Universidade de Aveiro Departamento de Matemática
2016

Eloísa Catarina Monteiro de Figueiredo Amaral e Macedo

Estudo Numérico de Regularidade em Programação Semidefinida e Aplicações

Numerical Study of Regularity in Semidefinite Programming and Applications

Universidade de Aveiro Departamento de Matemática
2016

Eloísa Catarina Monteiro de Figueiredo Amaral e Macedo

Estudo Numérico de Regularidade em Programação Semidefinida e Aplicações

Numerical Study of Regularity in Semidefinite Programming and Applications

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Matemática, realizada sob a orientação científica da Doutora Tatiana Tchemisova Cordeiro, Professora Auxiliar do Departamento de Matemática da Universidade de Aveiro.

acknowledgements / agradecimentos

First and foremost, I would like to express my thanks to my supervisor, Professor Tatiana Tchemisova, for her support during this journey. My gratitude extends to Professors Adelaide Freitas and Raquel Pinto, who helped me improve this work.
I am also indebted to the University of Aveiro, in particular Professor João Santos, for the opportunity of pursuing my dream of graduate studies, by granting me tuition exemption, and CIDMA (Center for Research & Development in Mathematics and Applications) for all the support to invest in my formation and participation in national and international conferences. Thanks also EURO (Association of European Operational Research Societies) for their bursary, allowing me to attend a Convex Optimization Course.
Thanks to the professors and colleagues with whom I had the pleasure of working at the Department of Mathematics of the University of Aveiro.
Special thanks are due to my friends Cristina, Elisabete, Isabel, Neusa, Paula, Sandra, Teresa, Tita and Milé, who are my company for countless good or bad days.
I would also like to express my warm thanks to Rui for his love, support and patience along these years. Last, but not least, I owe great thanks to my parents, Gabi and Albanito, for their love and continuous encouragement. They are a constant source of inspiration throughout my life. This thesis is dedicated to them.

palavras-chave

Programação semidefinida (SDP), regularidade, qualificações de restrições (CQ), condições de optimalidade, análise de dados, clusterização.

resumo

Esta tese é dedicada ao estudo de regularidade em programação semidefinida (SDP - semidefinite programming), uma importante área da optimização convexa com uma vasta gama de aplicações. A teoria de dualidade, condições de optimalidade e métodos para SDP assentam em certos pressupostos de regularidade que nem sempre são satisfeitos. A ausência de regularidade, isto é, não regularidade, pode afetar a caracterização da optimalidade de soluções e os solvers podem apresentar dificuldades numéricas, conduzindo a resultados pouco fiáveis.

Existem diferentes noções associadas a regularidade. Nesta tese, estudamos em particular, os conceitos de problemas bem-postos, bem comportados e condições de qualificação de restrições (CQ - constraint qualifications), bem como as relações entre eles. Uma das CQs mais utilizadas em SDP é a condição de Slater. Esta condição garante que as condições de optimalidade de primeira ordem, conhecidas como condições de Karush-Kuhn-Tucker, estão satisfeitas. Os solvers atuais não verificam se um problema a resolver satisfaz a condição de Slater, mas trabalham nesse pressuposto. Desenvolvemos e implementamos em MATLAB procedimentos numéricos para verificar se um dado problema de SDP é regular em termos da condição de Slater e determinar o grau de irregularidade no caso de problemas não regulares. Os resultados das experiências numéricas apresentados neste trabalho mostram que os procedimentos propostos são eficientes e confirmam as conclusões obtidas sobre a relação entre a condição de Slater e outras noções de regularidade.

Outra contribuição da tese consiste no desenvolvimento e na implementação em MATLAB de um procedimento numérico para gerar problemas de SDP não regulares com um determinado grau de irregularidade. A coleção de problemas não regulares construidos usando este gerador é de acesso livre e permite testar novos métodos e solvers para SDP.

Uma outra contribuição desta tese está relacionada com uma aplicação de SDP em análise de dados. Consideramos um modelo de SDP não linear, bem como as suas relaxações lineares para problemas de clusterização, e estudamos a sua regularidade. Mostramos que o modelo não linear é não regular, enquanto que as suas relaxações são regulares. Sugerimos um algoritmo baseado em modelos de SDP para resolver problemas de clusterização e redução de dimensionalidade, e implementámo-lo em R. Os testes numéricos usando vários conjuntos de dados confirmam a rapidez e eficiência deste procedimento numérico.

keywords Semidefinite programming (SDP), regularity, constraint qualifications (CQ), optimality conditions, data analysis, clustering.

abstract This thesis is devoted to the study of regularity in semidefinite programming (SDP), an important area of convex optimization with a wide range of applications. The duality theory, optimality conditions and methods for SDP rely on certain assumptions of regularity that are not always satisfied. Absence of regularity, *i.e.*, nonregularity, may affect the characterization of optimality of solutions and SDP solvers may run into numerical difficulties, leading to unreliable results.

There exist different notions associated to regularity. In this thesis, we study in particular, well-posedness, good behaviour and constraint qualifications (CQs), as well as relations among them. A widely used CQ in SDP is the Slater condition. This condition guarantees that the first order necessary optimality conditions in the Karush-Kuhn-Tucker formulation are satisfied. Current SDP solvers do not check if a problem satisfies the Slater condition, but work assuming its fulfilment. We develop and implement in MATLAB numerical procedures to verify if a given SDP problem is regular in terms of the Slater condition and to determine the irregularity degree in the case of nonregularity. Numerical experiments presented in this work show that the proposed procedures are quite efficient and confirm the obtained conclusions about the relationship between the Slater condition and other regularity notions.

Other contribution of the thesis consists in the development and MATLAB implementation of an algorithm for generating nonregular SDP problems with a desired irregularity degree. The database of nonregular problems constructed using this generator is publicly available and can be used for testing new SDP methods and solvers.

Another contribution of this thesis is concerned with an SDP application to data analysis. We consider a nonlinear SDP model and linear SDP relaxations for clustering problems and study their regularity. We show that the nonlinear SDP model is nonregular, while its relaxations are regular. We suggest a SDP-based algorithm for solving clustering and dimensionality reduction problems and implement it in R. Numerical tests on various real-life data sets confirm the fastness and efficiency of this numerical procedure.

"If you live each day as it was your last,
someday you'll most certainly be right..."

– *Steve Jobs*

Contents

List of Tables

Chapter 1

Introduction

1.1 Overview and motivation

Convex optimization deals with problems of minimizing a convex objective function over a convex set. Semidefinite programming (SDP), which refers to minimizing a linear function subject to linear matrix inequalities (LMIs), is an important area of convex optimization. SDP problems can also be considered as conic optimization problems, since they consist in optimizing a linear function over the intersection of an affine space and a closed convex cone. Sometimes, SDP is seen as a generalization of linear programming (LP), where the vector of variables is replaced by a matrix.

SDP is an active area of research mostly due to its many applications in mechanical and electrical engineering, combinatorial optimization, robust optimization, computational biology, quantum chemistry, atomic physics, structural optimization, approximation theory, systems and control theory, circuit design, sensor network location, signal processing, and data analysis, among others (see, *e.g.*, the surveys [8, 69, 146, 159]). Besides applications, there are other reasons for the increasing interest in SDP. Many convex optimization problems can be reformulated as SDP ones. Moreover, SDP relaxations of nonconvex optimization problems provide, in general, good approximations. The SDP models are specially attractive since there exist efficient methods for solving them in polynomial-time [8, 69].

The methods for solving SDP problems are based on optimality conditions. The most known and widely used optimality conditions for SDP are the first-order necessary optimality conditions, also called Karush-Khun-Tucker-type (KKT) conditions [159]. The KKT optimality conditions are usually derived under some special assumptions on the feasible set of the problem. These assumptions are called constraint qualifications (CQs) or regularity conditions [13, 55, 69, 159]. CQs play also an important role in deriving duality relations, sensitivity/stability analysis and convergence of computational methods [68]. One such CQs widely used in SDP is the Slater regularity condition. There exist other notions associated to regularity, such as well-posedness and good behaviour. The regularity of an optimization problem is specially important to derive optimality conditions, guarantee the efficiency of numerical methods and stability of the solution. In practice, the regularity conditions may be difficult to verify. Although conditions of regularity are known in optimization, and in particular, in SDP, some are often used in a very general sense, not permitting to verify them in a rigorous way. Sometimes, one regularity notion is substituted by another, nevertheless the relationship between regularity notions are not well established.

The most efficient SDP methods are based on generalizations of the interior point method, firstly proposed by Karmarkar [67] for LP. Current popular SDP methods are the primal-dual interior point methods, which require regularity akin to the Slater condition of both primal and dual problems. Although in [31] it is shown that the Slater condition holds generically for linear conic problems, including the SDP ones, in practice, there exist many SDP instances for which the Slater condition fails to hold (*e.g.*, [23, 38, 49, 61, 152]). For nonregular SDP problems, the interior point methods, as well as the majority of other known methods, can not be applied, standard SDP solvers may run into difficulties and return solutions that can be far from the true optimal ones. Although in these cases some special techniques can be applied, the truth is that, in practice, the SDP solvers may still run into numerical difficulties. Therefore, the topic on (ir)regularity, strange or bad behaviour of SDP problems on methods is timely. It is essential to develop numerical procedures, that can be considered as presolving tools, for verifying the regularity in some sense of a given problem, in order to guarantee the reliability of results. For testing and implementing such procedures, or new stopping criteria and methods, it is important to have libraries of nonregular SDP problems, or procedures that permit to generate nonregular SDP problem instances.

Over the past few years, new SDP applications to data analysis have appeared, *e.g.*, in computational biology, involving large data sets. This usually leads to large-scale problems. The SDP models are often nonlinear and very difficult to solve, but linear relaxations and methods based on heuristics revealed to be very efficient on solving such problems.

This thesis is three-fold. First, we study regularity of SDP problems from both theoretical and numerical viewpoints, and suggest a numerical procedure for testing the Slater condition. Second, we develop a generator of nonregular SDP instances failing the Slater condition. The third topic covered in this thesis is related to a SDP application to data analysis in particular to clustering and dimensionality reduction problems.

1.2 Literature review

The SDP history can be traced back to the work of Bellman and Fan [11], in 1963. It seems to be the earliest work on SDP theory, where a SDP problem and the associated dual were formulated, optimality conditions were described and certain results on duality were established. It was already shown in this work that regularity of problems is needed to prove strong duality in SDP. Since that time, the theory and methods of SDP were actively developing, and many papers and books have appeared. Among numerous surveys and books dedicated to SDP, its theory, applications and algorithms, we can mention [47, 69, 70, 83, 140, 146] and [8, 42, 159]. The connection of SDP to other areas of convex optimization, such as LP and Semi-Infinite Programming (SIP), is studied in, *e.g.*, [79, 80, 147]. Many SDP applications are reviewed in, *e.g.*, [8, 28, 43, 69, 86, 140, 141, 146, 149, 159]. SDP has emerged as a powerful tool in combinatorial optimization relaxations of NP-hard problems. One of the most widely known combinatorial problems is the max-cut problem. In [43], it was shown that a SDP relaxation can provide good approximations for this problem and the randomized SDP-based approximation algorithm was suggested. Since then, SDP models has been successfully used in the development of approximation algorithms for several classes of hard combinatorial optimization problems [83].

Establishing optimality conditions is fundamental for solving any optimization problem. Optimality conditions for SDP are studied in, *e.g.*, [13, 24, 36, 69, 75, 158, 159]. The KKT conditions provide first-order necessary optimality conditions to characterize the optimality of a feasible solution. These optimality conditions are formulated under assumption that the constraints of the problem satisfy certain regularity conditions, so-called CQs [13, 55, 69, 159]. For convex SDP problems, the KKT conditions are proved to be both necessary and sufficient optimality conditions [13, 69, 159].

Regularity plays an important role in deriving duality relations, sensitivity/stability analysis and convergence of computational methods [68, 134]. An optimization problem is usually considered to be regular if certain CQ is satisfied [55], and nonregular, otherwise. The Slater condition, also called Slater CQ or Slater regularity condition [69], is widely used in SDP and many authors assume in their studies that this condition holds (see, *e.g.*, [23, 69, 80, 126, 159]). Besides the Slater condition, there exist other CQs, such as the Robinson and the Mangasarian-Fromovitz CQs, and some works include indication on their relationships (*e.g.*, [97, 99, 132, 134, 159]). Many algorithms for convex optimization, in particular SDP, are based on solving the KKT conditions, and thus, rely on the assumption that the Slater condition holds.

Recently, much attention has been devoted to the failure of regularity in SDP, and specially with respect to the Slater condition, *e.g.*, [37, 38, 49, 61, 111, 152, 155]. When the Slater condition does not hold, the KKT optimality conditions may fail to characterize optimality of a feasible solution [23, 78, 156]. Therefore, there has been an increasing interest in studying new optimality conditions that do not require CQs, called CQ-free optimality conditions (see, *e.g.*, [48, 62, 63, 78, 125, 126]).

In [62], a new approach to closing duality gaps for SDP without the Slater condition is presented, strong duality results are derived under a general condition weaker than the Slater CQ, and a sequential form of optimality conditions without CQs are derived. No computational tests to illustrate the properties of this approach were reported. In [78], new CQ-free optimality conditions for linear SDP problems were proposed. These conditions are based on the notion of immobile index subspace for SDP problems and are proved to be efficient in the cases when the KKT optimality conditions can not be applied since the Slater condition fails to hold, and coincide with the KKT optimality conditions in the regular cases. Nevertheless these optimality conditions are proved to be equivalent to that in [48, 125, 126], they are more constructive, since an algorithm proposed in [78] for constructing the immobile index subspace permits to formulate these optimality conditions in the explicit form.

A common approach to deal with SDP problems failing the Slater condition is to fulfil a preprocessing or regularization technique [22, 47, 69]. The idea is to construct equivalent SDP problems satisfying the Slater condition. In [48], a presolving step to regularize SDP problems failing the Slater condition via minimal faces of a closed convex cone was proposed. Recently, in 2013, the paper of Cheung, Schurr and Wolkowicz [23] presented a backward stable preprocessing technique for SDP problems for which the Slater condition fails. The regularization procedure described in this paper applies the Borwein-Wolkowicz facial reduction process [17] and permits to reduce the SDP problem into a smaller one satisfying the Robinson condition. This involves finding the minimal face of the semidefinite cone that contains the feasible set of the SDP problem. In [69], an initialization strategy for obtaining a strictly feasible solution of a SDP problem, called self-dual embedding, was described. The idea of a self-dual embedding technique is to transform the SDP problem into a larger one, by embedding the primal problem with its dual. The resulting (larger) SDP problem satisfies the Slater condition and an initial solution is known [42, 69, 120]. Therefore, the failure of the Slater condition in a given SDP should not be an issue for the self-dual embedding technique.

In [31], it was shown that the Slater condition holds generically for linear conic programs, including SDP. This means that the set of SDP problem instances failing the Slater condition has measure zero [5, 31]. However, this interesting theoretical result does not mean that any given SDP problem should satisfy the Slater condition. In practice, there are several SDP instances for which the Slater condition fails to hold and many authors are drawing their attention to the study of theoretical and numerical difficulties that can occur in such cases (see, *e.g.*, [23, 37, 38, 49, 61, 111, 152, 155]). In the case of nonregular SDP problems, a nonzero duality gap can exist, and/or the dual (or primal) optimal value may not be attained, one of the primal or dual problems may be feasible and bounded while the other can be infeasible [111, 112, 126, 156]. In [152], it is noticed some *strange behaviours* of SDP solvers when the Slater condition fails to hold for at least one of the primal or dual problems. It is also pointed out that in these cases there exists a significant "discrepancy between the true and the computed optimal values". Therefore, it is very important to develop special procedures for testing the Slater condition on SDP problems. Since standard SDP methods require the Slater condition to hold, it is important to check if it is satisfied before solving the problem. According to [142], "in terms of worst-case performance, deciding whether Slater condition holds for a given SDP problem seems no easier than solving an SDP problem". This is one of the reasons why no efficient numerical procedure to verify the Slater condition has been proposed till now.

In the literature, other notions associated to regularity are often used, such as well-posedness and good behaviour of optimization problems. Well-posedness is in general, related to stability of the optimal solutions [29, 69, 77, 138]. In [77], different notions of well-posedness (in particular, the well-posedness in the sense of Hadamard, Tikhonov and Levitin-Polyak, and strong well-posedness) of convex problems are studied and compared. It is shown that under the Slater condition, Hadamard's well-posedness is equivalent to that of Tikhonov. Theoretical study of well-posedness of optimization problems can be rather difficult and its practical verification is not always easy. The problems that are not well-posed are called ill-posed. Ill-posed problems are quite common in applications and, according to [61], may occur due to the lack of precise mathematical formulations. With respect to the SDP case, “a feasible problem that does not satisfy the Slater condition is ill-posed in the sense that an arbitrary small perturbation of the problem can change its status from feasible to infeasible” [69]. In [127], the Renegar's condition number is defined as the scale-invariant reciprocal of the smallest data perturbation that will render the perturbed problem primal or dual infeasible. The Renegar's condition number describes the sensitivity of the problem and is used to test well-posedness. It is mentioned in [38, 59] that the computation of the Renegar's condition number can be rather expensive. In [38, 61], different characterizations of well-posedness of SDP problems were proposed. In the paper of Freund, Ordóñez and Toh [38], a numerical approach to characterization of the well or ill-posedness based on estimating lower and upper bounds of the Renegar's condition number for a SDP problem was described. It is based on solving several auxiliary SDP problems, in structure and size compatible with the original primal and dual SDP problems. In [61], another approach to characterization of the well or ill-posedness based on rigorous upper bounds for the optimal values of SDP problems was proposed. It is shown that such upper bounds are related to the Renegar's condition number and it is proposed an algorithm for computing rigorous upper bounds. This approach is constructive and based on obtaining rigorous upper (and lower) bounds and error bounds for the optimal values, by properly postprocessing the output of a SDP solver. The main feature of this approach is that it uses interval arithmetic and the computation of the rigorous bounds takes into account all rounding errors and possible small errors presented in the input data.

More recently, Pataki introduced in [111] the notion of good behaviour of a SDP problem and presented characterizations of well and badly-behaved problems from the standpoint of duality, showing that it is important to verify if a given problem is well or badly-behaved in order to avoid numerical difficulties. The proposed characterizations of good or bad behaviour may be difficult to apply on a given SDP problem, since they are based on special reformulations of the problems. The presented standard SDP reformulations have a strictly feasible block, and some vanishing variables. Such reformulations involve using a sequence of operations on the SDP system of constraints, such as deletion of some rows and corresponding columns, rotation and contraction of all constraint matrices. In [111], it is stated that “it is nontrivial to prove that the standard reformulation exists”.

For a long time, the most efficient method for solving SDP problems was the ellipsoid method [46]. However, its performance revealed to be very slow in practice, which has motivated the research community to seek more efficient methods. The success of interior point methods in LP has stimulated their application and generalization in SDP. The special structure of SDP programs turned possible their efficient solution by interior point methods. In the 1990s, the works of Nesterov and Nemirovski [107, 108], and Alizadeh [4] were extremely important in developing the theory of interior point methods for SDP [103]. In [107] and [108], interior point methods based on the concept of self-concordant barrier functions were extended for general convex programming problems. In [4], it was proposed an interior point algorithm for SDP based on an extension of the Ye's projective potential reduction method for LP. In [145], Vandenberghe and Boyd proposed a primal-dual potential reduction method for convex optimization problems involving LMIs, based on the theory developed by Nesterov and Nemirovski, and presented some numerical experiments on SDP problems arising from control theory. In 1996, Helmberg, Rendl, Vanderbei and Wolkowicz [54] developed a primal-dual interior point algorithm for SDP and showed its applicability to max-cut and min-max eigenvalue problems. Since then, primal-dual interior point methods become the leader choice for SDP [69, 120].

The primal-dual interior point methods attempt to solve both primal and dual SDP problems, in order to minimize the duality gap, and rely on assumptions of regularity, *i.e.*, it is assumed that both primal and dual problems satisfy the Slater condition [69]. These methods search for a primal-dual pair of optimal solutions that satisfy the KKT optimality conditions. This is done by using at each iteration a modified Newton method in the region where the matrix variables are positive definite [161]. The idea is to consider a primal-dual pair of SDP problems perturbed by a barrier parameter. Assuming that the Slater condition holds for both, the KKT optimality conditions are formulated for the perturbed SDP problems, resulting in a perturbed system of nonlinear equations. To get the solution of the perturbed system, a damped Newton method is often applied in conjunction with a reduction of the barrier parameter toward zero [8]. Forming and solving the system of equations at each iteration can be the most time-consuming parts of any primal-dual interior point method, in which Cholesky factorizations are usually performed [161].

Over the past few years, other algorithms have been proposed for SDP, such as, augmented Lagrangian methods [20, 21, 135, 165], bundle methods [53], new Newton-type methods [66], modified barrier methods [71], filter-trust-region methods [57], and methods based on quadratic regularization and augmented Lagrangian techniques [96]. In [37], a method with no regularity assumption in terms of the Slater condition was proposed to solve SDP problems. However, there is no report on its implementation and numerical results.

Actually, the most popular methods for solving SDP problems are the primal-dual interior point methods [8, 42] and current efficient SDP solvers, such as CSDP [14], SDPA [162], SDPT3 [143], and SeDuMi [136], are based on variants of these methods. The interior point methods provide, in general, accurate solutions for problems of moderate size. There are however some drawbacks, namely, in terms of (in)efficiency on solving large-scale and nonregular SDP problems [42, 152, 161]. The absence of the Slater condition may affect the performance of standard algorithms, which may present difficulties on their convergence and efficiency, and coupled with the use of floating point arithmetic, wrong solutions may be produced and the solvers may run into numerical difficulties (see, *e.g.*, [23, 37, 152, 155]), even when the self-dual embedding technique is implemented in the SDP solver, as in SeDuMi [23, 42].

The SDPLIB [15] is a well known collection of SDP instances that is often used to test and develop new methods or solvers. In [122], it was mentioned that it would be important to have a library of infeasible SDP instances. In [85], an algorithm for generating infeasible SDP instances is presented and in [155], a generator of *hard* SDP instances, for which strict complementary fails, is proposed. In this light, a generator of nonregular SDP instances failing the Slater condition with predefined properties would also be useful.

SDP has many applications that can often result in large-scale problems, such as those arising from quantum chemistry, sensor network localization and data analysis [161]. To handle large-scale problems, some parallel versions of current SDP solvers have been recently proposed, such as SDPARA and a parallel version of CSDP [8, 16, 161], but to benefit of the full capabilities of these solvers, one should make use of multiprocessor (with multicore architecture) computers. When the computational resources are limited, other methods and strategies should be developed. For example, approximation algorithms based on some SDP relaxations have been suggested for solving specific and possibly large-scale problems in data analysis.

Data analysis is specially important for obtaining meaningful information hidden on (large) data sets. There exist various data analysis techniques, such as principal component analysis and clustering. In [115, 116], an application of nonlinear SDP models to clustering problems and an approximation algorithm for solving such models were proposed. The algorithm is based on linear SDP relaxations and on a rounding procedure that uses some heuristic to obtain a feasible solution for the nonlinear model. It revealed to be very efficient for the tested data sets in [116]. Another approach to solve clustering problems based on the nonlinear SDP model from [115, 116] was proposed in [81]. The main idea is to obtain a low-rank SDP model and use a nonconvex optimization algorithm to solve it. The analysis of large data sets can be made easier by considering clustering and dimensionality reduction models. The study of regularity of such models is important to guarantee reliable results. In spite of there is no efficient SDP method to solve clustering and dimensionality reduction problems, an SDP-based approximation algorithm may be developed.

1.3 Aims and contributions

The purpose of this thesis is to study and classify different notions of regularity of linear SDP problems, establish the relationships among them, and investigate the numerical procedures that permit to state the regularity of a given SDP problem. The main aims of the research are:

- to study different notions of regularity of linear SDP problems;
- to establish relationships between different notions of regularity;
- to develop, implement and test presolving numerical procedures to verify regularity of SDP problems from the viewpoint of the fulfilment of the Slater condition;
- to develop and implement a generator of nonregular SDP problem instances and create a database of linear SDP problems failing the Slater condition;
- to develop, implement and test a numerical procedure based on SDP models to solve problems of data analysis.

The contributions of this work are as follows:

- establishment of the relationships among the regularity notions in SDP, particularly, between the Slater condition, well-posedness and good behaviour;
- the numerical procedure SDPreg and its MATLAB implementation by the routine `SDPreg` to verify the fulfilment of the Slater condition in linear SDP problems;
- the adaptation of the theoretical DIIS algorithm into the numerical procedure DIISalg and its MATLAB implementation by the routine `DIISalg` to compute the irregularity degree of SDP problems;
- results of testing numerically problems from the SDPLIB database in terms of the fulfilment of the Slater condition and computation of their irregularity degrees with the developed tools;
- the generator of nonregular SDP problem instances and its MATLAB implementation by the routine `nonregSDPgen`;
- the NONREGSDP database of nonregular linear SDP problems with different irregularity degrees;
- the Two-Step-SDP algorithm for solving clustering and dimensionality reduction problems and its implementation in `R` by the routine `TwostepSDPClust`.

1.4 Structure of the thesis

The thesis is organized as follows. In Chapter 2, we present notation and basic definitions and make a brief introduction to SDP, where primal-dual formulations, duality results and optimality conditions are presented. Moreover, this chapter contains an overview of the standard primal-dual interior point methods for solving SDP problems and a brief survey of SDP solvers. We also point out the current limitation of SDP solvers in terms of working only under the assumption of regularity of the problems. We finish this chapter by presenting an example of a SDP problem for which SDP solvers run into numerical difficulties, because of lack of regularity. Chapter 3 is devoted to the study of regularity in SDP. We study different notions usually associated to regularity in SDP, their relations, as well as procedures to test them. We describe the DIIS algorithm from [78] and on its basis develop a new procedure to test the fulfilment of the Slater condition on a given SDP problem. Another procedure, yet more complete, is proposed. It includes computation of the irregularity degree of the SDP problem for which the Slater condition does not hold. The computational experiments are presented in Chapter 4. We describe two presolving numerical tools to check regularity on a SDP problem in terms of the fulfilment of the Slater condition. Implementation details are provided and extensive numerical experiments are carried out in order to test the efficiency of the proposed procedures[1]. In Chapter 5, we develop a generator of nonregular SDP problem instances, which constructs SDP instances failing to satisfy the Slater condition with a prescribed irregularity degree and where the true optimal value is known. The NONREGSDP database of nonregular SDP test problems is created and used in our tests. Numerical experiments with different SDP solvers are presented and discussed. Chapter 6 presents an application of SDP in data analysis, in particular in clustering and dimensionality reduction problems. We first focus on a nonlinear SDP model for the clustering problem, consider its linear SDP relaxations and study their regularity. Then, we propose a new approach for clustering and dimensionality reduction using an approximation algorithmic framework. The resulting procedure called Two-Step-SDP algorithm is implemented in `R`, a free open source software. To show its efficiency, we present numerical experiments on several real-life data sets, including gene expression data sets from microarray experiments[2]. The final chapter of this work presents concluding remarks and topics of future research.

The Appendix A presents a detailed description on how to construct SDP problem instances in sparse SDPA format. A brief user guide of the developed MATLAB functions `SDPreg` and `DIISalg` is presented in Appendix B. The basic instructions to use the MATLAB function `nonregSDP` for generating nonregular SDP problem instances is presented in the Appendix C, and the NONREGSDP database is described in Appendix D. Finally, the last appendix contains a detailed description of how to use the `R` function `TwostepSDPClust` for solving clustering and dimensionality reduction problems.

[1]Some of the results presented in Chapter 4 are published in [91] and [95].

[2]The content of Chapter 6 reflects, for the most part, our results published in [92] and [94].

Chapter 2

Semidefinite programming

In this chapter, we introduce some notation, and summarize basic notions and results from linear algebra and convex analysis. Then, the primal SDP problem and its dual are formulated, as well as some duality results and optimality conditions for SDP are provided. The final section of this chapter presents an overview of standard SDP methods and solvers, and a discussion of their main characteristics.

2.1 Preliminaries

Let $\{x_1, \ldots, x_p\}$ be a set of $p \in \mathbb{N}$ vectors in $\mathbb{R}^n$ and $\{\lambda_1, \ldots, \lambda_p\}$ a set of p real scalars. A vector $v \in \mathbb{R}^n$ of the form $v = \sum\limits_{i=1}^{p} \lambda_i x_i$ is a linear combination of $x_1, \ldots, x_p$. If $\sum\limits_{i=1}^{p} \lambda_i = 1$, then v is an affine combination of $x_1, \ldots, x_p$. If, additionally, $\lambda_i \geq 0$, for all $i = 1, \ldots, p$, then v is a convex combination of $x_1, \ldots, x_p$. The set $\{x_1, \ldots, x_p\}$ is linearly independent if the only null linear combination is the trivial one (*i.e.*, all $\lambda_i = 0$, $i = 1, ..., p$).

A set $S \subset \mathbb{R}^n$ is said to be affine (resp. convex) if for all $x, y \in S$ and $\lambda \in \mathbb{R}$ (resp. $\lambda \in [0, 1]$) the convex combination $\lambda x + (1 - \lambda) y$ belongs to S.

Proposition 1 *A set $S \subset \mathbb{R}^n$ is convex if and only if it contains every convex combination of its elements.*

Examples of convex sets are hyperplanes and halfspaces, which are defined, respectively, as $\left\{x \in \mathbb{R}^n : v^T x = b\right\}$ and $\left\{x \in \mathbb{R}^n : v^T x \leq b\right\}$, where $v \in \mathbb{R}^n \backslash \{0\}$ and $b \in \mathbb{R}$. Notice that a hyperplane divides $\mathbb{R}^n$ into two halfspaces.

Evidently, the intersection of a finite number of convex sets is also a convex set. For example, a polyhedron, being an intersection of finitely many halfspaces and hyperplanes, is a convex set.

Let $S \subset \mathbb{R}^n$ be a convex set.

A function $f : S \to \mathbb{R}$ is said to be convex on S if for all $x, y \in S$ and $\lambda \in [0, 1]$ the inequality

$$f\left(\lambda x + (1 - \lambda)\right) \leq \lambda f\left(x\right) + (1 - \lambda) f\left(y\right)$$

is satisfied.

The epigraph of a function $f : S \to \mathbb{R}$ is the subset of $\mathbb{R}^{n+1}$ defined by

$$epi(f) = \{(x, r), x \in S, r \in \mathbb{R} : \ r \geq f(x)\}.$$

A characterization of a convex function can be made in terms of the convexity of its epigraph: a function is convex if and only if its epigraph is a convex set.

A simple example of a convex function is the linear function, since its epigraph is a halfspace, which is a convex set.

A nonempty set $K \subset \mathbb{R}^n$ is called a cone if for all $x \in K$ and $\lambda \geq 0$ we have $\lambda x \in K$. Additionally, K is convex if $x + y \in K$ for all $x, y \in K$. For example, the nonnegative orthant $\mathbb{R}^n_+$ is a convex cone.

A cone K is pointed if $x, -x \in K$ imply $x = 0$. A pointed convex cone K in $\mathbb{R}^n$ defines a partial order on $\mathbb{R}^n$ by $x \succeq_K y \Leftrightarrow x - y \in K$ for $x, y \in \mathbb{R}^n$.

Let $\mathcal{V}$ be a real finite dimensional vector space. A basis of $\mathcal{V}$ is a finite set of linearly independent vectors which span $\mathcal{V}$. The dimension of $\mathcal{V}$, denoted by $dim(\mathcal{V})$, is the cardinality of a basis.

Given two vectors $x, y \in \mathbb{R}^n$, their inner product, denoted by $\langle x, y\rangle$, is defined by

$$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i.$$

A basis of a vector space $\mathcal{V}$ is called canonical, or standard, if its elements are orthonormal vectors for the usual inner product. Any two vectors $x, y \in \mathcal{V}$ are orthonormal if $\langle x, y\rangle = 0$, $\langle x, x\rangle = 1$ and $\langle y, y\rangle = 1$.

Let K be a cone in $\mathcal{V}$. The set

$$K^* = \{z \in \mathcal{V} : \langle z, x\rangle \geq 0, \forall x \in K\}$$

is called the dual (or polar) cone of K. A dual cone K^* is always convex, even when the original cone K is not [148].

A cone K is said to be self-dual, if $K^* = K$. If K is self-dual, then K is convex, closed and full, *i.e.*, has nonempty interior.

A classic result in duality is the following ([84, 159]):

Lemma 1 *Let K be a closed convex cone. Then $(K^*)^* = K$.*

Given integers $m, n \in \mathbb{N}$, $\mathbb{R}^{m\times n}$ denotes the set of all $m \times n$ real matrices (whose elements are real numbers).

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the subspace of $\mathbb{R}^n$ spanned by the columns of $\mathbf{A}$ is called column (or range) space of the matrix $\mathbf{A}$ and is denoted here by $\mathcal{C}(\mathbf{A})$, and the subspace of $\mathbb{R}^m$ spanned by the rows of $\mathbf{A}$ is called its row space and is denoted by $\mathcal{R}(\mathbf{A})$.

Given a vector $v \in \mathbb{R}^n$, diag(v) denotes the $n \times n$ matrix whose diagonal elements are the components of the vector v and the remaining elements are zero.

The trace of a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, denoted by $\text{tr}(\mathbf{A})$, is the sum of all the elements of its diagonal, *i.e.*,

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}.$$

The operation trace of square matrices has the following property.

Property 1 *Given the matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ and scalars $\alpha, \beta \in \mathbb{R}$, the following equalities hold:*

- $\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^T)$*;*
- $\text{tr}(\alpha\mathbf{A} + \beta\mathbf{B}) = \alpha\text{tr}(\mathbf{A}) + \beta\text{tr}(\mathbf{B})$ *(linearity of trace);*
- $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$.

A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is symmetric if $\mathbf{A}^T = \mathbf{A}$, where $\mathbf{A}^T$ denotes the transpose of the matrix $\mathbf{A}$, *i.e.*, if $a_{ij} = a_{ji}$, for all $i = 1, ..., n$, and $j = 1, ..., n$.

The set of all real symmetric matrices of order n, denoted by $\mathcal{S}(n)$, is defined by

$$\mathcal{S}(n) = \left\{\mathbf{A} \in \mathbb{R}^{n \times n} : \mathbf{A}^T = \mathbf{A}\right\} \subseteq \mathbb{R}^{n \times n}.$$

The set $\mathcal{S}(n)$ can be considered as a vector space with the trace inner product defined by

$$\text{tr}(\mathbf{AB}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ji}, \tag{2.1}$$

for $\mathbf{A}$, $\mathbf{B} \in \mathcal{S}(n)$.

It is easy to see that $\mathcal{S}(n)$ is a $\frac{1}{2}n\,(n+1)$-dimensional vector space. The canonical basis of $\mathcal{S}(n)$ consists of $\frac{1}{2}n\,(n+1)$ orthonormal symmetric matrices of dimension $n \times n$ of the form $\mathbf{E}_{ij}$ such that for $i = 1, ..., n$, $j = 1, ..., n$,

> if $i = j$, then $\mathbf{E}_{ii}$ is a matrix whose entries are all zeros, except for the entry (i, i) which is 1;
>
> for the remaining matrices, $j > i$ and $i = 1, ..., n-1$, $\mathbf{E}_{ij}$ is a matrix whose entries are all zeros, except for the entries (i, j) and (j, i) which are $\frac{1}{\sqrt{2}}$.

Example 1 *The canonical basis of $\mathcal{S}(2)$ is the set formed by the matrices* $\mathbf{E}_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $\mathbf{E}_{12} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 \end{bmatrix}$ *and* $\mathbf{E}_{22} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$.

A matrix $\mathbf{A} \in \mathcal{S}(n)$ is positive semidefinite ($\mathbf{A} \succeq 0$), if $x^T\mathbf{A}x \geq 0$, for all $x \in \mathbb{R}^n$, and is positive definite ($\mathbf{A} \succ 0$), if $x^T\mathbf{A}x > 0$, for all nonzero $x \in \mathbb{R}^n$.

A matrix $\mathbf{A} \in \mathcal{S}(n)$ is negative semidefinite (respectively, negative definite), if the matrix $-\mathbf{A}$ is positive semidefinite (respectively, positive definite). In this case, we write $\mathbf{A} \preceq 0$ (respectively, $\mathbf{A} \prec 0$).

An interesting property of positive semidefinite matrices is the following ([56]).

Property 2 *Let $\mathbf{A} = [a_{ij}] \succeq 0$. If $a_{kk} = 0$ for some $k \in \{1....,n\}$, then $a_{ik} = a_{ki} = 0$ for each $i = 1,...,n$.*

Notice that a positive semidefinite matrix has nonnegative diagonal entries.

There exist different characterizations of positive semidefinite matrices that can be summarized as follows (see, *e.g.*, [56]):

Theorem 1 *Let $\mathbf{A} \in \mathcal{S}(n)$. The following conditions are equivalent:*

- *$\mathbf{A}$ is positive semidefinite;*
- *all eigenvalues of $\mathbf{A}$ are nonnegative;*
- *there exists a matrix $\mathbf{C} \in \mathbb{R}^{m\times n}$ such that $\mathbf{A} = \mathbf{C}^T\mathbf{C}$;*
- *all principal minors of $\mathbf{A}$ are nonnegative.*

The principal minors of a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, denoted here by δ_J, are the determinants of the principal submatrices of $\mathbf{A}$. The principal submatrix of a square matrix $\mathbf{A}$ is the matrix $\mathbf{A}_J$ whose rows and columns are indexed by the set J, where J is a subset of $\{1,...,n\}$.

Example 2 *Consider the matrix $\mathbf{A} \in \mathcal{S}(3)$ given by $\mathbf{A} = \begin{bmatrix} -1 & 2 & 1 \\ 2 & -6 & -4 \\ 1 & -4 & -3 \end{bmatrix}$. This is a negative semidefinite matrix, since the matrix $-\mathbf{A} = \begin{bmatrix} 1 & -2 & -1 \\ -2 & 6 & 4 \\ -1 & 4 & 3 \end{bmatrix}$ is positive semidefinite. Indeed:*

$$\delta_{\{1\}} = 1,\ \delta_{\{2\}} = 6,\ \delta_{\{3\}} = 3,\ \delta_{\{1,2\}} = 10,\ \delta_{\{1,3\}} = 2,\ \delta_{\{2,3\}} = 2,\ \delta_{\{1,2,3\}} = \det(\mathbf{A}) = 0,$$

where $\det(\mathbf{A})$ denotes the determinant of the matrix $\mathbf{A}$.

For positive definite matrices, one can state the following characterizations:

Theorem 2 *Let $\mathbf{A} \in \mathcal{S}(n)$. The following conditions are equivalent:*

- *$\mathbf{A}$ is positive definite;*
- *all eigenvalues of $\mathbf{A}$ are positive;*

- *all leading (i.e., top left) principal minors of* $\mathbf{A}$ *are positive.*

Example 3 *Consider the matrix* $\mathbf{A} \in \mathcal{S}(3)$ *given by* $\mathbf{A} = \begin{bmatrix} 3 & -2 & 0 \\ -2 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$. *It is positive definite, since*

$$\delta_{\{1\}} = 3, \quad \delta_{\{1,2\}} = 2, \quad \delta_{\{1,2,3\}} = \det(\mathbf{A}) = 4.$$

Denote by $\mathcal{P}(n)$ the set of all $n \times n$ real positive semidefinite symmetric matrices,

$$\mathcal{P}(n) = \{\mathbf{A} \in \mathcal{S}(n) : \mathbf{A} \succeq 0\}.$$

Proposition 2 *The set* $\mathcal{P}(n)$ *is a subset of* $\mathcal{S}(n)$ *and is a convex cone.*

Proof. By definition, $\mathcal{P}(n)$ is comprised of all symmetric positive semidefinite $(n \times n)$ matrices. $\mathcal{P}(n)$ is a convex set, since any positive combination of semidefinite matrices is semidefinite. To prove the proposition, let us show that $\forall \mathbf{A} \in \mathcal{P}(n), \forall \lambda \geq 0 \Rightarrow \lambda \mathbf{A} \in \mathcal{P}(n)$.

The following sequence of equivalences holds true.

$\mathbf{A} \in \mathcal{P}(n) \Leftrightarrow v^T\mathbf{A}v \geq 0, \forall v \in \mathbb{R}^n \Leftrightarrow \lambda v^T \mathbf{A} v \geq 0, \forall \lambda \geq 0 \Leftrightarrow v^T(\lambda \mathbf{A})v \geq 0.$

Therefore, $\lambda\mathbf{A} \in \mathcal{P}(n)$ and the proof is completed. ■

The cone $\mathcal{P}(n)$ induces a partial order on $\mathcal{S}(n)$ called the Löwner partial order as follows: $\mathbf{B} \succeq \mathbf{A}$ if $\mathbf{B} - \mathbf{A} \succeq 0$ [8, 125].

It is easy to see that the interior of the cone $\mathcal{P}(n)$ consists of all positive definite matrices.

The dual of the cone $\mathcal{P}(n)$ is a closed convex cone denoted by $\mathcal{P}(n)^*$ and is given by

$$\mathcal{P}(n)^* = \{\mathbf{A} \in \mathcal{S}(n) : \langle \mathbf{A}, \mathbf{B} \rangle \geq 0, \forall \mathbf{B} \in \mathcal{P}(n)\}.$$

In the cone of positive semidefinite symmetric matrices, $\mathcal{P}(n)$, the following property holds.

Property 3 *Let* $\mathbf{A}$, $\mathbf{B} \in \mathcal{P}(n)$. *Then* $\operatorname{tr}(\mathbf{AB}) \geq 0$ *and the equality holds if and only if* $\mathbf{AB} = \mathbf{0}$, *where* $\mathbf{0}$ *is the null matrix of order n.*

Proposition 3 *The set* $\mathcal{P}(n)$ *is a self-dual cone, i.e.,* $\mathcal{P}(n) = \mathcal{P}(n)^*$.

Proof. First, we will prove that $\mathcal{P}(n) \subseteq \mathcal{P}(n)^*$. Let $\mathbf{A} \in \mathcal{P}(n)$.

By the Property 3, it is easy to see that for all $\mathbf{B} \in \mathcal{P}(n)$, $\langle \mathbf{A}, \mathbf{B} \rangle = \operatorname{tr}(\mathbf{AB}) \geq 0$, meaning that $\mathbf{A} \in \mathcal{P}(n)^*$.

To prove that $\mathcal{P}(n)^* \subseteq \mathcal{P}(n)$, given $\mathbf{A} \in \mathcal{P}(n)^*$ let us show that $\mathbf{A} \succeq 0$. Let $x \in \mathbb{R}^n$. The matrix $\mathbf{B} = xx^T$ is positive semidefinite and thus, $\langle \mathbf{A}, \mathbf{B} \rangle \geq 0$. Since

$\langle \mathbf{A}, \mathbf{B} \rangle = \langle \mathbf{A}, xx^T \rangle = \operatorname{tr}(\mathbf{A}xx^T) = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} a_{ij}x_ix_j = x^T\mathbf{A}x$, it follows that $x^T\mathbf{A}x \geq 0$.

Hence, $\mathbf{A} \in \mathcal{P}(n)$. ■

Other necessary notation and definitions will be introduced in the corresponding sections.

2.2 Linear SDP Problem

Given $s \in \mathbb{N}$, consider the space $\mathcal{S}(s)$ of the $s \times s$ real symmetric matrices equipped with the trace inner product. Consider also the cone $\mathcal{P}(s) \subset \mathcal{S}(s)$ of $s \times s$ positive semidefinite symmetric matrices.

A linear SDP problem can be formulated as

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & c^T x \\ \text{s.t.} & \mathcal{A}(x) \preceq 0, \end{array} \tag{2.2}$$

where $x \in \mathbb{R}^n$ is the vector 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 $\mathbf{A}_i \in \mathcal{S}(s)$, $i = 0, 1, ..., n$. The inequality $\mathcal{A}(x) \preceq 0$ is usually called Linear Matrix Inequality (LMI).

Without loss of generality, we can assume that the matrices $\mathbf{A}_i$, $i = 1, ..., n$, are linearly independent, *i.e.*, $\mathbf{A}_i$, $i = 1, ..., n$, span a n-dimensional linear space in $\mathcal{S}(s)$.

Introducing a slack matrix variable $\mathbf{S} \in \mathcal{S}(s)$ in (2.2) we obtain an equivalent SDP problem:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \sum\limits_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{S} = -\mathbf{A}_0, \\ & \mathbf{S} \succeq 0. \end{array} \tag{2.3}$$

The constraint $\mathcal{A}(x) \preceq 0$ in (2.2) can be written in the form of the cone constraint $\mathcal{A}(x) \in -\mathcal{P}(s)$, and then, the SDP problem takes the form

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \mathcal{A}(x) \in -\mathcal{P}(s). \end{array} \tag{2.4}$$

There exist other alternative, but equivalent formulations for SDP problems [159]. A SDP problem can be written in the trace form as follows:

$$\begin{array}{lll} \min\limits_{\mathbf{X} \in \mathcal{S}(s)} & \operatorname{tr}(\mathbf{C}\mathbf{X}) & \\ \text{s.t.} & \operatorname{tr}(\mathbf{A}_i \mathbf{X}) = b_i, & \forall i = 1, \ldots, n, \\ & \mathbf{X} \succeq 0, & \end{array} \tag{2.5}$$

where $\mathbf{X} \in \mathcal{S}(s)$ is the matrix variable, $\mathbf{C}, \mathbf{A}_i \in \mathcal{S}(s)$ and $b_i \in \mathbb{R}$, $i = 1, ..., n$.

When there is more than one matrix variable, a SDP problem can be written as

$$\begin{array}{lll} \min & \sum\limits_{j=1}^{n} \operatorname{tr}(\mathbf{C}_j \mathbf{X}_j) & \\ \text{s.t.} & \sum\limits_{j=1}^{n} \operatorname{tr}(\mathbf{A}_{ij} \mathbf{X}_j) = b_i, & i = 1, ..., m, \\ & \mathbf{X}_j \succeq 0, & j = 1, ..., n, \end{array} \tag{2.6}$$

where $\mathbf{C}_j$, $\mathbf{A}_{ij}$ and also the matrix variables $\mathbf{X}_j$ are $s_j \times s_j$ real symmetric matrices, $j = 1, ..., n$, and $b_i \in \mathbb{R}$, $i = 1, ..., m$.

The SDP problem in the form (2.5) can be transformed to the form (2.2). The following example shows such transformation.

Example 4 *Consider the SDP problem given by*

$$\begin{array}{ll} \min & \operatorname{tr}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{X}\right) \\ \text{s.t.} & \operatorname{tr}\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{X}\right) = 0, \\ & \operatorname{tr}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{X}\right) = 1, \\ & \mathbf{X} \succeq 0. \end{array} \tag{2.7}$$

where $\mathbf{X} \in \mathcal{S}(3)$. *The problem* (2.7) *is equivalent to*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & x_4 = 0, \\ & x_1 + 2x_5 = 1, \\ & \mathbf{X} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_4 & x_5 \\ x_3 & x_5 & x_6 \end{bmatrix} \succeq 0. \end{array}$$

This problem can be rewritten in the matrix form

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & 0 & \frac{1-x_1}{2} \\ x_3 & \frac{1-x_1}{2} & x_6 \end{bmatrix} \succeq 0, \end{array}$$

which can be easily transformed in the form (2.2).

According to [159], one can transform the SDP problem (2.2) to the form (2.5) as follows. First, let us introduce the slack matrix variable $\mathbf{S}$ in (2.2), yielding the equivalent SDP problem (2.3).

We will show that such problem can be equivalently written as a problem in the form (2.5), in particular, with the specific form

$$\begin{array}{ll} \min & \operatorname{tr}(\mathbf{G}_0\mathbf{S}) \\ \text{s.t.} & \operatorname{tr}(\mathbf{G}_j\mathbf{S}) = g_j, \quad j = 1, ..., k, \\ & \mathbf{S} \succeq 0, \end{array} \tag{2.8}$$

where $\mathbf{G}_0$, $\mathbf{G}_j \in \mathcal{S}(s)$ and $g_j \in \mathbb{R}$, $j = 1, ..., k$, will be defined later.

