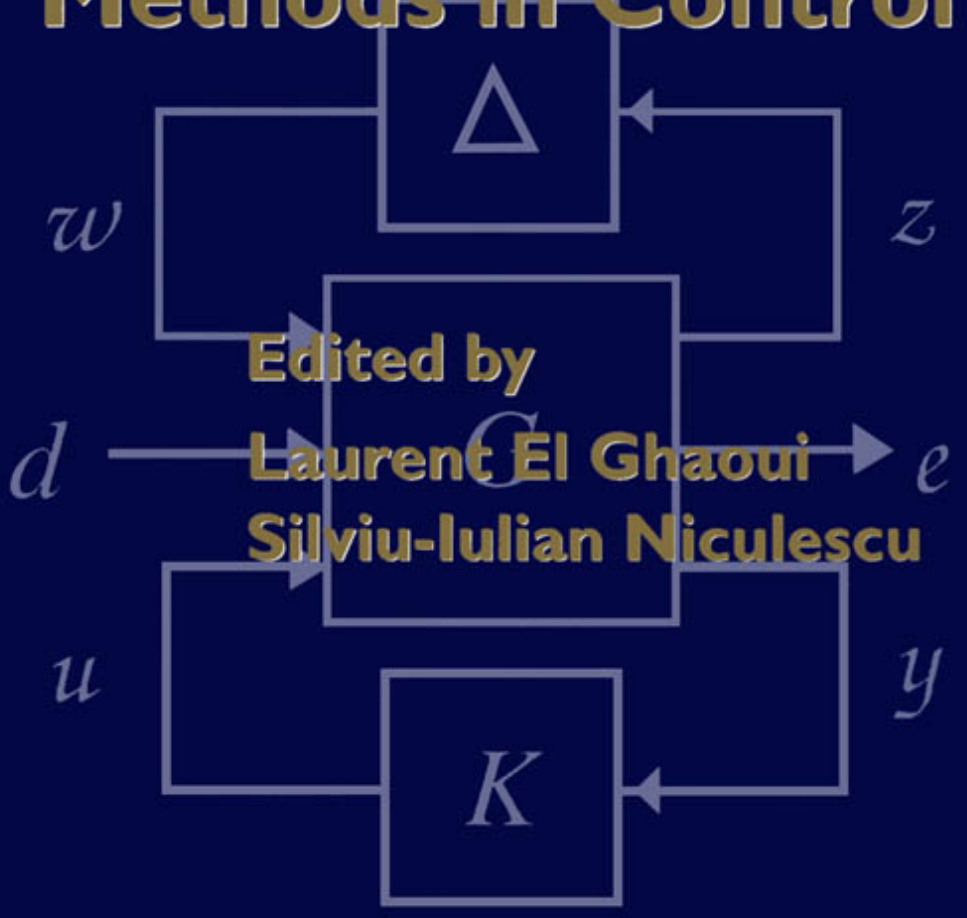
Advances in Linear Matrix Inequality Methods in Control



Edited by
Laurent El Ghaoui
Silviu-Iulian Niculescu

Advances in Design and Control
siam

Advances in Linear Matrix Inequality Methods in Control

Advances in Design and Control

SIAM's Advances in Design and Control series consists of texts and monographs dealing with all areas of design and control and their applications. Topics of interest include shape optimization, multidisciplinary design, trajectory optimization, feedback, and optimal control. The series focuses on the mathematical and computational aspects of engineering design and control that are usable in a wide variety of scientific and engineering disciplines.

Series Volumes
El Ghaoui, Laurent and Niculescu, Silviu-Iulian, eds., *Advances in Linear Matrix Inequality Methods in Control*
Helton, J. William and James, Matthew R., *Extending H^{∞} Control to Nonlinear Systems: Control of Nonlinear Systems to Achieve Performance Objectives*

Advances in Linear Matrix Inequality Methods in Control

Edited by
Laurent El Ghaoui
University of California, Berkeley

Silviu-Iulian Niculescu
Université de Technologie de Compiègne
Compiègne, France

siam
Society for Industrial and Applied Mathematics
Philadelphia

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data
Advances in linear matrix inequality methods in control / edited by
Laurent El Ghaoui, Silviu-Iulian Niculescu.
p.cm. -- (Advances in design and control)
Includes bibliographical references.
ISBN 0-89871-438-9 (pbk.)
1. Control theory. 2. Matrix inequalities. 3. Mathematical optimization. I. El Ghaoui, Laurent. II. Niculescu, Silviu-Iulian. III. Series.
QA402.3.A364 2000
629.8--dc21

99-39162

Contributors

R. J. Adams	Electrical Engineering Department, University of Washington, Seattle, WA 98195 Internet: rjadams@uwashington.edu
C. Andriot	Service Téléopération et Robotique, CEA, route du Panorama, 92 265 Fontenay aux Roses, France Internet: andriot@CYBORG.cea.fr
P. Apkarian	CERT/DERA, 2 Av. Ed. Belin, 31055 Toulouse, France Internet: apkarian@cert.fr
V. Balakrishnan	School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907-1285 Internet: ragu@ecn.purdue.edu
E. B. Beran	Axapta Financials Team, Damgaard International A/S Bregneroedvej 133, DK-3460 Birkeroed, Denmark Internet: ebb@dk.damgaard.com
S. Boyd	Information Systems Laboratory, Electrical Engineering Department, Stanford University, Stanford, CA 94305 Internet: boyd@stanford.edu
S. Dasgupta	Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242 Internet: soura-dasgupta@uiowa.edu
C. E. de Souza	Laboratório Nacional de Computação Científica–LNCC, Department of Systems and Control, Av. Getulio Vargas, 333, 25651-070 Petrópolis, RJ, Brazil Internet: csouza@lncc.br
S. Dussy	Aérospatiale-Matra Lanceurs 66, route de Verneuil, 78133 Les Mureaux Cédex, France Internet: stephane.dussy@espace.aerospatiale.fr
L. El Ghaoui	Laboratoire de Mathématiques Appliquées, ENSTA, 32, Bvd. Victor, 75739 Paris Cédex 15, France Internet: elghaoui@ensta.fr
E. Feron	Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 Internet: feron@mit.edu

