e − 2	∗ Run into numerical problems; Inaccurate; Solved
nonreg24	1	2	1	1	18	0.30	−1.7451e − 5	1.1614e − 5	1.7451e − 5	−5.84e − 6	∗ Solved
nonreg25	3	2	1	1	26	0.40	−4.1919e − 2	−4.1919e − 2	0.0000	−4.19e − 2	Solved
nonreg26	4	2	1	1	26	0.30	−9.0094e − 3	−9.0094e − 3	0.0000	−9.01e − 3	Solved
nonreg27	1	3	1	1	18	0.20	−5.0650e − 5	4.1677e − 5	5.0650e − 5	−8.97e − 6	∗ Solved
nonreg28	1	3	2	2	17	0.20	−6.5905e − 5	4.7263e − 5	6.5905e − 5	−1.86e − 5	∗ Solved
nonreg29	2	3	1	1	24	0.30	−1.4064e − 2	2.3344e − 9	1.4064e − 2	−1.41e − 2	∗ Solved
nonreg30	2	4	1	2	19	0.30	−1.7670e − 5	1.1010e − 5	1.7670e − 5	−6.66e − 6	∗ Solved
nonreg31	1	4	1	1	20	0.20	−3.2979e − 5	2.9286e − 5	3.2979e − 5	−3.69e − 6	∗ Solved
nonreg32	1	4	3	3	15	0.10	−2.8308e − 5	1.6821e − 5	2.8308e − 5	−1.15e − 5	∗ Solved
nonreg33	2	4	1	1	22	0.30	−1.7365e − 4	1.0958e − 4	1.7365e − 4	−6.41e − 5	∗ Solved
nonreg34	2	4	2	2	19	0.30	−3.2226e − 5	2.7211e − 5	3.2226e − 5	−5.01e − 6	∗ Solved
nonreg35	2	4	3	3	18	20	−8.0299e − 5	5.2231e − 5	8.0299e − 5	−2.81e − 5	∗ Solved
nonreg36	3	4	1	1	20	0.20	−2.5493e − 5	2.2435e − 5	2.5493e − 5	−3.06e − 6	∗ Solved
nonreg37	3	4	3	3	16	0.20	−4.2555e − 5	2.6337e − 5	4.2555e − 5	−1.62e − 5	∗ Solved
nonreg38	5	4	1	1	22	0.30	−4.1273e − 5	3.4208e − 5	4.1273e − 5	−7.06e − 6	∗ Solved
nonreg39	5	4	2	2	18	0.20	−3.7746e − 5	2.4570e − 5	3.7746e − 5	−1.32e − 5	∗ Solved
nonreg40	1	5	1	1	22	0.20	−2.8475e − 5	6.0001e − 1	6.0002e − 1	−1.00e − 5	∗ Solved
nonreg41	1	5	2	2	19	0.20	−1.5858e − 5	1.3866e − 5	1.5858e − 5	−1.99e − 6	∗ Solved
nonreg42	1	5	3	3	20	0.20	−4.7861e − 5	3.9980e − 5	4.7861e − 5	−7.88e − 6	∗ Solved
nonreg43	1	5	4	4	18	0.20	−2.1816e − 5	1.3399e − 5	2.1816e − 5	−8.42e − 6	∗ Solved
nonreg44	2	5	1	1	21	0.20	−3.4246e − 5	3.1213e − 5	3.4246e − 5	−3.03e − 6	∗ Solved
nonreg45	2	5	2	2	20	0.20	−1.4613e − 5	1.2855e − 5	1.4613e − 5	−1.76e − 6	∗ Solved
nonreg46	2	5	3	3	19	0.20	−7.0759e − 5	5.4775e − 5	7.0759e − 5	−1.60e − 5	∗ Solved
nonreg47	2	5	4	4	16	0.20	−1.7131e − 5	1.1590e − 5	1.7131e − 5	−5.54e − 6	∗ Solved
nonreg48	3	5	1	1	23	0.20	−3.6285e − 5	3.2545e − 5	3.6285e − 5	−3.74e − 6	∗ Solved
nonreg49	3	5	2	2	19	0.20	−3.3603e − 5	2.9011e − 5	3.3603e − 5	−4.59e − 6	∗ Solved
nonreg50	3	5	3	3	20	0.20	−6.7126e − 5	5.1655e − 5	6.7126e − 5	−1.55e − 5	∗ Solved
nonreg51	3	5	4	4	18	0.20	−1.8706e − 5	1.1907e − 5	1.8706e − 5	−6.80e − 6	∗ Solved
nonreg52	7	4	1	1	29	0.40	−2.2805e − 1	7.0992e − 9	2.2805e − 1	−2.28e − 1	∗ Solved
nonreg53	7	4	2	2	24	0.50	−3.0310e − 2	5.2935e − 8	3.0310e − 2	−3.03e − 2	∗ Run into numerical problems; Inaccurate; Solved
nonreg54	7	4	3	3	17	0.40	−2.8461e − 1	1.3761e − 6	2.8461e − 1	−2.85e − 1	∗ Run into numerical problems; Inaccurate; Solved