Since the matrices $\mathbf{A}_i, i = 1, ..., n$, are linearly independent, they span a n-dimensional linear subspace in $\mathcal{S}(s)$ and can be written as a combination of elements of the canonical basis of $\mathcal{S}(s)$. Consider the affine subset of $\mathcal{S}(s)$

$$\mathcal{V} = \left\{ \mathbf{S} = -\mathbf{A}_0 - \sum_{i=1}^{n} \mathbf{A}_i x_i : x \in \mathbb{R}^n \right\}. \tag{2.9}$$

According to [159], it can be shown that for $k = \frac{1}{2}s(s+1) - n$, there exist $\mathbf{G}_j \in \mathcal{S}(s)$, $j = 1, ..., k$, and $g = (g_1, ..., g_k) \in \mathbb{R}^k$, such that $\mathcal{V}$ can be written in the form

$$\mathcal{V} = \{ \mathbf{S} \in \mathcal{S}(s) : \mathrm{tr}(\mathbf{G}_j \mathbf{S}) = g_j, j = 1, ..., k \}. \tag{2.10}$$

The matrix $\mathbf{G}_0 \in \mathcal{S}(s)$ in (2.8) should satisfy the following conditions:

$$\begin{array}{l} \mathrm{tr}(\mathbf{G}_0 \mathbf{A}_i) = -c_i, i = 1, ..., n, \\ c^T x = \mathrm{tr}(\mathbf{G}_0 \mathbf{S}) + \mathrm{tr}(\mathbf{G}_0 \mathbf{A}_0). \end{array} \tag{2.11}$$

The second equality in (2.11) holds, since

$$\mathbf{S} = -\mathbf{A}_0 - \sum_{i=1}^{n} \mathbf{A}_i x_i \Leftrightarrow \sum_{i=1}^{n} \mathbf{A}_i x_i = -\mathbf{A}_0 - \mathbf{S}$$

and taking into account the first condition in (2.11), we get

$$c^T x = \sum_{i=1}^{n} c_i x_i = -\mathrm{tr}\left(\mathbf{G}_0 \sum_{i=1}^{n} \mathbf{A}_i x_i \right) = \mathrm{tr}(\mathbf{G}_0 \mathbf{S}) + \mathrm{tr}(\mathbf{G}_0 \mathbf{A}_0).$$

Finally, the vector $g = (g_1, ..., g_k) \in \mathbb{R}^k$ in (2.8) can be defined as

$$g_j = -\mathrm{tr}(\mathbf{G}_j \mathbf{A}_0), j = 1, ..., k. \tag{2.12}$$

Example 5 *Consider the following SDP program*

$$\begin{array}{ll} \min & x_{12} \\ \text{s.t.} & \begin{bmatrix} 0 & x_{12} & 0 \\ x_{12} & x_{22} & 0 \\ 0 & 0 & 1 + x_{12} \end{bmatrix} \succeq 0, \end{array} \tag{2.13}$$

which can be equivalently written as

$$\begin{array}{ll} \min & x_{12} \\ \text{s.t.} & \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} x_{12} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_{22} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \preceq 0. \end{array} \tag{2.14}$$

Notice that the last problem has the LMI form (2.2), with $c = (1, 0)^T$ and matrices $\mathbf{A}_i \in \mathcal{S}(3)$, $i = 0, 1, 2, 3$, given by

$$\mathbf{A}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \mathbf{A}_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \textit{ and } \mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let $\mathbf{S} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{12} & x_{22} & x_{23} \\ x_{13} & x_{23} & x_{33} \end{bmatrix} \in \mathcal{S}(3)$.

Following the above transformation, the matrix $\mathbf{G}_0 = \begin{bmatrix} g^0_{11} & g^0_{12} & g^0_{13} \\ g^0_{12} & g^0_{22} & g^0_{23} \\ g^0_{13} & g^0_{23} & g^0_{33} \end{bmatrix} \in \mathcal{S}(3)$ *must satisfy:*

$$\operatorname{tr}(\mathbf{G}_0\mathbf{A}_i) = -c_i, \; i = 1, 2 \tag{2.15}$$
$$c^T x = \operatorname{tr}(\mathbf{G}_0\mathbf{S}) + \operatorname{tr}(\mathbf{G}_0\mathbf{A}_0). \tag{2.16}$$

Considering $c_1 = 1$ *and* $c_2 = 0$*, it follows from condition* (2.15) *that*

$$\operatorname{tr}(\mathbf{G}_0\mathbf{A}_1) = -1 \Leftrightarrow -2g^0_{12} - g^0_{33} = -1 \Leftrightarrow g^0_{12} = \frac{1 - g^0_{33}}{2}$$
$$\operatorname{tr}(\mathbf{G}_0\mathbf{A}_2) = 0 \Leftrightarrow g^0_{22} = 0.$$

Since $\operatorname{tr}(\mathbf{G}_0\mathbf{A}_0) = -g^0_{33}$ *and*

$$\operatorname{tr}(\mathbf{G}_0\mathbf{S}) = \left(g^0_{11}x_{11} + \frac{1-g^0_{33}}{2}x_{12} + g^0_{13}x_{13}\right) + \left(\frac{1-g^0_{33}}{2}x_{12} + 0 + g^0_{23}x_{23}\right) + \\ + \left(g^0_{13}x_{13} + g^0_{23}x_{23} + g^0_{33}x_{33}\right),$$

from equality (2.16) *we get* $g^0_{11} = 0$, $g^0_{13} = 0$, $g^0_{23} = 0$ *and* $g^0_{33} = 0$.

Therefore, $g^0_{12} = \frac{1}{2}$ *and* $G_0 = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

The canonical basis of $S(3)$ *is* $\mathcal{B} = \{\mathbf{E}_{11}, \mathbf{E}_{12}, \mathbf{E}_{13}, \mathbf{E}_{22}, \mathbf{E}_{23}, \mathbf{E}_{33}\}$*, where*

$$\mathbf{E}_{11} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; \mathbf{E}_{12} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; \mathbf{E}_{13} = \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 \end{bmatrix},$$
$$\mathbf{E}_{22} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; \mathbf{E}_{23} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}, \; \mathbf{E}_{33} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Since $n = 2$ *and* $s = 3$*, then* $k = 4$*. In the basis* $\mathcal{B}$*, the matrices* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *have the form* $\mathbf{A}_1 = -\sqrt{2}\mathbf{E}_{12} - \mathbf{E}_{33}$ *and* $\mathbf{A}_2 = -\mathbf{E}_{22}$*. The remaining matrices of* $\mathcal{B}$ *do not contribute for* $\mathbf{A}_1$ *and* $\mathbf{A}_2$*: they will be used to obtain the matrices* $\mathbf{G}_1$*,* $\mathbf{G}_2$*,* $\mathbf{G}_3$ *such that* $\operatorname{tr}(\mathbf{G}_j\mathbf{A}_0) = -g_j$ *and* $g_j = 0$*, for* $j = 1, 2, 3$*. Hence, one can consider* $\mathbf{G}_1 = \mathbf{E}_{11}$*,*

$\mathbf{G}_2 = \mathbf{E}_{13}$, $\mathbf{G}_3 = \mathbf{E}_{23}$ or, to ease our calculus, we can just consider $\mathbf{G}_1 = \mathbf{E}_{11}$, $\mathbf{G}_2 = \sqrt{2}\mathbf{E}_{13}$, $\mathbf{G}_3 = \sqrt{2}\mathbf{E}_{23}$, and thus, $\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, $\mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$ and $\mathbf{G}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$.

A matrix $\mathbf{G}_4$ can be chosen to be a linear combination of the elements of $\mathcal{B}$ and g_4 must satisfy (2.12). So, consider $\mathbf{G}_4 = \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $g_4 = 1$.

Therefore, the problem (2.13) can be written in the trace form as

$$\begin{aligned} \min \quad & \operatorname{tr}\left(\begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{S}\right) \\ \text{s.t.} \quad & \operatorname{tr}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\mathbf{S}\right) = 0 \\ & \operatorname{tr}\left(\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}\mathbf{S}\right) = 0 \\ & \operatorname{tr}\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\mathbf{S}\right) = 0 \\ & \operatorname{tr}\left(\begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{S}\right) = 1 \\ & \mathbf{S} \succeq 0, \end{aligned} \tag{2.17}$$

where $\mathbf{S} \in S(3)$.

The SDP problem (2.2) is a convex problem. Indeed, its objective function is linear, hence convex and its feasible set given by

$$\mathcal{X} = \{x \in \mathbb{R}^n : \mathcal{A}(x) \preceq 0\} \tag{2.18}$$

is convex. To prove this, we will show that $\mathcal{A}(\lambda x + (1-\lambda)y) \preceq 0$ for all $x, y \in \mathcal{X}$ and for all $\lambda \in [0,1]$. In fact,

$$\begin{aligned} \mathcal{A}(\lambda x + (1-\lambda)y) &= \mathbf{A}_0 + \sum_{i=1}^{n} \mathbf{A}_i(\lambda x_i + (1-\lambda)y_i) \\ &= \mathbf{A}_0 + \lambda \sum_{i=1}^{n} \mathbf{A}_i x_i + (1-\lambda) \sum_{i=1}^{n} \mathbf{A}_i y_i \\ &= \lambda \mathbf{A}_0 + (1-\lambda)\mathbf{A}_0 + \lambda \sum_{i=1}^{n} \mathbf{A}_i x_i + (1-\lambda) \sum_{i=1}^{n} \mathbf{A}_i y_i \\ &= \lambda \mathcal{A}(x) + (1-\lambda)\mathcal{A}(y) \preceq 0, \text{ since } 1-\lambda \geq 0. \end{aligned}$$

The feasible set of a SDP problem can be considered as an intersection of the cone $\mathcal{P}(n)$ with an affine-linear space, and is called a spectrahedron.

SDP is a natural extension of linear programming (LP), which refers to minimizing a linear objective function over a convex polyhedron. Indeed, any LP problem of the form

$$\begin{array}{ll} \min\limits_{x} & c^T x \\ \text{s.t.} & \mathbf{B}x + b \leq 0, \end{array} \tag{2.19}$$

where $\mathbf{B} \in \mathbb{R}^{s\times n}$, $b, c \in \mathbb{R}^n$ and the variable is $x \in \mathbb{R}^n$, can be equivalently written as a SDP problem of the form (2.2). To prove it, suppose that $\mathbf{A}_0 = \operatorname{diag}(b)$ and $\mathbf{A}_i = \operatorname{diag}(\mathbf{B}^i)$, $i = 1, ..., n$, where $\mathbf{B}^i \in \mathbb{R}^s$ is the i-th column of the matrix $\mathbf{B}$.

2.3 Duality and optimality results in SDP

The optimality in SDP, as well as in the optimization theory in general, is closely related to the duality aspects.

Though there exist different duality approaches (see, *e.g.*, [126]), the dual problem is usually derived by applying the Lagrangian approach [52]. The Lagrange function (or Lagrangian) of the SDP problem (2.2) has the form

$$\mathcal{L}(x, \mathbf{Z}) = c^T x + \operatorname{tr}(\mathbf{Z}\mathcal{A}(x)) = c^T x + \operatorname{tr}\left(\mathbf{Z}\left(\mathbf{A}_0 + \sum_{i=1}^{n} \mathbf{A}_i x_i\right)\right), \tag{2.20}$$

where $\mathbf{Z} \in \mathcal{P}(s)$ and $x \in \mathbb{R}^n$. Using the properties of the trace of a matrix, the function (2.20) can be rewritten as

$$\mathcal{L}(x, \mathbf{Z}) = \operatorname{tr}(\mathbf{Z}\mathbf{A}_0) + \sum_{i=1}^{n} (\operatorname{tr}(\mathbf{Z}\mathbf{A}_i) + c_i)\, x_i.$$

Considering the min-max problem

$$\min_{x\in\mathbb{R}^n} \max_{\mathbf{Z}\in\mathcal{P}(s)} \mathcal{L}(x, \mathbf{Z}),$$

we can easily see that it is equivalent to the problem (2.2). Reversing the order of the min and max operations in the above problem, we get the dual problem to (2.2) in the form

$$\begin{array}{ll} \max & \operatorname{tr}(\mathbf{A}_0\mathbf{Z}) \\ \text{s.t.} & -\operatorname{tr}(\mathbf{A}_i\mathbf{Z}) = c_i, \quad \forall i = 1, \dots, n, \\ & \mathbf{Z} \succeq 0, \end{array} \tag{2.21}$$

where $\mathbf{Z} \in \mathcal{S}(s)$ is the dual matrix variable.

The feasible set of (2.21) is

$$\mathcal{Z} = \{\mathbf{Z} \in \mathcal{P}(s) : -\operatorname{tr}(\mathbf{A}_i\mathbf{Z}) = c_i, i = 0, 1, ..., n\}. \tag{2.22}$$

Notice that the dual problem (2.21) is a SDP problem in the form (2.5). Since the formulations (2.2) and (2.21) are dual to each other, there is no loss of generality in assuming a particular form for the primal or the dual problem [8]. We just use the transformations explained in the previous section and change the sign of the objective function when transforming maximization into minimization [149].

Unless stated otherwise, in what follows, we refer to (2.2) as the primal problem, and to the problem (2.21) as its dual.

Theorem 3 *[Weak Duality Property] Given a primal-dual pair of feasible solutions $x \in \mathcal{X}$ and $\mathbf{Z} \in \mathcal{Z}$ of the SDP problems* (2.2) *and* (2.21)*, the inequality $c^T x \geq \operatorname{tr}(\mathbf{A}_0\mathbf{Z})$ always holds.*

Proof. Considering $p = c^T x$ and $d = \operatorname{tr}(\mathbf{A}_0\mathbf{Z})$, and using the dual formulation, we have

$$p-d = \sum_{i=1}^{n} c_i x_i - \operatorname{tr}(\mathbf{Z}\mathbf{A}_0) = \sum_{i=1}^{n} (-\operatorname{tr}(\mathbf{Z}\mathbf{A}_i))\, x_i - \operatorname{tr}(\mathbf{Z}\mathbf{A}_0) = \operatorname{tr}\left(\left(-\mathbf{A}_0 - \sum_{i=1}^{n} \mathbf{A}_i x_i\right)\mathbf{Z}\right).$$

Since $-\mathbf{A}_0 - \sum_{i=1}^{n} \mathbf{A}_i x_i \succeq 0$ and $\mathbf{Z} \succeq 0$, then $\operatorname{tr}\left(\left(-\mathbf{A}_0 - \sum_{i=1}^{n} \mathbf{A}_i x_i\right)\mathbf{Z}\right) \geq 0$, and we obtain the inequality $p - d \geq 0$. Hence, $p \geq d$, for any primal feasible solution x and dual feasible solution $\mathbf{Z}$. This completes the proof. ■

Definition 1 *Given a primal-dual pair of feasible solutions $x \in \mathcal{X}$ and $\mathbf{Z} \in \mathcal{Z}$ of the SDP problems* (2.2) *and* (2.21)*, if $p = c^T x$ and $d = \operatorname{tr}(\mathbf{A}_0\mathbf{Z})$, the difference $p - d$ is called duality gap.*

Given any pair $(x, \mathbf{Z})$ of feasible solutions of the problems (2.2) and (2.21), if the duality gap is zero, then it can be proved that x is an optimal solution of (2.2) and $\mathbf{Z}$ is an optimal solution of (2.21) [52, 69]. The optimal solutions will be denoted by x^* and $\mathbf{Z}^*$, and the optimal values of the objective functions of the primal and dual problems (2.2) and (2.21) will be denoted here by p^* and d^*, respectively.

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 also called strict feasibility, Slater constraint qualification, or Slater regularity condition [69].

Definition 2 *The constraints of the problem* (2.2) *satisfy the Slater (regularity) condition if the interior of its feasible set $\mathcal{X}$ is nonempty, i.e.,*

$$\exists\ \bar{x} \in \mathbb{R}^n : \mathcal{A}(\bar{x}) \prec 0. \tag{2.23}$$

The analogous definition can be introduced for SDP problems in the dual form.

Definition 3 *The constraints of the dual SDP problem* (2.21) *satisfy the Slater condition if there exists a feasible matrix $\mathbf{Z}$ such that $\mathbf{Z} \succ 0$.*

The following duality results are known in SDP ([23, 156]):

Theorem 4 *[Strong Duality Property] Assume that the primal optimal value of the linear SDP problem* (2.2) p^* *is finite. Under the Slater condition, the duality gap vanishes and the (dual) optimal value of* (2.21) d^* *is attained.*

Corollary 1 *Let* p^* *be the optimal value of the objective function of the primal problem* (2.2) *and* d^* *the optimal value of the dual problem* (2.21).

1. *If the dual SDP problem* (2.21) *satisfies the Slater condition with* d^* *finite, then* $p^* = d^*$ *and this value is attained for the primal problem* (2.2);
2. *If both primal and dual SDP problems satisfy the Slater condition, then* $p^* = d^*$ *and this value is attained for both problems.*

Note 1 *The fulfilment of the Slater condition for the primal problem* (2.2) *does not imply that the constraints of its dual* (2.21) *satisfy the Slater condition, or vice-versa.*

Example 6 *Consider the primal-dual pair of SDP problems*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} x_1 & 1 \\ 1 & x_2 \end{bmatrix} \succeq 0, \end{array} \qquad\qquad \begin{array}{ll} \max & 2y_1 \\ \text{s.t.} & \begin{bmatrix} 1 & -y_1 \\ -y_1 & 0 \end{bmatrix} \succeq 0. \end{array}$$

The primal problem satisfies the Slater condition: its feasible solution with $x_1 = 1$ *and* $x_2 = 2$ *is strictly feasible:* $\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \succ 0$. *However, the dual problem does not have strictly feasible solutions. In fact,* $y_1 = 0$ *is the only feasible solution of the dual problem and the matrix* $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ *is not positive definite.*

Note 2 *If the dual SDP problem does not satisfy the Slater condition, the primal optimal value may not be attained even if it satisfies the Slater condition (see the Example 6, where the primal optimal value is zero, but it is not attained).*

The Slater condition is a condition that ensures strong duality. In the absence of the Slater condition, strong duality may not hold, or the optimum may not be attained, or one of the problems may be feasible and bounded, while the other is infeasible, as the following examples show.

Example 7 *Consider the primal SDP problem* (2.13) *given in the Example 5. Its dual is given by*

$$\begin{array}{ll} \max & y_1 \\ \text{s.t.} & \begin{bmatrix} -y_2 & \frac{1+y_1}{2} & -y_3 \\ \frac{1+y_1}{2} & 0 & -y_4 \\ -y_3 & -y_4 & -y_1 \end{bmatrix} \succeq 0 \end{array}$$

Both problems do not satisfy the Slater condition. The primal optimal value is $p^* = 0$, *while the dual is* $d^* = -1$, *i.e., there exists a nonzero duality gap, so strong duality does not hold.*

Example 8 *Consider the following primal SDP problem*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} x_1 + \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} x_2 \preceq 0 \end{array}$$

and its dual given by

$$\begin{array}{ll} \max & 0 \\ \text{s.t.} & \operatorname{tr}\left(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \mathbf{Z} \right) = -1, \\ & \operatorname{tr}\left(\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{Z} \right) = 0, \\ & \mathbf{Z} = \begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix} \succeq 0. \end{array}$$

It is easy to see that any feasible solution of the primal problem should satisfy

$$-{x_1}^2 \geq 0 \quad \textit{and} \quad x_2 \geq 0,$$

which implies that $x_1 = 0$. Therefore, the primal problem does not satisfy the Slater condition.

Considering the dual problem, we conclude that any feasible solution should satisfy

$$z_2 = \tfrac{1}{2}, \quad z_3 = 0, \quad z_1 \geq 0 \quad \textit{and} \quad z_1 z_3 - {z_2}^2 \geq 0.$$

Clearly, these conditions are incompatible, since the last condition does not hold for $z_2 = \frac{1}{2}$ and $z_3 = 0$. Therefore, the dual problem is infeasible.

Definition 4 *A pair $(x^*, \mathbf{Z}^*)$ is said to be a saddle point of the Lagrange function (2.20) if the conditions*

$$x^* \in \arg\min_{x \in \mathbb{R}^n} \mathcal{L}(x, \mathbf{Z}^*), \ \operatorname{tr}(\mathbf{Z}^* \mathcal{A}(x^*)) = 0, \ \mathcal{A}(x^*) \preceq 0, \ \mathbf{Z}^* \succeq 0 \tag{2.24}$$

hold.

Since the Lagrange function is convex, the first condition in (2.24) is equivalent to $\nabla_x \mathcal{L}(x^*, \mathbf{Z}^*) = 0$ [13], and thus, the conditions (2.24) can be rewritten as

$$\nabla_x \mathcal{L}(x^*, \mathbf{Z}^*) = 0, \ \operatorname{tr}(\mathbf{Z}^* \mathcal{A}(x^*)) = 0, \ \mathcal{A}(x^*) \preceq 0, \ \mathbf{Z}^* \succeq 0. \tag{2.25}$$

In the SDP duality theory the following result is established [13].

Proposition 4 *Given the primal-dual pair of SDP problems (2.2) and (2.21), $p^* = d^*$ and x^* and $\mathbf{Z}^*$ are optimal solutions of (2.2) and (2.21), respectively, if and only if $(x^*, \mathbf{Z}^*)$ is a saddle point of the Lagrange function (2.20).*

In convex SDP, given an optimal solution x^*, if the set $\Lambda(x^*)$ of all Lagrange multiplier matrices $\mathbf{Z}^*$ satisfying the conditions (2.25) is nonempty, then it coincides with the set of optimal solutions of the dual SDP problem (2.21) [13]. Let us formulate the following result from [13].

Theorem 5 *Let x^* be an optimal solution of the SDP problem* (2.2)*. The set $\Lambda(x^*)$ is nonempty and bounded if and only if the Slater condition holds for* (2.2)*.*

Considering the primal SDP problem in the form (2.3), its dual problem (2.21) and the corresponding optimal solutions $(x^*, \mathbf{S}^*)$ and $\mathbf{Z}^*$, notice that the duality gap is given by $\operatorname{tr}(\mathbf{S}^*\mathbf{Z}^*) = 0$. Using the Property 3, we can establish the following result.

Theorem 6 *Let $(x^*, \mathbf{S}^*)$ be a primal feasible solution of* (2.3) *and $\mathbf{Z}^*$ be a dual feasible solution of* (2.21)*. Then, $(x^*, \mathbf{S}^*)$ and $\mathbf{Z}^*$ are optimal solutions if and only if $\mathbf{S}^*\mathbf{Z}^* = 0$.*

The first order necessary and sufficient optimality conditions for (convex) SDP problems can be then formulated in the form of KKT-type conditions by the following well-known theorem (see, *e.g.*, [8, 69, 157, 159]).

Theorem 7 *Suppose that both the primal and dual SDP problems* (2.3) *and* (2.21) *satisfy the Slater condition. Then, the primal and dual solutions $(x^*, \mathbf{S}^*)$ and $\mathbf{Z}^*$, with $\mathbf{S}^*, \mathbf{Z}^* \succeq 0$, are optimal for* (2.3) *and* (2.21)*, respectively, if and only if the following conditions hold*

$$\mathbf{A}_0 + \sum_{i=1}^{n} \mathbf{A}_i x_i^* + \mathbf{S}^* = 0 \tag{2.26}$$

$$\operatorname{tr}(\mathbf{A}_i \mathbf{Z}^*) + c_i = 0, \ i = 1, ..., n \tag{2.27}$$

$$\mathbf{S}^*\mathbf{Z}^* = 0. \tag{2.28}$$

In the Theorem 7, the condition (2.26) is called primal feasibility, the condition (2.27) is called dual feasibility, and the condition (2.28) is called complementary condition.

The above KKT-type optimality conditions can be reformulated for the SDP problem (2.2) as follows ([13]):

Theorem 8 *Suppose that the SDP problem* (2.2) *satisfies the Slater condition. Then, $x^* \in \mathcal{X}$ is an optimal solution of* (2.2) *if and only if there exists a matrix $\mathbf{Z}^* \in \mathcal{P}(s)$ such that*

$$\operatorname{tr}(\mathbf{A}_i \mathbf{Z}^*) + c_i = 0, i = 1, ..., n, \quad \textit{and} \quad \operatorname{tr}(\mathcal{A}(x^*)\mathbf{Z}^*) = 0. \tag{2.29}$$

Example 9 *Let us show by applying the optimality conditions* (2.29) *that $x^* = 1$ is an optimal solution for the primal SDP problem*

$$\begin{array}{ll} \min & -2x \\ \text{s.t.} & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \preceq 0. \end{array}$$

First, notice that the constraints of this problem satisfy the Slater condition. For instance, $x = \frac{1}{2}$ *is a strictly feasible solution. So, we can apply the Theorem 8 to prove that* x^* *is optimal. Second,* $\mathcal{A}(x^*) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$.

Now, let us show that there exists a matrix $\mathbf{Z}^* = \begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix} \in \mathcal{P}(2)$ *satisfying the conditions* (2.29). *It follows that*

$$\operatorname{tr}\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix}\right) - 2 = 0 \Leftrightarrow z_2 = 1$$

and

$$\operatorname{tr}\left(\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix}\right) = 0 \Leftrightarrow z_1 = z_3 - 2.$$

Thus, $\mathbf{Z}^* = \begin{bmatrix} z_3 - 2 & 1 \\ 1 & z_3 \end{bmatrix}$ *with* $z_3 \in \mathbb{R}$. *It is easy to see that, for example,* $z_3 = 4$, $\mathbf{Z}^* = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix} \in \mathcal{P}(2)$.

Therefore, $x^* = 1$ *is optimal.*

In the absence of the Slater condition, the optimality conditions of the Theorem 8 may not hold. The following example illustrates such situation.

Example 10 *Consider the primal SDP problem*

$$\begin{array}{ll} \min & x \\ \text{s.t.} & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \preceq 0. \end{array}$$

The constraints of this problem do not satisfy the Slater condition. Clearly, $x^* = 1$ *is the optimal solution.*

It is easy to see that $\mathcal{A}(x^*) = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$.

Consider a matrix $\mathbf{Z}^* = \begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix} \in \mathcal{P}(2)$. *From the conditions* (2.29), *we get*

$$\operatorname{tr}\left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix}\right) + 1 = 0 \text{ and } \operatorname{tr}\left(\begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} z_1 & z_2 \\ z_2 & z_3 \end{bmatrix}\right) = 0.$$

It follows that $z_1 = 0$ *and* $z_2 = -\frac{1}{2}$. *Therefore,* $\mathbf{Z}^* = \begin{bmatrix} 0 & -\frac{1}{2} \\ -\frac{1}{2} & z_3 \end{bmatrix}$ *and for any* z_3 *the matrix* $\mathbf{Z}^* \notin \mathcal{P}(2)$.

We can conclude that the optimality conditions of the Theorem 8 are not satisfied.

Optimality conditions that do not require the Slater condition to hold have been formulated in, *e.g.*, [48, 62, 63, 78, 125, 126]. However, most of these conditions are rather difficult to verify in practice and, to the best of our knowledge, none of them are implemented in a SDP method.

Since most popular and efficient SDP methods are based on solving systems of type (2.26)-(2.28), the failure of the Slater condition yields numerical difficulties. Therefore, it is of great importance to know in advance if the constraints of a given SDP problem satisfy the Slater condition and in the case of failure, to choose some alternative strategy for solving the problem.

A numerical procedure that permits to verify if a given SDP problem satisfies the Slater condition will be discussed in Chapter 3. The implementation details of the proposed procedure, as well as numerical experiments will be presented in Chapter 4.

2.4 Interior point methods for solving SDP problems

Different methods have been proposed for solving SDP problems, such as primal-dual interior point methods, dual interior point methods, and augmented Lagrangian methods, some of them for specific problems [8, 16, 103].

Primal-dual interior point methods are currently the most successful known methods for solving (approximately) SDP problems [42].

In what follows, we will describe the general steps of a primal-dual (path-following) interior point method applied to a given SDP problem in the form (2.3).

The basic idea of the method is to replace the solution of (2.3) by a sequence of approximated solutions of some auxiliary barrier problems. These (perturbed) auxiliary problems are constructed by adding the barrier function $\log(\det(\mathbf{S}))$ to the objective function of (2.3), and introducing a parameter in the barrier function, given by μ, which is a positive real number called barrier parameter.

Consider the following perturbation to the SDP problem (2.3):

$$\begin{array}{ll} \min & c^T x - \mu \log(\det(\mathbf{S})) \\ \text{s.t.} & \sum\limits_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{S} = -\mathbf{A}_0 \\ & \mathbf{S} \succeq 0. \end{array} \tag{2.30}$$

For each $\mu > 0$, the Lagrange function associated to this problem is

$$\mathcal{L}(x, S, Z) = c^T x - \mu \log(\det(\mathbf{S})) + \operatorname{tr}\left(\mathbf{Z}\left(\sum_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{A}_0 + \mathbf{S}\right)\right). \tag{2.31}$$

Since this function is strictly convex [114], and under the assumption that the Slater condition holds, we can guarantee that the following optimality conditions are both necessary and sufficient:

$$\left\{\begin{array}{l} \sum\limits_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{A}_0 + \mathbf{S} = 0, \\ \operatorname{tr}(\mathbf{A}_i \mathbf{Z}) + c_i = 0, i = 1, ..., n, \\ -\mu \mathbf{S}^{-1} + \mathbf{Z} = 0. \end{array}\right. \tag{2.32}$$

The condition $-\mu\mathbf{S}^{-1} + \mathbf{Z} = 0$ reflects the perturbation of the barrier parameter μ and is equivalent to $\mathbf{ZS} - \mu\mathbf{I}_s = 0$, where $\mathbf{I}_s$ stands for the identity matrix of order s. This condition is called *perturbed complementary slackness* (see [157]). Hence, the minimizers of the problem (2.30) satisfy the following conditions.

$$\begin{cases} \sum_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{S} = -\mathbf{A}_0 \\ \operatorname{tr}(\mathbf{A}_i\mathbf{Z}) = -c_i, i = 1, ..., n \\ \mathbf{ZS} = \mu\mathbf{I}_s \\ \mathbf{S} \succ 0, \mathbf{Z} \succ 0 \end{cases} \tag{2.33}$$

Let $(x_\mu, \mathbf{S}_\mu, \mathbf{Z}_\mu)$ be a solution of the system (2.33). The set

$$\{(x_\mu, \mathbf{S}_\mu, \mathbf{Z}_\mu)\,,\ \mu > 0\} \tag{2.34}$$

is called *central path.* A solution $(x_\mu, \mathbf{S}_\mu, \mathbf{Z}_\mu)$ is usually called central path point. According to [69, 140], we formulate the following theorem.

Theorem 9 *Suppose that both primal SDP problem* (2.3) *and its dual* (2.21) *satisfy the Slater condition and that the matrices $\mathbf{A}_i$, $i = 1, ..., n$, are linearly independent. Then for any $\mu > 0$, the central path point $(x_\mu, \mathbf{S}_\mu, \mathbf{Z}_\mu)$ exists and is unique.*

Suppose that $(x^*, \mathbf{S}^*)$ and $\mathbf{Z}^*$ are the optimal solutions of the primal and the dual problems (2.3) and (2.21), respectively. In [52], it is shown that

$$\lim_{\mu\to 0}(x_\mu, \mathbf{S}_\mu, \mathbf{Z}_\mu) = (x^*, \mathbf{S}^*, \mathbf{Z}^*). \tag{2.35}$$

Thus, the idea of primal-dual interior point methods is to iteratively compute an approximate solution of the system (2.33), usually by using a modified Newton's method, followed by a reduction in the parameter μ, in order to minimize the perturbation. Therefore, interior point methods are also known in the literature as path-following methods [69, 159].

We would like to compute an approximate solution of (2.33). Let $i = 1, ..., n$, and

$$F_\mu(x, \mathbf{S}, \mathbf{Z}) = \begin{bmatrix} \sum_{i=1}^{n} \mathbf{A}_i x_i + \mathbf{S} + \mathbf{A}_0 \\ \operatorname{tr}(\mathbf{A}_i\mathbf{Z}) + c_i \\ \mathbf{ZS} - \mu\mathbf{I}_s \end{bmatrix} = \mathbf{0}. \tag{2.36}$$

On a general iteration of the method, say k, starting from a point $(x^k, \mathbf{S}^k, \mathbf{Z}^k)$ of the central path, we need to find a search direction $\triangle d = (\triangle x, \triangle\mathbf{S}, \triangle\mathbf{Z})$ such that a new point $(x^k + \triangle x, \mathbf{S}^k + \triangle\mathbf{S}, \mathbf{Z}^k + \triangle\mathbf{Z})$ lies in the central path for μ. One can obtain a search direction $\triangle d$ by solving the system

$$F_\mu(x^k, \mathbf{S}^k, \mathbf{Z}^k) + \nabla F_\mu(x^k, \mathbf{S}^k, \mathbf{Z}^k)\triangle d^T = 0$$

using a modified Newton's method [51].

The system (2.36) can be rewritten introducing the residuals R_P, R_D and R_C as follows:

$$\begin{array}{l} R_P := \sum\limits_{i=1}^{n} \mathbf{A}_i x_i^k + \mathbf{A}_0 + \mathbf{S}^k = 0 \\ R_D := \mathrm{tr}\left(\mathbf{A}_i \mathbf{Z}^k\right) + c_i = 0, i = 1, ..., n \\ R_C := \mathbf{Z}^k \mathbf{S}^k - \mu \mathbf{I}_s = 0 \end{array}$$

and its linearization leads to the following system in terms of the search direction $\triangle d$:

$$\left\{ \begin{array}{l} \sum\limits_{i=1}^{n} \mathbf{A}_i \triangle x_i + \triangle \mathbf{S} = -R_P \\ \mathrm{tr}\left(\mathbf{A}_i \triangle \mathbf{Z}\right) = -R_D, i = 1, ..., n \\ \mathbf{Z}^k \triangle \mathbf{S} + \triangle \mathbf{Z} \mathbf{S}^k = -R_C. \end{array} \right. \tag{2.37}$$

In general, the matrices $\mathbf{S}$ and $\mathbf{Z}$ of (2.36) do not commute (see [51, 52]) and thus, the third equation in (2.37) yields nonsymmetric directions. Therefore, some symmetrization process should be applied.

To ensure the positive definiteness of the matrices $\mathbf{S}^k$ and $\mathbf{Z}^k$, a line search can be performed to find constants α_P and α_D such that the matrices $\mathbf{S}^k + \alpha_P \triangle \mathbf{S}$ and $\mathbf{Z}^k + \alpha_D \triangle \mathbf{Z}$ are positive definite [54]. The new central path point can be written as

$$(x^{k+1}, \mathbf{S}^{k+1}, \mathbf{Z}^{k+1}) = (x^k + \alpha_P \triangle x, \mathbf{S}^k + \alpha_P \triangle \mathbf{S}, \mathbf{Z}^k + \alpha_D \triangle \mathbf{Z}).$$

The interior point method stops when $\mathrm{tr}(\mathbf{Z}^k \mathbf{S}^k) \leq \varepsilon$, *i.e.*, when the duality gap is sufficiently small.

Primal-dual interior point algorithms are proved to be polynomially convergent (see, *e.g.*, [8, 51, 54, 69, 104, 159]).

The basic steps of a primal-dual interior point algorithm can be outlined as follows ([161]).

Algorithm 1 General scheme of a primal-dual interior point method

```
1: Choose a tolerance $\varepsilon$ and an initial point $(x^0, \mathbf{S}^0, \mathbf{Z}^0)$, where $\mathbf{S}^0 \succ 0$ and $\mathbf{Z}^0 \succ 0$
2: general iteration $k$: given a current approximate solution $(x^k, \mathbf{S}^k, \mathbf{Z}^k)$:
3: while $\mathrm{tr}(\mathbf{Z}^k \mathbf{S}^k) > \varepsilon$ do
4: compute $\mu$
5: find $\triangle d = (\triangle x, \triangle \mathbf{S}, \triangle \mathbf{Z})$ using a modified Newton method on the system (2.37) in order to
   get smaller residuals
6: evaluate the maximum lengths of the steps
   $\alpha_P = \max\{\alpha : \mathbf{S}^k + \alpha \triangle \mathbf{S} \succ 0\}$ and $\alpha_D = \max\{\alpha : \mathbf{Z}^k + \alpha \triangle \mathbf{Z} \succ 0\}$
7: update the current point by $(x^{k+1}, \mathbf{S}^{k+1}, \mathbf{Z}^{k+1}) = (x^k + \alpha_P \triangle x, \mathbf{S}^k + \alpha_P \triangle \mathbf{S}, \mathbf{Z}^k + \alpha_D \triangle \mathbf{Z})$
8: set $k = k + 1$
9: return an approximate optimal solution $(x^k, \mathbf{S}^k, \mathbf{Z}^k)$.
```

There exist many variants of primal-dual methods that employ different strategies to update the parameter μ and symmetrization processes (see, *e.g.*, [8, 54, 69, 139]). Now, we briefly describe one possible approach introduced in [54].

For a given point in the central path $(x, \mathbf{S}, \mathbf{Z})$, we can obtain μ by applying the trace operator to the perturbed complementary condition of the system (2.36):

$$\operatorname{tr}(\mathbf{ZS}) = \operatorname{tr}(\mu \mathbf{I}_s) = s\mu,$$

where s is the dimension of the matrices $\mathbf{S}$ and $\mathbf{Z}$. Therefore, $\mu = \frac{\operatorname{tr}(\mathbf{ZS})}{s}$. According to [54], the parameter μ can be chosen smaller:

$$\mu = \frac{\operatorname{tr}(\mathbf{ZS})}{2s}. \tag{2.38}$$

Since the matrix $\mathbf{ZS} - \mu\mathbf{I}$ in (2.36) is not necessarily symmetric, many approaches have been proposed to overcome this issue. One possible approach is to consider the linear transformation $H_P(\mathbf{M}) := \frac{1}{2}\left[\mathbf{PMP}^{-1} + (\mathbf{PMP}^{-1})^T\right]$, for any matrix $\mathbf{M}$, where the *scaling* matrix $\mathbf{P}$ is invertible and determines the symmetrization strategy [54, 69]. There are different choices for $\mathbf{P}$ that lead to different search directions (see, *e.g.*, [69, 139]).

According to [54], the third equation of the system (2.36) can be replaced by $H_P(\mathbf{ZS}) = \mu\mathbf{I}_s$. Linearization results in a direction $\triangle\mathbf{Z}$ that can be written as

$$\triangle\mathbf{Z} = \frac{\triangle\mathbf{Z} + \triangle\mathbf{Z}^T}{2}. \tag{2.39}$$

This direction approach is referred in the literature as the "HRVW", since it was introduced by Helmberg, Rendl, Vanderbei and Wolkowicz in [54]. The idea is to replace $\triangle\mathbf{Z}$ in the system (2.37) by (2.39) and to solve the resulting system to find the direction $\triangle d = (\triangle x, \triangle\mathbf{S}, \triangle\mathbf{Z})$.

Theoretically, SDP problems can be efficiently solved using standard interior point methods (see [8, 69, 149, 159]). Therefore, almost all general purpose SDP solvers implement modifications of such methods [42]. The main difference between them is in how the barrier parameter μ is updated and in the approach used for search direction (see, *e.g.*, [139], where several search directions are studied, and [8, 69, 103, 159], for details on primal-dual interior point methods).

Notice that interior point methods require a strictly feasible solution to exist, which make them impossible to be applied on SDP instances failing the Slater condition. To overcome this difficulty, a self-dual embedding technique can be applied. Self-dual embedding is a specific initialization strategy for obtaining strictly feasible problems [47, 69]. The general idea is as follows ([69, 120]):

- transform the given SDP problem into a larger one, by embedding both the primal and dual problems into a single problem and adding some extra variables;
- the larger problem satisfies the Slater condition;

- the optimal solution of the original SDP problem can be recovered from the optimal solution of the larger problem.

In [69], a self-dual embedding is proposed. Given the primal-dual pair of SDP problems (2.3) and (2.21), consider the following strictly feasible embedding problem:

$$\begin{array}{ll} \min & \theta\beta \\ \text{s.t.} & \operatorname{tr}(\mathbf{A}_i\mathbf{Z}) + \tau c_i - \theta \bar{c}_i = 0 \\ & \sum\limits_{i=1}^{n} \mathbf{A}_i x_i - \tau \mathbf{A}_0 + \tau \bar{\mathbf{A}}_0 + \mathbf{S} = 0 \\ & c^T x - \operatorname{tr}(\mathbf{A}_0\mathbf{Z}) + \theta\alpha - \rho = 0 \\ & -\bar{c}^T x + \operatorname{tr}(\bar{\mathbf{A}}_0\mathbf{Z}) - \tau\alpha - \nu = -\beta \\ & x \in \mathbb{R}^n, \mathbf{S} \succeq 0, \mathbf{Z} \succeq 0, \tau \geq 0, \theta \geq 0, \rho \geq 0, \nu \geq 0, \end{array} \tag{2.40}$$

where, for $i = 1, ..., n$, $\bar{c}_i := c_i - \operatorname{tr}(\mathbf{A}_i)$, $\bar{\mathbf{A}}_0 := \mathbf{A}_0 + \mathbf{I}_s$, $\alpha := 1 + \operatorname{tr}(\mathbf{A}_0)$, and $\beta := n + 2$.

A strictly feasible initial solution is given by $x^0 = 0$, $\mathbf{S}^0 = \mathbf{Z}^0 = \mathbf{I}_s$ and $\tau^0 = \theta^0 = \rho^0 = \nu^0 = 0$.

According to [69], any solution $(x, \mathbf{S}, \mathbf{Z}, \tau, \theta, \rho, \nu)$ to (2.40) with $\tau > 0$ yields an optimal solution, which in turn yields a complementary solution $(x, \mathbf{S}, \mathbf{Z})$ to the primal-dual pair of SDP problems (2.3) and (2.21).

This technique is implemented in the SeDuMi solver [136].

It is worth mentioning that there exist variants of interior point methods that allow infeasible initial solutions. There are feasible-start methods, that require existence of strictly feasible initial solutions $\mathbf{S}$ and $\mathbf{Z}$ for the primal and dual problems (2.3) and (2.21), respectively, and infeasible-start methods, that require an initial primal-dual pair $(\mathbf{S}, \mathbf{Z})$ satisfying $\mathbf{S}, \mathbf{Z} \succ 0$ [69, 159]. An infeasible interior point method, "starts with an infeasible solution and works towards improving feasibility and optimality, simultaneously" [120].

There exist also some variants of primal-dual interior point methods that implement a predictor-corrector step. At each iteration, one determines how much of a decrease in the barrier parameter μ is possible and a predictor search direction is computed. Then, a corrector step is performed for keeping the iterates close to the central path (see, *e.g.*, [8, 69]).

According to [42], interior point methods can be rather slow on large-scale problems. However, they are easy to implement and work very well in practice for small to medium-scale problems.

Although interior point methods are the most popular choice for solving SDP problems, other methods are also known in the literature. Some nonlinear optimization methods can be generalized to solve SDP problems. For example, in [20], Burer and Monteiro proposed an augmented Lagrangian method for solving SDP problems.

Considering a primal SDP problem in the form (2.5) and the associated dual, in [20] it is assumed that both problems have nonempty optimal solution sets with zero duality gap. An equivalent nonlinear reformulation of the primal SDP problem is obtained by replacing the matrix variable $\mathbf{X}$ in (2.5) by an appropriate low-rank factorization of the form $\mathbf{X} = \mathbf{R}\mathbf{R}^T$, where $\mathbf{R}$ is a real $(s \times r)$ matrix, with $r \leq s$, yielding the following problem in the new matrix variable $\mathbf{R}$:

$$\begin{array}{ll} \min\limits_{\mathbf{R}\in\mathbb{R}^{s\times r}} & \operatorname{tr}\left(\mathbf{C}\mathbf{R}\mathbf{R}^T\right) \\ \text{s.t.} & \operatorname{tr}\left(\mathbf{A}_i\mathbf{R}\mathbf{R}^T\right) = b_i, \quad \forall i = 1, \ldots, n, \end{array} \tag{2.41}$$

Notice that the positive semidefiniteness constraint in (2.5) is implicit in (2.41), since any $\mathbf{X} \succeq 0$ can be factored as $\mathbf{R}\mathbf{R}^T$ for some $\mathbf{R}$. Such low-rank factorization is valid for some or all optimal solutions, and not for all feasible solutions and it reduces the number of variables.

Sufficient optimality conditions for the problem (2.41) can be formulated assuming that for a local minimum $\mathbf{R}^*$ the gradients of the constraints are linearly independent [20].

To solve the nonlinear problem (2.41), an augmented Lagrangian function with a penalty parameter is considered. To perform the unconstrained minimization of the augmented Lagrangian function w.r.t. $\mathbf{R}$, a first-order limited memory BFGS algorithm can be used [20]. Let $\mathbf{R}^*$ be an optimal solution for the nonlinear problem (2.41). Then, the optimal solution $\mathbf{X}^*$ for the original SDP problem can be easily obtained by $\mathbf{X}^* = \mathbf{R}^*\mathbf{R}^{*T}$.

It is worth mentioning that there exist efficient methods for solving general or particular classes of SDP problems, but all of them rely on assumptions of regularity (*e.g.*, in terms of the Slater condition) of the primal and/or the dual problems.

2.5 Numerical solution of SDP problems

2.5.1 Overview of existing SDP solvers

In [101] and [120], an overview of the major currently available solvers for conic problems, and in particular, for SDP, is presented. Almost all popular SDP solvers either have an interface or are written in MATLAB.

The most widely used publicly available software packages for solving SDP problems are CSDP [14], SDPA [162], SDPT3 [143], SeDuMi [136] and DSDP [12], which implement different modifications of interior point methods. They can handle small to medium-scale SDP problems with high accuracy, but some may require a considerable running time. The sparse structure of SDP problems and also their dimension (in the case of large-scale problems) can result into numerical difficulties. Experiments (*e.g.*, [23, 60]) also show that one can obtain different/wrong results using different solvers.

The solver CSDP [14] considers a SDP problem in the form (2.21) and its dual in the form (2.3). This solver implements a predictor-corrector variant of the primal-dual interior point method proposed in [54]. CSDP uses an infeasible interior point version and is quite competitive with other solvers. It has a MATLAB interface and is publicly available at [25].

SDPT3 [143] is another solver developed for solving conic problems in the form of semidefinite-quadratic-linear programs, which include the primal SDP problem in the form (2.5) and its dual (2.3) with maximization instead of minimization (see [101]). The most recent version of SDPT3 implements an infeasible primal-dual predictor-corrector path-following method, which is also an interior point method, and allows the user to choose a corrector step. This solver is one of the most widely used solvers for solving SDP problems. As well as the CSDP solver, it does not require feasible initial approximations. The basic code is written in MATLAB and is publicly available at [130].

The SeDuMi solver [136] (the latest version was released in 2010) considers a primal SDP problems in the form (2.5) and the corresponding dual in the form (2.3) with maximization instead of minimization. SeDuMi applies a self-dual embedding technique to get an initial solution. The existence of strictly feasible solutions is assumed, since this solver implements a modification of the primal-dual interior point method with a predictor-corrector scheme, in order to speed up the global convergence. This allows to admit iterates far from the central path and to use long steps. The SeDuMi solver is popular because it is publicly available at [131], can handle complex data, explore the sparsity of data, and solve relatively large-scale problems.

Both SeDuMi and SDPT3 solvers are included in CVX, a MATLAB working package for specifying and solving convex programs [26, 44], including SDP ones. Since CVX is a Matlab-based modelling system for convex optimization, it allows one to introduce a SDP problem in any form.

The solver DSDP [12] considers a SDP problem in the form (2.5) and its dual in the form (2.3) with maximization instead of minimization. DSDP implements a dual-scaling potential reduction interior point algorithm for SDP, assuming that the constraint matrices of the SDP problem are linearly independent and that there exist strictly primal and dual feasible solutions, *i.e.*, the Slater condition holds for both primal and dual problems. It uses only the dual solution to get a step direction [101]. The freely available code is written in C, but it has an MATLAB interface. DSDP reveals to be quite efficient and robust, exploiting the sparse structure of the problem. This solver can be found in [30].

The SDPA solver is one of the most efficient and stable SDP solvers [8, 162]. SDPA considers primal SDP problems in the form (2.3) and its dual in the form (2.21). SDPA implements a Mehrotra-type predictor-corrector infeasible primal-dual interior point method based on the method proposed in [54] and its main feature is that it fully exploits the sparsity of SDP problems. There exist some versions of the SDPA, including the SDPA-M which is suitable to work in the MATLAB environment. The SDPA can be found in [129].