J. P. Folcher	Laboratoire de Mathématiques Appliquées, ENSTA, 32, Bvd. Victor, 75739 Paris Cédex 15, France Internet: folcher@ensta.fr
M. Fu	Department of Electrical and Computer Engineering, The University of Newcastle, Newcastle, NSW 2308, Australia Internet: eemf@ee.newcastle.edu.au
K. M. Grigoriadis	Department of Mechanical Engineering, University of Houston, Houston, TX 77204 Internet: karolos@uh.edu
J.-P. A. Haeberly	Mathematics Department, Fordham University, Bronx, NY 10458 Internet: haeberly@murray.fordham.edu
S. R. Hall	Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 Internet: hall@mit.edu
T. Iwasaki	Department of Control Systems Engineering, Tokyo Institute of Technology, 2-12-1 Oookayama, Meguro-ku, Tokyo 152, Japan Internet: iwasaki@ctrl.titech.ac.jp
U. Jönsson	Division of Optimization and Systems Theory, Department of Mathematics, Royal Institute of Technology, S-100 44 Stockholm, Sweden Internet: ulfj@math.kth.se
C. Lemaréchal	INRIA Rhone-Alpes, ZIRST - 655 avenue de l'Europe 38330 Montbonnot Saint-Martin, France Internet: Claude.Lemarechal@inria.fr
M. Mesbahi	Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 Internet: mesbahi@hafez.jpl.nasa.gov
M. V. Nayakkankuppam	Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21250 Internet: madhu@cs.nyu.edu
S.-I. Niculescu	HEUDIASYC (UMR CNRS 6599), UTC-Compiègne, BP 20529, 60205, Compiègne, Cédex, France Internet: silviu@hds.utc.fr
J. Oishi	Production Engineering Department Laboratory, NEC Corporation, Tomagawa Plant 1753 Shimonumabe, Nakahara-ku Kawasaki, 211 Japan Internet: bigstone@msa.biglobe.ne.jp
F. Oustry	INRIA Rhone-Alpes, ZIRST - 655 avenue de l'Europe 38330 Montbonnot Saint-Martin, France Internet: Francois.Oustry@inria.fr
M. L. Overton	Computer Science Department, Courant Institute, New York University, New York, NY 10012 Internet: overton@cs.nyu.edu

F. Paganini
Department of Electrical Engineering,
UCLA, Los Angeles, CA 90095
Internet: paganini@ee.ucla.edu

G. P. Papavassilopoulos
Department of EE—Systems,
University of Southern California, Los Angeles, CA 90089
Internet: yorgos@bode.usc.edu

A. Rantzer
Lund Institute of Technology,
Box 118, s-221 00 Lund, Sweden
Internet: rantzer@control.lth.se

M. G. Safonov
Department of EE—Systems,
University of Southern California, Los Angeles, CA 90089
Internet: msafanov@usc.edu

C. W. Scherer
Mechanical Engineering Systems and Control Group,
Delft University of Technology,
Mekelweg 2, 2628 CD Delft, The Netherlands
Internet: c.w.scherer@wbmt.tudelft.nl

A. Trofino
Department of Automation and Systems,
Universidade Federal de Santa Catarina,
88040-900 Florianópolis, SC, Brazil
Internet: trofino@lcmi.ufsc.br

S.-P. Wu
Numerical Technology, Inc.,
Santa Clara, CA 95051

K. Y. Yang
Lincoln Laboratory,
Massachusetts Institute of Technology, Lexington, MA 02420
Internet: kyang@ll.mit.edu

This page intentionally left blank

Contents

Preface

Linear matrix inequalities (LMIs) have emerged recently as a useful tool for solving a number of control problems. The basic idea of the LMI method in control is to interpret a given control problem as a *semidefinite programming* (SDP) problem, i.e., an optimization problem with linear objective and positive semidefinite constraints involving symmetric matrices that are affine in the decision variables.

The LMI formalism is relevant for many reasons. First, writing a given problem in this form brings an efficient, numerical solution. Also, the approach is particularly suited to problems with "uncertain" data and multiple (possibly conflicting) specifications. Finally, this approach seems to be widely applicable, not only in control, but also in other areas where uncertainty arises.

Purpose and intended audience

Since the early 1990s, with the developement of *interior-point methods* for solving SDP problems, the LMI approach has witnessed considerable attention in the control area (see the regularity of the invited sessions in the control conferences and workshops). Up to now, two self-contained books related to this subject have appeared. The book *Interior Point Polynomial Methods in Convex Programming: Theory and Applications,* by Nesterov and Nemirovskii, revolutionarized the field of optimization by showing that a large class of nonlinear convex programming problems (including SDP) can be solved very efficiently. A second book, also published by SIAM in 1994, *Linear Matrix Inequalities in System and Control Theory,* by Boyd, El Ghaoui, Feron, and Balakrishnan, shows that the advances in convex optimization can be successfully applied to a wide variety of difficult control problems.

At this point, a natural question arises: Why another book on LMIs?

One aim of this book is to describe, for the researcher in the control area, several important advances made both in algorithms and software and in the important issues in LMI control pertaining to analysis, design, and applications. Another aim is to identify several important issues, both in control and optimization, that need to be addressed in the future.

We feel that these challenging issues require an interdisciplinary research effort, which we sought to foster. For example, Chapter 1 uses an optimization formalism, in the hope of encouraging researchers in optimization to look at some of the important ideas in LMI control (e.g., deterministic uncertainty, robustness) and seek nonclassical applications and challenges in the control area. Bridges go both ways, of course: for example, the "primal-dual" point of view that is so successful in optimization is also important in control.

Book outline

In our chapter classification, we sought to provide a continuum from numerical methods to applications, via some theoretical problems involving analysis and synthesis for uncertain systems. Basic notation and acronyms are listed after the preface.

After this outline, we provide some alternative keys for reading this book.

Part I: Introduction

Chapter 1: Robust Decision Problems in Engineering: A Linear Matrix Inequality Approach, by L. El Ghaoui and S.-I. Niculescu.