While the results provided by SDPT3 permit to consider many of them to be rather close to the true optimal value, notice that the results from SeDuMi can not be considered so good. Moreover, SeDuMi had never reported that it failed to solve some instances. A closer analysis on the results presented in the Table 3 permits to conclude that there are significant discrepancies between the computed optimal values and the true ones, even when the solver has reported "Solved". See, for example, the problems *nonreg2*, *nonreg3*, *nonreg7*, *nonreg22*, *nonreg25*, *nonreg26*, *nonreg29*, *nonreg33*, *nonreg52*. Notice that the closest value to zero in *val* is $-8.1019e-6$ for the problem *nonreg15*.

Regarding the computed value for p^*, only for the problem *nonreg3* SeDuMi had returned zero, and for almost all other instances, the computed optimal values are fairly far from the true ones. The closest value to zero corresponds to the problem *nonreg22*. Based on the results presented in the Table 3, there is no empirical evidence that there exists some correlation between the level of nonregularity and the number of iterations, or computational time spent by SeDuMi.

It is worth mentioning that in both tables, the problem *nonreg40* is particularly nasty, since both solvers behaved poorly, returning similar values for p^* and d^* (close to 0.6), and a different optimal value *val* of the given problem, which should be zero.

The following table summarizes the results obtained in this section.

Table 4: Summary of computational behaviour of SDPT3 and SeDuMi evaluated on 54 instances from NONREGSDP.

Solver	Primal problem solved	Dual problem solved	Accurate solutions 10^{-4}	Accurate solutions 10^{-6}	Accurate solutions 10^{-8}	Report of failures	Report of warnings
SDPT3	4	50	47	41	2	5	19
SeDuMi	4	50	47	6	4	0	5

Based on the numerical results presented in this section, we can conclude that they support the conclusion that standard SDP solvers applied to nonregular problems may be unable to provide accurate solutions.

7 Conclusions

In this paper, we presented an algorithm for generating nonregular SDP instances with a pre-specified irregularity degree. We have implemented this algorithm in MATLAB by the function `nonregSDPgen`. The routine `nonregSDPgen` is very simple to use and returns a dat-s file containing the generated nonregular SDP instance, that in turn can be used as input in popular SDP solvers. By construction, all the generated instances are feasible and have optimal value equal to zero. We have generated nonregular SDP instances and formed a new SDP database with nonregular SDP problems called NONREGSDP. The NONREGSDP library is described in the Table 1. This collection of nonregular SDP test problems was used to evaluate the performance and robustness of two popular SDP solvers.

The numerical experiments showed that the tested SDP solvers do not have a reliable behaviour on nonregular SDP instances. Although SeDuMi uses a self-dual embedding technique to regularize the nonregular problem, many examples showed that it may still return inaccurate solutions.

It should be noticed that it was not the aim of the paper to compare or test the efficiency of SDP solvers. We used two popular SDP solvers, SDPT3 and SeDuMi to analyse the solutions of nonregular SDP problems, and the testes showed that these solvers have not succeeded to find accurate solutions in many cases.

This work reinforces the needed of developing new SDP methods/solvers particularly suitable for nonregular problems, that is failing the Slater condition. Our future work will be dedicated to such a study based on the new CQ free optimality conditions for SDP problems formulated in [12]. Comparison of SDP optimization software on the basis of advanced metrics such as number of function evaluation, ratio of one solver's runtime to the best runtime and performance profiles (see [5] and the references therein) can be another interesting topic of study. To fulfill such comparison, one use the collections of benchmark SDP problems, including the NONREGSDP library.

Acknowledgment

The authors would like to thank the anonymous referees for their suggestions and valuable comments that have helped to improve the paper. This work was supported by Portuguese funds through the CIDMA - Center for Research and Development in Mathematics and Applications, and the Portuguese Foundation for Science and Technology (FCT - Fundação para a Ciência e a Tecnologia), within project UID/MAT/04106/2013.

References

[1] Anjos, M.F. and Lasserre, J.B. (Eds.), *Handbook of Semidefinite, Conic and Polynomial Optimization: Theory, Algorithms, Software and Applications*, International Series in Operational Research and Management Science, **166**, Springer, 2012.