There exist some commercial SDP solvers, such as Mosek [105], PENNON and PENSDP [72]. Mosek is a SDP solver based on a primal-dual interior point method with a predictor-corrector step, and PENNON and PENSDP are based on a generalized augmented Lagrangian method. Recently, a free MATLAB version of PENNON and PENSDP called PENLAB [35] was released.

Current SDP solvers are not very efficient on large-scale problems, and thus, there has been an increasing interest in developing new software that could handle the solution of large-scale SDP problems. Actually, the large-scale SDP problems arisen from practical applications are beyond the capabilities of a single processor, requiring a lot of computational time and memory. Therefore, many efforts have been applied to combine standard SDP methods with parallel computation [161]. For example, in [16], a new version of the CSDP is described as a parallel implementation of the primal-dual method on a shared memory system. Recently, new versions of the SDPA solver were presented to handle large-scale problems [8], namely, to improve the SDPARA solver, which was originally developed in 2003. This new version SDPARA 7.3.1 [161], is a parallel implementation to solve large-scale SDP problems and numerical results reported in [161] show that the new SDPARA finds solutions for extremely large SDP problems that other solvers can not solve.

2.5.2 An example of a nonregular SDP problem: numerical issues when the Slater regularity condition fails to hold

As it was already mentioned, to ensure strong duality in SDP, additional conditions on the constraints of the primal or dual problems, called constraint qualifications (CQ), are required. The most common CQ in SDP is the Slater condition. Under this condition, the solvers based on interior point methods can be applied. Otherwise, the application of these methods can result in numerical difficulties and the SDP solvers may fail to obtain an optimal solution. The following example shows that some numerical difficulties arise when the Slater condition does not hold, even for a very small SDP problem.

Example 11 *Consider the primal SDP problem*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} 0 & x_1 & 0 \\ x_1 & x_2 & 0 \\ 0 & 0 & x_1+1 \end{bmatrix} \succeq 0 \end{array} \tag{2.42}$$

and its dual

$$\begin{array}{ll} \max & -y_2 \\ \text{s.t.} & \begin{bmatrix} y_1 & \frac{1-y_2}{2} & 0 \\ \frac{1-y_2}{2} & 0 & 0 \\ 0 & 0 & y_2 \end{bmatrix} \succeq 0. \end{array} \tag{2.43}$$

Clearly, the feasible set of (2.42) *is* $\{x_1, x_2 \in \mathbb{R} : x_1 = 0 \wedge x_2 \geq 0\}$ *and the feasible set of the dual problem is* $\{y_1, y_2 \in \mathbb{R} : y_1 \geq 0 \wedge y_2 = 1\}$. *Evidently, both* (2.42) *and* (2.43) *do not satisfy the Slater condition.*

It is easy to see that the optimal solutions are $x_1^* = 0$, $x_2^* \geq 0$, *and* $y_1^* \geq 0$, $y_2^* = 1$ *and the optimal values of the above primal-dual pair of problems are* $p^* = 0$ *and* $d^* = -1$, *respectively. So, the duality gap is* $p^* - d^* = 1$.

The following tables show the results obtained while solving the primal problem (2.42) *using two different solvers, SDPT3 and SeDuMi, respectively, with the default options.*

Table 2.1: Numerical solution of the linear SDP problem (2.42) using SDPT3 4.0.

number of iterations	68
primal objective value	$-9.99999981e-1$
dual objective value	$-9.99998840e-1$
actual relative gap	$-3.80e-7$
x_1	$-1.1601e-6$
x_2	$2.8934e+8$
time (secs)	1.92

Table 2.2: Numerical solution of the linear SDP problem (2.42) using SeDuMi 1.34.

number of iterations	27
primal objective value	$-4.8109597859e-1$
dual objective value	$-3.3271398034e-1$
gap	$1.4838199825e-1$
x_1	-0.667286
x_2	$3.8005e+7$
time (secs)	0.9

Both solvers returned the warning message that they "solved the dual problem for improved efficiency".

A first observation is that although the primal objective function is x_1, the solvers return other values, but no information is provided in terms of how the objective function value was computed.

Observing the tables above, we can see that the solvers provided different results and the computed solutions are quite far from the true ones.

The SDP problems of the type presented in the example above are sometimes called "nasty" [45, 136], since they do not behave well on SDP solvers, *i.e.*, a standard SDP solver applied to such problems may be unable to provide accurate solutions.

Recall that the SDP problems in the example do not satisfy the Slater condition. It is easy to verify that strong duality does not hold here too.

According to [42], to guarantee that a given SDP problem is (approximately) solvable in polynomial time, one has to make sure that it is "well-behaved" in some sense. Current SDP solvers assume that the SDP problem must satisfy certain conditions and when such conditions fail to hold, one can not expect accurate results. These conditions are usually related to the problem regularity.

Motivated by the example above, the following question naturally arises:

> "What conditions guarantee that the given SDP problem can be correctly solved by a given solver?"

In this thesis, we will focus our attention on study such conditions. We will show that different regularity conditions can be formulated for SDP problems and that the strongest is the Slater condition. We will also study how to verify if a given SDP problem satisfies the Slater condition and how to proceed when the Slater condition fails to hold. We will provide a simple presolving numerical procedure to check the Slater condition for SDP problems that can then be incorporated into standard solvers. The procedure is implemented in a publicly available MATLAB code.

In the case of failure of the Slater condition, there are two possible ways to overcome this difficulty: to apply a presolving technique in order to transform the SDP problem into another one satisfying the Slater condition, such as a self-dual embedding technique or a preprocessing based on facial reduction (*e.g.*, [22, 23, 48, 69]), or to develop a SDP method with no regularity assumption, for instance, a method based on special optimality conditions that do not rely on a constraint qualification (usually called *CQ-free*). However, it is worth noting the following points. Despite the desirable theoretical properties of the self-dual embedding techniques [69] applied to SDP problems failing the Slater condition, in practice, the SDP solvers still run into numerical difficulties. On the other hand, the algorithmic implementation of facial reduction to obtain a smaller regularized problem for which the Slater condition holds is not yet extended for all classes of SDP problems (see, *e.g.*, [22, 23]). The last scenario of developing methods with no regularity assumptions is quite difficult to obtain. There exists the SDP method proposed in [37], but, unfortunately, neither numerical implementation nor results are reported. There have been proposed in the literature several CQ-free optimality conditions for SDP (*e.g.*, [48, 62, 63, 78, 125, 126]), however, to the best of our knowledge, no SDP method exists implementing this type of optimality conditions.

Chapter 3

Regularity in semidefinite programming

In SDP, there exist different notions of regularity: regularity from the viewpoint of the topology of the feasible set and the constraint functions – constraint qualifications; regularity from the viewpoint of stability and perturbation analysis – well-posedness; and regularity in terms of the strong duality – so-called good behaviour. The aim of this chapter is to study these notions of regularity and the relationships between them. We study the existing numerical procedures to test regularity of problems and present a theoretical algorithm that can be used to test the Slater condition.

3.1 Constraint qualifications

The term constraint qualification (CQ) was first introduced in [82]. Constraint qualifications are special conditions that the constraints of a given optimization problem should satisfy to guarantee that the first-order necessary optimality conditions – the KKT optimality conditions – are satisfied. The CQs are essential for deriving primal-dual characterizations of optimal solutions and play an important role in duality theory, sensitivity and stability analysis, and convergence properties of computational algorithms [34, 68, 134].

An optimization problem that satisfies a CQ is usually called regular [55] and therefore, a problem whose constraints do not satisfy any CQ is nonregular.

The most widely used CQ in SDP is the Slater condition. Other CQs can be found in, *e.g.*, [128, 132, 142, 159].

3.1.1 The Slater condition

In what follows, we will consider the linear SDP problems in the form (2.2) or in the equivalent forms (2.3) and (2.4), and the corresponding dual problems. We will show that important properties of these problems are guaranteed if the Slater condition is satisfied.

Given a SDP problem in the form (2.2), the Definition 2 defines the Slater condition for this problem, and the Definition 3 defines the Slater condition for its dual problem.

The Slater condition for the SDP problem in the form (2.4) can be given as follows.

Definition 5 *The SDP problem* (2.4) *satisfies the Slater condition if there exist* $\bar{x} \in \mathbb{R}^n$ *such that* $\mathcal{A}(\bar{x}) \in \operatorname{int}(-\mathcal{P}(s))$.

It was already mentioned in the Chapter 2 that when the Slater condition is satisfied for the problem (2.3) and its dual (2.21), then the KKT optimality conditions of the Theorem 7 are both necessary and sufficient optimality conditions.

The Slater condition ensures that the strong duality property holds. Thus, given primal and dual optimal solutions $(x^*, \mathbf{S}^*)$ and $\mathbf{Z}^*$ of the problems (2.3) and (2.21), respectively, the complementarity condition $\operatorname{tr}(\mathbf{S}^*\mathbf{Z}^*) = 0$ holds, and from the Theorem 6, we have $\mathbf{S}^*\mathbf{Z}^* = 0$. This implies that the matrices $\mathbf{S}^*$ and $\mathbf{Z}^*$ commute, sharing a set of orthonormal eigenvectors [5, 106].

Consider the following definitions from [106].

Definition 6 *A primal optimal solution* $(x^*, \mathbf{S}^*)$ *of a SDP problem in the form* (2.3) *and a dual solution* $\mathbf{Z}^*$ *of the dual problem are said to satisfy strict complementarity if*

$$\operatorname{rank}(\mathbf{S}^*) + \operatorname{rank}(\mathbf{Z}^*) = s.$$

Definition 7 *Let* $(x^*, \mathbf{S}^*)$ *and* $\mathbf{Z}^*$ *be primal and dual optimal solutions of the SDP problems* (2.3) *and* (2.21) *satisfying strict complementarity. Let* $\operatorname{rank}(\mathbf{S}^*) = r$ *and* $\mathbf{Q}$ *be a matrix whose columns form the orthonormal set of eigenvectors for* $\mathbf{S}^*$ *and* $\mathbf{Z}^*$. *Suppose that* $\mathbf{Q} = \left[\begin{array}{c:c} \mathbf{Q}_1 & \mathbf{Q}_2 \end{array}\right]$, *where* $\mathbf{Q}_1$ *is* $s \times (s-r)$ *and* $\mathbf{Q}_2$ *is* $s \times r$ *matrices corresponding to the zero and nonzero eigenvalues of* $\mathbf{S}^*$, *respectively. Then,*

- *a solution* $(x^*, \mathbf{S}^*)$ *is said to be primal nondegenerate if the matrices* $\mathbf{D}_i = \mathbf{Q}_1^T\mathbf{A}_i\mathbf{Q}_1$, $i = 1, ..., n$, *span* $\mathcal{S}(s-r)$, *and in this case we say that primal nondegeneracy holds;*
- *a solution* $\mathbf{Z}^*$ *is said to be dual nondegenerate if the matrices*

$$\mathbf{B}_i = \left[\begin{array}{c:c} \mathbf{Q}_1^T\mathbf{A}_i\mathbf{Q}_1 & \mathbf{Q}_1^T\mathbf{A}_i\mathbf{Q}_2 \\ \hdashline \mathbf{Q}_2^T\mathbf{A}_i\mathbf{Q}_1 & \mathbf{0} \end{array}\right], i = 1, ..., n,$$

  are linearly independent in $\mathcal{S}(s)$, *and in this case we say that dual nondegeneracy holds.*

Under the assumption that the primal and dual SDP problems have solutions satisfying strict complementarity and nondegeneracy, it can be proved that the primal and dual solutions are unique [5].

Theorem 10 *Let the SDP problem* (2.3) *and its dual* (2.21) *satisfy the Slater condition. If there exists a primal nondegenerate optimal solution* $(x^*, \mathbf{S}^*)$, *then there exists a unique optimal dual solution* $\mathbf{Z}^*$. *Analogously, if* $\mathbf{Z}^*$ *is a dual nondegenerate optimal solution, then there exists a unique primal optimal solution* $(x^*, \mathbf{S}^*)$.

Theorem 11 *Let the SDP problem* (2.3) *and its dual* (2.21) *satisfy the Slater condition. Suppose that* $(x^*, \mathbf{S}^*)$ *and* $\mathbf{Z}^*$ *are primal and dual optimal solutions satisfying strict complementarity. If the primal solution is unique, then dual nondegeneracy must hold, and if the dual solution is unique, then primal nongeneracy must hold.*

Therefore, if the primal-dual pair of linear SDP problems has optimal solutions satisfying strict complementarity, and primal and dual nondegeneracy, then the primal and dual optimal solutions are unique.

The known primal-dual interior point SDP methods assume that the Slater condition holds for both primal and dual problems [22, 37, 69, 161]. It was shown in the Chapter 2 (Theorem 9) that the Slater condition plays an important role on the well-definition of the central path (2.34), which is essential in interior point SDP methods, guaranteeing their stability.

Consider the following definition of genericity [5, 31, 113].

Definition 8 *Let the instances of a certain problem class can be parametrized in some way. A certain property is called generic if it holds for almost all instances, that is, the set of parameters describing the problem instances failing this property has measure zero.*

In [5], it is shown that the primal and dual nondegeneracy and strict complementarity hold generically for SDP problems.

Theorem 12 *Under the Slater condition, primal and dual nondegeneracy are generic properties of linear SDP problems.*

Theorem 13 *Under the Slater condition, strict complementarity is a generic property of linear SDP problems.*

These results were proved in [5] using the notion of transversality from differential topology. In [113], the genericity of strict complementarity and primal and dual nondegeneracy for general conic convex programs were proved using Hausdorff measures, certain properties of the boundaries of convex sets and assuming zero duality gap.

In [31] (Theorem 3.2), it is shown that the Slater condition is itself a generic property for linear conic problems, which include SDP, and thus, we can formulate the following theorem for the specific case of linear SDP.

Theorem 14 *For almost all instances of SDP problems in the form* (2.2)*, either one of the following holds:*

1. *the feasible set* $\mathcal{X}$ *of* (2.2) *is empty, i.e.,* (2.2) *is infeasible,*

2. *the Slater condition holds for* (2.2).

To prove that the Slater condition holds generically for linear conic problems, the authors in [31] used results from measure theory, in particular, the fact that the boundary of a convex set has measure zero.

Notice that given a feasible SDP problem for which the Slater condition fails to hold, it is clear that all its feasible solutions lie on the boundary of the feasible set.

However, the genericity of the Slater condition in linear conic programming, and in particular, in SDP, does not mean that a given SDP problem will satisfy this condition. In practice, there are many SDP instances for which the Slater condition fails to hold and many authors have drawn their attention to the study of both theoretical and numerical difficulties that occur due to the failure of the Slater condition (see, *e.g.*, [23, 38, 49, 61, 111, 152, 155]). To mention a few, in [22, 23], various SDP instances for which the Slater condition fails are provided and a reformulation based on facial reduction is described. In [152], it is shown that the failure of the Slater condition, at least for one of the problems of the SDP primal-dual pair, leads to numerical difficulties when using standard SDP solvers, presenting a "discrepancy between the true and computed optimal values".

When the Slater condition is not satisfied, the optimality conditions of the Theorem 7 may fail to characterize optimality of a feasible solution [23, 78, 156]. Consequently, SDP methods may run into numerical difficulties and the solutions obtained by these methods may be not correct.

Current SDP solvers do not check numerically the fulfilment of the Slater condition before solving the problem, but work under its assumption. The main aim of this thesis is to develop a presolving numerical tool to verify if a given SDP problem satisfies the Slater regularity condition to warn users that the computed solution will be trustful or not.

3.1.2 Other constraint qualifications in SDP

In [159], Shapiro introduced another CQ for SDP problems which he called the *regularity condition.*

Definition 9 *A SDP problem in the form* (2.4) *satisfies the regularity condition if*

$$0 \in \operatorname{int}\left(\{\mathcal{A}(\mathbb{R}^n) + \mathcal{P}(s)\}\right), \tag{3.1}$$

*where $\mathcal{A}(\mathbb{R}^n)$ denotes the set $\{\mathcal{A}(x) : \mathcal{A}(x) \in \mathcal{S}(s), \forall x \in \mathbb{R}^n\}$ and "*int*" stands for the interior of a set.*

The following results were obtained in [159] for a linear (convex) SDP problem (2.4).

Theorem 15 *Suppose that the regularity condition* (3.1) *holds for the primal SDP problem* (2.4)*. Then the duality gap between the primal and dual problems vanishes and, if their common optimal value is finite, then the set of optimal solutions of the dual problem is nonempty and bounded.*

Theorem 16 *Suppose that the dual to the SDP problem* (2.4) *has a nonempty and bounded set of optimal solutions. Then the regularity condition* (3.1) *holds and there is no duality gap between the primal and dual problems.*

For the dual SDP problem (2.21), the regularity condition introduced by Shapiro is as follows.

Definition 10 *The dual SDP problem* (2.21) *satisfies the regularity condition if*

$$0 \in \operatorname{int}\left(\{x \in \mathbb{R}^n : x_i = \operatorname{tr}(\mathbf{Z}\mathbf{A}_i) + c_i, i = 1, ..., n, \mathbf{Z} \succeq 0\}\right). \tag{3.2}$$

The following result is proved in [159].

Theorem 17 *Suppose that the dual SDP problem* (2.21) *satisfies the regularity condition* (3.2)*. Then there is no duality gap between the primal and the dual problems and the primal problem* (2.2) *has a nonempty and bounded set of optimal solutions. Conversely, if the primal SDP problem* (2.2) *has a nonempty and bounded set of optimal solutions, then its dual* (2.21) *satisfies the regularity condition* (3.2) *and the duality gap vanished.*

Notice that the CQ (3.2) holds for (2.21) if there exists a positive definite matrix $\mathbf{Z}$ such that $\operatorname{tr}(\mathbf{Z}\mathbf{A}_i) + c_i = 0$, $i = 1, ..., n$.

By the Theorems 15, 16 and 17, one can conclude that the regularity condition introduced by Shapiro ensures that strong duality holds for the primal-dual pair of SDP problems, and thus, the KKT optimality conditions can be applied.

Another CQ was introduced in [128] and called Robinson CQ. For a SDP problem in the form (2.4), the Robinson CQ can be defined as follows [159].

Definition 11 *A SDP problem in the form* (2.4) *satisfies the Robinson CQ at a feasible point* $\bar{x} \in \mathbb{R}^n$ *if*

$$0 \in \operatorname{int}\left(\{\mathcal{A}(\bar{x}) + D\mathcal{A}(\bar{x})\mathbb{R}^n + \mathcal{P}(s)\}\right), \tag{3.3}$$

where $D\mathcal{A}(\bar{x})$ *is the differential of* $\mathcal{A}(.)$ *at* $\bar{x}$*, that is* $D\mathcal{A}(\bar{x})h$ *is a linear function of* $h \in \mathbb{R}^n$ *given by* $D\mathcal{A}(\bar{x})h = \sum\limits_{i=1}^{n} h_i \mathcal{A}_i^d(\bar{x})$ *where* $\mathcal{A}_i^d(\bar{x}) := \frac{\partial \mathcal{A}(\bar{x})}{\partial x_i}$.

Shapiro showed in [159] that the Robinson CQ (3.3) can be derived by linearizing the regularity condition (3.1) at a point $\bar{x} \in \mathbb{R}^n$.

This result permits to conclude that the Robinson CQ also ensures a zero duality gap between the primal and the dual SDP problems, and thus, the KKT optimality conditions can be applied.

It is also shown in [159] that since $-\mathcal{P}(s)$ has a nonempty interior, the Robinson CQ (3.3) at $\bar{x} \in \mathbb{R}^n$ is equivalent to the existence of a vector $\bar{h} \in \mathbb{R}^n$ such that

$$\mathcal{A}(\bar{x}) + D\mathcal{A}(\bar{x})\bar{h} \in \operatorname{int}(-\mathcal{P}(s)),$$

which is equivalent to the Mangasarian-Fromovitz CQ for (2.4).

Definition 12 *A SDP problem in the form* (2.4) *satisfies the Mangasarian-Fromovitz CQ at a feasible point* $\bar{x} \in \mathbb{R}^n$ *if there exists a vector* $\bar{h} \in \mathbb{R}^n$ *such that*

$$\mathcal{A}(\bar{x}) + D\mathcal{A}(\bar{x})\bar{h} \prec 0. \tag{3.4}$$

It can be shown that, if the Mangasarian-Fromovitz CQ holds at a stationary point x^*, then the set $\Lambda(x^*)$ of all Lagrange multiplier matrices $\mathbf{Z}^*$ satisfying the conditions (2.25) is nonempty and bounded. Thus, the KKT optimality conditions can be applied.

3.1.3 Relationships between different constraint qualifications in SDP

The following proposition establishes the relationships existing among the constraint qualifications introduced above for SDP problems.

Proposition 5 *For a linear SDP problem in the form* (2.4)*, the following equivalences hold.*

a) Regularity condition (3.1) $\Leftrightarrow$ *Slater condition,*

b) Robinson CQ (3.1) $\Leftrightarrow$ *Mangasarian-Fromovitz CQ* (3.4)*,*

c) Slater condition (3.3) $\Leftrightarrow$ *Mangasarian-Fromovitz CQ* (3.4)*.*

Notice that some of these statements have already been proved in the literature. The equivalence *a)* was proved in [159] (Proposition 4.1.4):

Proposition 6 *For the particular case of convex SDP, the regularity condition* (3.1) *is equivalent to the Slater condition.*

This equivalence is valid since for the SDP problems in the form (2.4), the mapping $\mathcal{A}(x)$ is convex and the cone $-\mathcal{P}(s)$ has a nonempty interior.

We shall consider a reformulation of the SDP problem (2.2). Let us write the linear SDP problem (2.2) in the equivalent form

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & l^T \mathcal{A}(x) l \leq 0, \quad \forall l \in L := \{l \in \mathbb{R}^s : \|l\|_2 = 1\}, \end{array} \tag{3.5}$$

where the set L can be considered as an (infinite) index set. It is easy to see that this problem has an infinite number of constraints, and thus is a convex semi-infinite programming (SIP) problem.

It is easy to verify that the feasible set of the problem (3.5) coincides with the feasible set of the SDP problem (2.2):

$$\left\{x \in \mathbb{R}^n : l^T \mathcal{A}(x) l \leq 0, \forall l \in L\right\} = \{x \in \mathbb{R}^n : \mathcal{A}(x) \preceq 0\} = \mathcal{X}. \tag{3.6}$$

Considering that $g(x,l) = l^T \mathcal{A}(x) l$, we say that the problem (3.5) satisfies the Mangasarian-Fromovitz CQ at $\bar{x} \in \mathbb{R}^n$ if there exists a vector $\bar{h} \in \mathbb{R}^n$ such that $\bar{h}\nabla g(\bar{x},l) < 0$ for all $l \in \triangle(\bar{x})$, where $\triangle(\bar{x}) = \{l \in L : g(\bar{x},l) = 0\}$ is the set of active constraints at $\bar{x}$. The Slater condition holds for the problem (3.5) if there exists $\bar{x} \in \mathbb{R}^n$ such that $g(\bar{x},l) < 0$ for all $l \in L$. These CQs are equivalent according to the following proposition obtained by reformulation of the Proposition 3.1 in [132].

Proposition 7 *Suppose that the SIP problem* (3.5) *is convex. Then the Slater condition implies the Mangasarian-Fromovitz CQ at every solution $\bar{x}$. Conversely, if the Mangasarian-Fromovitz CQ is satisfied at a solution $\bar{x}$, then the Slater condition holds.*

Proof of the Proposition 5. The equivalence *a)* follows from the Proposition 6.

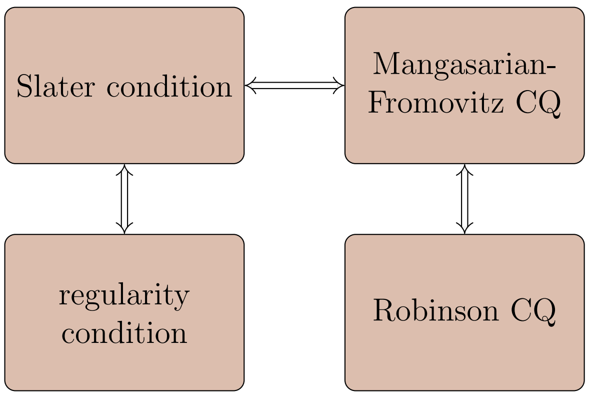
The equivalence *b)* is proved in [159]. Notice that the mapping $h \to \mathcal{A}(\bar{x}) + D\mathcal{A}(\bar{x})h$ is convex and the cone $-\mathcal{P}(s)$ has a nonempty interior, and thus, using the Proposition 6, it is easy to see that the Robinson CQ (3.1) can be written as (3.4), which is the MFCQ for the SDP problem (2.4).

To prove the equivalence *c)*, let us consider the SDP problem in the form (2.4). First, notice that this problem can be written in the form (2.2), which in turn is equivalent to the SIP problem (3.5).

If the SDP problem (2.4) satisfies the Slater condition, then it is easy to see that its reformulation as the SIP problem (3.5) also satisfies the Slater condition. Indeed, if problem (2.4), which can be equivalently written in the form (2.2), satisfies the Slater condition, then there exists $\bar{x} \in \mathbb{R}^n$ such that $\mathcal{A}(\bar{x}) \prec 0$. Since the feasible sets of the problems (2.4), (2.2) and (3.5) coincide, then (3.5) also satisfies the Slater condition. Hence, the problems (2.4), (2.2) and (3.5) satisfy the Slater condition simultaneously. Similarly, the Mangasarian-Fromovitz CQ holds for the equivalent problems (2.4), (2.2) and (3.5), simultaneously.

It immediately follows from the Proposition (7) that for the SIP problem (3.5), the Slater condition is equivalent to the Mangasarian-Fromovitz CQ. Since (3.5) is equivalent to (2.4), then for the SDP problem (2.4) the Slater condition is equivalent to the Mangasarian-Fromovitz CQ. ■

On the basis of the results obtained above, we can present the following diagram illustrating the relations between the studied CQs for convex SDP problems.



3.1.4 Testing the Slater condition

From the results of the previous section, we conclude that to verify any of the above CQs for linear SDP problems, it is enough to verify the Slater condition. The Slater condition is a topological property of the feasible set of a problem and is not connected with a given feasible solution, unlike the Robinson or the Mangasarian-Fromovitz CQs. In [142], it is mentioned that “in terms of worst-case performance, deciding whether Slater condition holds for a given SDP problem seems no easier than solving an SDP problem". In what follows, we will present an approach that permits to verify if a feasible SDP problem satisfies the Slater condition and describe a theoretical procedure that can be used for testing regularity.

Subspace of immobile indices for SDP

The suggested approach to verify the Slater condition for linear SDP problems is based on the notion of *subspace of immobile indices* for SDP proposed in [78].

Let us introduce the following definition from [78].

Definition 13 *Given the linear SDP problem* (2.2)*, the subspace of* $\mathbb{R}^s$ *defined by*

$$\mathcal{M} := \left\{ l \in \mathbb{R}^s : l^T \mathcal{A}(x) l = 0, \forall x \in \mathcal{X} \right\} \tag{3.7}$$

is called the subspace of immobile indices for (2.2)*.*

We will show that the fulfilment of the Slater condition in a given SDP problem is ensured if and only if the subspace of immobile indices is null.

The subspace $\mathcal{M}$ of immobile indices for SDP is closely related to the notion of *immobile indices* for SIP introduced in [33, 78].

Let us consider the SDP problem (2.2) equivalently written as a SIP problem in the form (3.5).

Definition 14 *Given a convex SIP problem in the form* (3.5)*, an index* $l^* \in L$ *is called immobile if* $l^{*T} \mathcal{A}(x) l^* = 0$*, for all* $x \in \mathcal{X}$*.*

The *set of immobile indices* for the SIP problem (3.5) is given by

$$L^* = \left\{ l \in L : l^T \mathcal{A}(x) l = 0, \forall x \in \mathcal{X} \right\}.$$

It is evident that, for a pair of equivalent problems (2.2) and (3.5), the set L^* can be presented in the form

$$L^* = L \cap \mathcal{M}, \tag{3.8}$$

where $\mathcal{M}$ is the subspace of immobile indices for the SDP problem (2.2) defined in (3.7). Notice that $\mathcal{M} \neq \emptyset$, since it has always the null vector as element.

Consider the following definition.

Definition 15 *The SIP problem* (3.5) *satisfies the Slater condition if there exists a feasible point* $\bar{x} \in \mathbb{R}^n$ *such that the inequalities* $l^T \mathcal{A}(\bar{x}) l < 0$ *hold for all indices* $l \in L$*.*

It is proved in the previous section that the equivalent problems (2.2) and (3.5) satisfy or not the Slater condition, simultaneously.

Using the fact that the feasible set of the SIP problem (3.5) coincides with the feasible set $\mathcal{X}$ of the SDP problem (2.2) and that it can be written in the form

$$\mathcal{X} = \left\{ x \in \mathbb{R}^n : l^T \mathcal{A}(x) l < 0, \forall l \in L \setminus L^*; l^T \mathcal{A}(x) l = 0, \forall l \in L^* \right\}, \tag{3.9}$$

the following propositions were proved in [78].

Proposition 8 *The convex SIP problem* (3.5) *satisfies the Slater condition if and only if the set L^* is empty.*

The following proposition is an immediate consequence of the Proposition 8.

Proposition 9 *The SDP problem* (2.2) *satisfies the Slater condition if and only if the set L^* in the equivalent SIP problem* (3.5) *is empty.*

Considering (3.8) and taking into account that $L = \{l \in \mathbb{R}^s : \|l\|_2 = 1\} \neq \emptyset$, we conclude that the set L^* of immobile indices in the SIP problem (3.5) is empty if and only if the subspace of immobile indices $\mathcal{M}$ is null. Combining this result with Proposition 9, we can then formulate the following theorem.

Theorem 18 *The SDP problem* (2.2) *satisfies the Slater condition if and only if the subspace of immobile indices $\mathcal{M}$ is null, i.e., $\mathcal{M} = \{0\}$.*

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 have called the *irregularity degree.*

Definition 16 *The dimension of a basis of the immobile index subspace $\mathcal{M}$ for the SDP problem* (2.2)*, denoted by s^*, is called irregularity degree of this problem.*

This definition permits to classify SDP problems in the form (2.2) 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 nonstrict feasibility, *i.e.*, it proves the failure of the Slater condition.

We have shown in the Section 2.3 that in the absence of the Slater condition, the characterization of optimality of solutions using the KKT conditions may fail. In what follows, optimality conditions that are valid for any SDP problem in the form (2.2), satisfying or not the Slater condition, are formulated.

Considering that $\mathbf{M} = (m_i, i = 1, ..., s^*)$ is a matrix whose columns form a basis of the subspace $\mathcal{M}$, according to [78], the feasible set of the SDP problem (2.2) can be presented in the form

$$\mathcal{X} = \left\{x \in \mathbb{R}^n : \mathcal{A}(x)m_i = 0, i = 1, ..., s^*, l^T\mathcal{A}(x)l \leq 0, \forall l \in \mathcal{M}^\perp\right\},$$

where $\mathcal{M}^\perp$ is the orthogonal complement of the subspace $\mathcal{M}$ in $\mathbb{R}^s$. It can be rewritten in the matrix form

$$\mathcal{X} = \left\{x \in \mathbb{R}^n : \mathcal{A}(x)\mathbf{M} = 0, \mathbf{N}^T\mathcal{A}(x)\mathbf{N} \preceq 0\right\},$$

where $\mathbf{N} \in \mathbb{R}^{s \times p^*}$, with $p^* = s - s^*$, is a basic matrix of $\mathcal{M}^\perp$.

In [78], a CQ-free optimality criterion is formulated based on the explicit determination of the subspace of immobile indices $\mathcal{M}$. For a SDP problem in the form (2.2), this criterion takes the form of the following theorem.

Theorem 19 *A feasible solution $x^* \in \mathbb{R}^n$ is optimal for the SDP problem* (2.2) *if and only if there exist vectors $\theta^k \in \Theta(x^*)$ and $\gamma_i \in \mathbb{R}^s$, with $i = 1, ..., s^*$, such that*

$$\sum_{k=1}^{p^*} \theta^{k^T}\mathbf{N}^T\mathbf{A}_j\mathbf{N}\theta^k + c_j + \sum_{i=1}^{s^*} \gamma_i^T\mathbf{A}_j m_i = 0, j = 1, ..., n, \tag{3.10}$$

where $\Theta(x^*) = \left\{\theta^k \in \mathbb{R}^{p^*} : \theta^k \neq 0, \theta^{k^T}\mathbf{N}^T\mathcal{A}(x^*)\mathbf{N}\theta^k = 0\right\}$.

Notice that this optimality criterion uses the vectors of the basis of both the subspace of immobile indices and its orthogonal complement. To find a basis of the subspace $\mathcal{M}$, a constructive algorithm called DIIS (Determination of the Immobile Index Subspace) is described and justified in [78].

In this thesis, we develop a numerical procedure based on the DIIS algorithm that permits to verify if a given SDP problem satisfies the Slater condition. This procedure is implemented in MATLAB and tested on several SDP instances, including instances from the SDPLIB suite [15].

The DIIS algorithm

Consider a linear SDP problem (2.2) with nonempty feasible set. The DIIS algorithm proposed in [78] constructs a basis of the subspace of immobile indices $\mathcal{M}$, which forms a matrix $\mathbf{M} = (m_i, i = 1, ..., s^*)$. At the k-th iteration, let I^k denote a set of indices and M^k denote a set of vectors. Suppose that $s > 1$, with $s \in \mathbb{N}$.

The brief description of the algorithm is as follows.

Algorithm 2 DIIS

```
input: $\mathbf{A}_j$, $j = 0, 1, ..., n$, $s \times s$ symmetric real matrices.
output: $\mathbf{M}$, basis of the subspace of immobile indices, whose elements are $m_i$, $i = 1, ..., s^*$.
 1: set $k := 1$, $I^1 := \emptyset$ and $M^1 := \emptyset$.
 2: repeat
 3:     given $k$, $I^k$, $M^k$:
 4:     set $p_k := s - |I^k|$
 5:     solve the quadratic system
            $$\begin{cases} \sum_{i=1}^{p_k} l_i^T \mathbf{A}_j l_i + \sum_{i \in I^k} \gamma_i^T \mathbf{A}_j m_i = 0, \ j = 0, 1, \dots, n, \\ \sum_{i=1}^{p_k} \|l_i\|^2 = 1, \\ l_i^T m_j = 0 \ , \ j \in I^k, \ i = 1, \dots, \ p_k, \end{cases} \quad (3.11)$$
        where $l_i \in \mathbb{R}^s, i = 1, ..., p_k$ and $\gamma_i \in \mathbb{R}^s, i \in I^k$
 6:     if system (3.11) does not have a solution, then stop
 7:     else given the solution $\{l_i \in \mathbb{R}^s, i = 1, \dots, p_k, \gamma_i \in \mathbb{R}^s, i \in I^k\}$ of (3.11):
 8:         construct the maximal subset of linearly independent vectors
                $$\{m_1, \dots, m_{s_k}\} \subset \{l_1, \dots, l_{p_k}\}$$
 9:         update:
10:             $\triangle I^k := \{|I^k| + 1, \dots, |I^k| + s_k\}$,
11:             $M^{k+1} := M^k \cup \{m_j, \ j \in \triangle I^k\}$, where for each $j \in \triangle I^k$, $m_j = m_i$, $i = 1, ..., s_k$,
12:             $I^{k+1} := I^k \cup \triangle I^k$.
13:         set $k := k + 1$
14: until system (3.11) does not have a solution
15: given $M^k$: construct $\mathbf{M}$, whose columns are the vectors from $M^k$
16: return $\mathbf{M}$.
```

In [78], it is proved that the DIIS algorithm founds a basis of the immobile index subspace $\mathcal{M}$ in a finite number of iterations. It should be noticed here that the DIIS algorithm is a theoretical algorithm and its numerical implementation would be an important tool for verifying the regularity of SDP problems and applying new CQ-free optimality conditions.

Considering a SDP problem in the form (2.2) and the results presented in the Section 3.1.4, we can make the following conclusions:

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

In [89], we have made a first attempt of a numerical implementation of the DIIS algorithm, but some difficulties have arisen in solving the system (3.11). The main procedure on each iteration of the DIIS algorithm consists in solving the system of quadratic equations (3.11). 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. Only one iteration of the DIIS algorithm is enough to verify if a given SDP problem satisfies the Slater condition and in this case, one has to solve a system with s vector variables and $n + 2$ equations. The DIIS algorithm should stop when the system (3.11) is inconsistent. On implementing the DIIS algorithm numerically, both the procedure of constructing a set of linearly independent vectors, and that of determining whether or not the system (3.11) is consistent can be difficult tasks.

In the next chapter, we will describe a numerical procedure for testing the Slater condition for SDP problems, but, before proceeding, we provide some examples illustrating how the DIIS algorithm works. We consider two SDP problem instances of small size.

Example 12 *Consider the SDP problem*

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^3} & x_1 + 2x_2 + 3x_3 \\ \text{s.t.} & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} x_1 + \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} x_2 + \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} x_3 + \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \preceq 0. \end{array} \tag{3.12}$$

Here, $\mathbf{A}_i$, $i = 0, 1, 2, 3$, are 2×2 symmetric matrices given by

$$\mathbf{A}_0 = \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix}, \ \mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \ \mathbf{A}_2 = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \text{ and } \mathbf{A}_3 = \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix}.$$

Set $k = 1$, $I^1 = \emptyset$ and $M^1 = \emptyset$. Then, $p_1 = s - |I^1| = 2$.
In this case, the system (3.11) *takes the form*

$$\begin{cases} \sum\limits_{i=1}^{2} l_i^T \mathbf{A}_j l_i = 0 \ , \quad j = 0, 1, 2, 3, \\ \sum\limits_{i=1}^{2} \|l_i\|^2 = 1, \end{cases}$$

i.e.,

$$\begin{cases} l_1^T \mathbf{A}_0 l_1 + l_2^T \mathbf{A}_0 l_2 = 0 \\ l_1^T \mathbf{A}_1 l_1 + l_2^T \mathbf{A}_1 l_2 = 0 \\ l_1^T \mathbf{A}_2 l_1 + l_2^T \mathbf{A}_2 l_2 = 0 \\ l_1^T \mathbf{A}_3 l_1 + l_2^T \mathbf{A}_3 l_2 = 0 \\ \|l_1\|^2 + \|l_2\|^2 = 1, \end{cases}$$

where $l_1 = (l_{11}, l_{12})^T$ and $l_2 = (l_{21}, l_{22})^T$.

This system can be rewritten in a more explicit form, in terms of components of vectors l_1 and l_2, resulting in the following one:

$$\begin{cases} \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} l_{21} \\ l_{22} \end{bmatrix} = 0 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{21} \\ l_{22} \end{bmatrix} = 0 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{21} \\ l_{22} \end{bmatrix} = 0 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{21} \\ l_{22} \end{bmatrix} = 0 \\ {l_{11}}^2 + {l_{12}}^2 + {l_{21}}^2 + {l_{22}}^2 = 1, \end{cases}$$

or, equivalently,

$$\begin{cases} -{l_{11}}^2 - 2l_{11}l_{12} - {l_{21}}^2 - 2l_{21}l_{22} = 0 \\ l_{11}l_{12} + l_{21}l_{22} = 0 \\ {l_{11}}^2 + l_{11}l_{12} + {l_{21}}^2 + l_{21}l_{22} = 0 \\ 3{l_{11}}^2 + 2l_{11}l_{12} + 3{l_{21}}^2 + 2l_{21}l_{22} = 0 \\ {l_{11}}^2 + {l_{12}}^2 + {l_{21}}^2 + {l_{22}}^2 = 1. \end{cases}$$

Solving this last system, we have

$$\begin{cases} l_{11} = 0 \\ l_{21} = 0 \\ l_{22}^2 = 1 - l_{12}^2 \\ l_{12} \in \mathbb{R}. \end{cases}$$

Supposing that $l_{12} = 0.1$*, we get* $l_{22} = 0.995$*, and conclude that the system admits a solution:*

$$l_1 = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}, \quad l_2 = \begin{bmatrix} 0 \\ 0.995 \end{bmatrix}.$$

It is evident that the maximal subset of linear independent vectors in $\{l_1, \ l_2\}$ *has a single vector. Therefore,* $s_1 = 1$ *and* $\triangle I^1 = \{|I^1| + 1\} = \{0 + 1\} = \{1\}$*. We can consider* $m_1 = l_1$*, hence* $\{m_i, \ i \in \triangle I^1\}$=$\{m_1\}$*. Therefore,* $M^2 = \left\{ \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \right\}$*,* $I^2 = I^1 \cup \triangle I^1$*, and so,* $I^2 = \emptyset \cup \{1\} = \{1\}$*.*

Set the next iteration to $k = 2$*. Compute* $p_2 = s - |I^2| = 1$*.*

Solve the following system w.r.t. variables l_1 *and* γ_1*:*

$$\begin{cases} l_1^T \mathbf{A}_0 l_1 + \gamma_1^T \mathbf{A}_0 m_1 = 0 \\ l_1^T \mathbf{A}_1 l_1 + \gamma_1^T \mathbf{A}_1 m_1 = 0 \\ l_1^T \mathbf{A}_2 l_1 + \gamma_1^T \mathbf{A}_2 m_1 = 0 \\ l_1^T \mathbf{A}_3 l_1 + \gamma_1^T \mathbf{A}_3 m_1 = 0 \\ \|l_1\|^2 + \|l_2\|^2 = 1 \\ l_1^T m_1 = 0. \end{cases}$$

For vectors $l_1 = (l_{11}, l_{12})^T$, $\gamma_1 = (\gamma_{11}, \gamma_{12})^T$ *and* m_1, *and matrices* $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, *we get*

$$\begin{cases} \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} \gamma_{11} & \gamma_{12} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} = 0 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} \gamma_{11} & \gamma_{12} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} = 0 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} \gamma_{11} & \gamma_{12} \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} = 0 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} \\ l_{12} \end{bmatrix} + \begin{bmatrix} \gamma_{11} & \gamma_{12} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} = 0 \\ {l_{11}}^2 + {l_{12}}^2 = 1 \\ \begin{bmatrix} l_{11} & l_{12} \end{bmatrix} \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} = 0. \end{cases}$$

From this system we obtain

$$\begin{cases} 0.1\gamma_{11} = -1 \\ 0.1\gamma_{11} = 0 \\ 0.1\gamma_{11} = -2 \\ 0.1\gamma_{11} = -3 \\ {l_{11}}^2 = 1 \\ l_{12} = 0 \end{cases}$$

and conclude that this system does not have a solution, so, the algorithm stops.

Hence, the basis of the subspace of immobile indices is given by

$$\mathbf{M} = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

and the subspace of immobile indices has dimension $s^* = 1$.

Based on the Theorem 25, we can conclude that the problem (3.12) *does not satisfy the Slater condition and has minimal irregularity degree.*

Example 13 *Consider the SDP problem from [15] given by*

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^3} & 48x_1 - 8x_2 + 20x_3 \\ \text{s.t.} & \begin{bmatrix} 10 & 4 \\ 4 & 0 \end{bmatrix} x_1 + \begin{bmatrix} 0 & 0 \\ 0 & -8 \end{bmatrix} x_2 + \begin{bmatrix} 0 & -8 \\ -8 & -2 \end{bmatrix} x_3 + \begin{bmatrix} 11 & 0 \\ 0 & -23 \end{bmatrix} \preceq 0. \end{array} \tag{3.13}$$

Here, the matrices $\mathbf{A}_i$, $i = 0, 1, 2, 3$, *are given by*

$$\mathbf{A}_0 = \begin{bmatrix} 11 & 0 \\ 0 & -23 \end{bmatrix}, \ \mathbf{A}_1 = \begin{bmatrix} 10 & 4 \\ 4 & 0 \end{bmatrix}, \ \mathbf{A}_2 = \begin{bmatrix} 0 & 0 \\ 0 & -8 \end{bmatrix} \textit{ and } \mathbf{A}_3 = \begin{bmatrix} 0 & -8 \\ -8 & -2 \end{bmatrix}.$$

Set $k = 1$, $I^1 = \emptyset$ *and let* $M^1 = \emptyset$. *Compute* $p_1 = s - |I^1| = 2$.

In this case, the system (3.11) *has the form*

$$\begin{cases} \sum_{i=1}^{2} l_i^T \mathbf{A}_j l_i = 0 \ , \quad j = 0, 1, 2, 3, \\ \sum_{i=1}^{2} \|l_i\|^2 = 1 \end{cases} \tag{3.14}$$

and we need to solve it w.r.t. the vector variables $l_1 = (l_{11}, l_{12})^T$ *and* $l_2 = (l_{21}, l_{22})^T$. *The system can be rewritten as*

$$\begin{cases} 11{l_{11}}^2 - 23{l_{12}}^2 + 11{l_{21}}^2 - 23{l_{22}}^2 = 0 \\ 10{l_{11}}^2 + 8l_{11}l_{12} + 10{l_{21}}^2 + 8l_{21}l_{22} = 0 \\ {l_{12}}^2 + {l_{22}}^2 = 0 \\ -16l_{11}l_{12} - 2{l_{12}}^2 - 16l_{21}l_{22} - 2{l_{22}}^2 = 0 \\ {l_{11}}^2 + {l_{12}}^2 + {l_{21}}^2 + {l_{22}}^2 = 1 \end{cases}$$

and we get

$$\begin{cases} l_{11} = 0 \\ l_{21} = 0 \\ l_{12} = 0 \\ l_{22} = 0 \\ {l_{11}}^2 + {l_{12}}^2 + {l_{21}}^2 + {l_{22}}^2 = 1. \end{cases}$$

Evidently, the system (3.14) *does not have a solution, and therefore, the DIIS algorithm stops at the first iteration* $k^* = 1$, *with* $I^1 = \emptyset$. *Therefore, the subspace of immobile indices* $\mathcal{M}$ *is null and by the Theorem 25, we can conclude that the problem* (3.13) *satisfies the Slater condition.*

3.2 Well-posedness

Another notion characterizing regularity of optimization problems is well-posedness. There exist different definitions of well-posedness, being the most common Hadamard's and Tikhonov's well-posedness [29, 68, 77, 102]. An optimization problem is said to be well-posed in the sense defined by Hadamard if it has a unique solution that depends continuously on data [29, 77]. A problem is said to be well-posed in the sense of Tikhonov if it has a unique solution toward which every minimizing sequence converges [29, 77, 138]. There are various other definitions of well-posedness, such as strong well-posedness and Levitin-Polyak well-posedness, and it has been shown in [77] that Hadamard's well-posedness implies Tikhonov's, Levitin-Polyak's and strong well-posedness. In particular, it was proved that under the Slater condition, Hadamard's well-posedness is equivalent to that of Tikhonov.

According to [69], a feasible SDP problem that does not satisfy the Slater condition is ill-posed "in the sense that an arbitrary small perturbation of the problem can change its status from feasible to infeasible". This is clear, since the feasible set of such SDP problem has an empty interior, hence, all the feasible solutions lie on the boundary of the feasible set and thus, it is very sensible even to small perturbations. Nevertheless, testing of well-posedness of convex optimization problems is usually based on a specific measure called Renegar's condition number introduced in [127].

3.2.1 Well-posedness in the sense of Renegar

In [38] and [61], constructive approaches to classify SDP problems in terms of well-posedness in the sense defined by Renegar were proposed. In [127], a specific measure, the Renegar condition number, was defined for convex optimization problems.

Consider a SDP primal problem in the form (2.5). Note that each SDP problem instance is characterized by its data d, which encompasses the matrices $\mathbf{A}_i$, $i = 1, ..., n$, $\mathbf{C}$ and the vector b. Evidently, if the given SDP problem has the form (2.2), then its data d involves the matrices $\mathbf{A}_i$, $i = 1, ..., n$, $\mathbf{A}_0$ and the vector c.

For a generic conic program with data d, Renegar introduced the following measures:

- the distance to primal infeasibility, which is defined as

$$\rho_P(d) := \inf \left\{ \frac{\|\triangle d\|}{\|d\|} : \text{problem } d + \triangle d \text{ is primal infeasible} \right\}, \tag{3.15}$$

- the distance to dual infeasibility, which is defined as

$$\rho_D(d) := \inf \left\{ \frac{\|\triangle d\|}{\|d\|} : \text{problem } d + \triangle d \text{ is dual infeasible} \right\}, \tag{3.16}$$

where $\triangle d$ is a small data perturbation and $\|.\|$ is a suitable norm [38, 59].