This chapter is an introduction to the "LMI method" in robust control theory and related areas. A large class of engineering problems with uncertainty can be expressed as optimization problems, where the objective and constraints are perturbed by unknown-but-bounded parameters. The authors present a robust decision framework for a large class of such problems, for which (approximate) solutions are computed by solving optimization problems with LMI constraints.

The authors emphasize the wide scope of the method, and the anticipated interplay between the tools developed in LMI robust control, and related areas where uncertain decision problems arise.

Part II: Algorithms and software

Chapter 2: Mixed Semidefinite–Quadratic–Linear Programs, by J.-P. A. Haeberly, M. V. Nayakkankuppam, and M. L. Overton.

The authors consider mixed semidefinite–quadratic–linear programs. These are linear optimization problems with three kinds of cone constraints, namely, the semidefinite cone, the quadratic cone, and the nonnegative orthant. The chapter outlines a primal-dual path-following method to solve these problems and highlights the main features of SDPpack, a Matlab package that solves such programs. Furthermore, the authors give some examples where such mixed programs arise and provide numerical results on benchmark problems.

Chapter 3: Nonsmooth Algorithms to Solve Semidefinite Programs, by C. Lemaréchal and F. Oustry.

Today, SDP problems are usually solved by interior-point methods, which are elegant, efficient, and well suited. However, they have limitations, particularly in large-scale or ill-conditioned cases. On the other hand, SDP is an instance of nonsmooth optimization (NSO), which enjoys some particular structure. This chapter briefly reviews the works that have been devoted to solving SDP with NSO tools and presents some recent results on bundle methods for SDP. Finally, the authors outline some possibilities for future work.

Chapter 4: `sdpsol`: A Parser/Solver for Semidefinite Programs with Matrix Structure, by S.-P. Wu and S. Boyd.

This chapter describes a parser/solver for a class of LMI problems, the so-called max-det problems, which arise in a wide variety of engineering problems. These problems often have matrix structure, which has two important practical ramifications: first, it makes the job of translating the problem into a standard SDP or max-det format tedious, and, second, it opens the possibility of exploiting the structure to speed up the computation.

In this chapter the authors describe the design and implementation of `sdpsol`, a parser/solver for SDPs and max-det problems. `sdpsol` allows problems with matrix structure to be described in a simple, natural, and convenient way.

Part III: Analysis

Chapter 5: Parametric Lyapunov Functions for Uncertain Systems: The Multiplier Approach, by M. Fu and S. Dasgupta.

This chapter proposes a *parametric multiplier approach* to deriving parametric Lyapunov functions for robust stability analysis of linear systems involving uncertain parameters. This new approach generalizes the traditional multiplier approach used in the absolute stability literature where the multiplier is independent of the uncertain parameters. The main result provides a general framework for studying *multiaffine Lyapunov functions.* It is shown that these Lyapunov functions can be found using LMI techniques. Several known results on parametric Lyapunov functions are shown to be special cases.

Chapter 6: Optimization of Integral Quadratic Constraints, by U. Jönsson and A. Rantzer.

A large number of performance criteria for uncertain and nonlinear systems can be unified in terms of integral quadratic constraints. This makes it possible to systematically combine and evaluate such criteria in terms of LMI optimization.

A given combination of nonlinear and uncertain components usually satisfies infinitely many integral quadratic constraints. The problem to identify the most appropriate constraint for a given analysis problem is convex but infinite dimensional. A systematic approach, based on finite-dimensional LMI optimization, is suggested in this chapter. Numerical examples are included.

Chapter 7: Linear Matrix Inequality Methods for Robust $\mathbf{H}_2$ Analysis: A Survey with Comparisons, by F. Paganini and E. Feron.

This chapter provides a survey of different approaches for the evaluation of $\mathbf{H}_2$-performance in the worst case over structured system uncertainty, all of which rely on LMI computation. These methods apply to various categories of parametric or dynamic uncertainty (linear time-invariant (LTI), linear time-varying (LTV), or nonlinear time-varying (NLTV)) and build on different interpretations of the $\mathbf{H}_2$ criterion. It is shown nevertheless how they can be related by using the language of LMIs and the so-called $\mathcal{S}$-procedure for quadratic signal constraints. Mathematical comparisons and examples are provided to illustrate the relative merits of these approaches as well as a common limitation.

Part IV: Synthesis

Chapter 8: Robust $\mathbf{H}_2$ Control, by K. Y. Yang, S. R. Hall, and E. Feron.

In this chapter, the problem of analyzing and synthesizing controllers that optimize the $\mathbf{H}_2$ performance of a system subject to LTI uncertainties is considered. A set of upper bounds on the system performance is derived, based on the theory of stability multipliers and the solution of an original optimal control problem. A Gauss–Seidel-like algorithm is proposed to design robust and efficient controllers via LMIs. An efficient solution procedure involves the iterative solution of Riccati equations both for analysis and synthesis purposes. The procedure is used to build robust and efficient controllers for a space-borne active structural control testbed, the Middeck Active Control Experiment (MACE). Controller design cycles are now short enough for experimental investigation.

Chapter 9: A Linear Matrix Inequality Approach to the Design of Robust $\mathbf{H}_2$ Filters, by C. E. de Souza and A. Trofino.

This chapter is concerned with the robust minimum variance filtering problem for linear continuous-time systems with parameter uncertainty in all the matrices of the system state-space model, including the coefficient matrices of the noise signals. The

admissible parameter uncertainty is assumed to belong to a given convex bounded polyhedral domain. The problem addressed here is the design of a linear stationary filter that ensures a guaranteed optimal upper bound for the asymptotic estimation error variance, irrespective of the parameter uncertainty. This filter can be regarded as a robust version of the celebrated Kalman filter for dealing with systems subject to convex bounded parameter uncertainty. Both the design of full-order and reduced-order robust filters are analyzed. We develop LMI-based methodologies for the design of such robust filters. The proposed methodologies have the advantage that they can easily handle additional design constraints that keep the problem convex.

Chapter 10: Robust Mixed Control and Linear Parameter-Varying Control with Full Block Scalings, by C. W. Scherer.