[2] Borchers, B., *SDPLIB 1.2, A library of Semidefinite Programming Test Problems*, Optimization Methods and Software, **11**(1-4), pp. 683-690, 1999.

[3] Cheung, Y., Schurr, S. and Wolkowicz, H., *Preprocessing and Reduction for Degenerate Semidefinite Programs*, Computational and Analytical Mathematics Springer Proceedings in Mathematics & Statistics, **50**, pp. 251-303, 2013.

[4] CVX Research, Inc., *CVX: Matlab Software for Disciplined Convex Programming, version 2.0*, `http://cvxr.com/cvx`, August, 2012.

[5] Dolan, Elizabeth D. and Moré, Jorge J., *Benchmarking optimization software with performance profiles*, Math. Program., Ser. A, **91**, pp. 201-213, 2002.

[6] Freund, R.M., Ordóñez, F. and Toh, K.C., *Behavioral Measures and their Correlation with IPM Iteration Counts on Semi-Definite Programming Problems*, Math. Programming, **109**(2), pp. 445-475, 2007.

[7] Gruber, G., Kruk, S., Rendl, F. and Wolkowicz, H., *Presolving for Semidefinite program without Constraint Qualifications*, Technical Report CORR 98-32, University of Waterloo, Waterloo, Ontario, 1998.

[8] Gruber, G. and Rendl, F., *Computational Experience with Ill-Posed Problems in Semidefinite Programming*, Computational Optimization and Applications, **21**, pp. 201-212, 2002.

[9] Hernández-Jiménez, B., Rojas-Medar, M.A., Osuna-Gómez, R., Beato-Moreno, A., *Generalized convexity in non-regular programming problems with inequality-type constraints*, J. Math. Anal. Appl., **352**, pp. 604-613, 2009.

[10] Jansson, C., Chaykin, D. and Keil, C., *Rigorous Error Bounds for the Optimal Value in SDP*, SIAM Journal on Numerical Analysis, **46**(1), pp. 180-200, 2007.

[11] Klerk, E. de, *Aspects of Semidefinite Programming - Interior Point Algorithms and Selected Applications*, Applied Optimization, **65**, Kluwer, 2004.

[12] Kostyukova, O.I. and Tchemisova, T.V., *Optimality Criterion without Constraint Qualification for Linear Semidefinite Problems*, Journal of Mathematical Sciences, Springer US, **182**(2), pp. 126-143, 2012.

[13] Liu, M. and Pataki, G., *Exact duals and short certificates of infeasibility and weak infeasibility in conic linear programming*, Math. Program., Ser. A, pp. 1-46, 2017.

[14] Macedo, E., *Testing Regularity on Linear Semidefinite Optimization Problems*, In: Almeida, J.P., Oliveira, J.F. and Pinto, A.A. (eds.) Operational Research, CIM Series in Mathematical Sciences, Springer, 4, pp. 213-236, 2015.

[15] Macedo, E., *Numerical study of regularity in Semidefinite Programming and applications*, PhD Thesis, University of Aveiro, Portugal, 2016.

[16] Pataki, G., *Bad semidefinite programs: they all look the same*, SIAM J. OPTIM., **27**(1), pp. 146-172, 2017.

[17] Polik, I. and Terlaky, T., *New stopping criteria for detecting infeasibility in conic optimization*, Springer, Optim. Lett., **3**(2), pp. 187-198, 2009.

[18] Solodov, M.V., *Constraint Qualifications*, Encyclopedia of Operations Research and Management Science, James J. Cochran, et al. (editors), John Wiley & Sons, Inc., 2010.

[19] Sturm, J.F., *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software, **11**, pp. 625-653, 1999.

[20] Tutuncu, R.H, Toh, K.C. and Todd, M.J., *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathematical Programming Ser. B, **95**, pp. 189-217, 2003.

[21] Waki, H., Nakata, M. and Muramatsu, M., *Strange behaviors of interior-point methods for solving semidefinite programming problems in polynomial optimization*, Computational Optimization and Applications, **53**(3), Springer, pp. 823-844, 2012.

[22] Wei, H. and Wolkowicz, H., *Generating and measuring instances of hard semidefinite programs*, Math. Program., **125**(1), Ser. A, pp.31-45, 2010.

[23] Wolkowicz, H., Saigal, R. and Vandenberghe, L., *Handbook of semidefinite programming: theory, algorithms, and applications*, Kluwer Academic Publishers, Boston, 2000.

[24] Yamashita, M., Fujisawa, K. and Kojima, M., *Implementation and evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0)*, Optimization Methods and Software **18**, pp. 491-505, 2003.