The Renegar condition number, denoted by $C(d)$, is defined by

$$C(d) := \frac{1}{\min \{\rho_P(d), \rho_D(d)\}}. \tag{3.17}$$

If the distance to infeasibility is zero, then $C(d) = \infty$ and the problem is said to be ill-posed; otherwise, if $C(d)$ is finite, then the problem is considered to be well-posed [38, 61].

Definition 17 *A problem instance with data d is called ill-posed if* $\min \{\rho_P(d), \rho_D(d)\} = 0$*, which is equivalent to $C(d) = \infty$.*

Obviously, if at least one of the primal or dual problems is infeasible, then the distance to primal or dual infeasibility is zero, and $C(d) = \infty$. The Renegar condition number can be regarded as a scale-invariant reciprocal of the distance to infeasibility (the smallest data perturbation that renders in either primal or dual infeasibility), and therefore, it describes the sensitivity of the problem [59].

Example 14 *Consider the SDP problem*

$$\begin{array}{ll} \min & x_{12} \\ \text{s.t.} & \begin{bmatrix} 0 & x_{12} \\ x_{12} & x_{22} \end{bmatrix} \succeq 0 \end{array} \tag{3.18}$$

and its dual

$$\begin{array}{ll} \max & 0 \\ \text{s.t.} & \begin{bmatrix} -y_1 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix} \succeq 0. \end{array} \tag{3.19}$$

It is easy to see that the primal problem is feasible and its optimal value is $p^* = 0$. Hence, the distance to primal infeasibility is $\rho_P = \infty$. However, the dual problem is infeasible and thus, the distance to dual infeasibility is $\rho_D = 0$. Therefore, the Renegar's condition number is $C(d) = \infty$, meaning that the problem is ill-posed.

Consider a SDP problem in the form (2.5). It has been shown in [61] that the Renegar condition number is related to a rigorous upper bound of the primal optimal value of such problems and it is proposed to compute this rigorous upper bound using interval arithmetic. The interval quantities can be introduced as follows. The data of a SDP problem instance is assumed to vary within certain interval bounds of the form $[A] = [\underline{A}, \bar{A}]$. Then, we denote by $[\mathbf{A}_i]$ and $[\mathbf{C}]$ the symmetric interval matrices and by $[b]$ an interval vector.

The following result is proved in [61] (Theorem 4.1).

Theorem 20 *Given a family of SDP problems in the form (2.5), suppose that there exist interval matrices $[\mathbf{X}]$ such that*

$$\forall b \in [b],\ \forall \mathbf{A}_i \in [\mathbf{A}_i],\ i = 1, ..., m, \exists \textit{ symmetric } \mathbf{X} \in [\mathbf{X}] : \operatorname{tr}(\mathbf{A}_i \mathbf{X}) = b_i, \tag{3.20}$$

and

$$\mathbf{X} \succeq 0 \textit{ for all symmetric } \mathbf{X} \in [\mathbf{X}]. \tag{3.21}$$

Then, the optimal value is bounded from above by

$$p^* \leq sup\{\operatorname{tr}([\mathbf{C}][\mathbf{X}])\} = \bar{p}^*. \tag{3.22}$$

Moreover, if all symmetric matrices $\mathbf{X} \in [\mathbf{X}]$ are positive definite and p^* is bounded from below, then the optimal value d^* of the corresponding dual to (2.5) is equal to p^* for all problem instances, and the dual supremum (the optimal dual solution) is attained.

It is shown that the upper bound (3.22), denoted by $\bar{p}^*$, is infinite when the distance to infeasibility is zero, *i.e.*, $C(d) = \infty$.

3.2.2 Testing well-posedness

In what follows, we describe two numerical approaches to test well-posedness of SDP problems.

The numerical approach to characterize the well-posedness of SDP problems proposed in [38] is based on the estimation of lower and upper bounds of the Renegar's condition number $C(d)$.

Consider a primal SDP problem in the form (2.5). Its dual is given by

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & \sum\limits_{i=1}^{n} \mathbf{A}_i y_i + \mathbf{Z} = \mathbf{C}, \\ & \mathbf{Z} \succeq 0, \end{array} \tag{3.23}$$

where $b, y \in \mathbb{R}^n$ and $\mathbf{A}_i$, $i = 1, ..., n$, $\mathbf{Z}$, $\mathbf{C}$ are $(s \times s)$ symmetric matrices.

To make the computations easier, in [38] the following matrix and vector norms were used: $\|\mathbf{X}\|_{E_p} := \left(\sum\limits_{j=1}^{s} |\lambda_j|^p \right)^{\frac{1}{p}}$, where λ_j, $j = 1, ..., s$ are the eigenvalues of the matrix $\mathbf{X}$, and $\|b\|_1 = \sum\limits_{i=1}^{n} |b_i|$ for $b \in \mathbb{R}^n$.

For computing $\rho_P(d)$, according to [38], a set of $2n$ auxiliary SDP problems are considered (see the Remark 6 of [40]):

$$\begin{array}{lll} \rho_P(d)^k & = \min\limits_{y,\mathbf{Z},u,\gamma} & \gamma \\ & \text{s.t.} & \sum\limits_{i=1}^{n} \mathbf{A}_i y_i + \mathbf{Z} = \gamma \mathbf{I}_s \\ & & -b^T y + u \leq \gamma \\ & & y_{\lceil \frac{k}{2} \rceil} = (-1)^k \\ & & \mathbf{Z} \succeq 0, y \in \mathbb{R}^n, \gamma \in \mathbb{R}, u \geq 0, \end{array} \tag{3.24}$$

where $k = 1, ..., 2n$ and $\mathbf{I}_s$ is the identity matrix of order s. Then, $\rho_P(d)$ is defined as follows:

$$\rho_P(d) = \min_{k=1,...,2n} \rho_P(d)^k.$$

Hence, the computation of the distance to primal infeasibility involves solving $2n$ conic convex problems of size and structure compatible with the dual SDP problem (3.23).

For computing $\rho_D(d)$, one has to solve the following problem (see the Theorem 2 from [40]):

$$\begin{array}{lll} \rho_D(d) & = \min\limits_{\mathbf{X},g,\gamma} & \gamma \\ & \text{s.t.} & \|\mathbf{A}\mathbf{X}\|_1 \leq \gamma \\ & & |\text{tr}(\mathbf{C}\mathbf{X}) + g| \leq \gamma \\ & & \text{tr}(\mathbf{X}) = 1 \\ & & \mathbf{X} \succeq 0, \gamma \in \mathbb{R}, g \geq 0, \end{array} \tag{3.25}$$

where here, $\mathbf{AX} = (\mathrm{tr}(\mathbf{A}_1\mathbf{X}) \dots \mathrm{tr}(\mathbf{A}_n\mathbf{X}))^T$.

According to [38], this problem can be converted into a conic convex problem whose size and structure is compatible with the primal problem (2.5).

The estimation of the norm of data can be done with the help of its upper and lower bounds using straightforward matrix norms and maximum eigenvalue computations. By using the Proposition 3 in [38], $\|d\|$ can be bounded as follows:

$$\max\{l, \|b\|_1, \|\mathbf{C}\|_{E_\infty}\} \leq \|d\| \text{ and } \|d\| \leq \max\{u, \|b\|_1, \|\mathbf{C}\|_{E_\infty}\},$$

where l and u are positive values specified in the Proposition 3 in [38].

The major difficulty of this approach is that to calculate all the three quantities $\rho_P(d)$, $\rho_D(d)$ and $\|d\|$, one has to solve several SDP problems, in structure and size compatible with the original primal and dual SDP problems. Therefore, as it is reported in [38] the computation of the Renegar condition number is rather expensive.

In [61], it is also considered that a problem is ill-posed if the Renegar condition number is infinite, but another approach to characterize the well-posedness of SDP problems is proposed. This approach is based on the calculus of a rigorous upper bound $\bar{p}^*$ of the optimal value of a given SDP problem.

Consider a primal SDP problem in the form (2.5). In [61], a procedure for computing the upper bound $\bar{p}^*$ for (2.5) is described in the Algorithm 4.1 in [61] (we will call it here upper bound algorithm). The procedure uses interval arithmetic. On its iterations, some auxiliary perturbed "midpoint" SDP problems are solved using a SDP solver and special interval matrices are constructed on the basis of their solutions. These interval matrices must contain a primal feasible solution of the perturbed "midpoint" problem and satisfy the conditions of the Theorem 20. If such interval matrix can be computed, then the optimal value is bounded from above by $\bar{p}^*$, which is the value of the objective function calculated using the interval matrix. The upper bound algorithm needs verified solvers to provide an interval result (enclosure) that surely contains the correct result for the solution of interval linear systems and eigenvalue problems. It also needs a SDP solver to solve the auxiliary perturbed problems.

The perturbed "midpoint" problem solved in the iterations of the algorithm has the form

$$\begin{array}{ll} \min & \mathrm{tr}\,(\mathbf{CX}) \\ \text{s.t.} & \mathrm{tr}\,(\mathbf{A}_i\mathbf{X}) = b_i, \quad i = 1, ..., m \\ & \mathbf{X} \succeq \epsilon \mathbf{I}_s, \end{array} \tag{3.26}$$

where $\epsilon > 0$, $\mathbf{I}_s$ is the identity matrix, $b_i \in \mathbb{R}$, $\mathbf{C}$, $\mathbf{A}_i$ and $\mathbf{X}$, $i = 1, ..., m$, are symmetric matrices of order s. The solution of (3.26), $\tilde{\mathbf{X}}$, is used as an initial solution for the upper bound algorithm.

In [61], the primal SDP problem (2.5) is equivalently written in the following vector form:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \mathbf{A}^{\mathrm{mat}} x = b, \\ & \mathbf{X} \succeq 0, \end{array} \tag{3.27}$$

where $c := \text{svec}((\mathbf{C}), 2)$, $x := \text{svec}((\mathbf{X}), 1)$ and the i-th row of the $\left(m \times \frac{s(s+1)}{2}\right)$ matrix $\mathbf{A}^{\text{mat}}$ is given by

$$\mathbf{A}^{\text{mat}}(i, :) := \text{svec}((\mathbf{A}_{\text{i}}), 2),$$

where for $\mu \in \mathbb{R}$ and a $(s \times s)$ symmetric matrix $\mathbf{M}$, $\text{svec}((\mathbf{M}), \mu) := (\mathbf{M}_{11}, \mu\mathbf{M}_{21}, ..., \mu\mathbf{M}_{s1}, \mathbf{M}_{22}, \mu\mathbf{M}_{32}, ..., \mu\mathbf{M}_{ss-1}, \mathbf{M}_{ss})^T$. The inverse operator of svec is denoted by smat.

For an interval matrix $[\mathbf{A}]$, the midpoint is defined as $\text{mid}[\mathbf{A}] := \frac{\underline{\mathbf{A}}+\bar{\mathbf{A}}}{2}$ [61].

The upper bound algorithm chooses an index set I such that the $m \times m$ submatrix $\text{mid}[\mathbf{A}]_{\text{I}}^{\text{mat}}$ is nonsingular. The index set I is chosen by performing an LU-decomposition on $(\text{mid}[\mathbf{A}]^{\text{mat}})^{\text{T}}$. Then, it computes an enclosure $\mathbf{x}_I$ of the solution set

$$\Sigma_I := \left\{ x_I \in \mathbb{R}^m : \mathbf{A}_I^{\text{mat}} x_I = b - \sum_{\gamma \in N} \mathbf{A}_N^{\text{mat}} \tilde{x}_N, \mathbf{A} \in [\mathbf{A}]^{\text{mat}}, b \in [b] \right\}, \tag{3.28}$$

where N denote the indices of columns of $\text{mid}[\mathbf{A}]^{\text{mat}}$ which are not in I, $[b]$ is the interval quantity for the vector b and $\tilde{x} := \text{svec}((\mathbf{X}(\epsilon)), 1)$. Then $\mathbf{x} := (\mathbf{x}_I; \tilde{x}_N)$ and $(\mathbf{X}) = \text{smat}(\mathbf{x}, 1)$ satisfies the condition (3.20). A special method for computing the rigorous lower bound for the smallest eigenvalue of a symmetric interval matrix is needed to verify the condition (3.21) of the Theorem 20.

The steps of the upper bound algorithm (Algorithm 4.1 in [61]) can be outlined as follows.

Algorithm 3 Computing the rigorous upper bound $\bar{p}^*$

```
input: $\tilde{\mathbf{X}}$, solution of the problem (3.26).
output: $\bar{p}^*$, rigorous upper bound.
1: set $\bar{p}^* = \infty$ and $k, \epsilon$ are $n$-dimensional zero vectors
2: choose an index set $I$ such that the submatrix $\text{mid}\mathbf{A}_{\text{I}}^{\text{mat}}$ is nonsingular
3: if there is no nonsingular submatrix then stop
4: while perturbed problem (3.26) is feasible do
5:     compute an enclosure $\mathbf{x}_I$ of (3.28) and set $\mathbf{x} := (\mathbf{x}_I; \tilde{x}_N)$
6:     set $(\mathbf{X}) = \text{smat}(\mathbf{x}, \mathbf{1})$ and compute rigorous bounds $\underline{\lambda} \le \lambda_{min}(\mathbf{X})$
7:     if $\underline{\lambda} \ge 0$ then compute $\bar{p}^* = \sup\{\mathbf{c}^T\mathbf{x}\}$
8:     update the perturbation $\epsilon$ by computing
           $k = \begin{cases} k+1, & \underline{\lambda} < 0 \\ k, & \text{otherwise} \end{cases}$ and $\epsilon = \begin{cases} -2^k\underline{\lambda} + \epsilon, & \underline{\lambda} < 0 \\ \epsilon, & \text{otherwise} \end{cases}$
9:     solve the perturbed problem (3.26), set $\tilde{\mathbf{X}} := \tilde{\mathbf{X}}(\epsilon)$ and set $\tilde{x} := \text{svec}((\tilde{\mathbf{X}}), 1)$
10: return $\bar{p}^*$.
```

If the lower eigenvalue bound is nonnegative, this algorithm returns a finite upper bound $\bar{p}^*$. In the particular case of the lower eigenvalue bound be positive, then the primal SDP problem of the form (2.5) is strictly feasible, that is its constraints satisfy the Slater condition. If a primal SDP problem is infeasible, then the upper bound $\bar{p}^*$ is infinite.

The rigorous bounds $\bar{p}^*$ for SDP problems in the form (2.5) recognize the difficulty of solving such problems and overestimate the optimal value only slightly, which depends on the quality of the computed approximations [61].

According to [61], a finite rigorous upper bound indicates well-posedness of the SDP problem and an infinite upper bound indicates ill-posedness in the sense of Renegar.

Example 15 *Consider the following class of SDP problems from [61]:*

$$\begin{array}{ll} \min & x_1 - \sigma x_2 - \sigma x_3 \\ \text{s.t.} & X = \begin{bmatrix} \varepsilon & -1 & 0 \\ -1 & x_2 & 0 \\ 0 & 0 & x_3 \end{bmatrix} \succeq 0, \end{array} \tag{3.29}$$

where σ and ε are real parameters. Its dual is given by

$$\begin{array}{ll} \max & y_1 + \varepsilon y_2 \\ \text{s.t.} & \begin{bmatrix} -y_2 & \frac{1+y_1}{2} & -y_3 \\ \frac{1+y_1}{2} & \sigma & -y_4 \\ -y_3 & -y_4 & \sigma \end{bmatrix} \succeq 0. \end{array} \tag{3.30}$$

Observe that the primal solution must satisfy $x_2 \geq 0$, $x_3 \geq 0$ and $\varepsilon x_2 - (-1)^2 \geq 0$.

If $\varepsilon \leq 0$, then the primal problem is infeasible, and if $\sigma < 0$, then the dual problem is infeasible. In the case of $\varepsilon \leq 0$ and $\sigma < 0$, it is clear that both problems do not satisfy the Slater condition and are ill-posed, since they are infeasible. It is easy to see that if $\varepsilon, \sigma > 0$, then both problems satisfy the Slater condition. If $\varepsilon, \sigma = 0$, then duality gap is nonzero. In this case, the primal problem is infeasible, hence $\bar{p}^* = \infty$, meaning that the primal problem (3.29) *is ill-posed.*

In [61], rigorous bounds for the primal SDP problem were computed considering five different values for ε and σ. The results of the numerical tests indicate that the values of the parameters ε and σ are important to make conclusions about well or ill-posedness.

3.3 Good behaviour

Another notion associated with regularity sometimes used in the literature is that of good behaviour. According to [42], a SDP problem needs to be "well-behaved" in some sense in order to guarantee that current SDP methods/solvers provide a reliable solution.

3.3.1 Good behaviour in the sense of Pataki

Assuming that a SDP problem is feasible, Pataki introduced the following definition in [111].

Definition 18 *The constraint system of a SDP problem in the form* (2.2) *is said to be well-behaved if for all objective functions, the optimal values of* (2.2) *and its dual* (2.21) *coincide and the dual optimal value is attained, when it is finite. Otherwise, the SDP system is said to be badly-behaved.*

Notice that a SDP system is well-behaved if strong duality holds for all objective functions.

By abuse of language, we say that a given SDP problem in the form (2.2) is either badly or well-behaved instead of saying that the SDP system of this problem is, respectively, badly or well-behaved.

In [111], "efficiently verifiable" characterizations of well or badly-behaved SDP problems are proposed. In what follows, we will describe such characterizations, whose proofs can be found in [111].

Consider a feasible SDP problem in the form (2.2) and the associated dual in the form (2.21). Consider also a slack matrix in (2.2) given by $\mathbf{S} := -\mathbf{A}_0 - \sum_{i=1}^{n} \mathbf{A}_i x_i \succeq 0$ and assume that there exists a slack matrix with maximum rank given by $\mathbf{S} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, 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.

The following characterization of badly-behaved problems is proposed in [111] (Theorem 4).

Theorem 21 *The SDP problem* (2.2) *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*

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} \\ \mathbf{V}_{12}^T & \mathbf{V}_{22} \end{bmatrix}, \tag{3.31}$$

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* $\mathcal{C}(\mathbf{V}_{12}^T)$ *is not contained in* $\mathcal{C}(\mathbf{V}_{22})$*.*

The above matrices $\mathbf{S}$ and $\mathbf{V}$ provide a certificate of the bad behaviour of the SDP problem (2.2) [111].

A characterization of well-behaved problems is also proposed (Theorem 5 in [111]).

Theorem 22 *The SDP problem* (2.2) *is well-behaved if and only if the following two conditions hold:*

1. *there is a* $(s \times s)$ *matrix* $\mathbf{U}$ *of the form*

$$\mathbf{U} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_{22} \end{bmatrix}, \tag{3.32}$$

   where $\mathbf{U}_{22} \succeq 0$ *and* $\mathrm{tr}(-\mathbf{A}_0\mathbf{U}) = \mathrm{tr}(\mathbf{A}_1\mathbf{U}) = ... = \mathrm{tr}(\mathbf{A}_n\mathbf{U}) = 0$*;*

2. *for all matrices* $\mathbf{V}$*, which are linear combination of the matrices* $\mathbf{A}_i$*, for* $i = 0, ..., n$*, and are of the form*

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} \\ \mathbf{V}_{12}^T & \mathbf{0} \end{bmatrix}, \tag{3.33}$$

   where $\mathbf{V}_{11} \in \mathcal{S}(r)$*, we must have* $\mathbf{V}_{12} = \mathbf{0}$*.*

It is also shown that one can verify if a SDP problem is badly or well-behaved without using the above theorems. For that purpose, standard reformulations for badly and well-behaved problems are suggested and proved to be badly or well-behaved when the original SDP problem is badly or well-behaved as well.

In [111], a reformulation of a SDP problem in the form (2.2), yielding another SDP problem in the same form, can be obtained by performing a sequence of the following operations (Definition 2 in [111]):

1. apply a rotation $\mathbf{T}^T()\mathbf{T}$ to all matrices $\mathbf{A}_i$, $i = 0, ..., n$, where $\mathbf{T} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix}$ and $\mathbf{M}$ is invertible;
2. replace $-\mathbf{A}_0$ by $-\mathbf{A}_0 + \sum_{j=1}^{n} \mu_j \mathbf{A}_j$, where $\mu_j \in \mathbb{R}^n$;
3. replace $\mathbf{A}_i$ by $\sum_{j=1}^{n} \lambda_j \mathbf{A}_j$, and c_i by $\sum_{j=1}^{n} \lambda_j c_j$, where $i \in \{1, ..., n\}$, $\lambda \in \mathbb{R}^n$ with nonzero elements;
4. exchange the pairs $(\mathbf{A}_i, c_i)$ and $(\mathbf{A}_j, c_j)$, where $i, j \in \{1, ..., n\}$.

The reformulation for a badly-behaved SDP problem can be stated as follows (Theorem 6 in [111]):

Theorem 23 *The SDP problem* (2.2) *is badly-behaved if and only if it has a reformulation of the form*

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \sum_{i=1}^{k} x_i \begin{bmatrix} \mathbf{F}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \sum_{i=k+1}^{n} x_i \begin{bmatrix} \mathbf{F}_i & \mathbf{G}_i \\ \mathbf{G}_i^T & \mathbf{H}_i \end{bmatrix} \preceq \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{S}, \end{array} \tag{3.34}$$

where

1. $\mathbf{S}$ *is the maximum rank slack matrix;*
2. *the matrices* $\begin{bmatrix} \mathbf{G}_i \\ \mathbf{H}_i \end{bmatrix}$, *for* $i = k+1, ..., n$, *are linearly independent;*
3. $\mathbf{H}_n \succeq 0$.

Another interesting result in [111] consists in the proof that any badly-behaved SDP problem can be reduced to

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x_1 \begin{bmatrix} \alpha & 1 \\ 1 & 0 \end{bmatrix} \preceq \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \end{array} \tag{3.35}$$

where $\alpha \in \mathbb{R}$, if we add to the previous operations a sequence of the following ones:

1. delete the i-th row and the i-th column from all matrices, where $i \in \{1, ..., s\}$;
2. delete a constraint matrix.

Example 16 *Consider the following SDP problem.*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} -2 & 3 & 0 \\ 3 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_1 + \begin{bmatrix} 1 & 2 & 0 \\ 2 & 3 & 0 \\ 0 & 0 & -6 \end{bmatrix} x_2 \preceq \begin{bmatrix} 3 & 0 & \frac{1}{2} \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}. \end{array} \tag{3.36}$$

This is a badly-behaved problem. Indeed, let us perform a sequence of the operations presented above: delete the first row and column in all the constraint matrices and then, delete the second constraint matrix. As result, we get a SDP problem of the form (3.35), *with* $\alpha = -4$.

The reformulation for a well-behaved SDP problem is stated as follows (Theorem 7 in [111]):

Theorem 24 *The SDP problem* (2.2) *is well-behaved if and only if it has a reformulation of the form*

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \sum\limits_{i=1}^{k} x_i \begin{bmatrix} \mathbf{F}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \sum\limits_{i=k+1}^{n} x_i \begin{bmatrix} \mathbf{F}_i & \mathbf{G}_i \\ \mathbf{G}_i^T & \mathbf{H}_i \end{bmatrix} \preceq \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{S}, \end{array} \tag{3.37}$$

where

1. $\mathbf{S}$ *is the maximum rank slack matrix;*
2. *the matrices* $\mathbf{H}_i$, $i = k+1, ..., n$, *are linearly independent;*
3. $\operatorname{tr}(\mathbf{H}_{k+1}\mathbf{I}) = ... = \operatorname{tr}(\mathbf{H}_n\mathbf{I}) = 0$.

3.3.2 Testing good behaviour

In [111], an algorithm to generate well-behaved SDP systems based on the reformulation of the Theorem 24 is suggested. But in terms of characterizing the behaviour of a given SDP problem, no algorithm is proposed.

Notice that to characterize well-behaved problems using the Theorem 22, no algorithmic procedure is suggested in [111] neither to obtain a matrix $\mathbf{U}$, so that it is orthogonal to all constraint matrices, nor to construct a matrix $\mathbf{V}$ satisfying the condition 2 of the Theorem 22.

Although the characterization of badly-behaved SDP problems proved in the Theorem 21 is considered to be "easy to spot" in [111], for constraint matrices of general dimension, the certificates of the bad behaviour of SDP problems may be not easy to obtain, since it is a nontrivial task to prove that the standard reformulation exists for badly-behaved problems [111], using the elementary algebra operations described in the previous section.

Therefore, testing of the good or bad behaviour of SDP problems can be rather difficult to implement in the form of a numerical procedure.

Motivated by the characterizations of badly-behaved problems from [111], we developed an algorithm for generating a class of SDP problems that are badly-behaved, that in turn fail to satisfy the Slater condition. This algorithm will be presented in the Chapter 5.

3.4 Relationships between different notions of regularity in SDP

In the previous sections, we have discussed different notions of regularity in SDP and showed that they are important for efficient solution of the problems. It was shown that these notions are not so easy to verify in practice and only few numerical procedures are proposed for checking the regularity in some sense of a given problem.

In this section, we will recall known results about relationships between such regularity notions and establish some new ones, that should permit to clarify the connections and when it is possible and useful to replace the check of one regularity condition by another.

Nevertheless the notions of regularity of SDP problems introduced above are different, there exist a deep connection between them. In what follows, we will show that the Slater condition is closely related to the notion of well-posedness and also of good behaviour.

According to [156], the lack of regularity in terms of the Slater condition is an indication of ill-posedness of the problem. In [154, 155], it is pointed out that measure of strict feasibility is also called distance to infeasibility, which in turn is used for computing the Renegar's condition number to check well-posedness of a problem. Therefore, the Slater condition is related to the Renegar's condition number.

The following lemma can be proved.

Lemma 2 *If the linear SDP problem in the form* (2.2) *does not satisfy the Slater condition, then the problem is ill-posed.*

Proof. Indeed, if the Slater condition is not satisfied, all the feasible solutions of the given SDP problem lie on the boundary of the feasible set. Hence, there exist arbitrarily small data perturbations that lead to the loss of feasibility, rendering an infeasible problem. Thus, for the given problem, the distance to infeasibility is zero and, therefore, the Renegar's condition number is infinite. Hence, according to the Definition 17, the problem is ill-posed. ■

Notice that the reciprocal of Lemma 2 is not true. The following example shows that there exist problems that are ill-posed, but do satisfy the Slater condition.

Example 17 *Consider the particular case of the primal SDP problem from Example 15,* *with* $\varepsilon = 1$ *and* $\sigma = -1$*. The problem can be easily written in the form* (2.2) *as follows.*

$$\begin{array}{ll} \min & x_1 - x_2 - x_3 \\ \text{s.t.} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} x_2 + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} x_3 + \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \preceq 0. \end{array} \tag{3.38}$$

The dual problem to (3.38) *has the form*

$$\begin{array}{ll} \max & y_1 + y_2 \\ \text{s.t.} & \begin{bmatrix} -y_2 & \frac{1+y_1}{2} & -y_3 \\ \frac{1+y_1}{2} & -1 & -y_4 \\ -y_3 & -y_4 & -1 \end{bmatrix} \succeq 0. \end{array} \tag{3.39}$$

The constraints of the primal problem (3.38) *satisfy the Slater condition, since there exists a strictly feasible solution: e.g.,* $x = [1\ 2\ 1]^T$*. However, the problem* (3.38) *is ill-posed, since its Renegar condition number is infinite. Indeed, it is easy to see that the dual problem* (3.39) *is infeasible, i.e., there is no possible feasible solution satisfying the constraints, and hence, the distance to dual infeasibility is zero.*

Notice that a SDP problem is well-behaved in the sense of Pataki [111] if strong duality holds, which can be ensured in SDP if a regularity condition, such as the Slater condition, holds. Therefore, the good behaviour of a SDP problem is closely connected to the Slater condition.

The following result was proved in [111] (Corollary 2).

Proposition 10 *If the constraints of the SDP problem* (2.2) *satisfy the Slater condition, then the problem is well-behaved.*

Evidently, we can formulate the following lemma.

Lemma 3 *If the SDP problem* (2.2) *is badly-behaved, then it does not satisfy the Slater condition.*

From Lemmas 2 and 3, we can then establish the following connection.

Lemma 4 *If the SDP problem* (2.2) *is badly-behaved, then it is ill-posed.*

Summarizing the relations between the notions of regularity addressed in this work, for a given SDP problem we can construct the following diagram.

$$\begin{tikzcd} \boxed{\text{well-posedness}} \arrow[r, Rightarrow] & \boxed{\text{Slater condition}} \arrow[r, Rightarrow] & \boxed{\text{good behaviour}} \end{tikzcd}$$

3.5 Conclusions

In this chapter, we have studied different notions usually associated to regularity in SDP, namely, CQs, well-posedness, and good behaviour. In terms of CQs, we proved that for the convex SDP problems considered in this thesis, the Slater condition is equivalent to the regularity condition of Shapiro, Robinson CQ and to the Mangasarian-Fromovitz CQ. We also established connections between different notions associated to regularity, and realized that all have some relation to the Slater condition: a badly-behaved SDP problem implies that the Slater condition fails to hold, which in turn implies that the given SDP problem is ill-posed.

It is important to emphasize once again that, in spite of the Slater condition has been shown to be a generic property for SDP, in practice, there exist many SDP instances failing this condition. Current SDP solvers require the Slater condition to hold for both primal and dual SDP problems, but do not check its fulfilment. Although the Slater condition seems to be not easy to verify on a given SDP problem, we have presented an approach to test the Slater condition in SDP based on the DIIS algorithm. We have shown that the nonvanishing dimension of a basis of the immobile index subspace of a SDP problem can be regarded as a certificate of the failure of the Slater condition.

In the next chapter, we will propose an original numerical tool to test the regularity of a given SDP problem in terms of the Slater condition and present numerical experiments.

Chapter 4

Testing of regularity in SDP

In this chapter, we suggest a numerical procedure to test if a given SDP problem is regular in terms of the fulfilment of the Slater condition and present details of its implementation. The procedure is based on the DIIS algorithm from [78] and uses two numerical approaches for solving the system of quadratic equations. Numerical experiments showing the efficiency of the proposed procedure are presented. These experiments are carried out using SDP instances from the available SDP literature and from the well known SDP database, SDPLIB. A comparative analysis with other results on regularity available in the literature is also made.

4.1 SDPreg: a numerical procedure to test the Slater condition in SDP problems

Based on the Theorem 18, we have constructed a procedure to determine if a given feasible SDP problem in the form (2.2) has a null subspace of immobile indices $\mathcal{M}$, and hence verify if the Slater condition holds. The proposed procedure focuses on a single iteration of the DIIS algorithm 2.

At the first iteration of the DIIS algorithm, the basis of the subspace $\mathcal{M}$ is empty and the number of vector variables is $p_1 = s$. The system of quadratic equations (3.11) takes the form

$$\begin{cases} \sum\limits_{i=1}^{s} l_i^T \mathbf{A}_j l_i = 0, \ j = 0, 1, \ldots, n, \\ \sum\limits_{i=1}^{s} \|l_i\|_2^2 = 1, \end{cases}$$

where $l_i \in \mathbb{R}^s$, $i = 1, ..., s$, are the vector variables, $\mathbf{A}_j \in S(s)$, $j = 0, 1, ..., n$, and n is the dimension of the variable space of the SDP problem (2.2).

If this system is consistent, then the dimension s^* of the subspace of immobile indices $\mathcal{M}$ is nonzero, otherwise, $\mathcal{M} = \{0\}$ and $s^* = 0$. In the first case, one can conclude that the Slater condition is violated, while in the latter, one can conclude that the Slater condition is satisfied.

Therefore, the algorithm for testing if a given SDP problem satisfies the Slater condition can be outlined as follows.

Basic Algorithm 4 Testing the Slater condition on SDP

```
input: $\mathbf{A}_j$, $j = 0, 1, ..., n$, $s \times s$ real constraint matrices of a SDP problem in the form (2.2);
output: result on regularity in terms of the fulfilment of the Slater condition.
1: solve the system of quadratic equations:
   $$\begin{cases} \sum\limits_{i=1}^{s} l_i^T \mathbf{A}_j l_i = 0, \ j = 0, 1, \dots, n, \\ \sum\limits_{i=1}^{s} \|l_i\|_2^2 = 1 \end{cases} \qquad (4.1)$$
   w.r.t. $l_i \in \mathbb{R}^s$, $i = 1, ..., s$
2: if the system (4.1) is consistent then
3:     return the SDP problem does not satisfy the Slater condition
4: else the system (4.1) is inconsistent
5:     return the SDP problem satisfies the Slater condition
```

Since the numerical methods for solving the system (4.1) provide only approximate solutions, the important question in the Step 2 of the Basic Algorithm 4 in terms of the consistency of the system (4.1) can only be answered within a certain degree of certainty.

In our case, we are interested in obtaining accurate solutions for the system (4.1), or guaranteeing that this system is not consistent. In what follows, we present two numerical approaches that are aimed to approximately decide whether the system (4.1) is consistent or not, within a desirable tolerance. These approaches are included into the numerical procedure for testing the Slater condition. This procedure is described in the Section 4.1.1 and tested in the Section 4.3.

Numerical Approach I

Let us rewrite the system (4.1) in the following componentwise form

$$\begin{cases} F_j(\ell) &= \sum\limits_{i=1}^{s} l_i^T \mathbf{A}_j l_i = 0, \ j = 0, 1, \dots, n, \\ F_{n+1}(\ell) &= \sum\limits_{i=1}^{s} \|l_i\|_2^2 \ - 1 \ = \ 0, \end{cases} \tag{4.2}$$

where the vector variable ℓ is constructed by stacking the vectors $l_1, l_2, \dots, l_s \in \mathbb{R}^s$, each one with s scalar variables, on top of each other as follows:

$$\ell = \begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_s \end{bmatrix} = \begin{bmatrix} \ell_1 \\ \ell_2 \\ \vdots \\ \ell_m \end{bmatrix} \in \mathbb{R}^m,$$

where $\ell_i \in \mathbb{R}$, $i = 1, ..., m$, and $m = s^2$.

The system (4.2) has $n + 2$ nonlinear equations and m unknowns. In general, we have $m \geq n + 2$ and thus, the system (4.2) is underdetermined. The functions F_j, $j = 0, 1, \dots, n+1$ are quadratic real valued functions defined in $\mathbb{R}^m$, hence they are continuous and smooth. For every $j = 0, 1, ..., n$, the function F_j is convex if and only if the symmetric matrix $\mathbf{A}_j$ is positive semidefinite, which may not hold. The function F_{n+1} is convex by construction.

Any solution of the system (4.2) is a minimizer of the function $\sum_{j=0}^{n+1} F_j^2(\ell)$ and thus, we can formulate the following unconstrained nonlinear least-squares problem:

$$\min_{\ell \in \mathbb{R}^m} \quad G(\ell) = \sum_{j=0}^{n+1} F_j^2(\ell). \tag{4.3}$$

Notice that $G(\ell) \geq 0$, $\forall \ell \in \mathbb{R}^m$.

Considering the unconstrained optimization problem (4.3) and denoting by ℓ^* its solution, one of the following two situations can occur:

1. if $G(\ell^*) = 0$, then ℓ^* is a solution of the system (4.2) and, consequently, the system (4.1) is consistent;

2. if $G(\ell^*) > 0$, then the system (4.2) is not consistent, as well as system (4.1).

The problem (4.3) is a global optimization problem and may have multiple local minima. Therefore, its solution may be not unique. There exist various optimization algorithms that can be used to solve the nonlinear least-squares problem (4.3), such as the Levenberg-Marquardt algorithm, the Gauss-Newton algorithm and the Trust-Region-Reflective algorithm [109]. These algorithms are included in the MATLAB function `lsqnonlin`, a specific routine to solve unconstrained nonlinear least-squares problems. These are iterative methods, that starting from a given initial approximation, find an approximation to a local minimum with a predefined tolerance. Since the problem (4.3) may have multiple local minima, the algorithms may not reach the global minimum for some initial approximation. Running the algorithm with several starting points may increase the degree of certainty that the system (4.2) is not consistent. Moreover, the cross-check of results obtained by different algorithms may also increase that certainty. We use the MATLAB routine `lsqnonlin` with the Levenberg-Marquardt algorithm for solving the problem (4.3).

Numerical Approach II

Another numerical approach to solve the system (4.1) is based on solving a nonlinear programming (NLP) problem with equality constraints.

Consider the following problem:

$$\begin{array}{lll} \min\limits_{y\in\mathbb{R}^{n+2}, l_i\in\mathbb{R}^s} & H(y) = \|y\|_2^2 & \\ \text{s.t.} & \sum\limits_{i=1}^{s} l_i^T \mathbf{A}_j l_i + y_{j+1} = 0, & j = 0, 1, \ldots, n, \\ & \sum\limits_{i=1}^{s} \|l_i\|_2^2 + y_{n+2} - 1 = 0, & \end{array} \tag{4.4}$$

where $y = \begin{bmatrix} y_1 & y_2 & \cdots & y_{n+2} \end{bmatrix}^T$.

Let $y^* \in \mathbb{R}^{n+2}$ be a solution of the problem (4.4). The following two situations can occur:

1. if $H(y^*) = 0$, then the system (4.1) is consistent;

2. if $H(y^*) > 0$, then the system (4.1) is not consistent.

There exist many algorithms for solving NLP problems, such as the interior point algorithm, trust-region-reflective algorithm, active-set algorithm, or a sequential quadratic programming algorithm [109]. Here, the problem (4.4) is solved using interior point methods, which have proved to be effective in practice on solving this type of nonlinearly constrained problems [109]. The algorithm may not reach the global minimum for some initial approximation, so it is recommended to run the algorithm with different starting points. We use the MATLAB routine `fmincon` for solving the problem (4.4).

4.1.1 Description of the SDPreg procedure

To test if the Slater condition holds for SDP problems in the form (2.2), we propose the numerical procedure SDPreg. Since this procedure uses methods that approximately solve a system of quadratic equations, we introduce a specific tolerance denoted by `SCQ` that is the desired accuracy on the test of the Slater condition. When the Approach I is being used, the condition $G(\ell^*) <$ `SCQ` guarantees with a given accuracy that the Slater condition does not hold for the SDP problem (2.2). When the Approach II is being used, one can conclude that the SDP problem (2.2) does not satisfy the Slater condition if $H(y^*) <$ `SCQ`. It is worth mentioning that the SDPreg procedure does not check the feasibility of the SDP problems: here, we always assume that their feasible sets are nonempty. The detailed algorithmic scheme of the SDPreg procedure can be outlined as follows.

Algorithm 5 SDPreg: Procedure for testing regularity on SDP problems

```
input: $\mathbf{A}_j$, $j = 0, 1, ..., n$, $s \times s$ real constraint matrices of a SDP problem in the form (2.2);
       SCQ, numerical tolerance.
output: result on regularity in terms of the fulfilment of the Slater condition.
 1: choose a numerical approach:
 2: if Approach I is selected, then
 3:     solve the nonlinear least squares problem (4.3), and get $\ell^*$
 4:     compute $G(\ell^*)$
 5:     if $G(\ell^*) <$ SCQ, then the system (4.1) is consistent
 6:         return the SDP problem does not satisfy the Slater condition
 7:     else the system (4.1) is inconsistent
 8:         return the SDP problem satisfies the Slater condition
 9: else Approach II is selected:
10:     solve the nonlinear problem (4.4), and get $y^*$
11:     compute $H(y^*)$
12:     if $H(y^*) <$ SCQ, then the system (4.1) is consistent
13:         return the SDP problem does not satisfy the Slater condition
14:     else the system (4.1) is inconsistent
15:         return the SDP problem satisfies the Slater condition
```

The two numerical approaches in the SDPreg procedure 5 are important for a cross-check, *i.e.*, comparison, of results. It should be emphasized that the cross-check of different numerical approaches permits to increase the reliability on the results.

4.1.2 Implementation details

Since many SDP solvers have an interface or work with MATLAB, the SDPreg procedure was implemented entirely in MATLAB language under the name `SDPreg`. We are focused on the practical use of our routine, so our intention was to construct a simple code, publicly available for MATLAB users. The `SDPreg` routine can be found in [90].

On the iterations of the SDPreg algorithm, the existence of solution of the system of quadratic equations (4.1) is verified using the Approaches I or II. The algorithm allows the user to specify which of the two approaches to use. A cross-check of results can be done by running the SDPreg algorithm for one approach, and then for the other, in order to establish their reasonableness.

The solvers `lsqnonlin` and `fmincon` incorporated into the SDPreg procedure stop when one of the following stopping criteria is satisfied:

- $||F(\ell^{(i)}) - F(\ell^{(i+1)})||_\infty <$ `TolFun`, where `TolFun` is a tolerance on the function value;
- $||\ell^{(i)} - \ell^{(i+1)}||_\infty <$ `TolX`, where `TolX` is a tolerance on the argument variable value;
- $||c(\ell)|| >$ `TolCon`, where `TolCon` is a tolerance for constraints violation (only for the Approach II).

The tolerances `TolFun`, `TolX` and `TolCon` might be specified by users. The SDPreg implemented in MATLAB has these tolerances set by default to 10^{-8}.

The proposed SDPreg procedure also needs a pre-specified value on the accuracy of the regularity test, which is given by the parameter `SCQ`. In `SDPreg` this parameter is set to 10^{-4} by default.

It should be stressed out once again that the SDPreg procedure is based on the approximate solution of the nonlinear least-squares problem (4.3) and the NLP problem (4.4). In order to increase the reliability on the results, the following techniques were used in an initial testing phase:

- each solver was run 10 times with different random starting points for each problem;
- each solver was restarted using the last computed approximation;
- different tolerances were used;
- both numerical and analytical Jacobians were used.

Experiments showed that the most powerful technique is running the solver several times with different starting points. Therefore, by default, our program uses randomly chosen starting points while performing the Approaches I and II. The user does not interact, unless the user decided to change the code.

The implementation of the SDPreg procedure in the form of the MATLAB routine `SDPreg` involves auxiliary functions to construct the problems (4.3) and (4.4), and we decided to use nested functions, *i.e.*, there is a main function that calls other functions that are inside it. This permits to use less memory.

Given a SDP problem in the form (2.2), the user needs to present the problem in SDPA sparse format [162] (`dat-s`), as it is explained in the Appendix A. The `dat-s` file with the problem data must be saved in the working directory, which is opened to choose a problem after calling the program `SDPreg`. To read the SDP instances in dat-s format, the `SDPreg` uses the function `read_data.m` from the SDPA package, publicly available in [129].

The user must define the desired accuracy on the regularity test specified by `SCQ`. The basic calling statement structure of the SDPreg procedure is:

```
> SDPreg(SCQ)
```

The working directory will be opened and the user chooses a SDP problem for testing regularity.

After choosing the particular SDP problem, the user must specify which approach to use:

- 1 – for the numerical Approach I,
- 2 – for the Approach II.

When `SDPreg` stops, it will deliver a message reporting that the tested problem satisfies or not the Slater condition.

In Section 4.3, we will test the SDPreg procedure on several SDP problems.

4.2 DIISalg: a numerical procedure to determine the irregularity degree of SDP problems

Testing of regularity on SDP problems can also be done by performing all iterations of the DIIS algorithm from [78]. In Section 3.1.4, we have shown that the dimension s^* of a basis of the immobile index subspace can be considered as an irregularity degree of a given SDP problem and if $s^* \neq 0$, then it can be regarded as a certificate of the failure of the Slater condition.

We have implemented the DIIS algorithm for our numerical tests. The MATLAB function is called `DIISalg` and can be found in [90].

On the first iteration of the `DIISalg` routine, one of the above presented Approach I or Approach II can be used. The user must specify which approach to use. Similar to `SDPreg`, the user needs to specify the desired accuracy on the regularity test using the parameter `SCQ`.

In MATLAB, the basic calling statement structure of the function `DIISalg` is as follows:

```
> DIISalg(SCQ)
```

If the dimension of the immobile index subspace is zero, then `DIISalg` delivers the message that the tested problem satisfies the Slater condition. Otherwise, the `DIISalg` routine will return a basis of the immobile index subspace, as well as its dimension and the conclusion about the nonregularity of the tested SDP problem.

4.3 Numerical experiments

In this section, we present several numerical experiments on testing the regularity of linear SDP problems using instances from a collection of 50 SDP problems found in the literature and also instances from the well know database SDPLIB. On the basis of these tests, we will make some conclusions about relationships between regularity notions in terms of the Slater condition and well-posedness in SDP.

4.3.1 Description of the experiments and numerical results

The numerical experiments were run 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 have implemented the SDPreg procedure and also the DIIS algorithm in MATLAB language under the functions `SDPreg` and `DIISalg`, respectively. Both implementations handle block diagonal matrices and the SDP problems to be tested must be in dat-s format.

For the numerical tests we have used problems from the literature and from the SD-PLIB suite, a collection of 92 linear SDP test problems, provided by Brian Borchers [15] and available at `http://euler.nmt.edu/~brian/sdplib/sdplib.html`. The SDPLIB is a

SDP database containing problems ranging in size from 6 variables and 13 constraints up to 7000 variables and 7000 constraints. The problems are drawn from a variety of applications, such as truss topology design, control systems engineering and relaxations of combinatorial optimization problems. Due to the limited computational resources, we were only able to test 26 small to medium-scale problems from SDPLIB. Notice that all the tested problems from SDPLIB are feasible. The test problems collected from the literature are feasible and were constructed or adapted from [15, 39, 41, 42, 52, 74, 76, 78, 87, 100, 109, 111, 114] and from [121, 137, 140, 149, 164], and can be found in [90]. Notice here that the dimensions of the problems from the literature are rather small. This is explained by the fact that these problems present mostly academic examples, small and easy to solve, but often not regular by construction.

We have chosen empirically the tolerances for the stopping criteria. In our computational experiments with the programs `SDPreg` and `DIISalg`, we have set the tolerances `TolFun`, `TolX` and `TolCon` to 10^{-8} for all the tests. The numerical experiments have shown that for the used tolerances, the numerical results get stabilized, *i.e.*, for termination tolerances less than 10^{-8} all the solvers stop at the same point. Therefore, the value 10^{-8} is considered to be safe to conclude about the fulfilment of the Slater condition. Notice that numerical tests with such small tolerance can be rather time consuming.

In our experiments, we have set the parameter `SCQ` to 10^{-4}. This choice for the value of `SCQ` was considered to be reasonable in practice when large scale problems are involved. If we force a smaller value for this parameter, our procedures will be less efficient, since the computation time of the experiments increases in a non acceptable way.

For a large system in the form (4.1), even with sparse matrices $\mathbf{A}_j$, $j = 0, 1, \ldots, n$, the memory needed to allocate the data is so large that solving such system is quite difficult in a common desktop computer. To reduce the computational time in such cases special procedures for solving systems with sparse data can be developed.

SDP instances from literature

First, we will present the numerical experiments using the MATLAB routine `SDPreg`. Since the test problems collected from the literature are small-scale SDP problems, we will also present the results using the MATLAB function `DIISalg`, in order to compute the irregularity degree of SDP problems when the Slater condition does not hold.

The results obtained using the SDPreg procedure to test the regularity on 50 problems collected from the literature are displayed in the Table 4.1. The first column of the table contains the instance's name. The second and third columns contain the number of variables, n, and the dimension of the constraint matrices, s, respectively. The next three columns represent the obtained results and conclusions about the fulfilment of the Slater condition using the Approach I, where the solver `lsqnonlin` was applied for solving the quadratic system (4.1). The last columns of the table contain the results and conclusions about the fulfilment of the Slater condition using the Approach II, where the solver `fmincon` was applied for solving the quadratic system.

Considering the results obtained using the routine `SDPreg` and reported in Table 4.1,

Table 4.1: Numerical results using `SDPreg` on problems collected from literature (computation time is in seconds).