This chapter considers systems affected by time-varying parametric uncertainties. The author devises a technique that allows us to equivalently translate robust performance analysis specifications characterized through a single Lyapunov function into the corresponding analysis test with multipliers. Out of the multitude of possible applications of this so-called full block $\mathcal{S}$-procedure, the chapter concentrates on a discussion of robust mixed control and of designing linear parameter-varying (LPV) controllers.

Chapter 11: Advanced Gain-Scheduling Techniques for Uncertain Systems, by P. Apkarian and R. J. Adams.

This chapter is concerned with the design of gain-scheduled controllers for uncertain LPV systems. Two alternative design techniques for constructing such controllers are discussed. Both techniques are amenable to LMI problems via a gridding of the parameter space and a selection of basis functions. The problem of synthesis for robust performance is then addressed by a new scaling approach for gain-scheduled control. The validity of the theoretical results is demonstrated through a two-link flexible manipulator design example. This is a challenging problem that requires scheduling of the controller in the manipulator geometry and robustness in the face of uncertainty in the high frequency range.

Chapter 12: Control Synthesis for Well-Posedness of Feedback Systems, by T. Iwasaki.

A wide variety of control synthesis problems reduces to the same type of computational problems involving LMIs. The main objective of this chapter is to show explicitly a general class of control problems that reduce to the same computational problem. The class is defined by the closed-loop specifications given in terms of the well-posedness of feedback systems. The author shows that this class includes many existing control problems as special cases, and such problems can be reduced to a search for structured positive definite matrices satisfying LMIs and possibly rank constraints.

Part V: Nonconvex problems

Chapter 13: Alternating Projection Algorithms for Linear Matrix Inequalities Problems with Rank Constraints, by K. M. Grigoriadis and E. B. Beran.

Recently a large class of fixed-order control design problems has been formulated in a unified way as LMI problems with additional coupling matrix rank constraints. Because of the nonconvexity of the rank constraints, efficient convex programming algorithms cannot be used to find a solution. In this chapter, the method of alternating projections along with efficient SDP algorithms are proposed to address these problems of combined LMIs and coupling matrix rank constraints. Alternating projection algorithms exploit the geometry of these problems to obtain feasible solutions. Directional alternating projection methods that enhance the computational efficiency of the algorithms

are also described. Extensive computational experiments are provided to indicate the effectiveness and complexity of the proposed algorithms.

Chapter 14: Bilinearity and Complementarity in Robust Control, by M. Mesbahi, M. G. Safonov, and G. P. Papavassilopoulos.

The authors present an overview of the key developments in the methodological, structural, and computational aspects of the bilinear matrix inequality (BMI) feasibility problem. In this direction, the chapter presents the connections of the BMI with robust control theory and its geometric properties, including interpretations of the BMI as a rank-constrained LMI, as an extreme form problem (EFP), and as a semidefinite complementarity problem (SDCP). Computational implications and algorithms are also discussed.

Part VI: Applications

Chapter 15: Linear Controller Design for the NEC Laser Bonder via Linear Matrix Inequality Optimization, by J. Oishi and V. Balakrishnan.

The authors describe a computer-aided control system design method for designing an LTI controller for the NEC Laser Bonding machine. The procedure consists of numerically searching over the set of Youla parameters that describe the set of stabilizing LTI controllers for an LTI model of the Laser Bonder; this search is conducted using convex optimization techniques involving LMIs. The design specifications include constraints on the transient and steady-state tracking of a specific input, as well as bounds on the norms of certain closed-loop maps of interest. The design procedure is shown to yield a controller that optimally satisfies the various performance specifications.

Chapter 16: Multiobjective Robust Control Toolbox for Linear-Matrix-Inequality-Based Control, by S. Dussy.

This chapter presents a collection of LMI-based synthesis tools for a large class of nonlinear uncertain systems, packaged in a set of Matlab functions. Most of them are based on a *cone complementarity problem* and provide solutions for the robust state- and output-feedback controller synthesis under various specifications. These specifications include performance requirements via α-stability or $\mathcal{L}_2$-gain bounds and command input and outputs bounds. Two numerical examples, namely, an inverted pendulum and a nonlinear benchmark, are provided.

Chapter 17: Multiobjective Control for Robot Telemanipulators, by J. P. Folcher and C. Andriot.

This chapter addresses the control of robot telemanipulators in the presence of uncertainties, disturbance, and measurement noise. The controller design problem can be divided into several multiobjective robust controller synthesis problems for LTI systems subject to dissipative perturbations. Synthesis specifications include robust stability, robust performance ($\mathbf{H}_2$ norm) bounds, and time-domain bounds (output and command input peak). Sufficient conditions for the existence of an LTI controller such that the closed-loop system satisfies all specifications simultaneously are derived. An efficient *cone complementarity linearization algorithm* enables us to solve numerically the associate optimization problem. This multiobjective synthesis approach is used to design the force controller of a single joint of a slave manipulator.

How to read the book

The book is the result of the work of many different authors, each with a different point of view. We have sought to provide a coherent and unitary text, without sacrificing the

individuality of each contribution. Some redundancy might result from this; however, we feel it is always beneficial to provide different views on similar problems.

There are several themes, emphasized as key words in the list below, that come across the sequential order of the book. An interested reader might read the corresponding chapters independently.

- Several chapters deal with the so-called S-procedure and its applications in control: Chapter 1 puts it in the context of Lagrange relaxations, Chapters 8 and 10 generalize the result to different settings. Also, Lyapunov functions (a theme recurrent in the book) can be interpreted in the context of Lagrange multipliers, as explained in Chapter 1.

- The *robust* $\mathbf{H}_2$ problem is evoked first in its analysis aspect in Chapter 7. Chapters 8 and 10 deal with the corresponding control synthesis problem in different ways; Chapter 9 addresses the issues of filter design in this context.

- *Nonconvex optimization problems* arise mainly in three forms in control, as LMI problems with rank constraints, as BMI problems, or as cone complementarity problems. The similarity between these points of view is outlined in Chapter 1 and are explored in more detail in Part V. The so-called cone complementarity formulation, mentioned in Chapters 1 and 14 (under the name SCDP) is used in a practical context in Chapters 16 and 17.

- *Linear parameter-varying* (LPV) systems are considered in Chapters 10 and (to a lesser extent) 12, where a general framework, encompassing robust control, is outlined. Chapter 11 is devoted to techniques improving the possible conservatism of the method.

- The search for *multipliers* in the context of robustness analysis is addressed in Chapter 5 under a "parametric Lyapunov function" point of view, while Chapter 6 develops the "integral quadratic constraint" (IQC) approach; Chapter 17 uses a very similar approach in a practical application. Note that Chapter 1 makes some connections between Lyapunov functions, multipliers (in the above "control" sense), and Lagrange relaxations in optimization.

- *Algorithms and software* issues are broadly mentioned in Chapter 1. Chapters 2 and 3 describe algorithms, and Chapter 4 describes a parser/solver for a general class of LMI problems. Chapter 16 uses a specialized Matlab toolbox for some robust control problems.

- Some *numerical results and applications* are mentioned throughout the book, coming across as illustrations of software performance (Part II), applications of system analysis (Chapters 6 and 7), or control synthesis methods (Chapters 11 and 13). Part VI is dedicated to realistic design examples.

Acknowledgments

This project began a long time ago (in October 1996!). During this long process, the book has benefited from the enthusiastic participation and feedback from the authors, whose patience is remarkable. We express our thanks to all of them.

Some of the authors have particularly helped our editorial job, improving the overall message. For example, Michael Overton and François Oustry have completely rewritten a section on LMI algorithms in the introductory chapter; Fernando Paganini and Eric Feron decided to merge their contribution on robust $\mathbf{H}_2$ analysis in a single survey chapter.

We also would like to thank the anonymous reviewers, who have carefully read the manuscript and provided us with very useful suggestions for improvement. We are grateful to John Burns, editor-in-chief of this series, for his visionary support, making this book a reality. Our SIAM correspondents, Vickie Kearn and Marianne Will, offered us precious help at all stages of this project.

This book would not have been possible without the support of EDF (Electricité de France), under contract P33L74/2M7570/EP777; special thanks goes to Clément-Marc Falinower, head of the "Service Automatique" at EDF. His support partially funded, in particular, several key visits of contributors to ENSTA during the project, including those of Silviu-Iulian Niculescu, Michael Overton, Stephen Boyd, and Eric Feron. Special thanks go to Jean-Luc Commeau, who helped us with various computer problems. Finally, we would like to thank our home institutions, ENSTA (Paris) and CNRS-HEUDIASYC (Compiègne), for their support.

Chapter 1 was partially funded by a support from EDF under contract P33L74/2M7570/EP777. Section 1.5 was written with precious help from Michael Overton and François Oustry. Chapter 2 was supported in part by National Science Foundation grant CCR-9731777 and in part by U.S. Department of Energy grant DE-FG02-98ER25352. Chapter 4 was partially supported by AFOSR contract F49620-95-1-0318, NFS contracts ECS-9222391 and EEC-9420565, and MURI contract F49620-95-1-0525. Chapter 15 was supported in part by the NEC Faculty Fellowship and a General Motors Faculty Fellowship.

Laurent El Ghaoui
Silviu-Iulian Niculescu

This page intentionally left blank

Notation

$\mathbf{R}$, $\mathbf{R}^k$, $\mathbf{R}^{m\times n}$ The real numbers, real k-vectors, real $m \times n$ matrices.

$\mathbf{R}_+$, $j\mathbf{R}$ The nonnegative real numbers and the imaginary numbers, respectively.

$\mathbf{C}$ The complex numbers.

$\mathbf{C}^0$ Unit circle of the complex plane.

$\mathbf{C}^+$, $\bar{\mathbf{C}}^+$, $\mathbf{C}^-$ The open right half, closed right half, and open left half complex planes, respectively.

a^*, $\Re(a)$ The complex conjugate and the real part of $a \in \mathbf{C}$, i.e., $(a+a^*)/2$.

$\mathbf{E}x$ Expected value of (the random variable) x.

$\mathbf{H}_p$ pth Hardy space (in this book, p is either 2 or ∞).

$\mathbf{L}_p(\mathbf{R}, \mathbf{R}^{n\times m})$ pth Hilbert space of functions mapping $\mathbf{R}$ to $\mathbf{R}^{n\times m}$. ($p = 2$ in this book; also, the abbreviation $\mathbf{L}_2$ is used.)

$\overline{S}$ The closure of a set S.

$\mathrm{int}S$ The interior of a set S.

S^* The dual cone of a set $S \subseteq \mathcal{S}^n$ is defined to be

$$S^* = \{Y \in \mathcal{S}^n \ : \ \mathbf{Tr}(XY) \geq 0 \ \ \forall X \in S\}.$$

$\mathcal{S}^n$ Space of real, symmetric matrices of order n.

$\mathcal{S}^n_+$ Space of real, positive semidefinite, symmetric matrices of order n.

$\mathbf{Co}S$ Convex hull of the set $S \subseteq \mathbf{R}^n$, given by

$$\mathbf{Co}S \triangleq \left\{ \sum_{i=1}^{p} \lambda_i x_i \,\middle|\, \lambda_k \geq 0, \right\}.$$

(Without loss of generality, we can take $p = n + 1$ here.)

I_k The $k \times k$ identity matrix. The subscript is omitted when k is not relevant or can be determined from context.

M^T Transpose of a matrix M: $(M^T)_{ij} = M_{ji}$.

M^* Complex-conjugate transpose of a matrix M: $(M^*)_{ij} = M^*_{ji}$, where α^* denotes the complex conjugate of $\alpha \in \mathbf{C}$.

$\mathbf{Tr}(M)$ Trace of $M \in \mathbf{R}^{n\times n}$; i.e., $\sum_{i=1}^n M_{ii}$. Sometimes the parentheses are omitted if context permits.

$\mathbf{Tr}(A^TB)$ Scalar product of two real matrices A, B. Again, the parentheses are omitted if context permits.