Problem	n	s	`lsqnonlin`		Slater	`fmincon`		Slater
			$G(\ell^*)$	time	condition	$H(y^*)$	time	condition
example3isa	2	2	$1.1363e-25$	0.036052	no	$6.9483e-15$	0.155928	no
FreundSun	2	3	$5.8581e-1$	0.070417	yes	$5.8581e-1$	0.172609	yes
helmberg1	2	3	$5.4540e-24$	0.060488	no	$1.5165e-13$	0.180284	no
janssondual	4	3	$4.5386e-23$	0.072950	no	$2.4628e-14$	0.281306	no
Jansson1	3	3	$3.2587e-2$	0.138249	yes	$3.2587e-2$	0.105394	yes
Jansson2	3	3	$3.2619e-3$	0.073290	yes	$3.2619e-3$	0.143587	yes
Jansson3	3	3	$3.3027e-1$	0.060724	yes	$3.3027e-1$	0.097528	yes
Jansson4	3	3	$1.4601e-1$	0.069686	yes	$1.4601e-1$	0.093127	yes
Jansson5	3	3	$3.2127e-1$	0.071036	yes	$3.2127e-1$	0.099324	yes
kojima1SDP2006	2	4	$9.5204e-1$	0.223964	yes	$9.5204e-1$	0.188890	yes
kojimaSDP2006	4	3	$2.5000e-1$	0.023337	yes	$2.5000e-1$	0.109484	yes
K-Tn1	1	2	$2.5841e-23$	0.023267	no	$1.3029e-14$	0.118685	no
K-Tn2	2	2	$3.4530e-24$	0.044250	no	$5.5507e-15$	0.216425	no
K-Tn3	3	2	$5.8301e-23$	0.039288	no	$6.3543e-15$	0.124848	no
K-Tn4	4	2	$1.8636e-23$	0.041151	no	$3.8451e-15$	0.159502	no
K-Tn5	5	2	$1.8835e-22$	0.042491	no	$1.1144e-15$	0.190099	no
K-Tn6	6	2	$3.8912e-21$	0.043691	no	$1.7715e-15$	0.183893	no
K-Tn7	7	2	$3.6967e-22$	0.048549	no	$1.8502e-15$	0.270957	no
K-Tn8	8	2	$3.4387e-21$	0.050035	no	$6.6118e-16$	0.214851	no
K-Tn9	9	2	$6.4710e-22$	0.055816	no	$2.1353e-16$	0.274397	no
K-Tn10	10	2	$1.5483e-20$	0.058125	no	$4.1765e-17$	0.340125	no
LuoSturmZhang	2	3	$1.0255e-23$	0.066404	no	$7.3305e-15$	0.404231	no
Mitchell2004	2	3	$5.0000e-1$	0.045751	yes	$5.0000e-1$	0.423425	yes
pataki1	1	2	$4.2345e-24$	0.025313	no	$1.6671e-14$	0.187986	no
pataki1alpha1	1	2	$9.1435e-24$	0.024894	no	$4.1820e-12$	0.048927	no
pataki1alpha2	1	2	$5.2021e-23$	0.024497	no	$2.8976e-14$	0.189384	no
pataki1alpha3	1	2	$7.6752e-25$	0.024970	no	$2.6538e-17$	0.066301	no
pataki1alpha4	1	2	$2.1773e-25$	0.031795	no	$1.4323e-14$	0.183482	no
pataki1alpha5	1	2	$1.5810e-22$	0.026075	no	$7.3292e-15$	0.213028	no
pataki1alpha-1	1	2	$6.0371e-25$	0.030338	no	$1.7755e-14$	0.148022	no
pataki1alpha-2	1	2	$5.5582e-25$	0.037817	no	$9.7994e-15$	0.145855	no
pataki1alpha-3	1	2	$4.2037e-26$	0.038157	no	$4.7202e-15$	0.151887	no
pataki1alpha-4	1	2	$2.3310e-26$	0.034479	no	$9.2661e-15$	0.227901	no
pataki1alpha-5	1	2	$2.7032e-23$	0.024885	no	$1.9230e-14$	0.195679	no
pataki2	2	3	$1.0996e-23$	0.068269	no	$4.6525e-15$	0.443608	no
pataki2.-1	2	3	$2.1120e-22$	0.060252	no	$1.0909e-14$	0.433127	no
pataki2.32	2	3	$5.3683e-21$	0.058523	no	$1.9661e-14$	0.284285	no
pataki2.33	2	3	$1.2313e-22$	0.065221	no	$1.5303e-14$	0.392578	no
polik1	1	2	$4.3130e-24$	0.024449	no	$4.0894e-15$	0.142294	no
polik2	2	3	$2.8493e-23$	0.060726	no	$3.2616e-14$	0.411495	no
polik3	1	2	$1.6265e-26$	0.026367	no	$5.6213e-15$	0.163265	no
polik4	2	3	$3.4310e-24$	0.063227	no	$9.2481e-15$	0.364975	no
polik5	4	3	$1.8360e-24$	0.078022	no	$1.7116e-14$	0.575205	no
polik6	2	2	$7.3225e-25$	0.040701	no	$2.6444e-14$	0.146335	no
polik7	2	2	$1.9659e-24$	0.028269	no	$1.7242e-16$	0.134994	no
polik8	2	2	$3.3333e-1$	0.010292	yes	$3.3333e-1$	0.068612	yes
SturmZhang	3	5	$6.4972e-2$	0.442867	yes	$6.4972e-2$	0.326788	yes
Todd	2	2	$3.1557e-23$	0.029669	no	$2.2196e-15$	0.140185	no
VandBoyd1	2	3	$1.5896e-23$	0.025929	no	$5.1099e-15$	0.137883	no
YumingZhang1995	4	4	$1.4186e-22$	0.125032	no	$8.9888e-17$	0.407860	no

we can see that for the feasible problems collected from the literature, both numerical approaches, Approach I and Approach II, performed very well and the obtained results coincide for the two proposed approaches. We can also see that `SDPreg` is quite fast.

From Table 4.1, we can see that 39 of the 50 tested problems do not satisfy the Slater condition, while 11 do satisfy.

To complement the information on (ir)regularity, we have also applied to these 50 problems the routine `DIISalg`. In this case, the computation time is expected to be larger than using the `SDPreg`. The Table 4.2 displays the results using all the iterations of the `DIISalg` for the 50 problems collected from the literature.

The first column in Table 4.2 contains the instance's name. The next two columns contain the number of variables, n, and the dimension of the constraint matrices, s. The remaining columns contain the numerical results using the Approach I and the Approach II in the DIIS algorithm. These columns contain the dimension of the immobile index subspace, s^*, the number of iterations, the computation time, in seconds, and the result on regularity.

Table 4.2: Numerical results using `DIISalg` on problems collected from literature (computation time is in seconds).

Problem	n	s	`lsqnonlin`			Slater	`fmincon`			Slater
			s^*	iter	time	condition	s^*	iter	time	condition
example3isa	2	2	1	2	0.226202	no	1	2	0.539775	no
FreundSun	2	3	0	1	0.146649	yes	0	1	0.259852	yes
helmberg1	2	3	1	2	0.178968	no	1	2	0.658152	no
janssondual	4	3	2	2	0.140688	no	2	2	0.507312	no
Jansson1	3	3	0	1	0.244887	yes	0	1	0.170550	yes
Jansson2	3	3	0	1	0.109592	yes	0	1	0.193711	yes
Jansson3	3	3	0	1	0.525396	yes	0	1	0.279601	yes
Jansson4	3	3	0	1	0.114246	yes	0	1	0.127950	yes
Jansson5	3	3	0	1	0.101950	yes	0	1	0.144949	yes
kojima1SDP2006	2	4	0	1	0.387002	yes	0	1	0.303760	yes
kojimaSDP2006	4	3	0	1	0.032463	yes	0	1	0.174287	yes
K-Tn1	1	2	1	2	0.072533	no	1	2	0.447973	no
K-Tn2	2	2	1	2	0.093047	no	1	2	0.447500	no
K-Tn3	3	2	1	2	0.083016	no	1	2	0.538221	no
K-Tn4	4	2	1	2	0.097496	no	1	2	0.637090	no
K-Tn5	5	2	1	2	0.120930	no	1	2	0.670066	no
K-Tn6	6	2	1	2	0.140890	no	1	2	0.762693	no
K-Tn7	7	2	1	2	0.142380	no	1	2	0.766286	no
K-Tn8	8	2	1	2	0.148806	no	1	2	0.877893	no
K-Tn9	9	2	1	2	0.145744	no	1	2	0.873110	no
K-Tn10	10	2	1	2	0.135638	no	1	2	0.907883	no
LuoSturmZhang	2	3	1	2	0.162486	no	1	2	0.518502	no
Mitchell2004	2	3	0	1	0.070062	yes	0	1	0.634781	yes
pataki1	1	2	1	2	0.121993	no	1	2	0.433417	no
pataki1alpha1	1	2	1	2	0.100748	no	1	2	0.806680	no
pataki1alpha2	1	2	1	2	0.129894	no	1	2	0.497163	no
pataki1alpha3	1	2	1	2	0.071018	no	1	2	0.458166	no
pataki1alpha4	1	2	1	2	0.127574	no	1	2	0.428617	no
pataki1alpha5	1	2	1	2	0.180141	no	1	2	0.479815	no
pataki1alpha-1	1	2	1	2	0.170281	no	1	2	0.419834	no
pataki1alpha-2	1	2	1	2	0.099228	no	1	2	0.525398	no
pataki1alpha-3	1	2	1	2	0.074760	no	1	2	0.555413	no
pataki1alpha-4	1	2	1	2	0.099995	no	1	2	0.824851	no
pataki1alpha-5	1	2	1	2	0.131180	no	1	2	0.445150	no
pataki2	2	3	1	2	0.177333	no	1	2	0.523884	no
pataki2.-1	2	3	1	2	0.201498	no	1	2	0.620947	no
pataki2.32	2	3	1	2	0.227159	no	1	2	0.559715	no
pataki2.33	2	3	1	2	0.191399	no	1	2	0.667981	no
polik1	1	2	1	2	0.091698	no	1	2	0.392195	no
polik2	2	3	1	2	0.165904	no	1	2	0.611956	no
polik3	1	2	1	2	0.084446	no	1	2	0.431411	no
polik4	2	3	1	2	0.151859	no	1	2	0.642326	no
polik5	4	3	1	2	0.319945	no	1	2	1.059643	no
polik6	2	2	1	2	0.097512	no	1	2	0.527270	no
polik7	2	2	1	2	0.114855	no	1	2	0.477063	no
polik8	2	2	0	1	0.020385	yes	0	1	0.110016	yes
SturmZhang	3	5	0	1	0.507338	yes	0	1	0.380577	yes
Todd	2	2	1	2	0.089365	no	1	2	0.561483	no
VandBoyd1	2	3	1	2	0.086054	no	1	2	0.422023	no
YumingZhang1995	4	4	3	2	0.301380	no	3	2	1.424340	no

Observing Table 4.2, we can see that all the nonregular problems, *i.e.*, for which the Slater condition fails to hold, exhibit minimal irregularity degree, with the exception of the problems *janssondual* and *YumingZhang1995* that present irregularity degrees equal to 2 and 3, respectively.

Based on the experiments on these small-scale SDP problems, we conclude that SDPreg is an efficient procedure to verify the Slater condition on SDP. We also conclude that the DIIS algorithm permits to complement the information about the (non)regularity of the SDP problem, providing an irregularity degree of problems that do not satisfy the Slater condition.

SDP instances from SDPLIB

Here, we present the numerical experiments with `SDPreg` and `DIISalg` on SDP instances from the SDPLIB database. The description of the contents of the tables is quite similar to the above ones, so, we omit it. The lack of results in Tables 4.3 and 4.4 corresponds to the cases when we were unable to complete the test, since the running time increased in an unacceptable way.

The Table 4.3 presents the numerical results obtained while testing the regularity of SDP problems using the SDPreg procedure on SDP instances from the SDPLIB suite.

Table 4.3: Numerical results using `SDPreg` on problems from SDPLIB (computation time is in seconds).

Problem	n	s	`lsqnonlin`		Slater	`fmincon`		Slater
			$G(\ell^*)$	time	condition	$H(y^*)$	time	condition
control1	21	15	$3.3333e-1$	601.1	yes	$3.3333e-1$	49.0	yes
control2	66	30	$3.3333e-1$	38720.5	yes	$3.3333e-1$	924.1	yes
control3	136	45	$3.3333e-1$	251020.6	yes	$3.3333e-1$	9921.5	yes
hinf1	13	14	$3.2289e-3$	390.3	yes	$3.2289e-3$	25.2	yes
hinf2	13	16	$1.1797e-6$	517.0	no	$1.1797e-6$	199.5	no
hinf3	13	16	$1.7767e-7$	606.7	no	$1.7767e-7$	99.1	no
hinf4	13	16	$4.4624e-5$	1050.7	no	$4.4622e-5$	62.7	no
hinf5	13	16	$2.7754e-9$	2587.2	no	$2.7754e-9$	203.1	no
hinf6	13	16	$6.9476e-9$	457.5	no	$6.9476e-9$	100.7	no
hinf7	13	16	$1.6784e-10$	176.9	no	$1.6786e-10$	153.9	no
hinf8	13	16	$1.2890e-7$	221.1	no	$1.2890e-7$	108.3	no
hinf9	13	16	$2.2907e-11$	166.4	no	$2.2022e-11$	185.6	no
hinf10	21	18	$4.2815e-5$	325.2	no	$4.2813e-5$	113.6	no
hinf11	31	22	$3.5093e-4$	446.8	yes	$3.5089e-4$	307.2	yes
hinf12	43	24	$1.8611e-5$	908.6	no	$1.8611e-5$	562.8	no
hinf13	57	30	$6.4114e-9$	25491.2	no	$4.3625e-9$	2998.5	no
hinf14	73	34				$2.3778e-6$	4327.7	no
hinf15	91	37				$7.9113e-10$	19670.0	no
qap5	136	26	$4.7906e-1$	114006.8	yes	$4.7904e-1$	2911.1	yes
qap6	229	37	$4.8191e-1$	429849.5	yes	$4.8191e-1$	17965.0	yes
qap7	358	50	$4.8414e-1$	428798.8	yes	$4.8048e-1$	52047.2	yes
qap8	529	65	$4.8192e-1$	516925.5	yes	$4.8193e-1$	120428.9	yes
theta1	104	50				$4.9999e-1$	15068.3	yes
truss1	6	13	$1.4285e-1$	4.1	yes	$1.4284e-1$	7.1	yes
truss3	27	31	$3.2258e-2$	180.1	yes	$3.2256e-2$	399.9	yes
truss4	12	19	$7.6923e-2$	19.9	yes	$7.6923e-2$	29.1	yes

From the Table 4.3, we can observe that the `SDPreg` with the Approach I, based on the `lsqnonlin` solver for solving (4.1), was able to test 23 instances, while `SDPreg` with the Approach II, based on the `fmincon` solver for solving (4.1), tested the regularity of all the 26 problems. Based on these experiments, the Approach II seems to be the most efficient, since the computation time is better for almost all problems and it was able to check the regularity of all the 26 SDP problems.

Considering the results in Table 4.3, we can see that for 23 of the 26 tested problems the results using both numerical approaches do coincide. Our experiments show that 12 tested problems satisfy the Slater condition, while 11 do not satisfy. Notice that the procedure with the Approach I was not able to check the regularity of 3 SDP problems, since the computation time have increased in an unacceptable way. The tests using `SDPreg` with the Approach II showed that 13 instances satisfy the Slater condition, while other 13 do not.

A more detailed analysis of the results presented in the Table 4.3 permits to observe that the SDPreg procedure may return slightly different numerical values of the objective functions of the problems (4.3) and (4.4), when the Approach I or the Approach II is used, respectively. Therefore, when using the SDPreg procedure to check the Slater condition we recommend to run the procedure a couple of times, for different starting points, in order to increase the quality of the results.

The following table presents the results obtained when the DIIS algorithm was applied to test regularity of SDPLIB instances.

Table 4.4: Numerical results using `DIISalg` on problems from SDPLIB (computation time is in seconds).

Problem	n	s	`lsqnonlin`			Slater	`fmincon`			Slater
			s^*	iter	time	condition	s^*	iter	time	condition
control1	21	15	0	1	939.7	yes	0	1	50.5	yes
control2	66	30	0	1	42405.0	yes	0	1	1287.2	yes
control3	136	45	0	1	259224.2	yes	0	1	12657.3	yes
hinf1	13	14	0	1	461.9	yes	0	1	26.1	yes
hinf2	13	16	16	2	816.8	no	16	2	230.6	no
hinf3	13	16	16	2	5301.1	no	16	2	135.0	no
hinf4	13	16	16	2	13675.2	no	16	2	86.2	no
hinf5	13	16	16	2	22167.8	no	16	2	221.6	no
hinf6	13	16	16	2	101429.0	no	16	2	122.3	no
hinf7	13	16	16	2	23130.5	no	16	2	202.4	no
hinf8	13	16	16	2	1825.3	no	16	2	151.5	no
hinf9	13	16	16	2	1562.2	no	16	2	827.9	no
hinf10	21	18					18	2	161.1	no
hinf11	31	22					0	1	375.0	yes
hinf12	43	24					24	2	1038.6	no
hinf13	57	30					30	2	5902.5	no
hinf14	73	34					34	2	10123.4	no
hinf15	91	37					37	2	55173.8	no
qap5	136	26	0	1	118064.7	yes	0	1	3508.2	yes
qap6	229	37	0	1	432101.6	yes	0	1	21827.1	yes
qap7	358	50	0	1	433254.7	yes	0	1	57981.6	yes
qap8	529	65	0	1	518749.0	yes	0	1	121362.1	yes
theta1	104	50					0	1	18840.7	yes
truss1	6	13	0	1	4.0	yes	0	1	7.5	yes
truss3	27	31	0	1	178.5	yes	0	1	408.5	yes
truss4	12	19	0	1	20.0	yes	0	1	30.6	yes

First of all, let us observe that the computation time is much better when we use the `fmincon` solver of the Approach II for solving the quadratic system (4.1). The only exceptions are the problems *truss1*, *truss3* and *truss4*.

Considering the results displayed in Table 4.4, we can observe that for 19 of the 26 tested problems the results using both numerical approaches do coincide. The experiments show that for both cases 11 tested problems satisfy the Slater condition, while 8 do not satisfy. Notice that for 7 SDP problems, the `lsqnonlin` solver, that is used in the implementation of the Approach I to provide a solution of the system (4.1), was unable to complete its task, since the computation time increased in an unacceptable way. From the Table 4.4, we can see that testing the regularity using `DIISalg` is more time consuming than `SDPreg`, since more iterations are performed.

Observing the Table 4.4, we can conclude that all the nonregular problems present maximal irregularity degree.

Based on these experiments, we can assert that the numerical procedure `SDPreg` is quite fast and efficient.

4.3.2 Comparison of regularity results

In what follows, we compare the results of the numerical testing of regularity in terms of the fulfilment of the Slater condition obtained in the previous section with the results on testing of well-posedness in the same SDP problems reported in [38] and [61]. The numerical results on well-posedness presented in [38] and [61] seem to be the only available results on testing regularity of SDP problems in some sense, and were obtained using implementations of the procedures described in the Section 3.2. While in [61] a detailed algorithm (the upper bound algorithm) was presented to check well-posedness of a SDP problem, in [38] no specific algorithm was included.

For the comparison, we use our numerical results obtained for 26 instances from SDPLIB presented in the previous section using `SDPreg` and `DIISalg` with the Approach II, which showed to be the most efficient numerical approach for solving the quadratic system (4.1). Notice that all the feasible problems from SDPLIB (including the above 26 instances) were tested in [38] and [61], since more powerful computational resources were used.

To compare our numerical results with those from [38] and [61], we present all these results in the Table 4.5. The first column of the table contains the instance's name used in the SDPLIB database. The next two columns refer to the number of variables, n, and the dimension of the constraint matrices, s. The fourth column contains the results obtained with the SDPreg procedure (*i.e.*, the problem satisfies or not the Slater condition). The next column contains the dimension of a basis of the immobile index subspace, s^*, found by the DIIS algorithm. Column 6 contains the lower and upper bounds of the condition number C reported in [38] and the last column presents the upper bound for the primal objective function from [61].

Table 4.5: Numerical results on testing regularity using: `SDPreg` to check the Slater condition, `DIISalg` (if $s^* = 0$, then the Slater condition holds for the SDP problem), the lower and upper bounds of the Renegar condition number from [38] and the rigorous upper bound of the optimal value from [61] (if C or $\bar{p}^*$ is finite, then the SDP problem is well-posed).

			`SDPreg`	`DIISalg`	Results from [38]		Results from [61]
Problem	n	s	Slater	s^*	C		$\bar{p}^*$
			condition		lower bound	upper bound	
control1	21	15	yes	0	8.3×10^5	1.8×10^6	-1.7782×10^1
control2	66	30	yes	0	3.9×10^6	1.3×10^7	-8.2909×10^0
control3	136	45	yes	0	2.0×10^6	1.2×10^7	-1.3615×10^1
hinf1	13	14	yes	0	∞	∞	∞
hinf2	13	16	no	16	3.5×10^5	5.6×10^5	-7.1598×10^0
hinf3	13	16	no	16	∞	∞	∞
hinf4	13	16	no	16	∞	∞	∞
hinf5	13	16	no	16	∞	∞	∞
hinf6	13	16	no	16	∞	∞	∞
hinf7	13	16	no	16	∞	∞	∞
hinf8	13	16	no	16	∞	∞	∞
hinf9	13	16	no	16	2.0×10^7	3.6×10^7	∞
hinf10	21	18	no	18	∞	∞	∞
hinf11	31	22	yes	0	∞	∞	∞
hinf12	43	24	no	24	∞	∞	∞
hinf13	57	30	no	30	∞	∞	∞
hinf14	73	34	no	34	∞	∞	∞
hinf15	91	37	no	37	∞	∞	∞
qap5	136	26	yes	0	∞	∞	∞
qap6	229	37	yes	0	∞	∞	∞
qap7	358	50	yes	0	∞	∞	∞
qap8	529	65	yes	0	∞	∞	∞
theta1	104	50	yes	0	2.0×10^2	2.1×10^2	-2.3000×10^1
truss1	6	13	yes	0	2.2×10^2	3.0×10^2	9.0000×10^0
truss3	27	31	yes	0	7.4×10^2	1.9×10^3	9.1100×10^0
truss4	12	19	yes	0	3.6×10^2	7.7×10^2	9.0100×10^0

Table 4.6: Summary of regularity tests in terms of the fulfilment of the Slater condition and well-posedness according to [38].

	Slater condition	
	Regular	*Nonregular*
well-posed	7	2
ill-posed	6	11

Table 4.7: Summary of regularity tests in terms of the fulfilment of the Slater condition and well-posedness according to [61].

	Slater condition	
	Regular	*Nonregular*
well-posed	7	1
ill-posed	6	12

As can be seen in the Table 4.5, the results on well-posedness obtained in [38] and [61] do not always coincide. In order to ease the comparison of results of regularity in terms of the Slater condition with that of well-posedness, we construct the Tables 4.6 and 4.7.

In Table 4.6, the lines correspond to well and ill-posed problems classified in the basis of the test from [38], and the columns correspond to regular and nonregular problems in terms of the Slater condition, *i.e.*, the results of our numerical tests with the SDPreg procedure. On the intersection, we have the number of problems that satisfy both corresponding conditions. Table 4.7 is constructed in a similar way, but the lines correspond to the number of the well and ill-posed problems classified on the basis of the experiments in [61].

From the Table 4.6, we can see that 18 of the tested problems either satisfy the Slater condition and are well-posed, or do not satisfy the Slater condition and are ill-posed, simultaneously, and 6 of the ill-posed problems do satisfy the Slater condition. The only exceptions are the problems *hinf2* and *hinf9* that are nonregular in terms of the fulfilment of the Slater condition and well-posed according to [38]. This contradiction to the Lemma 2 can be explained by the fact that our numerical procedures and those used in [38] are based on approximated calculus and may be not precise.

Comparing now our regularity results of testing the Slater condition with those from [61] where the same problems were tested in terms of well-posedness, observing the Table 4.7 we conclude that for 19 problems these results coincide, *i.e.*, the problems satisfy the Slater condition and are well-posed, or do not satisfy the Slater condition and are ill-posed, simultaneously. It can also be observed that there is also a contradiction to the Lemma 2: the problem *hinf2* does not satisfy the Slater condition, but is well-posed according to [61].

Moreover, notice that the numerical results of well-posedness obtained in [38] and in [61] do not coincide: the problem *hinf9* is well-posed according to [38] and ill-posed according to [61]. This can be connected with the fact that nevertheless the condition number C is finite, it is rather big and the problem is very close to be ill-posed. It may also be due to the tests were performed in nonexact arithmetic and/or with different numerical procedures.

Finally, notice that in [61], it is reported that the problem *hinf8* is well-posed, although the results presented in the same paper (and also in [38]) show that this problem is ill-posed. Our numerical tests show that this problem does not satisfy the Slater condition.

The comparison of our tests with those reported in [38] and [61] confirm the conclusions about the relationship between the regularity notions in SDP.

It is worthwhile mentioning that there are no reports of numerical tests of the good behaviour in the sense of Pataki on SDP problems.

4.4 Conclusions

In this chapter, we have presented the numerical procedure SDPreg for testing the Slater condition in SDP. We have implemented the SDPreg procedure in MATLAB, as well as the DIIS algorithm that determines the irregularity degree of SDP problems. Numerical experiments are carried out on several SDP instances, some collected from the literature, and other from the SDPLIB suite. These experiments permit to conclude that both the DIIS algorithm and the SDPreg procedure are efficient for testing the Slater condition. Furthermore, the outputs of our implementation of the DIIS algorithm can be applied to verify the optimality of some feasible solution using the optimality criterion formulated in the Theorem 19, and also in the development of new SDP methods. The majority of results obtained in the experiments confirm our conclusions about the relationships between the tested regularity notions. Nevertheless, some technical difficulties should be mentioned. Both the construction of a set of linearly independent vectors, and solution of the problems (4.3) and (4.4) presented in the Approaches I and II, are difficult tasks from the numerical viewpoint.

In the following chapter, we will present an algorithm for generating nonregular SDP problem instances, *i.e.*, for which the Slater condition fails to hold.

Chapter 5

Generating nonregular instances in semidefinite programming

In this chapter, we describe a class of nonregular SDP problems with optimal value zero and a prescribed irregularity degree, and construct an algorithm for generating such problems. The resulting set of nonregular SDP instances is used to numerically test the behaviour of popular SDP solvers in solving nonregular problems. Numerical results are presented and discussed.

5.1 An algorithm for generating nonregular SDP instances

In the previous chapters, we defined different notions of regularity and showed that they play an important role in solving SDP problems. The existing libraries of test problems do not have indication if their instances are regular or not. It is known that the behaviour of SDP solvers may be compromised in the case of nonregularity. Therefore, it is important to have access to information about certain regularity properties of the test problems of the existing SDP libraries, or to create new ones, for example, just containing nonregular SDP test problems. This idea of creating libraries of "bad" instances in some sense is not new. Thus, as remarked in [122], it would be important to have a library of infeasible SDP instances to test and develop new stopping criteria for SDP methods. In [85], an algorithm for generating infeasible SDP instances is proposed. In this light, we propose a generator of nonregular SDP problem instances with predefined properties.

In this chapter, we present a procedure for generating a class of SDP problems that fails the Slater condition, based on the characterization of badly-behaved systems introduced by Pataki in [111].

5.1.1 A class of nonregular SDP problems

Nonregular SDP instances are not so rare in practice and various studies show that state-of-the-art SDP solvers can run into numerical difficulties in obtaining their solution (see, *e.g.*, [23, 38, 49, 61, 152]). Although popular SDP solvers use some special techniques to handle SDP problems failing the Slater condition (such as self-dual embedding), they may still produce erroneous results. As shown in the Example 11, the SDP solvers SDPT3 and SeDuMi were used to solve a nonregular SDP instance and both have failed to obtain the true solution. Nonregular SDP problems are a challenge to current SDP solvers. It is important to have a set of nonregular SDP test problems to evaluate the performance and efficiency of SDP solvers.

Before proceeding further, let us consider some nonregular SDP instances collected from the SDP literature. For these instances, we have tested the regularity in terms of the Slater condition and determined their irregularity degrees in the Section 4.3 with our procedure `DIISalg`. We now take a closer look to their structure.

Example 18 *The problem helmberg1 is given by*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} 0 & x_1 & 0 \\ x_1 & x_2 & 0 \\ 0 & 0 & x_1+1 \end{bmatrix} \succeq 0. \end{array}$$

The constraint matrix has a diagonal entry equal to zero, and to be positive semidefinite, it should satisfy the condition $x_1 = 0$. Therefore, any feasible solution has the form $(0, x_2)$, $x_2 \in \mathbb{R}$, and the optimal value of this problem is 0.

Applying the `DIISalg` on the above problem, it shows that the problem is nonregular with irregularity degree $s^* = 1$.

Example 19 *The problem Janssondual of the set of problems collected from the SDP literature has the form*

$$\begin{array}{ll} \min & x_1 \\ \text{s.t.} & \begin{bmatrix} -x_2 & \frac{1+x_1}{2} & -x_3 \\ \frac{1+x_1}{2} & 0 & -x_4 \\ -x_3 & -x_4 & 0 \end{bmatrix} \succeq 0. \end{array}$$

Its constraint matrix has two null diagonal entries and it is easy to see that for any feasible solution we have $x_3 = x_4 = 0$ and $x_1 = -1$. Therefore, the optimal value is -1.

The procedure `DIISalg` shows that the problem is nonregular with irregularity degree $s^* = 2$.

Example 20 *The problem YumingZhang1995 can be written as*

$$\begin{array}{ll} \min & 10x_4 \\ \text{s.t.} & \begin{bmatrix} 0 & -1-x_4 & x_2 & 0 \\ -1-x_4 & x_1 & x_3 & 0 \\ x_2 & x_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \succeq 0. \end{array}$$

The constraint matrix has three null diagonal entries, and any feasible solution must satisfy $x_2 = x_3 = 0$ and $x_4 = -1$. Therefore, the optimal value is -10.

The irregularity degree of this nonregular problem computed by `DIISalg` *is $s^* = 3$.*

Empirical evidence in several examples of nonregular problems suggests that the number of zeros in the main diagonal of the constraint matrix is equal to the value of the irregularity degree of the problem.

Based on the Theorems 21 and 23, we propose an algorithm to construct a class of nonregular SDP problems with a desired irregularity degree s^*, $1 \leq s^* \leq s-1$, where its optimal value is $p^* = 0$. The following proposition establishes conditions to construct such nonregular problems.

Proposition 11 *Suppose that a SDP problem in the form (2.2) satisfies the conditions:*

1. *integers $s \geq 2$, $1 \leq n \leq \frac{s(s+1)}{2}$ and $r = 1, ..., s-1$,*
2. *c is a n-dimensional vector with $c^T = \left[\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right]$,*
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 $\mathbf{D}_r = \operatorname{diag}(\beta_1, \dots, \beta_r)$ with $\beta_i \in \mathbb{R}^+$, $i = 1, ..., r$,*
4. *for $i = 1, ..., n$,*
$$\mathbf{A}_i = \left[\begin{array}{c:c} \mathbf{F}_i & \mathbf{G}_i \\ \hdashline \mathbf{G}_i^T & \mathbf{H}_i \end{array}\right]_{s \times s}, \tag{5.1}$$
where $\mathbf{F}_i \in S(r)$, $i = 1, ..., n$; $\mathbf{G}_i \in \mathbb{R}^{r \times (s-r)}$, $i = 1, ..., n-s$, are linearly independent matrices chosen to be multiples of a vector of the canonical basis of $\mathbb{R}^{r \times (s-r)}$; $\mathbf{G}_1 \neq \mathbf{0}$, but it is allowed to be a linear combination of the vectors of the canonical basis of $\mathbb{R}^{r \times (s-r)}$; for $i \geq s$, $\mathbf{G}_i = \mathbf{0}$; $\mathbf{H}_i \in S(s-r)$, $i = 1, ..., n$, has null diagonal and $\mathbf{H}_1 = \mathbf{0}$.

Then, this problem is nonregular with irregularity degree $s^* = s - r$ and optimal value $p^* = 0$, since by construction any feasible solution has $x_1 = 0$.

Using this proposition we can construct a generator of nonregular SDP instances in the form (2.2), with a pre-specified irregularity degree, for which the optimal value is known to be equal to zero.

5.1.2 Generating nonregular SDP instances

In this section, we present an algorithm for constructing nonregular SDP instances with a pre-specified irregularity degree, based on the Proposition 11. We can outline the algorithm as follows.

Algorithm 6 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 $\mathbf{A}_0 = -\left[\begin{array}{c:c} \mathbf{D}_r & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} \end{array}\right]$
 4: generate randomly symmetric $(r \times r)$ matrices $\mathbf{F}_i$, $i = 1, ..., n$
 5: generate the canonical basis of $\mathbb{R}^{r \times s^*}$, $\mathbf{T} = \{\mathbf{T}_j, j = 1, ..., rs^*\}$
 6: choose the matrix $\mathbf{G}_1 \neq \mathbf{0} \in \mathbb{R}^{r \times s^*}$ to be a linear combination of the elements of the basis $\mathbf{T}$
    with arbitrary coefficients
 7: choose the matrices $\mathbf{G}_i \in \mathbb{R}^{r \times s^*}$, $i = 2, ..., n - s$, such that $\mathbf{G}_i = \alpha \mathbf{T}_j$, $\mathbf{T}_j \in \mathbf{T}$, $\alpha \in \mathbb{R}$, and
    the set formed by the matrices $\mathbf{G}_i$ and $\mathbf{G}_1$ is linearly independent
 8: for $i \geq s$ do
 9:     $\mathbf{G}_i = \mathbf{0}$
10: set $\mathbf{H}_1 = \mathbf{0}$
11: choose $\mathbf{H}_i \in S(s^*)$, $i = 2, ..., n$, having a null diagonal
12: for $i = 1, ..., n$ do
13:     $\mathbf{A}_i = \left[\begin{array}{c:c} \mathbf{F}_i & \mathbf{G}_i \\ \hdashline \mathbf{G}_i^T & \mathbf{H}_i \end{array}\right]$
14: set the coefficient vector of the objective function to $c_1 = 1$ and $c_i = 0$, for $i = 2, ..., n$
15: return $\mathbf{A}_i$, $i = 0, 1, ..., n$, and $c$.
```

We can establish the following results.

Theorem 25 *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 6, the following properties hold for any problem of the form* (2.2) *generated by the Algorithm 6:*

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 6 is a SDP problem of the form (2.2). It is feasible, since it admits the trivial solution.

By construction, the constraint matrices $\mathbf{A}_i$, $i = 1, ..., n$, have the form (5.1) 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 2 for positive semidefinite matrices, it follows that any feasible solution has $x_1 = 0$. Since the objective function is x_1, then the objective value of the generated problem is always zero. Hence, it is easy to see that all feasible solutions are optimal and the optimal value is $p^* = 0$.

It remains to show the failure of the Slater condition on such SDP problem. The essential observation is that, by construction, the constraint matrix of the generated problem is always negative semidefinite for any feasible solution, since its main diagonal has s^* elements equal to zero. Therefore, the problem can not satisfy the Slater condition. ■

It is evident that the SDP instances generated by the Algorithm 6, which are specially constructed to fail the Slater condition and to have optimal value zero, can be used to test SDP solvers and new methods, as well as procedures that verify regularity properties, such as SDPreg and DIISalg. In the following section, we present some numerical experiments.

5.2 Implementation and numerical experiments

We begin this section by describing the implementation of the Algorithm 6. Then, we present a new test set of nonregular SDP instances called NONREGSDP. Numerical experiments on randomly generated instances are carried out to evaluate the performance of popular SDP solvers, namely SDPT3 and SeDuMi, available on the CVX package.

5.2.1 `nonregSDPgen`: a nonregular SDP instance generator

We have implemented the Algorithm 6 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 $s-1$. The `nonregSDPgen` function is publicly available in [90].

In the steps of the Algorithm 6, 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 a linear combination 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, *e.g.*, `examplename.dat-s`, should be specified as well. The basic calling statement structure of the `nonregSDPgen` function is as follows.

```
> 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.

5.2.2 NONREGSDP: a nonregular SDP database

It is known that 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" [15]. The SDPLIB [15] 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 [38], 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.

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 publicly available at [90] and is described in the Appendix D. 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 the developed MATLAB function `DIISalg` in order to confirm the irregularity degree of the SDP instances.

In the following section, we used 54 instances from the NOREGSDP database to test the computational behaviour of the popular SDP solvers SDPT3 and SeDuMi.

5.2.3 Numerical results and discussion

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, SDPT3 and SeDuMi, and we used their default precision or tolerance values.

The numerical results of the tests are displayed in the Tables 5.1 and 5.2. 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 5.1 and 5.2 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 *obs* 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.

Table 5.1: 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	obs
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 5.2: 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	obs
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 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 5.1, 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 some 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 5.2, it can be observed that SeDuMi reported 5 warning messages about numerical problems on solving the given SDP instances.

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 5.2 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 5.2, 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.

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.

5.3 Conclusions

In this chapter, we have 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. 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.

We have also used `DIISalg` on such generated nonregular instances to test their nonregularity and compute the associated irregularity degree. The routine `DIISalg` confirmed that all instances fail the Slater condition and returned the correct values of the irregularity degrees.

In the next chapter, we will focus on a particular SDP application to data analysis, where large-scale data is expected.

Chapter 6

Application of semidefinite programming in data analysis

In this chapter, we are concerned specifically with one interesting application of semidefinite programming in data analysis for clustering and dimensionality reduction techniques. These tasks are important to the analysis of (large) data sets, where it is often needed not only to reduce the dimension of the attribute space (dimensionality reduction), but also to reveal some patterns among the objects (clustering). Here, we describe some models of a minimum sum-of-squares clustering problem, where the distances between pairs of objects are used to measure (dis)similarities. We will focus on a nonlinear SDP model and its linear SDP relaxations. We study the regularity of these SDP models and methods for solving them. We propose a SDP-based approximation algorithm for solving clustering and dimensionality reduction problems. Numerical experiments are carried out using various data sets.

6.1 Brief introduction and motivation

The advances of computer technology have enabled to store large databases, such as, for example, data sets of sequenced genomes. When dealing with real data sets, it is often needed not only to reduce the dimension of the attribute space (dimensionality reduction), but also to reveal some patterns hidden on data (clustering).

Clustering methods and principal component analysis (PCA) are powerful techniques, very important for data visualization. These techniques have been widely studied and applied to many real-life data, and in areas such as statistics, data mining, machine learning, pattern recognition, engineering, computational biology and image processing [7, 73, 160].

The reduction of the object space is usually done by applying a clustering method to a given data set. Clustering is an unsupervised learning technique that aims to partition a given finite set of objects into a finite number of subsets, called clusters, based on some similarity criterion. The clusters are constructed in such a way that the objects within a cluster are more similar to one another, than the objects belonging to different clusters.

There are different types of clustering techniques, such as hierarchical and partitional clustering. We focus on partitional clustering, where nonoverlapping clusters are constructed. In such a problem, each object can be considered as a point in a n-dimensional space and each cluster can be identified by its centre, called centroid, a non-observable object calculated by taking the mean of all the objects assigned to this cluster [73, 151, 160]. To express similarity between objects, *i.e.*, homogeneity inside a cluster, several similarity measures have been proposed, such as a metric defined on the data set [6, 10]. One of the most used (dis)similarity measures is the squared Euclidean distance [7, 3, 50, 73, 160]. Given a number of objects, the idea is to minimize the sum of the squared distances between each object and the corresponding cluster centroid. The resulting problem is called in the literature the minimum sum-of-squares clustering (MSSC) problem (see, *e.g.*, [3, 6, 9, 50, 73, 116, 153, 160]). The MSSC problem is usually formulated as a binary integer programming problem [6, 115, 116, 153], that in turn can be rewritten either as a (0,1) - semidefinite programming (SDP) problem [81, 115, 116], or as an unconstrained nonsmooth and nonconvex nonlinear problem [3, 9, 10, 153]. The MSSC problem is NP-hard [3, 58]. Many clustering algorithms have been developed to solve it, the most popular of them being, by far, the K-means algorithm [7, 9, 50, 73, 160]. Here, we focus on a nonlinear SDP-based model for MSSC and its linear SDP relaxations proposed in [115, 116], and study their regularity and appropriate methods that can be used to solve them.

PCA is a common statistical technique for unsupervised dimension reduction of data. It finds linear combinations of all the original attributes, called components or principal components, that are able to explain the maximum variability of the data, *i.e.*, the data compression based on correlated attributes is done with minimum information loss [27, 64]. PCA uses an orthogonal projection of the data onto a lower dimensional space along the direction where the data present the highest variability. This technique can be performed in an equivalent form as either an eigendecomposition of the data covariance (or correlation) matrix, or a Singular Value Decomposition (SVD) of the column-centred (or standardized) data matrix. The resulting components are mutually uncorrelated and can be ordered by variance [50]. In PCA, there are as many principal components as the number of the original attributes and typically, the coefficients of the components, also called loadings, are nonzero, which can be considered a shortcoming for interpretation [50, 64]. Various PCA-based methodologies have been proposed to obtain disjoint or sparse components, *i.e.*, with zero loadings (see, *e.g.*, [7, 27, 32, 65, 88, 151, 166]). Some of these approaches to get more interpretable components involve an attribute clustering (see, *e.g.*, [7, 32, 151]). The resulting components are disjoint, meaning that each attribute contributes at most to a single component.

A new methodology of PCA called clustering and disjoint principal component analysis (CDPCA) was proposed by Vichi and Saporta in [151]. CDPCA permits to cluster the objects along a set of centroids and, at the same time, partition the attributes into a reduced set of components, in order to maximize the between cluster deviance. The resulting problem is formulated as a quadratic mixed integer programming problem and an alternating least-squares (ALS) algorithm is proposed to solve it in [151]. The ALS algorithm can be considered as a heuristic that guarantees only local solutions. In [93], we have implemented the ALS algorithm to perform CDPCA in R [124], a widely used open source software for data analysis, very popular in statistics. Recently, in [94], we have developed a new scheme of the ALS algorithm to improve the implementation of the CDPCA model in terms of estimating the parameters.

The main aim of this chapter is to present a new SDP-based approach to clustering and dimensionality reduction. Inspired by recent work [94, 115, 116, 151], we propose a new approach to CDPCA. This approach results in a SDP-based approximation algorithm, called Two-Step-SDP algorithm, that permits to cluster objects and attributes. The Two-Step-SDP algorithm can be considered as an improvement on the ALS algorithm proposed in [151]. The Two-Step-SDP algorithm is implemented in a easy-to-use software application using R and several numerical experiments are carried out to evaluate its efficiency.

6.1.1 Clustering: preliminaries

In what follows, by *objects* we mean entities to be clustered, and *attributes* are characteristics of the objects.

Clustering consists in partitioning a given set of m objects into p nonempty and nonoverlapping clusters C_j, $j = 1, ..., p$, where $2 \leq p < m$ is a given integer. These clusters are constructed in such a way that each object is assigned to a single cluster.

Each object i is characterized by a n-dimensional row vector d_i of n attributes (also called variables or features).

A data matrix is a $(m \times n)$ matrix $\mathbf{D} = [d_{ij}]$, where the m rows correspond to the objects and the n columns correspond to the attributes characterizing these objects.

Given a $(m \times n)$ data matrix $\mathbf{D}$, if we want to assign the m objects into p, $2 \leq p < m$, clusters, the assignments can be stored in a $(m \times p)$ binary matrix $\mathbf{U} = [u_{ij}]$ defined as follows:

$$u_{ij} = \begin{cases} 1, & \text{if object } i \in C_j, \\ \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

This matrix is called *assignment* or *cluster indicator* matrix. By construction, $\mathbf{U}$ is a binary row stochastic matrix, that has only one nonzero element per row and the sum of all entries in each row is equal to 1, *i.e.*,

$$\mathbf{U}e^p = e^m,$$

where $e^k \in \mathbb{R}^k$ is a vector with all entries equal to 1, $k \in \mathbb{N}$. Notice that $\text{rank}(\mathbf{U}) = p$ and that an assignment matrix is not unique, since one can permute columns.

Each cluster can be identified by its cluster centre, called centroid, usually, a non-observable object that is the mean of all the objects assigned to this cluster [73, 151, 160]. Therefore, if the objects are assigned with an assignment matrix $\mathbf{U}$, then the centroid $c_j \in \mathbb{R}^n$ of the cluster C_j, for $j = 1, ..., p$, can be defined as

$$c_j = \frac{1}{\sum_{t=1}^{m} u_{tj}} \sum_{t=1}^{m} u_{tj} d_t. \tag{6.2}$$

The $(p \times n)$ object cluster centroid matrix, whose rows correspond to the clusters presented by their centroids c_j, is denoted here by $\bar{\mathbf{D}}$. This matrix can be written as follows ([151]):

$$\bar{\mathbf{D}} = \left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T\mathbf{D}. \tag{6.3}$$

The similarity measure that we use here is the Euclidean distance between a pair of objects. In the minimum sum-of-squares clustering (MSSC) problem [3, 6, 9, 50, 73, 116, 153, 160] one is interested in minimizing the sum of the distances between the objects in each cluster. The sum of the squared distances between each object and the centroid of the cluster to which it belongs is equal to $\sum\limits_{j=1}^{p}\sum\limits_{i=1}^{m} u_{ij}\|d_i - c_j\|_2^2$, where the cluster centroid c_j is defined in (6.2).

Given a $(m \times n)$ data matrix $\mathbf{D}$, a partition of the set of n attributes into k, $2 \leq k < n$, disjoint subsets S_j, $j = 1, ..., k$, called components, can be considered as a clustering problem over the attributes. The assignment of attributes can be stored in a $(n \times k)$ binary assignment matrix $\mathbf{V} = [v_{ij}]$ where

$$v_{ij} = \begin{cases} 1, \text{ if attribute } i \in S_j, \\ \\ 0, \text{ otherwise.} \end{cases} \tag{6.4}$$

Notice that $\mathbf{V}$ is a binary row stochastic matrix satisfying $\mathbf{V}e^k = e^n$ and $\text{rank}(\mathbf{V}) = k$.

6.2 Integer programming model for clustering and its solution

6.2.1 Integer programming model

Consider the clustering problem introduced in the previous section. The minimum sum-of-squares clustering (MSSC) model can be formulated w.r.t. the variables u_{ij}, $i = 1, ..., m$, and $j = 1, ..., p$, as follows ([3, 6, 73, 81, 115, 116, 160]):

$$\begin{array}{ll} \min\limits_{u_{ij}} & \sum\limits_{j=1}^{p}\sum\limits_{i=1}^{m} u_{ij}\left\| d_i - \frac{\sum\limits_{t=1}^{m} u_{tj}d_t}{\sum\limits_{t=1}^{m} u_{tj}}\right\|_2^2 \\ \text{s.t.} & \sum\limits_{j=1}^{p} u_{ij} = 1, \quad i = 1, ..., m, \\ & \sum\limits_{i=1}^{m} u_{ij} \geq 1, \quad j = 1, ..., p \\ & u_{ij} \in \{0, 1\}. \end{array} \tag{6.5}$$

The first constraint in (6.5) ensures that each object is assigned to a single cluster and the second constraint ensures that each cluster has at least one object assigned. Any feasible solution of problem (6.5) is an assignment matrix $\mathbf{U}$.

The problem (6.5) is a binary integer programming problem and is known to be NP-hard (see [3, 6, 58, 116, 153]).

6.2.2 K-means algorithm

The MSSC problem (6.5) is very difficult to solve due to its variables take only discrete values and the nonlinearity and nonconvexity of its objective function [115, 116]. Many approaches have been proposed to solve it either by exact algorithms, or heuristics (*e.g.*, [3, 6, 9, 81, 115, 116]). By far, the most popular algorithm to solve the MSSC problem (6.5) is the K-means algorithm [50, 58].

In essence, the classical K-means algorithm starts by randomly assigning each object to some cluster. Then, the algorithm performs a re-assignment of the objects to the clusters, based on minimizing the sum of the squared distances between the objects and the cluster centroids.

These steps can be outlined as follows ([58]):

Basic Algorithm 7 K-means

```
1: Choose an initial partition of $p$ clusters (or generate it randomly) and find the corresponding
   centroids,
2: Assign each object $i$, $i = 1, ..., m$, to the closest centroid,
3: Update the centroids using the current assignments.
```

Steps 2 and 3 are repeated until the within cluster sum of squares is no longer reduced. Numerical tests show that the K-means algorithm returns well-separated clusters having a convex-shaped geometry, *i.e.*, spherical or elliptical [50, 58].

It is worthwhile mentioning that the K-means algorithm should be used on scaled data, since it relies on the Euclidean distances. Moreover, the K-means algorithm returns a local optimum and depends on the initial choice of the centroids [9, 58, 73]. To overcome this drawback, it is recommended to consider several random initializations [58]. For example, in [151], a “tandem analysis” (*i.e.*, PCA followed by applying the K-means algorithm using only the first few components) was carried out with 10000 random starts of the K-means algorithm using the first two principal components of a particular data set. The data set consists of 20 objects and 6 attributes, and the aim was to obtain 3 clusters of objects.

There exist extensions and modifications of the K-means algorithm, such as the K-medoids, where each cluster is represented by its *medoid*, which is the most centrally located object in the cluster, or fuzzy algorithms, that allows nonconvex shapes of the clusters (see, *e.g.*, [9, 50, 58, 73]).