$\mathbf{KerM}$ Nullspace of a matrix M.

ImM Range of a matrix M.

RankM Rank of a matrix M.

$M^\perp$ Orthogonal complement of M, a matrix of maximal rank such that $M^T M^\perp = 0$; the rows of $M^\perp$ form a basis for the nullspace of M^T.

$M^\dagger$ Moore–Penrose pseudoinverse of M.

$M \geq 0$ M is symmetric and positive semidefinite; i.e., $M = M^T$ and $z^T M z \geq 0$ for all $z \in \mathbf{R}^n$.

$M > 0$ M is symmetric and positive definite; i.e., $z^T M z > 0$ for all nonzero $z \in \mathbf{R}^n$.

$M > N$ M and N are symmetric and $M - N > 0$.

$M^{1/2}$ For $M > 0$, $M^{1/2}$ is the unique $Z = Z^T$ such that $Z > 0$, $Z^2 = M$.

sect$X$ Bilinear sector transformation of a square matrix, defined (when applicable) as $(I - X)(I + X)^{-1}$.

$\lambda_{\max}(M)$ The maximum eigenvalue of the matrix $M = M^T$.

$\lambda_{\max}(P, Q)$ For $P = P^T$, $Q = Q^T > 0$, $\lambda_{\max}(P, Q)$ denotes the maximum eigenvalue of the symmetric pencil (P, Q); i.e., $\lambda_{\max}(Q^{-1/2} P Q^{-1/2})$.

$\lambda_{\min}(M)$ The minimum eigenvalue of $M = M^T$.

$\|M\|$ The spectral norm of a matrix or vector M, i.e., $\sqrt{\lambda_{\max}(M^T M)}$. It reduces to the Euclidean norm, i.e., $\|x\| = \sqrt{x^T x}$, for a vector x.

$\|M\|_F$ The Frobenius norm of a matrix or vector M, i.e., $\sqrt{\mathbf{Tr}(M^T M)}$. It reduces to the Euclidean norm, i.e., $\|x\| = \sqrt{x^T x}$, for a vector x.

diag$(\cdots)$ Block-diagonal matrix formed from the arguments, i.e.,

$$\mathbf{diag}\,(M_1, \ldots, M_m) \triangleq \begin{bmatrix} M_1 & & \\ & \ddots & \\ & & M_m \end{bmatrix}.$$

Vec(M) vector obtained by stacking up the columns of matrix M.

$A \otimes B$ The Kronecker product of matrices A, B.

Herm(M) The Hermitian part of matrix M, i.e., the matrix $\frac{1}{2}(M + M^T)$.

Partitioned matrix notation for transfer functions:

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = D + C(sI - A)^{-1} B.$$

$\mathcal{F}_U(M, \Delta)$, $\mathcal{F}_L(\Delta, M)$ The upper and lower linear fractional transformations of a matrix Δ and a partitioned matrix M, defined as

$$\mathcal{F}_U\left(\left[\begin{array}{c|c} M_{11} & M_{12} \\ \hline M_{21} & M_{22} \end{array}\right], \Delta\right) = M_{22} + M_{21}\Delta(I - M_{11}\Delta)^{-1} M_{12}$$

and

$$\mathcal{F}_L\left(\Delta, \left[\begin{array}{c|c} M_{11} & M_{12} \\ \hline M_{21} & M_{22} \end{array}\right]\right) = M_{11} + M_{12}\Delta(I - M_{22}\Delta)^{-1} M_{21}.$$

$\mathcal{F}(M, N)$ The linear fractional transformation (LFT) or star product of two partioned matrices M, N, defined as

$$\mathcal{F}\left(\left[\begin{array}{c|c} M_{11} & M_{12} \\ \hline M_{21} & M_{22} \end{array}\right], \left[\begin{array}{c|c} N_{11} & N_{12} \\ \hline N_{21} & N_{22} \end{array}\right]\right) = \left[\begin{array}{c|c} \mathcal{F}_L(M, N_{11}) & M_{12}(I - N_{11}M_{22})^{-1}N_{12} \\ \hline N_{21}(I - M_{22}N_{11})^{-1}M_{21} & \mathcal{F}_U(N, M_{22}) \end{array}\right].$$

Δ Uncertainty matrix.

$\mathbf{\Delta}$ Uncertainty set, including the zero matrix in general.

This page intentionally left blank

Part I

Introduction

This page intentionally left blank

Chapter 1

Robust Decision Problems in Engineering: A Linear Matrix Inequality Approach

L. El Ghaoui and S.-I. Niculescu

1.1 Introduction

1.1.1 Basic idea

The basic idea of the LMI method is to formulate a given problem as an optimization problem with linear objective and *linear matrix inequality* (LMI) constraints. An LMI constraint on a vector $x \in \mathbf{R}^m$ is one of the form

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i \geq 0, \tag{1.1}$$

where the symmetric matrices $F_i = F_i^T \in \mathbf{R}^{N \times N}$, $i = 0, \ldots, m$, are given. The minimization problem

$$\text{minimize } c^T x \text{ subject to } F(x) \geq 0, \tag{1.2}$$

where $c \in \mathbf{R}^m$, and $F \geq 0$ means the matrix F is symmetric and positive semidefinite, is called a semidefinite program (SDP). The above framework is particularly attractive for the following reasons.

Efficient numerical solution. SDP optimization problems can be solved very efficiently using recent interior-point methods for convex optimization (the global optimum is found in polynomial time). This brings a numerical solution to problems when no analytical or closed-form solution is known.

Robustness against uncertainty. The approach is very well suited for problems with uncertainty in the data. Based on a deterministic description of uncertainty (with detailed structure and hard bounds), a systematic procedure enables us to formulate an SDP optimization problem that yields a robust solution. This statement has implications for a wide scope of engineering problems, where measurements, modelling errors, etc., are often present.

Multicriteria problems. The approach enables us to impose many different (possibly conflicting) specifications in the design process, allowing us to explore trade-offs and analyze limits of performance and feasibility. This offers a drastic advantage over design methods that rely on a single criterion deemed to reflect all design constraints; the choice of a relevant criterium is sometimes a nontrivial task.

Wide applicability. The techniques used in the approach are relevant far beyond control and estimation. This opens exciting avenues of research where seemingly very different problems are analyzed and solved in a unified framework. For example, the method known in LMI-based control as the $\mathcal{S}$-procedure can be successfully applied in combinatorial optimization, leading to efficient relaxations of hard problems.

1.1.2 Approximation and complexity

At this point we should comment on a very important aspect of the LMI approach, which puts in perspective the above.

The LMI approach is deeply related to the branch of theoretical computer science that seeks to classify problems in terms of their *computational complexity* [323]. In a sense, the approach is an approximation technique for a class of (NP-) hard problems, via polynomial-time algorithms, and thus hinges on a dividing line between "hard" and "easy" in the sense of computational complexity.

This point of view shows that the LMI approach is not only a numerical approach to practical problems. It also requires one to consider the complexity analysis of a given problem and to seek an approximation in the case of a "hard" problem. This concern is quite new in control; see [53] for a survey and references on complexity analysis of control problems.

1.1.3 Purpose of the chapter

In this chapter, we describe a general LMI-based method to cope with engineering decision problems with uncertainty and illustrate its relevance, both in control and in other areas. This chapter is intended to serve two kinds of objectives.

The first is to *introduce* the reader to the LMI method for control and help understand some of the issues treated in more detail in the subsequent chapters. References that would usefully complement this reading include a book on the LMI approach in system and control theory by Boyd, El Ghaoui, Feron, and Balakrishnan [64], a book on algorithms for LMI problems by Nesterov and Nemirovskii [296], and [404]. Also, the course notes [66] contain many results on convex optimization and applications.

Another purpose of this chapter is to *open* the method to other areas. In our presentation, we have tried to give an "optimization point of view" of the method in an effort to put it in perspective with classical relaxation methods. This effort is mainly motivated by the belief that the approach could be useful in many other fields, especially regarding robustness issues. This topic of robustness is receiving renewed attention in the field of optimization, as demonstrated by a series of recent papers [42, 127, 39].

1.1.4 Outline

Section 1.2 defines robust solutions to problems with uncertain data and outlines the general ideas involved in Lagrange relaxations for these problems. Section 1.3 is devoted to models of uncertain systems. In section 1.4, we concentrate on SDP problems with uncertain data matrices and detail a general methodology for computing appropriate bounds on these problems. Section 1.5 briefly describes algorithms, software, and related

issues for solving general SDPs. (This section, written with the help of Michael Overton and François Oustry, can be read independently of the rest of this chapter.) Some illustrative examples taken from the control area, which complement those put forth in the other chapters, are given in section 1.6. Section 1.7 presents a number of examples taken out of control. Finally, section 1.8 explores some perspectives and challenges that lie ahead.

1.2 Decision problems with uncertain data

1.2.1 Decision problems

Many engineering analysis and design problems can be seen as decision problems. In control engineering, one must decide which controller gains to choose in order to satisfy the desired specifications. This decision involves several trade-offs.

In this context, decision problems are based on two "objects": a set of *decision* variables that reflect the engineer's choices (controller gains, actuator location, etc.) and a set of *constraints* on these decision variables that reflect the specifications imposed by the problem (desired closed-loop behavior, etc.). Such problems can be sometimes translated in terms of mathematical programming problems of the form

$$\text{minimize } f_0(x) \text{ subject to } x \in \mathcal{X}, \quad f_i(x) \leq 0, \quad i = 1, \ldots, p, \tag{1.3}$$

where $f_0, f_1, \ldots, f_p$ are given scalar-valued functions of the decision vector $x \in \mathbf{R}^m$, and $\mathcal{X}$ is a subset of $\mathbf{R}^m$. In some problems, $\mathcal{X}$ is infinite dimensional, and the decision vector x is a function.

In the language of control theory, we can interpret the above as a *multispecification* control problem, where several (possibly conflicting) constraints have to be satisfied.

Let us comment on the limitations of the above mathematical framework. The desired specifications are sometimes easy to describe mathematically as in the above model (for example, when we seek to ensure a desired maximum overshoot). Others are more difficult to handle and not less important, such as economic constraints (cost of design), reliability and ease of implementation (time needed to obtain a successful design), etc. These "soft" constraints can be neglected or taken into account implicitly (for example, choosing a PID controller structure, as opposed to a more complicated one). A classical way to handle these constraints is by hierarchical decision, where soft constraints are taken into account at a higher level of managerial decision (e.g., allowing a constrained budget for the whole control problem).

1.2.2 Using uncertainty models

In some applications, the "data" in problem (1.3) is not well known; the previous "optimization model" falls short of handling this case, so it would be useful to consider decision problems with uncertainty. For this, it is necessary to introduce *models* for uncertain systems; this is the purpose of section 1.3.

There are other motivations for us to use such models. Real-world phenomena are very complex (nonlinear, time-varying, infinite-dimensional, etc.). To cope with analysis and design problems for such systems, we almost always have to use (finite-dimensional, linear, stationary) approximations. We believe that a finer way to handle complexity is to use approximations with uncertainty.

We thus make the distinction between "physical" and "engineering" uncertainties. The first are due to ignorance of the precise physical behavior; the second come from a voluntary simplifying assumption. (This technique can be referred to as *embedding*.)

Assume, for instance, that in a complex system two variables p, q are linked by a nonlinear relation $p = \delta(q)$. The function δ may be very difficult to identify (this is a physical uncertainty). Moreover, we may choose to "embed" this nonlinearity in a *family* of linear relations of the form $p = \delta q$, where the scalar δ is inside a known sector. (The latter may be inferred from physical knowledge.) We thus obtain a linear model with (engineering) uncertainty that is more tractable than the nonlinear one. The technique of embedding may be conservative, but it usually gives rise to tractable problems.

1.2.3 Robust decision problems

In some applications, the "data" in problem (1.3) is not well known; that is, every f_i depends not only on x but also on a real "uncertainty matrix"[1] Δ. The latter is only known to belong to a set of admissible perturbations $\mathbf{\Delta}$. We may then define a variety of *robust decision problems* and associated solutions. These problems are based on a worst-case analysis of the effect of perturbations.