6.3 Semidefinite programming-based model for clustering and its properties

This section is devoted to the study of an equivalent formulation for the clustering problem (6.5) that was proposed in [115, 116]. We will first outline some basic definitions.

6.3.1 SDP-based model

Consider an assignment matrix $\mathbf{U}$ defined in (6.1). By construction, the columns u_i, $i = 1, ..., p$, of the matrix $\mathbf{U}$ are linearly independent and orthogonal. Therefore, $\mathbf{U}$ has a zero nullspace and $\operatorname{rank}(\mathbf{U}) = p$. Hence, $\mathbf{U}^T\mathbf{U}$ is a $(p \times p)$ diagonal and invertible matrix, where each diagonal entry is the sum of the elements of the corresponding column of $\mathbf{U}$:

$$\mathbf{U}^T\mathbf{U} \;=\; \operatorname{diag}\left(\sum_{i=1}^{m} u_{i1}, \sum_{i=1}^{m} u_{i2}, ..., \sum_{i=1}^{m} u_{ip}\right) \;=\; \operatorname{diag}\left(\|u_1\|_2^2, \|u_2\|_2^2, ..., \|u_p\|_2^2\right). \tag{6.6}$$

Given an assignment matrix $\mathbf{U}$, the subspace of $\mathbb{R}^m$ spanned by its columns is denoted by $\mathcal{C}(\mathbf{U})$.

Let $\mathbf{M}$ be a $(m \times p)$ matrix. Consider an orthogonal projection of the space $\mathbb{R}^m$ onto the subspace spanned by the columns of $\mathbf{M}$. The following theorem is valid [163].

Theorem 26 *Let $\mathbf{M}$ be a $(m \times p)$ matrix whose p columns are linearly independent. The matrix $\mathbf{P}$ of the orthogonal projection of the space $\mathbb{R}^m$ onto the subspace spanned by the columns of $\mathbf{M}$ has the form $\mathbf{P} = \mathbf{M}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$.*

It is known from linear algebra [1, 98, 163] that an orthogonal projection matrix $\mathbf{P}$ satisfies the following properties:

1. $\mathbf{P}^2 = \mathbf{P}$, ensuring that $\mathbf{P}$ is a projection matrix,

2. $\mathbf{P}$ is symmetric, *i.e.*, $\mathbf{P}^T = \mathbf{P}$,

3. $\operatorname{rank}(\mathbf{P}) = \operatorname{tr}(\mathbf{P})$,

4. $\operatorname{rank}(\mathbf{P}) = p$,

5. the eigenvalues of $\mathbf{P}$ are only 0 or 1.

From the last property, it follows that any orthogonal projection matrix is positive semidefinite.

Let us consider a $(m \times m)$ matrix $\mathbf{Z} = [z_{ij}]$ in the following form ([115, 116]):

$$\mathbf{Z} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T, \tag{6.7}$$

where $\mathbf{U}$ is defined in (6.1). It is easy to verify that $\mathbf{Z}$ has nonnegative elements. Evidently, $\mathbf{Z}$ is the orthogonal projection matrix onto the space $\mathcal{C}(\mathbf{U})$ spanned by the columns of the matrix $\mathbf{U}$. Hence, $\mathbf{Z}$ satisfies the properties (1)-(5). Moreover, since $\mathcal{C}(\mathbf{U})$ is a p-dimensional subspace of $\mathbb{R}^m$, it follows that $\operatorname{rank}(\mathbf{Z}) = \operatorname{rank}(\mathbf{U}) = p$.

In spite of the fact that matrix $\mathbf{U}$ is not unique (*e.g.*, one can permute the columns of $\mathbf{U}$ getting another assignment matrix), the matrix $\mathbf{Z}$ defined in (6.7) is.

From the definition of the matrix $\mathbf{Z}$ and since each object is assigned to a single cluster, each row of the matrix $\mathbf{Z}$ has sum equal to 1, *i.e.*, it should satisfy $\mathbf{Z}e^m = e^m$.

Example 21 *Consider the assignment matrix*

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

specifying the assignment of $m = 6$ *objects into* $p = 3$ *clusters. By* (6.7)*, the matrix* $\mathbf{Z}$ *has the form*

$$\mathbf{Z} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}.$$

Evidently, the matrix $\mathbf{Z}$ *is symmetric,* $\mathbf{Z}^2 = \mathbf{Z}$*,* $\operatorname{rank}(\mathbf{Z}) = \operatorname{tr}(\mathbf{Z}) = 3$*, and the sum of the elements in each row is equal to* 1*.*

Notice that both matrices $\mathbf{U}$ *and* $\mathbf{Z}$ *represent the same partition: the objects* 1 *and* 2 *are assigned to one cluster,* 3 *and* 4 *belong to another cluster, and the objects* 5 *and* 6 *are assigned to a third cluster.*

In [115, 116], it is shown that the objective function in the MSSC problem (6.5) is equal to the function $\operatorname{tr}\left(\mathbf{D}\mathbf{D}^T(\mathbf{I_m} - \mathbf{Z})\right)$ and the following optimization problem is formulated:

$$\begin{array}{lll} \min\limits_{\mathbf{Z}} & \operatorname{tr}\left(\mathbf{D}\mathbf{D}^T(\mathbf{I_m} - \mathbf{Z})\right) & \\ \text{s.t.} & \mathbf{Z}^T = \mathbf{Z}, & \\ & \mathbf{Z}^2 = \mathbf{Z}, & \\ & \operatorname{tr}(\mathbf{Z}) = p, & \\ & \mathbf{Z}e^m = e^m, & \\ & z_{ij} \geq 0, & \forall i, j = 1, 2, ..., m. \end{array} \tag{6.8}$$

The first two constraints in (6.8) imply that $\mathbf{Z}$ is an orthogonal projection matrix and the next constraint ensures that there are exactly p clusters. The last equality constraint in problem (6.8) means that each object is assigned to a single cluster, *i.e.*, each row of the matrix $\mathbf{Z}$ has sum equal to 1. The inequality constraint ensures that $\mathbf{Z}$ has nonnegative elements.

In [116], it is proved that problem (6.8) is equivalent to (6.5). In [115, 116], the model (6.8) is called 0-1 SDP model, since the first two constraints imply that $\mathbf{Z}$ is positive semidefinite, with eigenvalues 0 or 1. Nevertheless, (6.8) does not have a standard SDP form, and thus, we will call it the *SDP-based model* of the clustering problem (6.5).

The problem (6.8) can be rewritten as

$$\begin{array}{ll} \max\limits_{\mathbf{Z}} & \operatorname{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}) \\ \text{s.t.} & \mathbf{Z}^T = \mathbf{Z}, \\ & \mathbf{Z}^2 = \mathbf{Z}, \\ & \operatorname{tr}(\mathbf{Z}) = p, \\ & \mathbf{Z}e^m = e^m, \\ & z_{ij} \geq 0, \qquad \forall i,j = 1,2,...,m. \end{array} \tag{6.9}$$

Both problems, (6.8) and (6.9), are difficult to solve due to the nonlinearity of the second constraints. Moreover, since these problems do not have the standard SDP form, current SDP methods can not be applied to solve them. Therefore, other strategies should be applied.

For example, in [81], it is showed that the problem (6.9) is equivalent to

$$\begin{array}{ll} \max\limits_{\mathbf{Z}} & \operatorname{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}) \\ \text{s.t.} & \operatorname{tr}(\mathbf{Z}) = p, \\ & \mathbf{Z}e^m = e^m, \\ & \operatorname{rank}(\mathbf{Z}) = p, \\ & \mathbf{Z} \succeq 0, \\ & z_{ij} \geq 0, \qquad \forall i,j = 1,2,...,m. \end{array} \tag{6.10}$$

and this problem is solved using nonconvex algorithms.

Notice that to obtain this problem, the first two constraints in (6.9) were replaced by the nonconvex rank constraint and the positive semidefiniteness condition on the matrix $\mathbf{Z}$. Due to the rank constraint, (6.10) is a low-rank SDP problem that can not be solved using standard SDP methods, and thus, special methods are needed. In [81], the authors developed a general nonconvex optimization algorithm based on an algorithm that uses low-rank factorizations to optimize full-rank SDP problems proposed by Burer and Monteiro (see all the details in [81]).

6.3.2 General properties of the SDP-based model

The nonlinear SDP-based model (6.9) is very difficult to solve [115, 116]. The problem can possibly be solved using commercial and open-source solvers, such as the LGO Solver Package for Global and Local Nonlinear Optimization [118], PENNON and PENLAB [35], but it is required to rewrite the problem either explicitly, or using polynomial matrix inequalities. This augments crucially the number of variables and constraints.

According to [116], there are two groups of algorithms that can be used for solving the nonlinear model (6.9) in the implicit form: feasible iterative algorithms and approximation algorithms based on LP or SDP relaxations. The feasible iterative algorithms use some kind of heuristics, such as the classical K-means algorithm. The second group of algorithms is the most popular in the literature, and is based on constructing and solving LP or SDP relaxations for the nonlinear SDP-based model, and then use some rounding procedure to obtain a feasible solution to (6.9).

Given a clustering problem, it is always possible to get a randomly chosen assignment matrix $\mathbf{U}$ and a matrix $\mathbf{Z}$ using (6.7). Then the SDP-based problem (6.9) is always feasible.

Example 22 *Suppose we are interested in clustering a set of* 6 *objects into* 4 *clusters, i.e.,* $m = 6$ *and* $p = 4$*. Consider the following random assignment matrix*

$$\mathbf{U}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Therefore, a feasible solution to problem (6.9) *may be given by* (6.7)*, yielding the matrix*

$$\mathbf{Z}_1 = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}.$$

It is easy to see that matrix $\mathbf{Z}_1$ *satisfies all the constraints of the problem* (6.9).

6.3.3 Study of regularity of the SDP-based model

In this section, we study the regularity in terms of the Slater condition of the SDP-based model (6.9). Let us consider the analogous definition of the Slater condition for the nonlinear SDP-based model.

Definition 19 *The nonlinear SDP-based problem* (6.9) *satisfies the Slater condition if there exists* $\bar{\mathbf{Z}} \in \mathbb{R}^{m\times m}$ *with nonnegative entries, such that* $\bar{\mathbf{Z}} \succ 0$ *and the equality constraints in* (6.9) *are satisfied.*

The following theorem states that problem (6.9) is nonregular.

Theorem 27 *The SDP-based problem* (6.9) *does not satisfy the Slater condition.*

Proof. Let a $(m \times m)$ matrix $\mathbf{Z}$ be a feasible solution of the problem (6.9).

Since $\mathbf{Z}$ can be considered as an orthogonal projection matrix, then it satisfies the condition $\text{rank}(\mathbf{Z}) = \text{tr}(\mathbf{Z}) = p$, with $2 \leq p < m$, and its eigenvalues are either 0 or 1, which implies $\mathbf{Z} \succeq 0$.

Hence, $\mathbf{Z}$ has exactly p eigenvalues equal to 1 and $m-p$ zero eigenvalues. Since $p < m$, then the matrix $\mathbf{Z}$ can not be positive definite. Notice that $\mathbf{Z} \succ 0$ if and only if all eigenvalues are positive, and this only happens for $p = m$. Therefore, the SDP-based problem (6.9) does not satisfy the Slater condition. ■

This result about the nonregularity of the SDP-based problem (6.9) shows that special attention should be given to the numerical methods for solving it. Since the model is nonregular, then it is important to have specific efficient methods to handle such problems.

6.3.4 Recovering the assignments

Suppose that the problem (6.9) is solved and matrix $\mathbf{Z}$ is its solution. Now, let us discuss how to recover a cluster assignment matrix $\mathbf{U}$.

It can be pointed out that $\mathbf{Z}$ can be considered as a block similarity matrix, where each block reflects some kind of similarity between the corresponding objects. This fact can be used to obtain the desired p clusters of objects.

The recovering of an assignment matrix $\mathbf{U}$ can be done by considering a factorization of $\mathbf{Z}$ in a particular form, or simply by reducing the matrix $\mathbf{Z}$ to its row echelon form. In what follows, we will describe both procedures.

The first procedure is based on the following lemma.

Lemma 5 *Let $\mathbf{Z}$ be a $(m \times m)$ matrix defined in (6.7). Then there exists a $(m \times p)$ matrix $\mathbf{Q}$, satisfying $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_p$, such that $\mathbf{Z}$ can be decomposed in the form $\mathbf{Z} = \mathbf{Q}\mathbf{Q}^T$ and the columns of $\mathbf{Q}$ form an orthonormal basis for the range space of $\mathbf{Z}$.*

Proof. Let $\mathbf{Z}$ be the matrix of rank p defined in (6.7). Consider a basis of the column space of $\mathbf{U}$, say $v_1, v_2, ..., v_p$ (not necessarily the basis formed by the columns of $\mathbf{U}$). Consider also an orthonormal basis given by the vectors $q_1, q_2, ..., q_p$, where $q_i = \frac{v_i}{\|v_i\|_2}$, $i = 1, ..., p$, and the matrix $\mathbf{Q}$, whose columns are these vectors. By construction, $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_p$.

The matrix $\mathbf{U}$ can be written as

$$\mathbf{U} = \mathbf{Q}\mathbf{R}, \tag{6.11}$$

where $\mathbf{R}$ is a full rank square $(p \times p)$ matrix given by

$$\mathbf{R} = \text{diag}\left(\|v_1\|_2, \|v_2\|_2, ..., \|v_p\|_2\right).$$

Therefore, taking into account (6.1) and (6.11), we get

$$\begin{aligned}
\mathbf{Z} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T &= (\mathbf{Q}\mathbf{R})\left[(\mathbf{Q}\mathbf{R})^T(\mathbf{Q}\mathbf{R})\right]^{-1}(\mathbf{Q}\mathbf{R})^T \\
= \mathbf{Q}\mathbf{R}\left(\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\right)^{-1}\mathbf{R}^T\mathbf{Q}^T &= \mathbf{Q}\mathbf{R}\left(\mathbf{R}^T\mathbf{I}_p\mathbf{R}\right)^{-1}\mathbf{R}^T\mathbf{Q}^T \\
&= \mathbf{Q}\mathbf{R}\left(\mathbf{R}\right)^{-1}\left(\mathbf{R}^T\right)^{-1}\mathbf{R}^T\mathbf{Q}^T = \mathbf{Q}\mathbf{Q}^T.
\end{aligned}$$

Since $\operatorname{rank}(\mathbf{Q}) = p = \operatorname{rank}(\mathbf{Z})$, then the columns of $\mathbf{Q}$ form an orthonormal basis for the range space of matrix $\mathbf{Z}$. This completes the proof of Lemma 5. ■

According to [123], the matrix $\mathbf{Q}$ from Lemma 5 can be considered as a weighted cluster indicator matrix, giving information of the allocation of the m objects into p clusters. Therefore, it is related to the assignment matrix $\mathbf{U}$.

The matrix $\mathbf{Q}$ is not unique, nevertheless it has a particular form. For example, its elements can be defined as

$$q_{ij} = \begin{cases} \frac{1}{\sqrt{|C_j|}}, & \text{if } i \in C_j, \\ \\ 0, & \text{otherwise,} \end{cases} \tag{6.12}$$

where $|C_j|$ is the number of objects in cluster C_j.

According to [81], the structure of the matrix $\mathbf{Q}$ is given by (6.12), but if $\mathbf{Q}$ is a column orthonormal matrix, whose columns are q_i, $i = 1, ..., p$, then the matrix $\bar{\mathbf{Q}}$, whose columns are $\bar{q}_i = \alpha_i q_i$, is also a column orthonormal matrix if and only if $\alpha_i = \pm 1$. This is the reason why we say that $\mathbf{Q}$ is not unique, but has a special form.

Notice that the nonzero entries in $\mathbf{Q}$ correspond to the nonzero entries in $\mathbf{U}$. In order to get an assignment matrix $\mathbf{U}$, we can replace each nonzero element in $\mathbf{Q}$ by 1.

The matrix $\mathbf{Q}$ satisfying the conditions of Lemma 5 can be obtained via Singular Value Decomposition (SVD) of the matrix $\mathbf{Z}$ ([98, 115, 133]):

$$\mathbf{Z} = \mathbf{C}\mathbf{\Sigma}\mathbf{N}^T,$$

where $\mathbf{C}$ and $\mathbf{N}$ are orthogonal matrices and $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, ..., \sigma_m)$ contains the singular values of $\mathbf{Z}$ in decreasing order. The singular values of $\mathbf{Z}$ are the square roots of the eigenvalues of the matrix $\mathbf{Z}^T\mathbf{Z}$. Since $\operatorname{rank}(\mathbf{Z}) = p$, $\mathbf{Z}$ has p nonzero singular values.

The columns of the matrix $\mathbf{C}$ corresponding to the nonzero singular values of $\mathbf{Z}$ form an orthonormal basis of the range space of $\mathbf{Z}$. Therefore, it follows from Lemma 5 that these columns form the matrix $\mathbf{Q}$. The columns of the matrix $\mathbf{N}$ corresponding to the zero singular values of $\mathbf{Z}$ form an orthonormal basis of the null space of Z.

The SVD is a standard decomposition in numerical analysis, and many algorithms exist for its computation [50].

We can outline the following procedure to recover an assignment matrix $\mathbf{U}$.

Basic Algorithm 8 Recovering the assignments via SVD

```
input: $\mathbf{Z}$, solution of the problem (6.9).
output: $\mathbf{U}$, an assignment matrix.
1: Apply SVD to the matrix $\mathbf{Z}$ and obtain $\mathbf{Q}$,
2: Replace each nonzero element of $\mathbf{Q}$ by 1 to obtain the assignment matrix, $\mathbf{U}$.
```

Notice that the matrix $|\mathbf{Q}|$, whose elements are the absolute values of the elements of $\mathbf{Q}$, has the structure defined in (6.12).

Example 23 *Considering the matrix* $\mathbf{Z}$ *from Example 21 and using the above procedure, let us obtain an assignment matrix from* $\mathbf{Z}$. *Applying the SVD to* $\mathbf{Z}$, *we get* $\mathbf{\Sigma} = \operatorname{diag}(1,1,1,0,0,0)$ *and*

$$\mathbf{C} = \mathbf{N} = \begin{bmatrix} -0.7071 & 0 & 0 & -0.7071 & 0 & 0 \\ -0.7071 & 0 & 0 & 0.7071 & 0 & 0 \\ 0 & 0 & -0.7071 & 0 & -0.7071 & 0 \\ 0 & 0 & -0.7071 & 0 & 0.7071 & 0 \\ 0 & -0.7071 & 0 & 0 & 0 & -0.7071 \\ 0 & -0.7071 & 0 & 0 & 0 & 0.7071 \end{bmatrix}.$$

Then

$$\mathbf{Q} = \begin{bmatrix} -0.7071 & 0 & 0 \\ -0.7071 & 0 & 0 \\ 0 & 0 & -0.7071 \\ 0 & 0 & -0.7071 \\ 0 & -0.7071 & 0 \\ 0 & -0.7071 & 0 \end{bmatrix}.$$

This matrix already gives us an idea of the assignment. We can see that the elements of the matrix

$$|\mathbf{Q}| = \begin{bmatrix} 0.7071 & 0 & 0 \\ 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 \\ 0 & 0 & 0.7071 \\ 0 & 0.7071 & 0 \\ 0 & 0.7071 & 0 \end{bmatrix}$$

have the special form (6.12).

Replacing each nonzero element of the matrix $\mathbf{Q}$ *by 1 yields the assignment matrix*

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Therefore, we get the following assignments: the first two objects are assigned to one cluster, objects 5 *and* 6 *are assigned to a second one, and a third cluster contains the objects* 3 *and* 4.

Although the common procedure of recovering an assignment matrix is using SVD, the same result can be obtained by applying the Gauss elimination method to the matrix $\mathbf{Z}$. Recall that the Gauss elimination method of a matrix is a row reduction method that uses a sequence of elementary row operations and returns a matrix with all the linearly independent rows on top (row echelon form of **Z**). The vectors on the top form the basis of the column space of **Z** that can be used to recover **U**. Let **L** be the $(m \times p)$ matrix whose columns are these vectors. This matrix can be considered as a weighted cluster indicator matrix.

An assignment matrix **U** can be recovered by replacing each nonzero element in **L** by 1.

We can then describe the recovering procedure as follows.

Basic Algorithm 9 Recovering the assignments via row echelon form

```
input: Z, solution of the problem (6.9).
output: U, an assignment matrix.
1: Get a row echelon form of the matrix Z and construct the matrix L whose columns
   are the linearly independent rows of Z,
2: Replace each nonzero element of L by 1 to obtain the assignment matrix, U.
```

Example 24 *Considering the matrix* **Z** *from Example 21, in the Step 2 of the above procedure, we can see that its row echelon form is*

$$\begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The weighted cluster indicator matrix **L** *as the form:*

$$\mathbf{L} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 0.5 \end{bmatrix}.$$

Replacing each nonzero element of **L** *by 1, we get the assignment matrix*

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

It is easy to see that this matrix represents the same assignments obtained in Example 23, and after a permutation of the second and third columns, the assignment matrices coincide.

6.4 Approximation algorithm for solving the SDP-based model

Approximation algorithms are widely used to solve hard optimization problems (see, *e.g.*, [3, 42, 43, 116]). The general idea of an approximation algorithm is to first solve a relaxation of the hard problem, and then apply some rounding procedure to the obtained solution to finally find a feasible solution of the original problem. It is known that SDP relaxations provide good approximations [8, 83]. The first attempt to use a SDP-based approximation algorithm dates back to 1995, when Goemans and Williamson [43] suggested the randomized approximation algorithm for the max-cut problem, a well known NP-hard problem [83]. Since then, SDP has been successfully applied in the development of approximation algorithms for several classes of hard combinatorial optimization problems. It is also known that standard interior point SDP methods, such as the primal-dual interior point methods [8, 42, 149], are not so efficient for large scale problems and thus, different strategies have been developed [35, 81, 115].

In [115], an approximation algorithm that uses a linear SDP relaxation is proposed to obtain an approximate solution of (6.9).

6.4.1 Linear SDP relaxations of the SDP-based model

In [115, 116], some relaxations of the SDP-based problem (6.9) are suggested, including linear SDP relaxations. One of such relaxations was obtained by replacing the first two constraints in (6.9) by a condition of positive semidefiniteness of matrix $\mathbf{Z}$, yielding the following linear SDP problem

$$\begin{array}{ll} \max\limits_{\mathbf{Z}} & \operatorname{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}) \\ \text{s.t.} & \operatorname{tr}(\mathbf{Z}) = p, \\ & \mathbf{Z}e^m = e^m, \\ & \mathbf{Z} \succeq 0, \\ & z_{ij} \geq 0, \qquad \forall i,j = 1,2,...,m. \end{array} \tag{6.13}$$

This is a convex problem, since its objective function is linear, and its feasible set is the intersection of the convex cone of the positive semidefinite matrices with the convex set described by a finite number of linear equalities (notice that the last constraint can be incorporated in the positive semidefinite constraint).

Theoretically, the SDP problem (6.13) can be solved using standard SDP methods (see [8, 81, 115, 116, 149]), but, in practice, such methods are not so efficient for large-scale problems [42], because of the *curse of dimensionality*, which encompasses storage issues. In what follows, we will discuss this phenomenon.

From the computational viewpoint, the constraint $\mathbf{Z}e^m = e^m$ is equivalent to $\operatorname{tr}(\mathbf{B}_i\mathbf{Z}) =$

1, where for each $i = 1, ..., m$, $\mathbf{B}_i = [b_{rq}]$ is a $m \times m$ matrix defined as

$$b_{rq} = \begin{cases} 1, & \text{if } r = q = i, \\ \frac{1}{2}, & \text{if } r = i \text{ and } q \neq i, \\ \frac{1}{2}, & \text{if } q = i \text{ and } r \neq i, \\ 0, & \text{otherwise} \end{cases} \tag{6.14}$$

and the constraint on the nonnegativity of the elements of $\mathbf{Z}$ can be incorporated in the positive semidefinite constraint. Therefore, the problem (6.13) can be written in terms of a $\left(\frac{m(m+1)}{2} \times \frac{m(m+1)}{2}\right)$ matrix variable $\bar{\mathbf{Z}}$ in a block diagonal form as follows:

$$\begin{array}{lll} \max\limits_{\bar{\mathbf{Z}}} & \operatorname{tr}\left(\left[\begin{array}{c:c} \mathbf{D}\mathbf{D}^T & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} \end{array}\right] \bar{\mathbf{Z}}\right) & \\ \text{s.t.} & \operatorname{tr}\left(\left[\begin{array}{c:c} \mathbf{I}_m & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} \end{array}\right] \bar{\mathbf{Z}}\right) = p, & \\ & \operatorname{tr}\left(\left[\begin{array}{c:c} \mathbf{B}_i & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} \end{array}\right] \bar{\mathbf{Z}}\right) = 1, & i = 1, 2, ..., m, \\ & \bar{\mathbf{Z}} = \left[\begin{array}{c:c} \mathbf{Z}_{m \times m} & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{D}_k \end{array}\right] \succeq 0, & \end{array} \tag{6.15}$$

where $k = \frac{m(m-1)}{2}$, $\mathbf{D}_k = \operatorname{diag}(z_{ij})$, for all $i = 1, ..., m-1$ and $j > i$, and $\mathbf{B}_i$, $i = 1, ..., m$, are defined in (6.14).

Recall that typically, in a clustering problem the dimension of the data can be extremely large, which results in huge matrices in (6.15). Therefore, the numerical solution of such problems can be computationally expensive, or even impossible. It is known that standard interior point SDP methods suffer from lack of performance for large-scale problems.

Another relaxation of the problem (6.9) can be obtained by replacing the last two constraints in (6.13) by the matrix inequality $\mathbf{I}_m \succeq \mathbf{Z} \succeq 0$, yielding the following SDP problem [115, 116]:

$$\begin{array}{ll} \max\limits_{\mathbf{Z}} & \operatorname{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}) \\ \text{s.t.} & \operatorname{tr}(\mathbf{Z}) = p, \\ & \mathbf{Z}e^m = e^m, \\ & \mathbf{I}_m \succeq \mathbf{Z} \succeq 0. \end{array} \tag{6.16}$$

The linear SDP problem (6.16) is convex, since its objective function is linear, and its feasible set is the intersection of a compact convex subset of the linear space of the real symmetric matrices [83, 110] with a convex set described by a finite number of linear equalities.

Observation 1 *The solutions of the relaxed problems* (6.13) *and* (6.16) *may not coincide with the solution of the original problem* (6.9).

Observation 2 *The relaxed problems* (6.13) *and* (6.16) *can also be considered as relaxations of* (6.10), *since problems* (6.9) *and* (6.10) *are equivalent.*

In the following sections, we study the regularity of the presented relaxations and then, we focus on the SDP problem (6.16) and describe a procedure to solve it and to extract a feasible solution to the SDP-based problem (6.9).

6.4.2 Study of regularity of the SDP relaxations

It was shown above that the SDP relaxations (6.13) and (6.16) are convex. It was also shown that any matrix $\mathbf{Z}$ in the form (6.7) is a feasible solution of (6.9), and thus, it is feasible for the relaxed problems (6.13) and (6.16). Therefore, the SDP relaxations (6.13) and (6.16) are always feasible.

Theoretically, the SDP problems (6.13) and (6.16) can be solved in polynomial time using standard SDP methods (see [8, 81, 115, 116, 149]), such as interior point methods, which perform very well on small to medium scale SDP problems, but are not so efficient for large scale problems. All standard SDP methods rely on assumptions of regularity [4, 8, 23, 37, 54]. Therefore, it is very important to know in advance if a given SDP problem is regular.

The following theorem states that the relaxed problem (6.13) is regular.

Theorem 28 *The Slater condition holds for the SDP problem* (6.13).

Proof. Let us show that there exists a feasible solution of the problem (6.13) that is a positive definite matrix.

For this purpose, consider a positive $(m \times m)$ symmetric matrix $\bar{\mathbf{Z}}$ defined as follows:

$$\bar{z}_{ij} = \begin{cases} \frac{p}{m}, \ i = j, \\ \\ \frac{m-p}{m(m-1)}, \ i \neq j, \end{cases} \tag{6.17}$$

where $2 \leq p < m$.

It is easy to see that the matrix $\bar{\mathbf{Z}}$ is a feasible solution of (6.13).

The matrix $\bar{\mathbf{Z}}$ has m positive eigenvalues: one equal to 1, and the remaining equal to $\frac{p-1}{m-1}$. Let us prove this result.

Each eigenvalue λ of $\bar{\mathbf{Z}}$ is a solution of the equation $\det(\bar{\mathbf{Z}} - \lambda \mathbf{I}_m) = 0$, *i.e.*,

$$\det \left(\begin{bmatrix} \frac{p}{m} - \lambda & \frac{m-p}{m(m-1)} & \frac{m-p}{m(m-1)} & \cdots & \frac{m-p}{m(m-1)} \\ \frac{m-p}{m(m-1)} & \frac{p}{m} - \lambda & \frac{m-p}{m(m-1)} & \cdots & \frac{m-p}{m(m-1)} \\ \frac{m-p}{m(m-1)} & \frac{m-p}{m(m-1)} & \frac{p}{m} - \lambda & \cdots & \frac{m-p}{m(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{m-p}{m(m-1)} & \frac{m-p}{m(m-1)} & \frac{m-p}{m(m-1)} & \cdots & \frac{p}{m} - \lambda \end{bmatrix} \right) = 0.$$

Let us find the expression for $\det(\bar{\mathbf{Z}} - \lambda \mathbf{I}_m)$. First, subtract the last row from the other rows of the matrix and then, sum the last column of the obtained matrix with all the previous columns. Taking into account the properties of the determinants, we get:

$$\det(\bar{\mathbf{Z}} - \lambda \mathbf{I}_m) = \det\left(\begin{bmatrix} \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & 0 & 0 & \cdots & \frac{m-p}{m(m-1)} - \left(\frac{p}{m} - \lambda\right) \\ 0 & \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & 0 & \cdots & \frac{m-p}{m(m-1)} - \left(\frac{p}{m} - \lambda\right) \\ 0 & 0 & \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & \cdots & \frac{m-p}{m(m-1)} - \left(\frac{p}{m} - \lambda\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{m-p}{m(m-1)} & \frac{m-p}{m(m-1)} & \frac{m-p}{m(m-1)} & \cdots & \frac{p}{m} - \lambda \end{bmatrix}\right)$$

$$= \frac{m-p}{m(m-1)} \det\left(\begin{bmatrix} \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & 0 & 0 & \cdots & \frac{m-p}{m(m-1)} - \left(\frac{p}{m} - \lambda\right) \\ 0 & \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & 0 & \cdots & \frac{m-p}{m(m-1)} - \left(\frac{p}{m} - \lambda\right) \\ 0 & 0 & \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & \cdots & \frac{m-p}{m(m-1)} - \left(\frac{p}{m} - \lambda\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & \left(\frac{p}{m} - \lambda\right) \frac{m(m-1)}{m-p} \end{bmatrix}\right)$$

$$= \frac{m-p}{m(m-1)} \det\left(\begin{bmatrix} \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & 0 & 0 & \cdots & 0 \\ 0 & \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & 0 & \cdots & 0 \\ 0 & 0 & \frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & \left(\frac{p}{m} - \lambda\right) \frac{m(m-1)}{m-p} + m - 1 \end{bmatrix}\right)$$

and since $\frac{p}{m} - \lambda - \frac{m-p}{m(m-1)} = \frac{p-1}{m-1} - \lambda$, then the last determinant can be rewritten in the form

$$\frac{m-p}{m(m-1)}\det\left(\begin{bmatrix} \frac{p-1}{m-1}-\lambda & 0 & 0 & \cdots & 0 \\ 0 & \frac{p-1}{m-1}-\lambda & 0 & \cdots & 0 \\ 0 & 0 & \frac{p-1}{m-1}-\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & \left(\frac{p}{m}-\lambda\right)\frac{m(m-1)}{m-p}+m-1 \end{bmatrix}\right),$$

which is equal to

$$\frac{m-p}{m(m-1)}\left(\frac{p-1}{m-1}-\lambda\right)^{m-1}\left(\left(\frac{p}{m}-\lambda\right)\frac{m(m-1)}{m-p}+m-1\right).$$

Hence, the eigenvalues of $\bar{\mathbf{Z}}$ are the solutions of the equation

$$\frac{m-p}{m(m-1)}\left(\frac{p-1}{m-1}-\lambda\right)^{m-1}\left(\left(\frac{p}{m}-\lambda\right)\frac{m(m-1)}{m-p}+m-1\right)=0.$$

Since $m(m-1) \neq 0$ and $m-p \neq 0$, we get the equivalent equation

$$\begin{aligned} &\left(\frac{p-1}{m-1}-\lambda\right)^{m-1}\left(\left(\frac{p}{m}-\lambda\right)\frac{m(m-1)}{m-p}+m-1\right)=0 \\ \Leftrightarrow &\left(\frac{p-1}{m-1}-\lambda\right)^{m-1}=0 \quad \vee \quad (p-\lambda m)(m-1)+(m-1)(m-p)=0 \\ \Leftrightarrow &\ \lambda=\frac{p-1}{m-1} \quad \vee \quad \lambda=1. \end{aligned}$$

Hence, the eigenvalues of the matrix $\bar{\mathbf{Z}}$ are $\lambda = \frac{p-1}{m-1}$ with algebraic multiplicity equal to $m-1$, and $\lambda = 1$, with algebraic multiplicity 1. Since $p < m$, then the smallest eigenvalue of $\bar{\mathbf{Z}}$ is $0 < \frac{p-1}{m-1} < 1$ and the largest eigenvalue is 1.

Therefore, we have constructed a feasible matrix $\bar{\mathbf{Z}}$ such that $\bar{\mathbf{Z}} \succ 0$ and we can conclude that the SDP problem (6.13) satisfies the Slater condition. ∎

Example 25 *Consider the problem* (6.13) *for* $m = 5$ *and* $p = 2$. *It is easy to verify that the* (5×5) *matrix*

$$\bar{\mathbf{Z}} = \begin{bmatrix} 0.4 & 0.15 & 0.15 & 0.15 & 0.15 \\ 0.15 & 0.4 & 0.15 & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.4 & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.15 & 0.4 & 0.15 \\ 0.15 & 0.15 & 0.15 & 0.15 & 0.4 \end{bmatrix}$$

is positive definite and satisfies the constraints of the problem (6.13). *Indeed,* $\mathrm{tr}(\bar{\mathbf{Z}}) = 2$ *and* $\bar{\mathbf{Z}}$ *is row stochastic, with 5 positive eigenvalues: 1.0, 0.25, 0.25, 0.25, 0.25. Therefore, we can conclude that the interior of the feasible set of problem* (6.13) *is nonempty, i.e., the problem satisfies the Slater condition.*

In the following example, we use the SDPreg procedure 5 introduced in Section 4.1 to test the regularity of the SDP problem (6.13).

Example 26 *Let us consider the feasible SDP problem* (6.13) *for $m = 4$ and $p = 2$.*

The problem (6.13) *is written in the trace form and, to check the regularity using the MATLAB routine* **`SDPreg`**, *we have to write its constraints in the LMI form. For that purpose, we use the reformulation* (6.15) *and consider matrices of order* 10, *where the variable is a block diagonal matrix. In this case, the constraints are*

$$\begin{aligned} z_{11}+z_{22}+z_{33}+z_{44} &= 2\\ z_{11}+z_{12}+z_{13}+z_{14} &= 1\\ z_{12}+z_{22}+z_{23}+z_{24} &= 1\\ z_{13}+z_{23}+z_{33}+z_{34} &= 1\\ z_{14}+z_{24}+z_{34}+z_{44} &= 1\\ \bar{\mathbf{Z}} = \left[\begin{array}{c:c} \mathbf{Z}_{4\times4} & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{D}_6 \end{array}\right] &\succeq 0, \end{aligned}$$

where $\mathbf{Z}_{4\times4} = \begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14}\\ z_{12} & z_{22} & z_{23} & z_{24}\\ z_{13} & z_{23} & z_{33} & z_{34}\\ z_{14} & z_{24} & z_{34} & z_{44} \end{bmatrix}$ *and* $\mathbf{D}_6 = \operatorname{diag}(z_{12}, z_{13}, z_{14}, z_{23}, z_{24}, z_{34})$, *which in turn can be written in the LMI form as*

$$\mathbf{A}_1 z_{22} + \mathbf{A}_2 z_{33} + \mathbf{A}_3 z_{44} + \mathbf{A}_4 z_{13} + \mathbf{A}_5 z_{14} + \mathbf{A}_0 \succeq 0,$$

where

$$\mathbf{A}_0 = \left[\begin{array}{cccc:c} 2 & -1 & 0 & 0 & \\ -1 & 0 & 1 & 1 & \mathbf{0}\\ 0 & 1 & 0 & 0 & \\ 0 & 1 & 0 & 0 & \\ \hdashline & \mathbf{0} & & & \operatorname{diag}(-1,0,0,1,1,0) \end{array}\right]$$

$$\mathbf{A}_1 = \left[\begin{array}{cccc:c} -1 & 1 & 0 & 0 & \\ 1 & 1 & -1 & -1 & \mathbf{0}\\ 0 & -1 & 0 & 1 & \\ 0 & -1 & 1 & 0 & \\ \hdashline & \mathbf{0} & & & \operatorname{diag}(1,0,0,-1,-1,1) \end{array}\right]$$

$$\mathbf{A}_2 = \left[\begin{array}{cccc:c} -1 & 1 & 0 & 0 & \\ 1 & 0 & -1 & 0 & \mathbf{0}\\ 0 & -1 & 1 & 0 & \\ 0 & 0 & 0 & 0 & \\ \hdashline & \mathbf{0} & & & \operatorname{diag}(1,0,0,-1,1,0) \end{array}\right]$$

$$\mathbf{A}_3 = \left[\begin{array}{cccc:c} -1 & 1 & 0 & 0 & \\ 1 & 0 & 0 & -1 & \mathbf{0} \\ 0 & 0 & 0 & 0 & \\ 0 & -1 & 0 & 1 & \\ \hdashline & \mathbf{0} & & & \operatorname{diag}(1,0,0,0,-1,0) \end{array}\right]$$

$$\mathbf{A}_4 = \left[\begin{array}{cccc:c} 0 & -1 & 1 & 0 & \\ -1 & 0 & 0 & 1 & \mathbf{0} \\ 1 & 0 & 0 & -1 & \\ 0 & 1 & -1 & 0 & \\ \hdashline & \mathbf{0} & & & \operatorname{diag}(-1,1,0,0,1,-1) \end{array}\right]$$

$$\mathbf{A}_5 = \left[\begin{array}{cccc:c} 0 & -1 & 0 & 1 & \\ -1 & 0 & 1 & 0 & \mathbf{0} \\ 0 & 1 & 0 & -1 & \\ 1 & 0 & -1 & 0 & \\ \hdashline & \mathbf{0} & & & \operatorname{diag}(-1,0,1,1,0,-1) \end{array}\right].$$

The SDPreg procedure with the tolerance ***SCQ*** *equal to* 10^{-4} *and the remaining tolerances equal to* 10^{-8}*, reported that the problem satisfies the Slater condition, and thus, is regular.*

Considering now the relaxation model (6.16), we can also show that it is regular.

Theorem 29 *The Slater condition holds for the SDP problem* (6.16)*.*

Proof. Let us show that there exists a feasible solution of the problem (6.16) that is a positive definite matrix.

For that purpose, consider again a positive $(m \times m)$ symmetric matrix $\bar{\mathbf{Z}}$ defined as in (6.17). The matrix $\bar{\mathbf{Z}}$ is positive definite with entries lying in $]0, 1[$, and its largest entry is in the diagonal. It is easy to see that $\mathbf{I}_m - \bar{\mathbf{Z}} \succeq 0$. Indeed, the matrix $\mathbf{I}_m - \bar{\mathbf{Z}}$ has m nonnegative eigenvalues: $m-1$ equal to $1 - \frac{p-1}{m-1}$ and one equal to zero.

Therefore, there exists a feasible matrix $\bar{\mathbf{Z}} \succ 0$ and we can conclude that the SDP problem (6.16) satisfies the Slater condition. ■

Since the SDP relaxations (6.13) and (6.16) satisfy the Slater condition, then standard SDP solvers can be used to solve them, at least for small to medium scale problems. For large problems more efficient strategies should be used.

6.4.3 SDP-based approximation algorithm

Before proceeding, consider an example consisting of synthetic data specially chosen to be a *well clusterable data* set [2]. The example shows that if the data matrix represents a *well clusterable data* set, then the solution of the SDP relaxation (6.13) is also the solution of the SDP-based problem (6.9).

Example 27 *Consider the data set consisting of* $m = 11$ *objects and* $n = 3$ *attributes with data matrix* $\mathbf{D}$ *given as follows:*

$$\mathbf{D} = \begin{bmatrix} 5.03 & -4.99 & -3.08 \\ 5.40 & -4.84 & -2.59 \\ -5.66 & 3.96 & 2.81 \\ -4.84 & 3.01 & 1.96 \\ -4.53 & 4.63 & 3.92 \\ -5.33 & 4.07 & 2.17 \\ 4.91 & 4.09 & 2.52 \\ 5.55 & 4.55 & 3.70 \\ 4.87 & 3.88 & 3.11 \\ -4.24 & 4.92 & 2.66 \\ -4.47 & 4.81 & 1.99 \end{bmatrix}.$$

Suppose we are interested in obtaining a partition of these 11 *objects into* 3 *clusters.*

We have constructed the SDP model (6.13)*. Then, we solved it numerically by using the CSDP solver in* R*, with the tolerance* 10^{-8}*, and get the following solution, displayed here with only* 4 *decimals:*

$$\mathbf{Z} = \begin{bmatrix} 0.5 & 0.5 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.5 & 0.5 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0 & 0.0 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.1667 \\ 0.0 & 0.0 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.1667 \\ 0.0 & 0.0 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.1667 \\ 0.0 & 0.0 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.1667 \\ 0.0 & 0.0 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.3333 & 0.3333 & 0.3333 & 0.0000 & 0.0000 \\ 0.0 & 0.0 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.3333 & 0.3333 & 0.3333 & 0.0000 & 0.0000 \\ 0.0 & 0.0 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.3333 & 0.3333 & 0.3333 & 0.0000 & 0.0000 \\ 0.0 & 0.0 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.1667 \\ 0.0 & 0.0 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.0000 & 0.0000 & 0.0000 & 0.1667 & 0.1667 \end{bmatrix}.$$

The solution matrix $\mathbf{Z}$ *has three eigenvalues equal to 1 and eight eigenvalues equal to 0. Therefore, it is positive semidefinite.*

It is easy to see that, within a given tolerance, the matrix $\mathbf{Z}$ *satisfies the nonlinear constraint of the 0-1 SDP model* (6.9)*, thus, it is also a solution of problem* (6.9)*. Notice that the structure of* $\mathbf{Z}$ *already permits to see the relations between objects. Applying the recovering method suggested in Section 6.3 using the row echelon form of* $\mathbf{Z}$*, we get the assignment matrix*

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Therefore, the objects 1 *and* 2 *can be assigned to one cluster, objects* 3*,* 4 5*,* 6*,* 10 *and* 11 *to another, and the objects* 7*,* 8 *and* 9 *to a third cluster.*

In the following example, we use a real data set, presented in [150] and also analysed in [151], describing a short-term macroeconomic scenario of 20 OECD countries characterized by 6 economic indicators, with the aim of clustering the data into 3 clusters. It can be shown that the solution of the SDP relaxation (6.13) is not feasible for the SDP-based model (6.9).

Example 28 *Consider the* (20×6) *data matrix from [150] given by*

$$\mathbf{D} = \begin{bmatrix}
1.81760 & -0.03755 & 2.13484 & 0.09760 & -1.2 & -0.25160 \\
-0.88325 & -0.38487 & -0.25765 & -0.74250 & 0.014 & -0.53603 \\
-0.66426 & -0.43180 & -0.47237 & 0.46823 & 0.782 & 0.57978 \\
0.64966 & -0.10795 & 0.32513 & 0.17173 & -1.1 & -0.05470 \\
-0.95625 & -0.32855 & 0.01840 & -0.59424 & -1.1 & 0.31724 \\
1.16063 & -0.44119 & -0.74842 & 1.01182 & -0.42 & 1.52056 \\
-0.00730 & -0.42007 & -0.22698 & 0.98711 & -0.52 & 0.44851 \\
-1.10224 & -0.40834 & -1.05515 & 0.44352 & -0.42 & -0.07658 \\
0.64966 & 1.45968 & -0.25765 & 0.64119 & -0.34 & -2.22068 \\
-1.02924 & 0.14081 & -0.56438 & 1.13537 & -0.29 & 0.53603 \\
-1.61321 & -1.11236 & 1.21465 & -0.86604 & 1.47 & -0.14221 \\
-0.00730 & 3.63982 & 1.27600 & -1.11313 & 0.859 & -0.40475 \\
0.43068 & -0.42007 & 0.04908 & -0.86604 & 1.65 & 1.12675 \\
-0.66426 & -0.23702 & -0.16563 & -1.08842 & 1.47 & 1.14863 \\
0.35768 & -0.15019 & -2.74217 & -0.69308 & 1.19 & -2.30819 \\
0.94165 & -0.15254 & 0.32513 & 2.79085 & 0.065 & -0.14221 \\
1.30663 & -0.30038 & -0.10429 & 0.29527 & -1.4 & 1.12675 \\
-0.88325 & -0.85422 & 0.20244 & -0.96488 & 0.986 & 0.55790 \\
-0.81025 & 0.52098 & 1.06129 & -0.32245 & -1.2 & -0.51415 \\
1.30663 & 0.02581 & -0.01227 & -0.79191 & -0.60 & -0.71106
\end{bmatrix}.$$

Let us assign these 20 *objects to* 3 *clusters. The numerical solution of the linear SDP model* (6.13)*, obtained using the CSDP solver in* R *with the tolerance* 10^{-8}*, is displayed here with* 4 *decimals. It is a* (20×20) *matrix* $\mathbf{Z}$ *given by*

$$\mathbf{Z} = \begin{bmatrix}
0.1330 & 0.0000 & 0.0000 & 0.1044 & 0.0341 & 0.0874 & 0.0792 & 0.0081 & 0.0551 & 0.0372 & 0.0000 & 0.0877 & 0.0000 & 0.0000 & 0.0000 & 0.0948 & 0.1040 & 0.0000 & 0.0867 & 0.0882 \\
0.0000 & 0.0985 & 0.0855 & 0.0049 & 0.0598 & 0.0075 & 0.0126 & 0.0706 & 0.0468 & 0.0430 & 0.1046 & 0.0000 & 0.1046 & 0.1046 & 0.0993 & 0.0065 & 0.0053 & 0.1046 & 0.0148 & 0.0264 \\
0.0000 & 0.0855 & 0.0995 & 0.0189 & 0.0749 & 0.0286 & 0.0345 & 0.0735 & 0.0000 & 0.0652 & 0.1094 & 0.0000 & 0.1094 & 0.1094 & 0.0077 & 0.0249 & 0.0203 & 0.1094 & 0.0232 & 0.0059 \\
0.1044 & 0.0049 & 0.0189 & 0.1109 & 0.0497 & 0.1099 & 0.1035 & 0.0341 & 0.0156 & 0.0668 & 0.0000 & 0.0176 & 0.0000 & 0.0000 & 0.0000 & 0.1119 & 0.1145 & 0.0000 & 0.0673 & 0.0697 \\
0.0341 & 0.0598 & 0.0749 & 0.0497 & 0.0682 & 0.0559 & 0.0579 & 0.0608 & 0.0000 & 0.0667 & 0.0760 & 0.0000 & 0.0760 & 0.0760 & 0.0000 & 0.0542 & 0.0521 & 0.0760 & 0.0364 & 0.0250 \\
0.0874 & 0.0075 & 0.0286 & 0.1099 & 0.0559 & 0.1168 & 0.1119 & 0.0469 & 0.0034 & 0.0800 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1158 & 0.1146 & 0.0000 & 0.0610 & 0.0603 \\
0.0792 & 0.0126 & 0.0345 & 0.1035 & 0.0579 & 0.1119 & 0.1080 & 0.0504 & 0.0023 & 0.0809 & 0.0067 & 0.0000 & 0.0067 & 0.0067 & 0.0000 & 0.1102 & 0.1081 & 0.0067 & 0.0583 & 0.0555 \\
0.0081 & 0.0706 & 0.0735 & 0.0341 & 0.0608 & 0.0469 & 0.0504 & 0.0770 & 0.0352 & 0.0664 & 0.0646 & 0.0000 & 0.0646 & 0.0646 & 0.0822 & 0.0421 & 0.0362 & 0.0646 & 0.0264 & 0.0316 \\
0.0551 & 0.0468 & 0.0000 & 0.0156 & 0.0000 & 0.0034 & 0.0023 & 0.0352 & 0.2105 & 0.0000 & 0.0000 & 0.2106 & 0.0000 & 0.0000 & 0.2233 & 0.0051 & 0.0072 & 0.0000 & 0.0780 & 0.1066 \\
0.0372 & 0.0430 & 0.0652 & 0.0668 & 0.0667 & 0.0800 & 0.0809 & 0.0664 & 0.0000 & 0.0805 & 0.0471 & 0.0000 & 0.0471 & 0.0471 & 0.0051 & 0.0756 & 0.0703 & 0.0471 & 0.0426 & 0.0311 \\
0.0000 & 0.1046 & 0.1094 & 0.0000 & 0.0760 & 0.0000 & 0.0067 & 0.0646 & 0.0000 & 0.0471 & 0.1437 & 0.0000 & 0.1437 & 0.1437 & 0.0000 & 0.0000 & 0.0000 & 0.1437 & 0.0166 & 0.0000 \\
0.0877 & 0.0000 & 0.0000 & 0.0176 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2106 & 0.0000 & 0.0000 & 0.4407 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1550 & 0.0884 \\
0.0000 & 0.1046 & 0.1094 & 0.0000 & 0.0760 & 0.0000 & 0.0067 & 0.0646 & 0.0000 & 0.0471 & 0.1437 & 0.0000 & 0.1437 & 0.1437 & 0.0000 & 0.0000 & 0.0000 & 0.1437 & 0.0166 & 0.0000 \\
0.0000 & 0.1046 & 0.1094 & 0.0000 & 0.0760 & 0.0000 & 0.0067 & 0.0646 & 0.0000 & 0.0471 & 0.1437 & 0.0000 & 0.1437 & 0.1437 & 0.0000 & 0.0000 & 0.0000 & 0.1437 & 0.0166 & 0.0000 \\
0.0000 & 0.0993 & 0.0077 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0822 & 0.2233 & 0.0051 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.4636 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1188 \\
0.0948 & 0.0065 & 0.0249 & 0.1119 & 0.0542 & 0.1158 & 0.1102 & 0.0421 & 0.0051 & 0.0756 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1161 & 0.1165 & 0.0000 & 0.0621 & 0.0639 \\
0.1040 & 0.0053 & 0.0203 & 0.1145 & 0.0521 & 0.1146 & 0.1081 & 0.0362 & 0.0072 & 0.0703 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1165 & 0.1190 & 0.0000 & 0.0635 & 0.0684 \\
0.0000 & 0.1046 & 0.1094 & 0.0000 & 0.0760 & 0.0000 & 0.0067 & 0.0646 & 0.0000 & 0.0471 & 0.1437 & 0.0000 & 0.1437 & 0.1437 & 0.0000 & 0.0000 & 0.0000 & 0.1437 & 0.0166 & 0.0000 \\
0.0867 & 0.0148 & 0.0232 & 0.0673 & 0.0364 & 0.0610 & 0.0583 & 0.0264 & 0.0780 & 0.0426 & 0.0166 & 0.1550 & 0.0166 & 0.0166 & 0.0000 & 0.0621 & 0.0635 & 0.0166 & 0.0904 & 0.0678 \\
0.0882 & 0.0264 & 0.0059 & 0.0697 & 0.0250 & 0.0603 & 0.0555 & 0.0316 & 0.1066 & 0.0311 & 0.0000 & 0.0884 & 0.0000 & 0.0000 & 0.1188 & 0.0639 & 0.0684 & 0.0000 & 0.0678 & 0.0923
\end{bmatrix}$$

Despite the obtained matrix $\mathbf{Z}$ *satisfies the constraints of the relaxed SDP problem* (6.13)*, it can be verified that it does not satisfy the nonlinear constraint of the SDP-based problem* (6.9)*.*

The considerations and the examples above show that, generally, the nonlinear SDP-based model (6.9) is very difficult to solve using direct methods, and its linear relaxations can provide infeasible solutions. One of the reasons of this issue may be the lack of regularity. Therefore, specific strategies should be used to solve the problem (6.9). One such strategies is to use approximation algorithms.

In [115], an approximation algorithm that uses a linear SDP relaxation is proposed to obtain an approximate solution of the problem (6.9). First, the SDP relaxation is solved using a procedure based on the characterization of the sum of the largest eigenvalues of a symmetric matrix introduced in [110]. Then, a rounding procedure is used to extract a feasible solution of (6.9). In what follows, we consider the two steps of this approximation algorithm.

First step: solving the linear SDP relaxation

Consider a feasible solution $\mathbf{Z}$ of the SDP problem (6.16). First, notice that one of its eigenvalues is equal to 1 and the corresponding eigenvector is $\frac{1}{\sqrt{m}}e^m$. Moreover, any matrix $\mathbf{Z}$ can be written as

$$\mathbf{Z} = \mathbf{Z}_1 + \frac{1}{m}e^m(e^m)^T, \tag{6.18}$$

where $\mathbf{Z}_1$ is a $(m \times m)$ column and row centred matrix by the orthogonal projection matrix $\left(\mathbf{I}_m - \frac{1}{m}e^m(e^m)^T\right)$, *i.e.*,

$$\mathbf{Z}_1 = \left(\mathbf{I}_m - \frac{1}{m}\mathbf{e}^m(\mathbf{e}^m)^T\right)\mathbf{Z} = \left(\mathbf{I}_m - \frac{1}{m}\mathbf{e}^m(\mathbf{e}^m)^T\right)\mathbf{Z}\left(\mathbf{I}_m - \frac{1}{m}\mathbf{e}^m(\mathbf{e}^m)^T\right). \tag{6.19}$$

Notice that $\left(\mathbf{I}_m - \frac{1}{m}e^m(e^m)^T\right)$ is a $(m \times m)$ matrix that satisfies

$$\left(\mathbf{I}_m - \frac{1}{m}e^m(e^m)^T\right)^2 = \left(\mathbf{I}_m - \frac{1}{m}e^m(e^m)^T\right).$$

Since $\text{tr}(\mathbf{Z}) = p$, then by using (6.18) we get

$$\text{tr}\left(\mathbf{Z}_1\right) + \text{tr}\left(\frac{1}{m}e^m(e^m)^T\right) = p,$$

and it is easy to see that

$$\text{tr}\left(\mathbf{Z}_1\right) = p - 1. \tag{6.20}$$

Applying the same transformation to the matrix $\mathbf{D}\mathbf{D}^T$, we get the matrix $\mathbf{W}_1$, which is given by

$$\mathbf{W}_1 = \left(\mathbf{I}_m - \frac{1}{m}e^m(e^m)^T\right)\mathbf{D}\mathbf{D}^T\left(\mathbf{I}_m - \frac{1}{m}e^m(e^m)^T\right). \tag{6.21}$$

Second, notice that the objective function in (6.16) can be written as

$$\mathrm{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}) = \mathrm{tr}\left(\mathbf{D}\mathbf{D}^T\mathbf{Z}_1 + \frac{1}{m}\mathbf{D}\mathbf{D}^T\right)$$

and since the last term is constant, then maximizing $\mathrm{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z})$ is equivalent to maximizing $\mathrm{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}_1)$.

Now, we need to establish a relation between $\mathrm{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}_1)$ and $\mathrm{tr}(\mathbf{W}_1\mathbf{Z}_1)$. For this purpose, notice that

$$\begin{aligned}\mathrm{tr}\left(\mathbf{W}_1\mathbf{Z}_1\right) &= \mathrm{tr}\left(\mathbf{D}\mathbf{D}^T\left(\mathbf{I}_m - \tfrac{1}{m}e^m(e^m)^T\right)^2\mathbf{Z}\left(\mathbf{I}_m - \tfrac{1}{m}e^m(e^m)^T\right)^2\right)\\ &= \mathrm{tr}\left(\mathbf{D}\mathbf{D}^T\left(\mathbf{I}_m - \tfrac{1}{m}e^m(e^m)^T\right)\mathbf{Z}\left(\mathbf{I}_m - \tfrac{1}{m}e^m(e^m)^T\right)\right)\\ &= \mathrm{tr}\left(\mathbf{D}\mathbf{D}^T\mathbf{Z}_1\right).\end{aligned}$$

Hence, maximizing the objective function in (6.16) is equivalent to maximizing $\mathrm{tr}(\mathbf{W}_1\mathbf{Z}_1)$. Then, the SDP problem (6.16) can be rewritten in the form of the following problem w.r.t. the matrix variable $\mathbf{Z}_1$:

$$\begin{array}{ll}\max\limits_{\mathbf{Z}_1} & \mathrm{tr}(\mathbf{W}_1\mathbf{Z}_1)\\ \text{s.t.} & \mathrm{tr}(\mathbf{Z}_1) = p-1,\\ & \mathbf{I}_m \succeq \mathbf{Z}_1 \succeq 0.\end{array} \tag{6.22}$$

Based on the results from [110] and [115], it can be shown that the optimal solution of the SDP problem (6.22) can be achieved if and only if

$$\mathrm{tr}(\mathbf{W}_1\mathbf{Z}_1) = \sum_{i=1}^{p-1}\lambda_i,$$

where $\lambda_1, ..., \lambda_m$ are the eigenvalues of the matrix $\mathbf{W}_1$ listed in decreasing order.

Any solution $\mathbf{Z}_1$ of the SDP problem (6.22) has the form $\mathbf{Z}_1 = \mathbf{F}\mathbf{F}^T$, where $\mathbf{F}$ is the $(m \times (p-1))$ matrix whose columns are the eigenvectors associated to the $p-1$ largest eigenvalues of the matrix $\mathbf{W}_1$ [83, 110, 115].

Hence, after obtaining a solution $\mathbf{Z}_1$ of the SDP problem (6.22), we replace it in (6.18) to get the solution of the SDP relaxed problem (6.16).

Therefore, a procedure to obtain a solution $\mathbf{Z}$ of the relaxed problem (6.16) can be outlined as follows ([115]):

Algorithm 10 Solving the SDP relaxed problem (6.16)

```
1: Compute the matrix $\mathbf{W}_1$ by using (6.21),
2: Apply SVD to obtain the $p-1$ largest eigenvalues of the matrix $\mathbf{W}_1$ and the associated eigenvectors, $v^1, ..., v^{p-1}$, and construct the matrix $\mathbf{F}$, whose columns are these eigenvectors,
3: Set $\mathbf{Z} = \mathbf{F}\mathbf{F}^T + \frac{1}{m}e^m(e^m)^T$.
```

The solution of the SDP problem (6.16), obtained using the Algorithm 10, can be used as an approximate solution of the SDP-based problem (6.9). This solution is not necessarily feasible. To find a feasible solution of (6.9), a rounding procedure is needed.

Although in [116] it was proposed another form to get the solution $\mathbf{Z}$ of the relaxed problem (6.16), we think that presumably it may have some mistyped steps, specifically when it is considered a diagonal matrix variable $\mathbf{\Gamma}$, which according to their procedure was only symmetric and not diagonal. So, we have followed the approach proposed in [115]. Nevertheless, we will use here the rounding procedure to get the feasible solution for (6.9) suggested in [116].

Second step: Rounding the approximate solution

It can be observed that if $\mathbf{Z}$ is a solution of problem (6.9), then $\mathbf{ZD}$ has at least p different rows. Notice that $\mathbf{ZD}$ can be considered as an object centroid based matrix, where each object is identified by the cluster centroid to which it belongs. Based on this observation, in [116], the following rounding procedure is suggested.

Algorithm 11 Rounding procedure for the SDP-based problem (6.9)

```
1: Given the data matrix $\mathbf{D}$ and the solution $\mathbf{Z}$ of the relaxed problem (6.16), select $p$ different
   rows of $\mathbf{ZD}$ and define the initial centroids,
2: Apply K-means to problem (6.5) using the initial centroids to obtain $\mathbf{U}$,
3: Set $\mathbf{Z} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$.
```

Notice that the SDP-based approximation algorithm, described by the Algorithms 10 and 11, performs a PCA (principal component analysis) step, by reducing the problem into (6.22) and solving it using SVD, and a K-means step.

6.5 Clustering and dimensionality reduction

High-dimensional data sets are challenging and it is often needed not only to cluster objects, but also reduce the dimension of the attribute space to facilitate the data analysis. Although principal component analysis (PCA) is a widely used tool in statistics for dimensionality reduction, an alternative way is to consider an attribute clustering and many authors have used this approach (see, *e.g.*, [7, 32, 151]). In this section, we describe a statistical technique proposed in [151] that provides not only clusters of objects, but also attributes.

6.5.1 Clustering and disjoint PCA

A new methodology called Clustering and Disjoint Principal Component Analysis (referred to hereafter as CDPCA) was proposed by Vichi and Saporta in [151] for obtaining not only nonoverlapping clusters of objects, but also a partition of the attribute space into disjoint subsets.

Given a data set, CDPCA groups m objects into p, $2 \leq p < m$, clusters identified by their centroids and, simultaneously, partitions n attributes into k, $2 \leq k < n$, disjoint subsets of attributes, called components. Basically, the idea is to perform K-means in the reduced space of the components obtained via PCA. In what follows, we briefly describe the CDPCA methodology (see [94, 151] for further details).

The CDPCA model results from applying PCA to the transformed data matrix $\mathbf{U}\bar{\mathbf{D}}$, where each object is replaced by its cluster centroid, obtained as a result of applying the K-means algorithm to the original data matrix $\mathbf{D}$. Hence, the data matrix $\mathbf{D}$ would be fitted by the CDPCA model as follows ([151]):

$$\mathbf{D} = \mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T + \mathbf{E}, \tag{6.23}$$

where

- $\mathbf{U}$ is the object assignment matrix introduced in (6.1),
- $\bar{\mathbf{Y}}$ is a $(p \times k)$ object centroid matrix in the reduced space of the components,
- $\mathbf{A}$ is the $(n \times k)$ component loading matrix,
- $\mathbf{E}$ is a $(m \times n)$ error matrix.

In [151], it is proposed to consider a decomposition of the matrix $\mathbf{A}$ in order to include a binary and row stochastic matrix $\mathbf{V}$, specifying the partition of n attributes into k disjoint components. The positions of the nonzero elements in $\mathbf{A}$ are identified by the positions of the unit elements in $\mathbf{V}$. The component loading matrix $\mathbf{A}$ is given by $\mathbf{A} = \mathbf{B}\mathbf{V}$, where $\mathbf{B} = \sum_{q=1}^{k} \text{diag}(v_q)\text{diag}(c_q)$ is a diagonal matrix of order n specifying the loadings of each component, where v_q is the q-th column of $\mathbf{V}$, c_q is the eigenvector associated to the largest eigenvalue of the matrix $(\mathbf{U}\bar{\mathbf{D}}\text{diag}(v_q))^T\mathbf{U}\bar{\mathbf{D}}\text{diag}(v_q)$ and $\bar{\mathbf{D}}$ is defined in (6.3). The matrix $\mathbf{A}$ has a unique nonzero element per row, *i.e.*, each row (attribute) contributes to a single column (component), and satisfies the conditions $\text{rank}(\mathbf{A}) = k$ and $\mathbf{A}^T\mathbf{A} = \mathbf{I}_k$.

The object centroid matrix in the reduced space $\bar{\mathbf{Y}}$ is given by $\bar{\mathbf{Y}} = \bar{\mathbf{D}}\mathbf{A}$.

We shall include one more definition that will be used later. The $(m \times k)$ component score matrix $\mathbf{Y} = [y_{iq}]$ is given by $\mathbf{Y} = \mathbf{D}\mathbf{A}$, where y_{iq} is the value of the i-th object for the q-th component.

It is natural to try to minimize the error in the CDPCA model (6.23), *i.e.*, minimize the norm of the error matrix: $\|\mathbf{D} - \mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T\|_2^2$. It is easy to see that this problem is equivalent to maximizing $\|\mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T\|_2^2$. Since the columns $\mathbf{A}_i$, $i = 1, ..., k$, of the matrix $\mathbf{A}$ are orthogonal, *i.e.*, $\mathbf{A}^T\mathbf{A} = \mathbf{I}_k$, the matrix $\bar{\mathbf{Y}}$ is given by $\bar{\mathbf{Y}} = \bar{\mathbf{D}}\mathbf{A}$, and the trace of a square matrix equals the trace of its transpose, then maximizing $\|\mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T\|_2^2$ is the same as maximizing $\|\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\|_2^2$, the between cluster deviance in the reduced space. Indeed, $\|\mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T\|_2^2 = \text{tr}\left((\mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T)(\mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T)^T\right) = \text{tr}\left(\mathbf{U}\bar{\mathbf{Y}}\mathbf{A}^T\mathbf{A}\bar{\mathbf{Y}}^T\mathbf{U}^T\right) = \text{tr}\left(\mathbf{U}\bar{\mathbf{Y}}\bar{\mathbf{Y}}^T\mathbf{U}^T\right) =$

$\text{tr}\left(\mathbf{U}(\bar{\mathbf{D}}\mathbf{A})(\bar{\mathbf{D}}\mathbf{A})^T\mathbf{U}^T\right) = \text{tr}\left((\mathbf{U}\bar{\mathbf{D}}\mathbf{A})(\mathbf{U}\bar{\mathbf{D}}\mathbf{A})^T\right) = \|\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\|_2^2$. Hence, we get the optimization problem

$$\max_{\mathbf{U},\bar{\mathbf{D}},\mathbf{A}} \quad \|\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\|_2^2, \tag{6.24}$$

where $\mathbf{U}$ is a binary and row stochastic matrix, $\mathbf{A}$ is a columnwise orthonormal matrix and $\bar{\mathbf{D}}\mathbf{A}$ is an object centroid matrix in the reduced space.

Considering that we seek nonempty clusters of objects and nonempty disjoint components, the CDPCA problem can be formulated as the following quadratic mixed integer programming problem:

$$\begin{array}{lll} \max\limits_{\mathbf{U},\mathbf{V},\mathbf{A}} & \|\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\|_2^2 & \\ \text{s.t.} & u_{ij} \in \{0,\ 1\}, & i = 1,...,m;\ j = 1,...,p \\ & \sum\limits_{j=1}^{p} u_{ij} = 1, & i = 1,...,m \\ & \sum\limits_{i=1}^{m} u_{ij} > 0, & j = 1,...,p \\ & v_{ij} \in \{0,\ 1\}, & i = 1,...,n;\ j = 1,...,k \\ & \sum\limits_{j=1}^{k} v_{ij} = 1, & i = 1,...,n \\ & \sum\limits_{i=1}^{n} v_{ij} > 0, & j = 1,...,k \\ & \sum\limits_{i=1}^{n} a_{ij}^2 = 1, & j = 1,...,k \\ & \sum\limits_{i=1}^{n} (a_{ij}a_{ir})^2 = 0, & j = 1,...,k-1;\ r = j+1,...,k \\ & \sum\limits_{j=1}^{k} a_{ij}^2 > 0, & i = 1,...,n. \end{array} \tag{6.25}$$

The first two constraints in the problem (6.25) correspond to the assignment of m objects into p clusters and the next two constraints represent the assignment of n attributes into k disjoint components. The remaining constraints are associated to the PCA implementation: guarantee that we get a columnwise orthonormal matrix $\mathbf{A}$ where for any two different columns one of the corresponding entries is zero, and that all the original attributes are included in the new components. The objective function value given by $\|\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\|_2^2$ can be computed either by $\text{tr}\left(\mathbf{U}\bar{\mathbf{D}}\mathbf{A}(\mathbf{U}\bar{\mathbf{D}}\mathbf{A})^T\right)$, which corresponds to the between cluster distances, or by $\text{tr}\left((\mathbf{U}\bar{\mathbf{D}}\mathbf{A})^T\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\right)$, which represents the total variance of the data in the reduced space, where the objects are identified by their centroids.

6.5.2 An alternating least-squares algorithm

The problem (6.25) is difficult to solve due to the presence of discrete variables. In [151], a four step Alternating Least-Squares (ALS) algorithm was proposed to solve this problem. In [94], we showed that each iteration of the ALS algorithm can be summarily described by two basic steps: the assignment of objects using K-means, and the reduction of the attribute space applying PCA to the resulting centroids. The simplified version of the ALS algorithm can be described as follows.

Algorithm 12 Alternating Least-Squares (Two-Step version of ALS) ([94])

```
input:  $\mathbf{D}$, data matrix of $m$ objects and $n$ attributes;
        $p$ and $k$, the desired numbers of clusters of objects and attributes, respectively;
        $\varepsilon$, numerical tolerance.
output: $\mathbf{U}$, object assignment matrix;
        $\mathbf{V}$, attribute assignment matrix;
        $\mathbf{A}$, component loading matrix;
        $\|\mathbf{U}\bar{\mathbf{D}}\mathbf{A}\|_2^2$, the between cluster deviance in the reduced space of the components.
  repeat
    1. K-means step (for the objects):
       • assign $m$ objects into $p$ clusters, obtaining matrix $\mathbf{U}$,
       • calculate the centroids in the space of the observed attributes and find matrix $\bar{\mathbf{D}}$,
       • identify the objects by their cluster centroids in the space of the observed attributes
         and get $\mathbf{U}\bar{\mathbf{D}}$.
    2. PCA step (for the attributes):
       • assign $n$ attributes into $k$ subsets and obtain matrix $\mathbf{V}$,
       • obtain the loadings of the CDPCA components and get matrix $\mathbf{A}$,
       • calculate the centroids in the reduced space of $k$ CDPCA components and define
         matrix $\bar{\mathbf{Y}} = \bar{\mathbf{D}}\mathbf{A}$,
       • identify the objects in the reduced space of $k$ CDPCA components and calculate
         matrix $\mathbf{Y} = \mathbf{D}\mathbf{A}$.
  until the difference between two consecutive computations of the between cluster deviance in
  the reduced space is smaller than $\varepsilon$.
```

At the beginning, the m objects are assigned into p clusters by means of the matrix $\mathbf{U}$. Next, each row of the data matrix is replaced by its corresponding object centroid, yielding the matrix $\mathbf{U}\bar{\mathbf{D}}$. The assignment of n attributes into k disjoint subsets is specified in the matrix $\mathbf{V}$ and the CDPCA component loadings are specified in the matrix $\mathbf{A}$. These two matrices are obtained using an iterative procedure working row-by-row and column-by-column on the matrices $\mathbf{V}$ and $\mathbf{A}$ in order to maximize the objective function of problem (6.25). Such iterative procedure for updating $\mathbf{V}$ and $\mathbf{A}$ is described in detail in [94] and can be briefly summarized as follows. For updating $\mathbf{V}$, the assignment of each original attribute to a component will be evaluated in order to find which component leads to a higher value of the objective function of (6.25), assuming that all remaining attributes are fixed. The first row of $\mathbf{V}$ is updated by detecting for which column j, with $j = 1, \cdots, k$, the assignment of its nonzero element yields better results in the sense of the maximization of the objective function. Concretely, fixing the first row (attribute), the *best* column (component) among k is selected by solving k PCA subproblems associated to the submatrices of $\mathbf{U}\bar{\mathbf{D}}$ restricted to the original attributes assigned into the j-th column of $\mathbf{V}$, assuming the k possible positions of the nonzero element into the first row of the potential updated matrix of $\mathbf{V}$. In the j-th PCA subproblem, the first principal component is calculated determining the update of the j-th column of $\mathbf{A}$. At this point, $\bar{\mathbf{Y}}$ and the objective function value of (6.25) are evaluated using $\mathbf{A}$. This process is done repeatedly to select the *best* component to assign the first row (attribute) in $\mathbf{V}$, which will coincide with the component that yields the highest value of the objective function value of (6.25). Then, the same process is repeated for the remaining rows of $\mathbf{V}$. Hence, for each original attribute there are solved k assignment subproblems. In each subproblem, a subspace of attributes is considered and the best direction (eigenvector) with maximum variability explained is obtained performing a PCA step. Each attribute will be included into a component associated to the subproblem that maximizes the objective function. Since there are n original attributes, then there are $n \times k$ subproblems to be solved in order to obtain $\mathbf{V}$ and $\mathbf{A}$. The best assignment will maximize the between cluster deviance in the reduced space of the components. Having obtained the component loading matrix $\mathbf{A}$, the component score matrix $\mathbf{Y}$, the object centroid matrix in the reduced space $\bar{\mathbf{Y}}$, and the value of the objective function of (6.25) are computed. Therefore, at the end of each iteration of the algorithm, m objects of the data matrix are assigned to p clusters, and simultaneously displayed in a reduced space of k disjoint components. The value of the between cluster deviance is also computed to evaluate the quality of the clustering. The iterative procedure of the Algorithm 12 stops when the difference between two consecutive computations of the value of the objective function of problem (6.25) is smaller than a pre-specified tolerance.

Since the objective function of problem (6.25) is bounded above, the algorithm converges to a stationary point, which is a local maximum of problem [151]. The ALS procedure can be considered as a heuristic and thus, to increase the possibility to achieve the global maximum, it has been suggested to run the algorithm several times for different initial assignment matrices $\mathbf{U}$ and $\mathbf{V}$, randomly chosen at the beginning of each run.

The difference between the four step ALS algorithm proposed in [151] and its simplified version described in [94], and summarized in Algorithm 12, consists in the way matrices $\mathbf{V}$ and $\mathbf{A}$ are updated. In the general iteration of the Two-Step version of ALS, these matrices are sequentially constructed by an iterative procedure applied to their rows and columns, in order to maximize the between cluster deviance in the reduced space, while in the four step ALS algorithm the matrices $\mathbf{V}$ and $\mathbf{A}$ are updated separately. Numerical experiments show that the ALS algorithm in its simplified Two-Step version 12 is faster than the original four step version.

6.6 A new SDP-based approach to clustering and dimensionality reduction

In this section, a new approach to clustering and dimensionality reduction that uses SDP models is presented. The new Two-Step-SDP algorithm based on a modification of the ALS algorithm to solve the CDPCA problem (6.25) is described and some differences between these algorithms are discussed. Numerical experiments using an R implementation of the Two-Step-SDP algorithm are also included.

6.6.1 Description of the new approach

In this section, we propose a new approach to solve the CDPCA problem (6.25) by combining SDP models and the CDPCA methodology. The new approach is called Two-Step-SDP, since two clustering problems are considered and solved using a SDP-based approximation algorithmic framework, and since we modify and improve the ALS algorithm in its Two-Step version summarized in Algorithm 12 in order to use SDP models for clustering objects and attributes. There are two main modifications on the Algorithm 12. In the first modification, we suggest to apply the SDP-based approximation algorithm described in Section 6.4 to construct the initial matrices **U** and **V**, instead of a random choice. It is expected that this approximation algorithm returns *almost optimal solutions* quite fast. The second modification on the Algorithm 12 consists in updating the component loading matrix **A** using the current assignment matrix **V**. Both modifications improve the Algorithm 12 not only in terms of computational time, but also in efficiency.

The idea of the Two-Step-SDP approach for solving the problem (6.25) is to obtain not only a solution of the SDP-based problem defined in (6.9) for clustering objects, but also a solution of a SDP-based problem for clustering attributes into disjoint subsets, using an approximation algorithm based on the framework described in the previous section. The solutions of such problems are computed in order to maximize the between cluster deviance in the reduced space of the components. Therefore, the Two-Step-SDP approach can be considered as an alternative to CDPCA.

Consider the SDP-based model (6.9) for the clustering problem. Recall that for clustering objects into groups, in (6.9) we use the matrix **Z** defined in (6.7) as an unknown orthogonal projection matrix defined by the assignment matrix **U**. For clustering attributes into disjoint subsets, we use the same model. For this purpose, we consider the assignment matrix **V** and the unknown $(n \times n)$ matrix $\mathbf{H} = [h_{ij}]$ defined as

$$\mathbf{H} = \mathbf{V}(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T, \qquad (6.26)$$

which is an orthogonal projection matrix of $\mathbb{R}^n$ onto $\mathcal{C}(\mathbf{V})$. The matrix **H** satisfies the properties (1)-(5). Using the model (6.9) with the matrix variable **H**, we get a SDP-based problem for clustering attributes.

As it was mentioned above, the SDP-based models of the form (6.9) are difficult to solve. Let us relax the nonlinear constraints of the SDP-based problems using the approach described in the previous section. Therefore, for the SDP-based problem for clustering objects, we get the relaxed SDP problem (6.16),

$$\begin{array}{ll}\max\limits_{\mathbf{Z}} & \operatorname{tr}(\mathbf{D}\mathbf{D}^T\mathbf{Z}) \\ \text{s.t.} & \operatorname{tr}(\mathbf{Z}) = p, \\ & \mathbf{Z}e^m = e^m, \\ & \mathbf{I}_m \succeq \mathbf{Z} \succeq 0,\end{array}$$

and for the SDP-based problem for clustering attributes, we get the following relaxation:

$$\begin{array}{ll}\max\limits_{\mathbf{H}} & \operatorname{tr}(\mathbf{D}^T\mathbf{D}\mathbf{H}) \\ \text{s.t.} & \operatorname{tr}(\mathbf{H}) = k, \\ & \mathbf{H}e^n = e^n, \\ & \mathbf{I}_n \succeq \mathbf{H} \succeq 0.\end{array} \tag{6.27}$$

To solve the linear SDP problems (6.16) and (6.27) one can use the approach described in Section 6.4, and the solution of the original SDP-based models can be obtained using a rounding procedure.

The rounding procedure used in the Two-Step-SDP approach can be summarized as follows. Given assignment matrices $\mathbf{U}$ and $\mathbf{V}$, apply the CDPCA methodology to obtain the component loading matrix $\mathbf{A}$, as well as the component score matrix $\mathbf{Y}$, and the object centroid matrix in the reduced space, $\bar{\mathbf{Y}}$. Next, perform the K-means algorithm for clustering the objects in the reduced space, *i.e.*, apply K-means to the matrix $\mathbf{Y}$ and use $\bar{\mathbf{Y}}$ as initial centroids. Finally, compute the between cluster deviance in the reduced space of the components. The algorithm stops when the between cluster deviance is no longer increased.

Motivated by the works [94, 115, 116] and [151], we propose a new approach to clustering and dimensionality reduction by combining SDP models and the CDPCA methodology. We modify and improve the ALS algorithm in order to use SDP models on the steps for clustering objects and attributes. The new algorithm is called Two-Step-SDP.

6.6.2 Two-Step-SDP algorithm

The algorithmic scheme of the Two-Step-SDP algorithm for clustering and dimensionality reduction can be described as follows.

Algorithm 13 Two-Step-SDP

```
input:  $\mathbf{D}$, data matrix of $m$ objects and $n$ attributes;
        $p$ and $k$, the desired number of clusters of objects and attributes, respectively;
        $\varepsilon$, numerical tolerance.
output: $\mathbf{U}$, object assignment matrix;
        $\mathbf{V}$, attribute assignment matrix;
        $\mathbf{A}$, component loading matrix;
        $\|\mathbf{ZDA}\|_2^2$, the between cluster deviance in the reduced space of the components.
1: Solve approximately the SDP-based model for clustering objects:
    a) Considering the relaxed model (6.16), compute the matrix $\mathbf{W}_1$ by using (6.21),
    b) Use SVD to obtain the $p-1$ largest eigenvalues of the matrix $\mathbf{W}_1$ and the associated
       eigenvectors, $v^1, ..., v^{p-1}$, and construct the matrix $\mathbf{F}$, whose columns are these eigen-
       vectors,
    c) Set the approximate solution: $\bar{\mathbf{Z}} = \mathbf{F}\mathbf{F}^T + \frac{1}{m}e^m(e^m)^T$.
2: Solve approximately the SDP-based model for clustering attributes:
    a) Considering the relaxed model (6.27), compute the matrix $\mathbf{W}_2$ by using (6.21) on the
       matrix $\mathbf{D}^T\mathbf{D}$,
    b) Use SVD to obtain the $k-1$ largest eigenvalues of the matrix $\mathbf{W}_2$ and the associ-
       ated eigenvectors, $w^1, ..., w^{k-1}$, and construct the matrix $\mathbf{G}$, whose columns are these
       eigenvectors,
    c) Set the approximate solution: $\bar{\mathbf{H}} = \mathbf{G}\mathbf{G}^T + \frac{1}{n}e^n(e^n)^T$.
3: Rounding procedure:
    Initialization: randomly select $p$ different rows from $\bar{\mathbf{Z}}\mathbf{D}$, and $k$ different rows from
    $\bar{\mathbf{H}}\mathbf{D}^T$, as initial centroids on the K-means algorithm to get the initial matrices $\mathbf{U}$, and $\mathbf{V}$,
    respectively.
    a) Compute $\mathbf{Z}^* = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$ and $\bar{\mathbf{D}} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{D}$.
    b) Compute $\mathbf{A}$ by fixing each column $v_q$, $q = 1, ..., k$, of $\mathbf{V}$ and replacing the nonzero
       elements in $v_q$ by the elements of the eigenvector associated to the largest eigenvalue
       of the matrix $(\mathbf{Z}^*\mathbf{D}[c])^T\,\mathbf{Z}^*\mathbf{D}[c]$, where $c$ is a row vector containing the row indices of
       nonzero elements of $v_q$, i.e., the attributes that contribute to the component $q$, and $[c]$
       means that only the columns specified in $c$ are used.
    c) Update $\mathbf{U}$ by performing the K-means algorithm in the reduced space, i.e., applying it
       to the matrix $\mathbf{Y} = \mathbf{DA}$ with initial centroids $\bar{\mathbf{Y}} = \bar{\mathbf{D}}\mathbf{A}$. Randomly select $k$ different
       rows from $\bar{\mathbf{H}}\mathbf{D}^T$ as initial centroids on K-means and get $\mathbf{V}$.
    d) Compute the between cluster deviance in the reduced space, $\|\mathbf{Z}^*\mathbf{DA}\|_2^2$.
    Stopping criterion: The rounding procedure stops when the difference between two con-
    secutive computations of the between cluster deviance in the reduced space is smaller than
    the pre-specified tolerance value $\varepsilon$.
```

Since the Two-Step-SDP algorithm uses the approximate solutions of the nonlinear SDP-based problems for clustering objects and attributes, which are close to the final solutions, the clustering process is expected to be finite.

The main feature of the Two-Step-SDP algorithm is that it constructs the clusters of attributes and uses them to cluster the objects by maximizing the between cluster deviance in the reduced space of the components, *i.e.*, the K-means algorithm is performed in lower dimensions. Moreover, the Two-Step-SDP algorithm provides not only the allocation of objects and attributes into clusters, but also the component loadings of such allocation of attributes. The quality of the clustering can be measured by considering the between cluster deviance in the reduced space of the components, or the within sum of squared distances between each object and the cluster centroids.

6.6.3 Two-Step-SDP and ALS algorithms

Notice that both the Two-Step-SDP and the ALS algorithms use iterative schemes to obtain the best solution. Nevertheless, there are some differences. In particular, in the ALS algorithm, the update of the matrices $\mathbf{V}$ and $\mathbf{A}$ is done using an alternating procedure that works row-by-row and column-by-column on these matrices by obtaining components that explain the largest variance in order to maximize the between cluster deviance in the reduced space. Such alternating procedure has to be performed nk times at each iteration of the main algorithm. Therefore, it may be quite expensive in terms of computation time and memory. In the Two-Step-SDP algorithm, the matrix $\mathbf{V}$ is obtained using the solution of the SDP relaxation problem (6.27), and therefore, it is expected to be close to the optimal assignment. Hence, at each iteration of the rounding procedure, the update of the matrix $\mathbf{A}$ requires only one step. Notice that in the Two-Step-SDP algorithm, the dimensionality reduction is done by finding a partition of the attributes specified by $\mathbf{V}$, and then the component loadings specified in the matrix $\mathbf{A}$ for this particular partition are computed. It follows that the dimensionality reduction may not explain the largest variance. Hence, the Two-Step-SDP algorithm can be regarded as a simplified version of the ALS algorithm, which results in a new version with a considerable improvement on the efficiency.

6.6.4 Implementation and numerical results

In what follows, several numerical experiments using the statistical software R [124], which is an open source software widely used by the statistical community, are carried out. Details on how to use our implementation of the Two-Step-SDP algorithm, as well as some examples are presented. A comparison of the results obtained using the proposed Two-Step-SDP algorithm and other existing techniques is made.

Implementation details

In [94], we have implemented the two-step version of the ALS algorithm for CDPCA in a easy-to-use software application using R. This function is available from the authors upon request. The R function was called `CDpca` and its main features include plotting the data's projection onto the two dimensional space defined by the first two CDPCA components, and constructing a pseudo-confusion matrix, which is important to show the (mis)classification of the objects when the true classification is known. We will use the function `CDpca` to compare the efficiency of implementations.

Here, we have implemented the Two-Step-SDP algorithm 13 and in order to solve the problems more efficiently, we used two strategies concerning computation of eigenvalues and eigenvectors. In the steps of the algorithm, we need to compute eigenvalues and eigenvectors of possibly huge covariance matrices of the form $\mathbf{M}^T\mathbf{M}$, where $\mathbf{M}$ is a matrix with more columns than rows. In these cases, we used the approach suggested in [119] and considered the smaller matrix $\mathbf{M}\mathbf{M}^T$. It is shown that the nonzero eigenvalues of $\mathbf{M}^T\mathbf{M}$ and $\mathbf{M}\mathbf{M}^T$ coincide, and each unit eigenvector $\mathbf{w}$ of $\mathbf{M}^T\mathbf{M}$ associated to the eigenvalue λ is related to the unit eigenvector $\mathbf{v}$ of the smaller matrix $\mathbf{M}\mathbf{M}^T$ by $\mathbf{w} = \mathbf{M}^T\frac{\mathbf{v}}{\sqrt{\lambda}}$. In the rounding procedure step of the Two-Step-SDP algorithm, we need to compute the largest eigenvalue and the corresponding eigenvector of a matrix. Here, we have implemented the Power Method [18], which proved to be faster than the standard methods for computing eigenvalues and eigenvectors.

The Two-Step-SDP algorithm was implemented in R by the function `TwostepSDPClust`, and is available in [90]. This function is suitable for data matrices with numeric elements and starts by standardizing the data (the description of the standardize step can be found in [94]). To use the routine `TwostepSDPClust`, the user needs to specify some arguments: `data`, the data matrix; `p` and `k`, the number of clusters of objects and attributes, respectively. It is also needed to define a parameter `class` which is a vector of integers with the true classification of objects, or 0, when the true class is unknown. The stopping criteria are specified in the parameters `tol`, a small convergence tolerance value, and `maxit`, which is the maximum number of iterations of the algorithm.

The routine `TwostepSDPClust` provides much more information than the standard K-means function in R, *e.g.*, besides returning the clusters of both objects and attributes, it returns the loadings of each original attribute to each component in the reduced space, and, when the true objects classification is known, a pseudo-confusion matrix. See all the output list in the Appendix E.

In the R environment, after defining the input arguments, the user can use the following command to apply the function `TwostepSDPClust`:

```
> example <- TwostepSDPClust(data,p,k,class,tol,maxit)
```

Data sets

The experiments were made using some standard problems from cluster analysis literature. Both data sets with more objects than attributes and with more attributes than objects have been selected. The following real-world data sets are available in the UCI repository [144], in the R package `plsgenomics` and in [150]:

- *OECD countries* – the data set of 20 objects and 6 attributes from [150], representing the short-term macroeconomic scenario (1999) of OECD countries; notice that, although the true classes are not known, the experiments reported in [150, 151] suggest to divide the objects into 3 clusters.

- *Breast Cancer* – the Winsconsin Breast Cancer data from UCI repository; contains 683 instances (originally, there were 699 instances, but 16 of them were excluded, since they contain missing values), each instance is described by 9 attributes with integer values in the range $1 - 10$ and a real binary class label, which divides the instances into two clusters, representing the type of tumor (benign or malignant).

- *Diabetes* – the Pima Indians Diabetes data set from UCI repository, contains 768 objects and 8 attributes, divided into two classes, representing the results on testing diabetes (positive or negative).

- *Colon* – the data set from microarray experiments on colon tissue samples, available in the the R package `plsgenomics`; contains 62 samples and 2000 genes, divided into 2 classes, representing the type of tumor (benign or malignant).

- *leukemia* – the leukemia data set, available in the package `plsgenomics`; contains 38 samples and 3051 genes, also divided into two classes, representing the type of tumor (benign or malignant).

- *SRBCT* – the gene expression data set from microarray experiments on small round blue cell tumors, also available in package `plsgenomics`; contains 83 objects (called samples) and 2308 attributes (genes), divided into 4 groups, representing four cancer variants.

- *Iris* – the Fisher's Iris data set, available in the UCI repository, the most famous data set for clustering experiments; contains 150 objects described by 4 attributes, and the true classification of the objects in 3 clusters is known, representing the three species of the Iris flower.

- *Soybean* – the Soybean data set from UCI repository contains 47 objects described by 35 attributes, where the four true classes of objects are also known, and represent four soybean diseases.

We have also tested the Two-Step-SDP algorithm on a synthetic data set, called *Synthetic*, specially constructed in [94] to satisfy the CDPCA model (6.23). It contains 15 objects and 3 attributes, divided into 3 classes.

Numerical results

To test the efficiency of the Two-Step-SDP algorithm implemented in the R function `TwostepSDPClust`, several numerical experiments have been carried out on a computer with an Intel Core i7-2630QM processor CPU@2.0GHz,, with Windows 7 (64 bits) and 12GB RAM, using R version 3.1.2 (2014). It is of interest to make a comparison of the performance of the Two-Step-SDP algorithm with other freely-available R functions. One such function is `kmeans`, the standard K-means algorithm which is available in R. The other function is our R implementation of the ALS algorithm, under the name `CDpca`, firstly implemented in [93] and recently improved in [94].

For the numerical tests using the Two-Step-SDP algorithm we have set the tolerance value to 10^{-8} and the maximum number of iterations to 100. The K-means heuristic was executed using all the data sets previously scaled, the multiple random start was set to 1000, and the maximum number of iterations was 100000. The tolerance for the ALS algorithm was set to 10^{-5} and the maximum number of iterations was 100. To augment the efficiency of the ALS algorithm, the numerical tests were run 1000 times, with the exception of the gene expression data *Colon*, *leukemia* and *SRBCT*. For the data sets *Colon* and *leukemia*, 20 runs were performed, and for *SRBCT*, only 10 runs were performed, because the computation time of the ALS algorithm increased in an unreasonable way in these cases. For the *Soybean* data set, the standardize step on the experiments was not executed, since the data matrix presents null columns.

Following the analysis presented in [151] for the *OECD countries* data set, let us consider only two principal components for all the tested data sets, *i.e.*, $k = 2$. Table 6.1 summarizes the characteristics of the data sets. It also contains the desired number of clusters of objects, represented by p.

Table 6.1: Summary of the characteristics of the data sets and the number of clusters of objects used in the experiments.

Dataset	Number of objects, m	Number of attributes, n	Desired number of clusters of objects, p
OECD countries	20	6	3
Breast Cancer	683	9	2
Diabetes	768	8	2
Colon	62	2000	2
leukemia	38	3051	2
SRBCT	83	2308	4
Iris	150	4	3
Soybean	47	35	4
Synthetic	15	3	3

For all the presented data, the true classes are known, with exception for the *OECD countries* data set.

The detailed numerical results are presented in Tables 6.2, 6.3 and 6.4. The Tables 6.2 and 6.3 contain the results obtained using the Two-Step-SDP algorithm and the K-means algorithm, respectively, while Table 6.4 contains the results obtained using the ALS algorithm. The first column of Tables 6.2, 6.3 and 6.4 contains the name of the data set, and the second column contains the computational time in seconds. In these tables, *iter* is the number of iterations, *wssd* is the within sum of squared distances, *i.e.*, within cluster deviance, *bcd* is the between cluster deviance, *bcdr* is the between cluster deviance in the reduced space of the components, *bcdrp* is the between cluster deviance of the total variance, e is the error associated to the CDPCA model (6.23), *Usize* and *Vsize* represent the sizes of the clusters of objects and attributes, respectively, and *Exp.var.* is the explained variance of the components. It should be noticed that, according to the information provided by the algorithms, the Tables 6.2, 6.3 and 6.4 contain a different number of columns.

Table 6.2: Numerical results using the Two-Step-SDP algorithm.

Data set	time	iter	wssd	bcd	bcdr	bcdrp	*error*	Usizes	Vsizes	explained var.
Breast Cancer	1.19	2	2728.15	3418.85	3418.85	75.6	0.07	230, 453	8, 1	62.52, 11.13
Diabetes	1.22	2	5121.9	1022.09	1022.09	38.7	0.09	309, 459	2, 6	26.62, 16.45
Iris	0.03	2	141.03	458.96	454.5	80.5	0.08	48, 50, 52	1, 3	69.60, 25.17
Soybean	0.05	6	453.12	2760.87	2557.18	99.6	0.54	9, 11, 13, 14	10, 25	1.51, 1.38
OECD countries	0.00	2	72.15	47.84	31.57	70.0	0.47	3, 7, 10	4, 2	22.39, 17.17
Colon	14.66	2	84631.79	39368.21	39368.21	61.4	4.69	18, 44	581, 1419	37.56, 15.03
leukemia	60.78	2	102681.7	13256.27	13256.27	75.9	8.43	17, 21	1374, 1677	7.83, 7.63
SRBCT	28.97	2	156947.7	34616.26	19438.99	86.2	4.99	13, 16, 18, 36	1293, 1015	6.28, 5.63
Synthetic	0.00	2	12.96	32.03	31.35	85.6	0.24	4, 5, 6	2, 1	51.47, 35.71

Table 6.3: Numerical results using the K-means algorithm.

Data set	time	wssd	bcd	Usizes	Vsizes
Breast Cancer	0.49	2728.15	3418.85	230, 453	8, 1
Diabetes	0.73	5121.9	1022.09	309, 459	2, 6
Iris	0.17	139.82	460.17	47, 50, 53	1, 3
Soybean	0.21	205.96	484.2	10, 10, 13, 14	12, 23
OECD countries	0.12	72.15	47.84	3, 7, 10	4, 2
Colon	10.27	84631.79	39368.21	18, 44	581, 1419
leukemia	9.83	102681.7	13256.27	17, 21	1374, 1677
SRBCT	25.76	151603.6	39960.37	17, 18, 22, 26	1278, 1030
Synthetic	0.10	12.96	32.03	4, 5, 6	2, 1

Table 6.4: Numerical results using the ALS algorithm.

Data set	time	loop	iter	bcdr	bcdrp	*error*	Usizes	Vsizes	explained var.
Breast Cancer	107.62	329	4	3418.85	79.8	0.07	230, 453	5, 4	39.11, 30.67
Diabetes	276.54	7	14	1022.09	44.5	0.09	309, 459	7, 1	24.96, 12.52
Iris	58.33	566	10	454.5	80.5	0.08	48, 50, 52	3, 1	69.60, 25.17
Soybean	175.76	1	4	2736.31	99.3	0.4	10, 11, 13, 13	14, 21	13.11, 1.31
OECD countries	26.17	98	5	45.98	77.8	0.43	3, 6, 11	3, 3	28.65, 23.18
Colon	3198.53	17	4	39368.21	71.1	4.69	18, 44	1019, 981	23.39, 21.96
leukemia	3388.19	13	5	13256.27	78.4	8.43	17, 21	1559, 1492	7.75, 7.23
SRBCT	12786.95	5	15	25873.22	90.4	4.9	16, 17, 24, 26	1302, 1006	8.75, 6.37
Synthetic	7.04	51	3	31.35	85.6	0.24	4, 5, 6	2, 1	51.47, 35.71

Comparing the results obtained using the Two-Step-SDP algorithm presented in the Table 6.2 and the standard K-means algorithm displayed in the Table 6.3, we can conclude that in general, in terms of computational time and solution, both approaches are quite efficient. For the *leukemia* data set, the K-mean algorithm performs quite faster. Notice that the K-means had to be executed for clustering attributes and objects, thus, the computational time is the sum of the running times of both procedures. With respect to the clusters of attributes, the K-means algorithm does not provide further information on the variance explained by the resulting components, while the Two-Step-SDP algorithm does. It can be observed that in almost all cases, the values of the within and between cluster deviances obtained using the Two-Step-SDP or the K-means algorithms are quite similar. The major difference in the performance of these algorithms is for the *Soybean* data set. The Two-Step-SDP algorithm returned the within cluster deviance equal to 453.12 and the between cluster deviance equal to 2760.87, while the K-means algorithm returned the within cluster deviance equal to 205.96 and the between cluster deviance equal to 484.2. Regarding the clusters of the attributes, it can be observed that the Two-Step-SDP and the K-means algorithms return clusters of equal sizes, with exception for the *Soybean* and *SRBCT* data sets. With respect to the clusters of objects, these algorithms have returned different clusters for the *Iris*, *Soybean* and *SRBCT* data sets.

Comparing the results presented in Tables 6.2 and 6.4, we can conclude that the Two-Step-SDP algorithm is faster than the ALS algorithm. In a couple of iterations, it founds a solution that either yields the same value of the between cluster deviance in the reduced space as that returned by the ALS algorithm, or is very close to it. It can also be observed that the errors of solving the CDPCA model (6.23) using the Two-Step-SDP and the ALS algorithms are very similar. Regarding the results of the clustering of objects, it can be observed that for the *Breast Cancer*, *Diabetes*, *Iris*, *Colon*, *leukemia* and *Synthetic* data sets, the sizes of each cluster coincide for both approaches. With respect to the clusters of attributes, there are some differences, which induce differences in the variance explained by the obtained components. The ALS algorithm provides better results for the *Soybean*, *OECD countries* and *SRBCT* data sets in terms of the value of the between cluster deviance in the reduced space and in the explained variance by the components. It can also be observed that for the *Breast Cancer* and the *Colon* data sets, the variances presented by the first component are, respectively, 62.52% and 37.56% for the Two-Step-SDP algorithm, and 39.11% and 23.39% for the ALS algorithm.

In order to compare the quality of the object clusters obtained using the three algorithms, we present pseudo-confusion matrices, summarizing the (mis)classification of the objects when the true classes are known. In a pseudo-confusion matrix, the rows correspond to the true classes and the columns correspond to the classes returned by algorithms. Notice that the order of the predicted classes may be different to that chosen for the true ones. The analysis of the performance of each algorithm can be complemented by computing its accuracy. Here, we measure the accuracy of an algorithm by the number of the correct predictions of objects divided by the total number of objects.

Table 6.5: Pseudo-confusion matrices for classification of objects of real data sets obtained using the R functions `TwostepSDPClust`, `kmeans` and `CDpca`. The *OECD countries* data is not included, since the true classes of this data set are unknown.

Data set	`TwostepSDPClust`					`kmeans`					`CDpca`				
		Two-Step-SDP					K-means					CDPCA			
	True	1	2	3	4	True	1	2	3	4	True	1	2	3	4
Breast Cancer	1	10	434			1	10	434			1	434	10		
	2	220	19			2	220	19			2	19	220		
Diabetes	1	156	356			1	156	356			1	156	356		
	2	153	103			2	153	103			2	153	103		
Iris	1	50	0	0		1	50	0	0		1	0	0	50	
	2	0	41	9		2	0	11	39		2	9	41	0	
	3	0	11	39		3	0	36	14		3	39	11	0	
Soybean	1	0	4	0	6	1	0	10	0	0	1	5	0	5	0
	2	8	0	2	0	2	0	0	0	10	2	4	0	6	0
	3	0	1	5	4	3	5	0	5	0	3	1	4	0	5
	4	3	8	2	4	4	9	0	8	0	4	0	9	0	8
Colon	1	6	16			1	6	16			1	16	6		
	2	12	28			2	12	28			2	28	12		
leukemia	1	16	11			1	11	16			1	11	16		
	2	1	10			2	10	1			2	10	1		
SRBCT	1	4	13	9	3	1	12	8	6	3	1	7	6	4	12
	2	0	0	8	3	2	0	4	4	3	2	0	4	7	0
	3	0	0	12	6	3	0	3	9	6	3	4	9	5	0
	4	9	3	7	6	4	5	7	7	6	4	13	7	0	5

Considering the pseudo-confusion matrices presented in the Table 6.5, we can conclude that all the approaches to clustering provide the same results in terms of classification of the objects for the *Breast Cancer* data, and the corresponding accuracy is very high, 96%; the *Diabetes* data, presenting the accuracy equal to 66%; the *Colon* data, with the accuracy equal to 55% and the *leukemia* data, with the accuracy equal to 68%. For the *SRBCT* and *Soybean* data sets, all the approaches present different cluster structures. The K-means presents the lowest accuracy for the *SRBCT* data, which is 37%, while the ALS algorithm had the highest accuracy, 47%, that is quite close to that obtained using the Two-Step-SDP algorithm, equal to 45%. For the *Soybean* data, the Two-Step-SDP algorithm yielded an accuracy of 57%, the ALS algorithm had an accuracy of 53% and the

K-means algorithm presented the highest accuracy: 72%. For the *Iris* data set, the Two-Step-SDP and ALS algorithms obtained the same classifications, presenting an accuracy of 87%, while the K-means algorithm returned an accuracy of 83%.

The results of the numerical experiments on these seven real data sets, where the true class of objects is known, show that for four of them, namely, the *Breast Cancer*, *Diabetes*, *Colon*, and *leukemia*, the accuracy of the Two-Step-SDP algorithm is equal to that of the ALS and K-means algorithms. For the *Iris* data set, the Two-Step-SDP and the ALS algorithms provided the best solution on clustering objects with only 20 out of 150 objects misclassified, comparing to the standard K-means algorithm, in which 25 objects were misclassified. The K-means algorithm provided the best solution for the *Soybean* data, with 13 out of 47 objects misclassified, while 20 objects were misclassified using the Two-Step-SDP algorithm. For the *SRBCT* data set, the ALS algorithm provided the best solution, with 42 out of 83 objects misclassified, while the Two-Step-SDP algorithm had misclassified 46 objects. The K-means algorithm returned the worst solution, with 52 misclassified objects.

6.7 Conclusions

In this chapter, we have presented a new approach to clustering and dimensionality reduction based on SDP models. The Two-Step-SDP algorithm is presented for clustering both objects and attributes. The new algorithm is implemented in a easy-to-use software application using R, and we have included tests that show the efficiency of the Two-Step-SDP algorithm. A comparison with other clustering algorithms, namely, the ALS and the K-means algorithms, was performed.

The experiments show that the Two-Step-SDP algorithm is comparable to the K-means algorithm in terms of the execution time, and that the Two-Step-SDP algorithm is significantly faster than the ALS algorithm. Although the computational time of K-means is slightly better than the Two-Step-SDP algorithm in several cases, the K-means algorithm ends up returning less information than the Two-Step-SDP algorithm. In particular, K-means does not provide further information on the partition of the attribute space, namely, about the component loadings and the variance explained by the components, while the Two-Step-SDP algorithm does. We can conclude that the Two-Step-SDP algorithm for clustering and dimensionality reduction can be considered as a modified or simplified version of the ALS algorithm, since it provides almost the same information, but it is much faster in obtaining the assignments, and the loss on the information of the explained variance by the components is quite insignificant.

Finally, it can also be mentioned that another way to obtain more tight approximations for the clustering problem is to develop specific numerical methods to handle the nonlinear SDP-based models. However, it encompasses two main difficulties: the nonlinearity and nonregularity of the model.

Chapter 7

Concluding remarks and future research topics

This thesis studies the phenomenon of regularity in SDP. Regularity is usually associated with notions such as constraint qualifications (CQs), well-posedness and good behaviour of the problems. With the aim of clarifying the notion of regularity, in this work we have studied these notions and established relations. Regarding CQs, we have proved that all the studied CQs are actually equivalent to each other for linear SDP problems, so we focused our study on the Slater condition, a widely used CQ in SDP. We have established relationships between different notions of regularity, in particular, we proved that the Slater condition (and hence, other CQs) is a necessary condition for well-posedness and a sufficient condition for good behaviour. CQs play an important role in duality theory, optimality conditions and sensitivity analysis in SDP. Popular SDP solvers use primal-dual interior point methods, which require that both primal and dual SDP problems satisfy the Slater condition. Although it was proved that the Slater condition is a generic property of linear conic problems, which include SDP, the truth is that one can not guarantee that a given SDP problem satisfies it. In practice, there exist many SDP instances failing the Slater condition (*e.g.*, in our numerical experiments, 78% of the tested SDP instances collected from literature fail the Slater condition, and half of the tested instances from the SDPLIB suite do not satisfy this condition as well). When the Slater condition fails to hold, the solvers run into numerical difficulties and may provide unreliable or wrong solutions. Therefore, it is very important to verify the fulfilment of the Slater condition before solving the SDP problem, but, as far as we know, there is no efficient procedure for doing this. One of the aims of this thesis was to develop a presolving numerical tool to easily verify if a given SDP problem satisfies the Slater regularity condition, warning the users that the computed solution with standard SDP solvers will be trustful or not.

We have developed here two numerical presolving tools, implemented in MATLAB: the `SDPreg` and `DIISalg` routines. We chose to use MATLAB since popular SDP solvers have interface with it. The `SDPreg` just verifies the Slater condition, and `DIISalg` determines the irregularity degree of a given SDP problem. These finite procedures can be applied to any linear SDP problem with constraints in the form of LMI. With the help of simple transformations, the problem in a trace form, can be rewritten as a LMI form as well. We have proposed two numerical approaches to handle the system of quadratic equations that lies in the basis of the SDPreg procedure. To carry out numerical experiments, we used the SDPLIB suite and we have created a collection of small dimension, mostly nonregular, SDP problems found in literature. Our numerical experiments show that the numerical presolving tool `SDPreg` is quite fast and efficient on checking regularity on SDP problems from both literature and SDPLIB collections. Numerous SDP instances were classified as nonregular. Based on our numerical experiments, we conclude that all nonregular problems from SDPLIB present maximal irregularity degree and only two iterations of `DIISalg` were fulfilled. We think that the DIIS algorithm stopped at the second iteration since our computations were performed in nonexact arithmetic, thus, the construction of sets of linearly independent vectors may be compromised. We have compared the numerical results on regularity with others available in the literature regarding well-posedness of SDP problems and can make the following conclusions:

– the presolving procedures `SDPreg` and `DIISalg` developed in this thesis perform well and permit to verify the Slater condition for small to medium sized SDP problems in a reasonable time;

– the numerical experiments confirm the conclusions about the relationships between different notions of regularity in SDP;

– the output of the numerical procedure `DIISalg` permits to develop new numerical methods for solving SDP problems that are not based on the assumption of some CQ.

The main advantage of the developed routines is that one does not need to solve auxiliary SDP problems to test the regularity of a given problem in terms of the Slater condition, while testing well-posedness requires the use of SDP solvers. Our numerical tests on the fulfilment of the Slater condition and the available numerical results of testing well-posedness show the efficiency of the SDPreg procedure.

The results of this part of the work motivate some future research topics. A first direction for further research is to improve the SDPreg and DIIS procedures by implementing a pre-step to verify the feasibility of SDP problems, since at the moment these procedures can only be applied on feasible SDP problems. Another subject not fully addressed in this thesis is to focus on the solution of nonlinear (quadratic) systems of equations by developing a special and more efficient algorithm, and implementing it in our procedures. The other numerical issue is whether there is the need to check linear independence of vectors in the developed MATLAB routine. It would also be important to improve our implementations `SDPreg` and `DIISalg` with respect to accuracy requirements and computation time, specially for large-scale problems. Another interesting topic of research is connected with the development of a CQ-free SDP method that could be safely applied to nonregular SDP problems, based on the optimality criterion that uses the information of the basis of the immobile index subspace returned by our `DIISalg` routine. Although we restricted our study of regularity to SDP, taking into account that some SIP problems can be formulated in terms of LMIs, working on presolving tools to verify regularity in SIP is also a future insight. So, we intend to extend our approach to verify regularity in terms of the Slater condition to other classes of problems.

One more contribution of this thesis consists in the development of a generator of nonregular SDP problem instances. Based on a reformulation of badly-behaved SDP problems and on empirical evidence of several SDP instances, we have developed an algorithm for generating nonregular SDP instances with a pre-specified irregularity degree. There is a lack of available libraries or generators of nonregular SDP instances with particular properties. These are very important tools not only for testing new stopping criteria and SDP methods, but also procedures that permit to check regularity of problems. Here, we have implemented a MATLAB generator of nonregular SDP instances called `nonregSDPgen` and created the NONREGSDP, a collection of 100 nonregular SDP instances of small to moderate sizes and different irregularity degrees. We analysed the behaviour of popular SDP solvers on some instances from NONREGSDP and our numerical results reinforce that further improvements should be developed on solvers in order to handle nonregular problems. The instances from NONREGSDP have a special structure that may be exploited by new solvers. All the SDP instances from NONREGSDP, or generated by `nonregSDPgen`, are in sparse SDPA format, a very common input format in SDP software.

In the final part of the work, we have given a particular attention to a SDP application in data analysis, more specifically in clustering and dimensionality reduction problems. First, we focused on a nonlinear SDP model for clustering and proved that it fails to satisfy the Slater condition, but its linear SDP relaxations are regular. We have proposed a Two-Step-SDP-based approximation algorithmic approach to analyse possibly large data sets. These approach permits to obtain clusters of both objects and attributes, in order to maximize the between cluster deviance in a reduced space. A new algorithm for clustering and dimensionality reduction is developed and called Two-Step-SDP algorithm. It is based on the ALS algorithm for performing CDPCA, an efficient statistical technique to analyse data. We have implemented the new algorithm in a easy-to-use software application using `R` by the function `TwostepSDPClust`. The algorithm was programmed in `R`, because it is a widely used open source software with lots of specific routines for different types of problems. We have compared the performance of the Two-Step-SDP algorithm with that of the ALS and K-means heuristics applied to real data sets, and can conclude that all of them present quite similar solutions. The main conclusion we can make here is that the Two-Step-SDP algorithm appears to be competitive in terms of computational time, and is more complete than K-means in terms of data analysis, since it provides additional important information about the partition of the attribute space, namely, the component loadings and the variance explained by the components. We can also conclude that the Two-Step-SDP algorithm is a modified version of ALS for performing CDPCA. Our method is an efficient tool for data analysis and can be used in different application areas such as economics and marketing, *e.g.*, in product recommendation based on sales and/or costumer choices, telecommunications, *e.g.*, in localization of wireless sensors, computational biology, *e.g.*, in protein structure prediction (very important for drug design) and gene or cancer classification,...

Appendix A

Sparse SDPA format

In what follows, we describe how to create a sparse SDPA (dat-s) file in MATLAB (see also the details in [41]).

Consider the following SDP problem adapted from the Example 3 in [111], which we have used in our tests under the name *patakibadalpha2.dat-s*.

$$\begin{array}{ll} \min & 5x_1 + 8x_1 + 2x_1 \\ \text{s.t.} & x_1 \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -3 \\ 1 & -3 & 8 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 1 & -3 \\ 1 & 0 & 1 \\ -3 & 1 & -6 \end{bmatrix} + x_3 \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -2 \\ -1 & -2 & 2 \end{bmatrix} \preceq \begin{bmatrix} 2 & 2 & -3 \\ 2 & 2 & -4 \\ -3 & -4 & 4 \end{bmatrix}. \end{array}$$

To write the data of the SDP problem in sparse SDPA format and create the corresponding dat-s file in MATLAB, we just need to open a new M-file and, after writing down the data, save it with the extension `dat-s`.

The first line of the file contains a comment and it must be written with quotation marks.

The next three lines contain the number of variables in the SDP problem, the number of diagonal blocks and the dimension of each block.

The fifth line contains the coefficient vector of the objective function.

The remaining lines should contain the nonzero elements on each of the symmetric matrices $\mathbf{A}_i$, $i = 0, 1, ..., n$. Each one of these lines has the form:

$i \;\; b \;\; j \;\; k \;\; value,$

where i is the index of the matrix $\mathbf{A}_i$, b is the number of the block, j and k identify the entry (j, k) of the matrix, and *value* correspond to the element on that entry of the matrix.

We should only include the nonzero elements of the matrices and since they are symmetric, then we only need to store their upper (or lower) triangular part.

The dat-s file corresponding to the above SDP problem is as follows:

"Example patakibadalpha2: mDim = 3, nBLOCK = 1, {3}"

```
3 = mDIM
1 = nBLOCK
3 = bLOCKsTRUCT
{ 5 8 2 } = obj func coefs
0 1 1 1 -2.00
0 1 1 2 -2.00
0 1 2 2 -2.00
0 1 1 3 3.00
0 1 2 3 4.00
0 1 3 3 -4.00
1 1 2 2 1
1 1 1 3 1
1 1 2 3 -3
1 1 3 3 8
2 1 1 2 1
2 1 1 3 -3
2 1 2 3 1
2 1 3 3 -6
3 1 1 1 1
3 1 1 2 1
3 1 2 2 1
3 1 1 3 -1
3 1 2 3 -2
3 1 3 3 2
```

Appendix B

The MATLAB functions SDPreg and DIISalg

The MATLAB function `SDPreg` implements the SDPreg procedure 5 and the function `DIISalg` implements the DIIS algorithm 2 to determine the irregularity degree of SDP problems. Both functions are publicly available in [90]. In what follows, we give a brief guidance for using our functions.

First, the user must copy the functions `SDPreg`, 2, as well as the auxiliary functions `read_data` and `rounddec` (also available in [90]) into a working directory. The former auxiliary function is part of the SDPA project [129] and the latter function was constructed by Peter J. Acklam, but apparently, the site where we have found it is no longer available.

The SDP instance for testing regularity must be in sparse SDPA format and supplied on the working directory.

For testing the regularity in terms of the Slater condition of a given SDP problem using `SDPreg`, the user must specify the following input arguments:

- `SCQ` – a small numerical tolerance for conclusion on regularity, by default `SCQ`=10^{-4},
- `tolparam` – a small numerical tolerance for the solvers, by default `tolparam`=10^{-8}.

After specifying these arguments, the `SDPreg` function can be called as follows:

```
>>SDPreg(SCQ,tolparam)
```

Next, two options, 1 and 2, for specifying the numerical approach to use in the regularity test will appear. The user must choose one of them. Then, the function provides information on the regularity of the tested SDP problem and the computation time as outputs.

With respect to the use of the `DIISalg` function, the user must specify the same above input arguments.

After calling the function using

```
>>DIISalg(SCQ,tolparam)
```

two options will appear. Here, the user must choose one of them by selecting 1 or 2. Notice that only the first iteration may use the option 2. All the remaining iterations will use the option 1.

Then, the following output parameters are provided:

- **M**, a basis of the immobile index subspace,
- s^*, the irregularity degree of the problem,
- information on regularity of the SDP problem,
- number of iterations of the DIIS algorithm,
- computation time.

Appendix C

The MATLAB function `nonregSDPgen`

The MATLAB function for generating nonregular SDP instances `nonregSDPgen` is publicly available at [90].

The user only needs to specify the following parameters:

- `n` – the number of variables in the SDP problem,
- `s` – the dimension of the symmetric constraint matrices,
- `d` – the desired irregularity degree, which should take integer values from 1 to s-1,
- `'examplename.dat-s'` – where *examplename* is a name for saving the generated data in sparse SDPA format (dat-s).

Then, one can call the generator as follows:

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

As output, the generator returns a dat-s file with the SDP problem data. The generated SDP instance is known to have zero optimal value and fails the Slater condition, having an irregularity degree equal to `d`.

Appendix D

NONREGSDP: a collection of nonregular SDP test problems

The NONREGSDP is a collection of nonregular linear SDP instances created using the generator `nonregSDPgen`. These SDP test problems failing the Slater condition are stored in sparse SDPA format, which is a common input data format for SDP solvers. The NONREGSDP database is publicly available at [90].

The nonregular SDP problems from NONREGSDP have the form (2.2). Currently, the NONREGSDP database contains a total of 100 SDP instances of different values of n and s, with n varying from 1 to 12, s from 1 to 29, and with irregularity degree d varying from 1 up to 29. These problems have optimal value p^* equal to zero. All the problems of the NONREGSDP database were tested using the DIISalg procedure to confirm their nonregularity and irregularity degrees.

Table D.1: SDP instances from the NONREGSDP database and the values of their irregularity degrees computed with `DIISalg`.

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

Appendix E

The R function `TwostepSDPClust`

The R function `TwostepSDPClust` implements the Two-Step-SDP algorithm 13 and is available in [90]. The user must specify the following input arguments:

- `data` – the numeric data matrix,
- `p` – the number of clusters of objects,
- `k` – the number of clusters of attributes,
- `class` – the vector of integers with the true classification of objects, or 0,
- `tol` – a small convergence tolerance value,
- `maxit` – the maximum number of iterations.

The R function `TwostepSDPClust` returns the following information:

- `Dscale` – the scaled data matrix,
- `U` – the object assignment matrix,
- `cluster` – the vector of integers identifying the clusters of objects,
- `Usizes` – the vector whose coordinates are the sizes of each cluster of objects,
- `Z0` – the approximate solution of the SDP-based problem for clustering objects,
- `Z` – the solution to the SDP-based problem for clustering objects,
- `bcdr` – the between cluster deviance in the reduced space of the k components,
- `bcdrp` – the between cluster deviance of the total variance,
- `wssd` – the within cluster deviance for clustering of objects,
- `ofrelaxZ` – the value of the objective function of the SDP problem (6.16),

- `iter` – the number of iterations in the refinement solution step,
- `V` – the attribute assignment matrix,
- `H0` – the approximate solution of the SDP-based problem for clustering attributes,
- `H` – the solution of the SDP-based problem for clustering attributes,
- `Vsizes` – the vector with the sizes of each cluster of attributes,
- `wssdat` – the within cluster deviance for clustering of attributes,
- `bcdat` – the between cluster deviance for clustering of attributes,
- `Ybar` – the object centroid matrix in the reduced space,
- `Dbar` – the object centroid matrix,
- `Y` – the component score matrix,
- `A` – the component loading matrix,
- `explvar` – the vector of explained variance of each component,
- `E` – the error associated to the CDPCA model,
- `compt` – the computational time,
- `pcm` – the pseudo-confusion matrix, when the true classes are known.

After specifying the input arguments in the R environment, the basic call of the function `TwostepSDPClust` is as follows:

```
> example <- TwostepSDPClust(data,p,k,class,tol,maxit)
```

To access the output matrices, say the component loading matrix `A`, one just need to write

```
> example$A
```

Bibliography

[1] Abadir, K.M. and Magnus, J.R., *Matrix Algebra*, Cambridge University Press, 2005.

[2] Ackerman, M. and Ben-David, S., *Clusterability: A theoretical study*, Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, JMLR: W&CP, **5**, pp. 1-8, 2009.

[3] Akteke-Ozturk, B., Weber, G.-W. and Kropat, E., *Continuous Optimization Approaches for Clustering via Minimum Sum of Squares*, Proc. 20th Mini-EURO Conf. Continuous Optimization and Knowledge-Based Technologies, Lithuania, pp. 253-258, 2008.

[4] Alizadeh, F., *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM J.Opt., **5**, pp. 13-51, 1995.

[5] Alizadeh, F., Haeberly, J.-P.A. and Overton, M.L, *Complementarity and nondegeneracy in semidefinite programming*, Mathematical Programming, **77**(1), Springer, pp. 111-128, 1997.

[6] Aloise, D. and Hansen, P., *A branch-and-cut sdp-based algorithm for minimum sum-of-squares clustering*, Pesquisa Operacional, **29**(3), pp. 503-516, 2009.

[7] Ames, B.P.W., *Guaranteed clustering and biclustering via semidefinite programming*, Mathematical Programming, **147**(1-2), Springer Berlin Heidelberg, pp. 429-465, 2014.

[8] 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.

[9] Bagirov, A.M., *Modified global k-means algorithm for minimum sum-of-squares clustering problems*, Pattern Recognition, **41**, pp. 3192-3199, 2008.

[10] Bagirov, A.M., Rubinov, A.M., Soukhoroukova, N.V. and Yearwood, J., *Unsupervised and Supervised Data Classification via Nonsmooth and Global Optimization*, TOP, **11**(1), pp. 1-75, 2003.

[11] Bellman, R. and Fan, K., *On systems of linear inequalities in Hermitian matrix variables*, In: Convexity, Proceedings of Symposia in Pure Mathematics, **7**, American Mathematical Society, Providence, RI, pp. 1-11, 1963.

[12] Benson, S.J. and Ye, Y., *DSDP5 user guide - software for semidefinite programming*, Technical Report, Mathematics and Computer Science Division, Argonne National Laboratory, 2005.

[13] Bonnans, J.F. and Shapiro, A., *Perturbation Analysis of Optimization Problems*, Springer-Verlag, New-York, 2000.

[14] Borchers, B., *CSDP: a C library for semidefinite programming*, Optimization Methods and Software, **11**(1-4), pp. 613-623, 1999.

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

[16] Borchers, B. and Young, J., *Implementation of a Primal-Dual Method for SDP on a Shared Memory Parallel Architecture*, Comp. Opt. Appl., **37**, pp. 355-369, 2007.

[17] Borwein, J.M. and Wolkowicz, H., *Facial reduction for a cone-convex programming problem*, J. Austral. Math. Soc. Ser. A, **30**(3), pp. 369-380, 1980/81.

[18] Bronson, R. and Costa, G.B., *Matrix Methods: Applied Linear Algebra*, 3rd Ed., Academic Press, Elsevier, 2009.

[19] Burer, S., *SDPlr code*, available at `http://sburer.github.io/projects.html`, 2009.

[20] Burer, S. and Monteiro, R.D.C., *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., Ser. B, **95**, pp. 329-357, 2003.

[21] Burer, S., Monteiro, R.D.C. and Zhang, Y., *Solving a class of semidefinite programs via nonlinear programming*, Mathematical Programming, **93**, pp. 97-122, 2002.

[22] Cheung, Y., *Preprocessing and Reduction for Semidefinite Programming via Facial Reduction: Theory and Practice*, PhD Thesis, University of Waterloo, Canada, 2013.

[23] 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.

[24] Craven, B. and Mond, B., *Linear programming with matrix variables*, Linear Algebra Appl., **38**, pp. 73-80, 1981.

[25] CSDP solver `https://projects.coin-or.org/Csdp`

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

[27] d'Aspremont, A., El Ghaoui, L., Jordan, M.I., Lanckriet, G.R.G. *A Direct Formulation for Sparse PCA Using Semidefinite Programming*, SIAM, **49**(3), pp. 434-448, 2007.

[28] Donath, W.E. and Hoffman, A.J., *Lower bounds for the partitioning of graphs*, IBM J. of Research and Development, **17**, pp. 420-425, 1973.

[29] Dontchev, A. L., Zolezzi, T., *Well-Posed Optimization Problems*, in Lecture Notes in Mathematics, **1543**, Springer-Verlag, Berlin, 1993.

[30] DSDP site `http://www.mcs.anl.gov/hs/software/DSDP/`

[31] Dür, M., Jargalsaikhan, B. and Still, G., *The Slater condition is generic in linear conic programming*, Nov 2012, available at `http://www.optimization-online.org/DB_FILE/2012/11/3675.pdf`

[32] Enki, D.G., Trendafilov, N.T. and Jolliffe, I.T., *A clustering approach to interpretable principal components*, Journal of Applied Statistics, **40**(3), pp. 583-599, 2013.

[33] Fajardo, M.D. and López, M.A., *Some results about the facial geometry of convex semi-infinite systems*, Optimization, **55**(5-6), pp. 661-684, 2006.

[34] Fiacco, A.V., *Sensitivity and Stability in NLP*, Floudas, C. A. and Pardalos, P. M. (Eds.): Encyclopedia of Optimization, Second Edition, Springer, pp. 3450-3454, 2009.

[35] Fiala, J., Kočvara, M. and Stingl, M., *PENLAB: A MATLAB solver for nonlinear semidefinite optimization*, Nov 2013, available at `http://arxiv.org/pdf/1311.5240v1.pdf`

[36] Fletcher, R., *Semidefinite matrix constraints in optimization*, SIAM J. Control Optim., **23**, pp. 493-513, 1985.

[37] Freund, R.M., *Complexity of an Algorithm for Finding an Approximate Solution of a Semi-Definite Program, with no Regularity Condition*, Technical Report OR 302-94, Op. Research Center, MIT, 1994, Revised December 1995.

[38] 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.

[39] Freund, R.M. and Sun, J., *Semidefinite Programming I: Introduction and minimization of polynomials*, System Optimization, available at `http://www.myoops.org/cocw/mit/NR/rdonlyres/Sloan-School-of-Management/15-094Systems-Optimization--Models-and-ComputationSpring2002/1B59FD11-A822-4C80-9301-47B127500648/0/lecture22.pdf`, 2002.

[40] Freund, R.M. and Vera, J.R., *Some characterization and properties of the "distance to ill-posedness" and the condition measure of a conic linear system*, Mathematical Programming, **86**(2), pp. 225-260, 1999.

[41] Fujisawa, K., Futakata, Y., Kojima, M., Matsuyama, S., Nakamura, S., Nakata, K. and Yamashita, M., *SDPA-M (SemiDefinite Programming Algorithm in MATLAB) User's Manual-Version 6.2.0*, Series B: Operations Research Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, May 2005.

[42] Gärtner, B. and Matoušek, J., *Approximation Algorithms and Semidefinite Programming*, Springer-Verlag, 2012.

[43] Goemans, M.X. and Williamson, D.P., *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming*, J. ACM, **42**(6), pp. 1115-1145, 1995.

[44] Grant, M. and Boyd, S., *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control, Lecture Notes in Control and Information Sciences Series, eds. Blondel, V., Boyd, S. and Kimura, H., Springer-Verlag Limited, pp. 95-110, 2008, available at `http://stanford.edu/~boyd/graph_dcp.html`

[45] Grant, M. and Boyd, S., *The CVX Users' Guide, Release 2.1*, 2015, available at `http://cvxr.com/cvx/doc/CVX.pdf`

[46] Grötschel, M., Lovász, L. and Schrijver, A., *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[47] Gruber, G., *On Semidefinite Programming and Applications in Combinatorial Optimization*, PhD Thesis, University of Technology, Graz, Austria, 2000.

[48] 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.

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

[50] Hastie, T., Tibshirani, R. and Friedman, J., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer Series in Statistics, 2009.

[51] Helmberg, C., *Semidefinite programming*, European Journal of Operational Research **137**, pp. 461-482, 2002.

[52] Helmberg, C., *Semidefinite Programming for Combinatorial Optimization*, ZIB Report, Berlin, 2000.

[53] Helmberg, C., Rendl, F., *A Spectral Bundle Method for Semidefinite Programming*, SIAM Journal on Optimization, **10**, pp. 673-696, 1997.

[54] Helmberg, C., Rendl, F., Vanderbei, R.J. and Wolkowicz, H., *An Interior-Point Method for Semidefinite Programming*, SIAM J. Optim., **6**(2), pp. 342-361, 1996.

[55] 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.

[56] Horn, R.A. and Johnson, C.R., *Matrix Analysis*, 2nd edn., Cambridge University Press, 2012.

[57] Huang, A. and Xu, C., *A trust region method for solving semidefinite programs*, Springer, Comput. Optim. Appl., **55**, pp. 49-71, 2013.

[58] Jain, A.K., *Data Clustering: 50 Years Beyond K-means*, Pattern Recogn. Lett. **31**(8), Elsevier Science Inc., pp. 651-666, 2010.

[59] Jansson, C., *On Verified Numerical Computations in Convex Programming*, Springer-Verlag, Japan J. Indust. Appl. Math., **26**(2-3), pp. 337-363, 2009.

[60] Jansson, C., *VSDP: a Matlab software package for Verified Semidefinite Programming*, NOLTA, pp. 327-330, 2006.

[61] 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.

[62] Jeyakumar, V. and Dihn, N., *Avoiding Duality Gaps in Convex Semidefinite Programming without Slater's Condition*, Applied Mathematics Report AMR04/6, University of New SouthWales, Sydney, Australia, 2004.

[63] Jeyakumar, V. and Nealon, M.J., *Complete Dual Characterizations of Optimality for Convex Semidefinite Programming*, in Théra, M. (ed) "Constructive, Experimental and Nonlinear Analysis", CMS Conference Proceedings American Mathematical Society, **27**, pp. 165-174, 2000.

[64] Jolliffe, I.T., *Principal Component Analysis*, Second edition, Springer-Verlag, New York, 2002.

[65] Jolliffe, I.T., Trendafilov, N.T., Uddin, M., *A modified principal component technique based on the lasso*, J. of Computational and Graphical Statistics, **12**(3), pp. 531-547, 2003.

[66] Kanzow, C. and Nagel, C., *Semidefinite programs: new search directions, smoothing-type methods, and numerical results*, SIAM Journal on Optimization, **13**(1), pp. 1-23, 2002.

[67] Karmarkar, N.K., *A new polynomial-time algorithm for linear programming*, Combinatorica, **4**, pp. 373-395, 1984.

[68] Klatte, D., *First Order Constraint Qualifications*, Floudas, C. A. and Pardalos, P. M. (Eds.): Encyclopedia of Optimization, 2nd edn, Springer, pp. 1055-1060, 2009.

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

[70] Klerk, E. de, Roos, C. and Terlaky, T., *A short survey on semidefinite programming*, Ten Years LNMB, PhD Research and Graduate Courses of the Dutch Network of Operations Research (W.K.K.H. et al., ed.), **122**, Amsterdam, The Netherlands: Centrum for Mathematics and Informatics (CWI), pp. 323-339, 1997.

[71] Kočvara, M. and Stingl, M., *On the solution of large-scale SDP problems by the modified barrier method using iterative solvers*, Mathematical Programming, **109**(2-3), pp. 413-444, 2007.

[72] Kočvara, M. and Stingl, M., *PENNON - A Code for Convex Nonlinear and Semidefinite Programming*, Optimization Methods and Software **18**(3), pp. 317-333, 2003.

[73] Kogan, J., Nicholas, C., Teboulle, M. (Eds.), *Grouping Multidimensional Data: Recent Advances in Clustering*, Springer, XII, 2006.

[74] Kojima, M., *Introduction to Semidefinite Programs (Semidefinite Programming and Its Application)*, Institute for Mathematical Sciences National University of Singapore, 2006.

[75] Kojima, M., Kojima, S. and Hara, S., *Linear algebra for semidefinite programming*, Technical report, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1994.

[76] Kolman, B. and Beck, R.E., *Elementary Linear Programming with Applications* , 2nd edn, Academic Press, San Diego, 1995.

[77] Konsulova, A.S. and Revalski,J.P., *Constrained convex optimization problems - well-posedness and stability*, Numerical Functional Analisys and Optimization, **15**(7-8), pp. 889-907, 1994.

[78] 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.

[79] Krishnan, K., *Linear Programming Approaches to Semidefinite Programming problems*, PhD thesis, Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, July 2002.

[80] Krishnan, K. and Mitchell, J., *Semi-Infinite Linear programming approaches to SDP problems*, In: Novel approaches to hard discrete optimization problems, edited by Pardalos, P. and Wolkowicz, H., Fields Institute Communications Series, AMS, **37**, pp. 123-142, 2003.

[81] Kulis, B., Surendran, A.C. and Platt, J.C., *Fast low-rank semidefinite programming for embedding and clustering*, in Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, pp. 235-242, 2007.

[82] Kuhn, H.W. and Tucker, A.W., *Nonlinear programming*, in Neyman, J., ed. Proceedings of Second Berkeley Symposium of Mathematical Statistics and Probability, University of California Press, Berkeley, pp. 481-492, 1950.

[83] Laurent, M. and Rendl, F., *Semidefinite programming and integer programming*, In Nemhauser, G., Aardal, K. and Weismantel, R., editors, Handbook on Discrete Optimization, Elsevier, pp. 393-514, 2005.

[84] Laurent, M. and Vallentin, F., *Semidefinite Optimization*, Lecture Notes, available at `http://page.mi.fu-berlin.de/fmario/sdp/laurentv.pdf`, 2012.

[85] Liu, M. and Pataki, G., *Exact duals and short certificates of infeasibility and weak infeasibility in conic linear programming*, July 2015, available at `http://arxiv.org/pdf/1507.00290v1.pdf`

[86] Lovász, L., *On the Shannon capacity of a graph*, IEEE Transactions on Information Theory, **25**(1), pp. 1-7, 1979.

[87] Luo, Z., Sturm, J. and Zhang, S., *Duality results for conic convex programming*, Econometric Institute Report No. 9719/A, 1997.

[88] Ma, Z., *Sparse principal component analysis and iterative thresholding*, The Annals of Statistics **41**(2), pp. 772-801, 2013.

[89] Macedo, E., *Estudo prático de regularidade de problemas de Programação Semidefinida*, MSc. Thesis, University of Aveiro, 2010.

[90] Macedo, E., `https://sites.google.com/site/macedoelocat/`

[91] 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.

[92] Macedo, E., *Two-Step-SDP Approach to Clustering and Dimensionality Reduction*, Stat., Optim. Inf. Comput., **3**(3), pp. 294-311, 2015.

[93] Macedo, E. and Freitas, A., *Statistical Methods and Optimization in Data Mining*, In: III International Conference of Optimization and Applications OPTIMA2012, pp. 164-169, 2012.

[94] Macedo, E. and Freitas, A., *The Alternating Least-Squares Algorithm for CDPCA*, In: Plakhov, A. et al (eds.) Optimization in the Natural Sciences, Communications in Computer and Information Science (CCIS), Springer, **499**, pp. 173-191, 2015.

[95] Macedo, E. and Sá Esteves, J., *A Least-Squares Approach for Testing the Slater Condition in Semidefinite Programs*, Proceedings of the 12th International Conference on Computational and Mathematical Methods in Science and Engineering - CMMSE2012, July 2-5, Múrcia, Spain, **2**, pp. 773-784, 2012.

[96] Malick, J., Povh, J., Rendl, F. and Wiegele, A., *Regularization methods for semidefinite programming*, SIAM Journal on Optimization, **20**(1), pp. 336-356, 2009.

[97] Mangasarian, O.L. and Fromovitz, S., *The Fritz-John necessary optimality conditions in presence of equality and inequality constraints*, J. Math. Anal. Appl. **17**, pp. 37-47, 1967.

[98] Meyer, C., *Matrix analysis and applied linear algebra*, Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[99] Minchenko, L. and Stakhovski, S., *On relaxed constant rank regularity condition in mathematical programming*, Optimization, **60**(4), pp. 429-440, 2011.

[100] Mitchell, J. and Krishnan, K., *A unifying framework for several cutting plane methods for semidefinite programming*, Technical Report, Dept. of Computational and Applied Mathematics, Rice University, December 2003.

[101] Mittelmann, H.D., *The State-of-the-Art in Conic Optimization Software*, in Anjos, M.F. and Lasserre, J.B. (eds) Handbook on Semidefinite, Conic and Polynomial Optimization, International Series in Operations Research & Management Science, **166**, Springer, pp. 671-686, 2012.

[102] Moldovan, A. and Pellegrini, L., *On Regularity for Constrained Extremum Problems Part 1: Sufficient Optimality Conditions*, JOTA, **142**(1), pp. 147-163, 2009.

[103] Monteiro, R.D.C., *First- and second-order methods for semidefinite programming*, Math. Program., Ser. B **97**, pp. 209-244, 2003.

[104] Monteiro, R.D.C. and Tsuchiya, T., *Polynomial convergence of a new family of primal-dual algorithms for semidefinite programming*, SIAM J. OPTIM., **9**(3), pp. 551-577, 1999.

[105] Mosek ApS, available at `http://www.mosek.com`

[106] Nayakkankuppam, M.V. and Overton, M.L., *Conditioning of semidefinite programs*, Math. Program., **85**, Springer, pp. 525-540, 1999.

[107] Nesterov, Y.E. and Nemirovski, A.S., *Conic formulation of a convex programming problem and duality*, Optim. Methods Software, **1**(2), pp. 95-115, 1992.

[108] Nesterov, Y.E. and Nemirovski, A.S., *Interior Point Polynomial Algorithms in Convex Programming*, SIAM, Philadelphia, USA, 1994.

[109] Nocedal, J. and Wright, S.J., *Numerical Optimization*, Springer, 1999.

[110] Overton, M.L. and Womersley, R.S., *Optimality Conditions and Duality Theory for Minimizing Sums of the Largest Eigenvalues of Symmetric Matrices*, Mathematical Programming, **62**, pp. 321-357, 1993.

[111] Pataki, G., *Bad semidefinite programs: they all look the same*, available at `http://arxiv.org/pdf/1112.1436.pdf`, july, 2014.

[112] Pataki, G., *Strong Duality in Conic Linear Programming: Facial Reduction and Extended Duals*, in Bailey, D.H. et al, Springer Proceedings in Mathematics & Statistics, Computational and Analytical Mathematics, **50**, Springer New York, pp. 613-634, 2013.

[113] Pataki, G. and Tunçel, L., *On the generic properties of convex optimization problems in conic form*, Math. Program., Ser. A **89**, Springer, pp. 449-457, 2001.

[114] Pedregal, P., *Introduction to Optimization*, Springer, 2004.

[115] Peng, J. and Wei, Y., *Approximating k-means-type clustering via semidefinite programming*, SIAM J. OPTIM., **18**(1), pp. 186-205, 2007.

[116] Peng, J. and Xia, Y., *A New Theoretical Framework for K-Means-Type Clustering*, Foundations and Advances in Data Mining Studies in Fuzziness and Soft Computing, **180**, pp. 79-96, 2005.

[117] Petrov, Y.P. and Sizikov, V.S., *Well-posed, Ill-posed, and Intermediate Problems with Applications (Inverse and Ill-Posed Problems)*, Brill Academic Publishers, 2005.

[118] Pintér, J.D., *LGO - A Model Development and Solver System for Nonlinear (Global and Local) Optimization, Users Guide*, Distributed by Pintér Consulting Services, Inc., Canada, `http://www.pinterconsulting.com/`, 2014.

[119] Pinto Da Costa, J.F., Alonso, H. and Roque, L., *A weighted principal component analysis and its application to gene expression data*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, **8**(1), pp. 246-252, 2011.

[120] Polik, I., *Conic optimization software*, Wiley Encyclopedia of Operations Research and Management Science, 2010.

[121] Polik, I., *Semidefinite programming Feasibility and duality*, available at `http://imre.polik.net/wp-content/uploads/IE496/POLIK_IE496_04_duality.pdf`, 2009.

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

[123] Pompili, F., Gillis, N, Absil, P.A. and Glineur, F., *Two algorithms for orthogonal nonnegative matrix factorization with application to clustering*, Neurocomputing **141**, pp. 15-25, 2014.

[124] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, `http://www.R-project.org`, 2015.

[125] Ramana, M.V., *An Exact Duality Theory for Semidefinite Programming and its Complexity Implications*, DIMACS Technical report 95-02R, RUTCOR, Rutgers University, New Brunswick, NJ, 1995.

[126] Ramana, M.V., Tunçel, L. and Wolkowicz, H., *Strong Duality for Semidefinite Programming*, SIAM J. Optimization, **7**(3), 1997.

[127] Renegar, J., *Some Perturbation-Theory for Linear-Programming*, Mathematical Programming, **65**(1) pp.73-91, 1994.

[128] Robinson, S.M., *First order conditions for general nonlinear optimization*, SIAM J. Appl. Math., **30**(4), pp. 597-607, 1976.

[129] SDPA Project available at `http://sdpa.sourceforge.net/`

[130] SDPT3 site `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`

[131] SeDuMi site `http://sedumi.ie.lehigh.edu/?page_id=58`

[132] Shapiro, A., *Directional differentiability of the optimal value function in convex semi-infinite programming*, Mathematical Programming **70**, pp. 149-157, 1995.

[133] Shores, T., *Applied Linear Algebra and Matrix Analysis*, Undergraduate Texts in Mathematics, Springer, 2007.

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

[135] Stingl, M., *On the Solution of Nonlinear Semidefinite Programs by Augmented Lagrangian Methods*, PhD thesis, University of Erlangen-Nürnberg, 2006.

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

[137] Sturm, J.F. and Zhang, S., *On Sensitivity of Central Solutions in Semidefinite Programming*, Mathematical Programming, **90**(2), Springer, pp. 205-227, 2001.

[138] Tikhonov, A.N. and Arsenin, V.Y., *Solutions of ill-posed problems*, John Wiley and Sons, 1977.

[139] Todd, M.J., *A study of search directions in primal-dual interior-point methods for semidefinite programming*, Optimization Methods and Software, **11**(1-4), pp. 1-46, 1999.

[140] Todd, M.J., *Semidefinite Optimization*, in Acta Numerica, Cambridge University Press, Cambridge, UK, **10**, May, pp. 515-560, 2001.

[141] Tunçel, L., *Some Applications of Semidefinite Optimization from an Operations Research Viewpoint*, Iranian Journal of Operations Research **1**(2), pp. 1-29, 2009.

[142] Tunçel, L. and Wolkowicz, H., *Strong duality and minimal representations for cone optimization*, Computational Optimization and Applications, **53**(2), Springer US, pp. 619-648, 2012.

[143] 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.

[144] UCI Repository: http://archive.ics.uci.edu/ml/

[145] Vandenberghe, L. and Boyd, S., *A Primal-Dual Potential Reduction Method for Problems Involving Matrix Inequalities*, Mathematical Programming, Series B, **69**(1), pp. 205–236, 1995.

[146] Vandenberghe, L. and Boyd, S., *Applications of Semidefinite Programming*, Applied Numerical Mathematics, **29**(3), pp. 283-299, 1999.

[147] Vandenberghe, L. and Boyd, S., *Connection between Semi-Infinite and Semidefinite Programming*, in Reemtsen, R. and Ruckmann, J.J., Eds, chapter 8 of *Semi-Infinite Programming*, Kluwer Academic Publishers, pp. 277-294, 1998.

[148] Vandenberghe, L. and Boyd, S., *Convex Optimization*, Cambridge University Press, 2009.

[149] Vandenberghe, L. and Boyd, S., *Semidefinite Programming*, SIAM REVIEW, **38**(1), pp. 49-95, 1996.

[150] Vichi, M. and Kiers, H.A.L., *Factorial k-means analysis for two-way data*, Computational Statistics & Data Analysis, **37**(1), pp. 49-64, 2001.

[151] Vichi, M. and Saporta, G., *Clustering and Disjoint Principal Component Analysis*, Computational Statistics & Data Analysis **53**, pp. 3194-3208, 2009.

[152] 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.

[153] Weber, G.-W., Taylan, P., Ozogur, S. and Akteke-Ozturk, B., *Statistical Learning and Optimization Methods in Data Mining*, in Ayhan, H. O. and Batmaz, I. (eds.): Recent Advances in Statistics, Turkish Statistical Institute Press, Ankara, pp. 181–195, 2007.

[154] Wei, H., *Numerical Stability in Linear Programming and Semidefinite Programming*, PhD Thesis, University of Waterloo, Ontario, Canada, 2006.

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

[156] Wolkowicz, H., *Duality for semidefinite programming*, edited by Floudas, C. A. and Pardalos, P. M., in Encyclopedia of Optimization, Springer, 2nd edn, pp. 811-813, 2009.

[157] Wolkowicz, H., *Semidefinite Programming*, Research Report CORR 2002-04, 2002, available at `http://www.math.uwaterloo.ca/~hwolkowi/henry/reports/sdpstats.pdf`

[158] Wolkowicz, H., *Some applications of optimization in matrix theory*, Linear Algebra and Appl. **40**, pp. 101-118, 1981.

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

[160] Xu, R. and Wunsch, D., *Survey of Clustering Algorithms*, IEEE Transactions on Neural Networks, **16**(3), pp. 645-648, 2005.

[161] Yamashita, M., Fujisawa, K., Fukuda, M., Nakata, K., and Nakata, M., *Algorithm 925: Parallel solver for semidefinite programming problem having sparse Schur complement matrix*, ACM Transactions on Mathematical Software, **39**(1), Article 6 (November), 2012.

[162] 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.

[163] Yanai, H., Takeuchi, K. and Takane, Y., *Projection Matrices, Generalized Inverse Matrices, and Singular Value Decomposition*, Statistics for Social and Behavioral Sciences, Springer, 2011.

[164] Zhang, Y., *Semidefinite Programming*, Lecture 2, available at `http://rutcor.rutgers.edu/~alizadeh/CLASSES/95sprSDP/NOTES/lecture2.ps`, 1995.

[165] Zhao, X.-Y., Sun, D. and Toh, K.-C., *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM Journal on Optimization, **20**(4), pp. 1737-1765, 2010.

[166] Zou, H., Hastie, T., Tibshirani, R., *Sparse principal component analysis*, J. of Computational and Graphical Statistics, **15**(2), pp. 262-286, 2006.

Index