Robustness analysis

We define the *robust feasibility* set

$$\mathcal{X}(\mathbf{\Delta}) = \{x \in \mathcal{X} \mid f_i(x, \Delta) \leq 0, \quad i = 1, \ldots, p \text{ for every } \Delta \in \mathbf{\Delta}\}\,. \tag{1.4}$$

The analysis problem is to check if a given candidate solution x is robustly feasible.

Robust synthesis

A robust synthesis problem is to find a vector x that satisfies robustness constraints. We may distinguish the following:

- A *robust feasibility problem*, where we seek a vector x that belongs to $\mathcal{X}(\mathbf{\Delta})$.
- A *robust optimality problem*, defined as

  $$\text{minimize } \max_{\Delta \in \mathbf{\Delta}} f_0(x, \Delta) \text{ subject to } x \in \mathcal{X}(\mathbf{\Delta}). \tag{1.5}$$

  Here, we seek to minimize the worst-case value of the objective $f_0(x, \Delta)$ over the set of robustly feasible solutions. (We note that the objective function f_0 may be assumed independent of the perturbation without loss of generality.)
- *Parameter-scheduled synthesis.* In such a problem, we allow the decision vector to be a function of the perturbation. In this case, we assume that the decision vector is not finite dimensional but evolves in a set $\mathcal{X}$ of functions (of the perturbation Δ). The problem reads formally as the robust synthesis ones, but with $\mathcal{X}$ a set of functions.
- *Mixed parameter-scheduled/robust problems.* In some problems, the decision vector is a mixture of a set of real variables and of functions of some parameters. In control, this kind of problem occurs when only some parameters are available for measurement.

[1] We take the uncertainty to be a matrix instead of a vector for reasons to be made clear later.

In control theory, we interpret the above problems as *robust multispecification* control problems, where several (possibly conflicting) constraints have to be satisfied in a robust manner. We note that these problems are in general very hard to solve (noncomputationally tractable). References on complexity of such problems in control include [86, 291, 72, 52, 393, 144].

Robust decision problems first arose (to our knowledge) in the field of automatic control. In fact, robustness is presented in the classical control textbooks as the main reason for using feedback. Good references on robust control include the book by Zhou, Doyle, and Glover [446]; Dahleh and Diaz-Bobillo [88]; and Green and Limebeer [175].

In other areas of engineering, uncertainty is generally dealt with using stochastic models for uncertainty. In *stochastic optimization*, the uncertainty is assumed to have a known distribution, and a typical problem is to evaluate the distribution of the optimal value, of the solution, etc.; a good reference on this subject is the book by Dempster [97].

Models with deterministic uncertainty are less classical in engineering optimization. Ben Tal and Nemirovski consider a truss topology design problem with uncertainty on the loading forces [41, 40]. In a more recent work [42], they introduce and study the complexity of robust optimality problems in the sense defined above in the context of convex optimization. Their approach is based on ellipsoidal bounds for the perturbation. The paper [127] addresses the computation of bounds for a class of robust problems and uses a control-based approach to solve for these bounds via linear-fractional representations (LFRs) and SDP.

1.2.4 Lagrange relaxations for min-max problems

Robust decision problems such as problem (1.5) belong to the class of min-max problems. To attack them, we can thus use a versatile technique, called *Lagrange relaxation*, that enables us to approximate a set of complicated constraints by a "more tractable" set.

To understand the technique, let us assume that the uncertainty set $\mathbf{\Delta}$ is a subset of $\mathbf{R}^l$ that can be described by a finite number of (nonlinear) constraints

$$\mathbf{\Delta} = \left\{\delta \in \mathbf{R}^l \mid g_i(\delta) \leq 0, \quad i = 1, \ldots, q\right\},$$

where the g_i's are given scalar-valued functions of the perturbation vector δ. Now, observe that the property

$$g_0(\delta) \leq 0 \text{ for every } \delta, \quad g_i(\delta) \leq 0, \quad i = 1, \ldots, q,$$

holds if there exist nonnegative scalars $\tau_1, \ldots, \tau_q$ such that

$$\text{for every } \delta, \quad g_0(\delta) \leq \sum_{i=1}^{q} \tau_i g_i(\delta).$$

The above condition is rewritten

$$\max_{\delta} \left\{ g_0(\delta) - \sum_{i=1}^{q} \tau_i g_i(\delta) \right\} \leq 0. \tag{1.6}$$

The above idea can be used for general min-max problems of the form (1.5). The basic idea is to approximate a min-max problem by a (hopefully easier) minimization problem. To motivate this idea, we take an example of the robust optimality problem (1.5) with no constraints on x ($p = 0$). That is, we consider

$$\min_{x \in \mathcal{X}} \max_{\delta} \left\{ f_0(x, \delta) \mid g_i(\delta) \leq 0, \quad i = 1, \ldots, q \right\}. \tag{1.7}$$

The Lagrange relaxation technique yields an *upper bound* on the above problem:

$$\begin{array}{ll} \text{minimize} & \max_{\delta} \left(f_0(x,\delta) - \sum_{i=1}^{q} \tau_i g_i(\delta) \right) \\ \text{subject to} & x \in \mathcal{X}, \quad \tau_i \geq 0, \quad i = 1, \ldots, q. \end{array} \tag{1.8}$$

In some cases, computing the (unconstrained) maximum in the above problem is very easy; then the min-max problem is approximated (via an upper bound) by a minimization problem (the minimization variables are x and τ). (The scalars τ_i appearing in the above are often referred to as *Lagrange multipliers.*) Note that, for x fixed, the above problem is convex in τ; if f_0 is convex in x, then we may use (subgradient-based) convex optimization to solve the relaxed problem. The technique thus requires that computing values and subgradient of the "max" part (a function of x, τ) is computationally tractable.

The upper bounds found via Lagrange relaxations can be improved as long as we are able to solve and compute corresponding subgradients for a constrained "max" problem in (1.8). For example, we may choose to relax only a subset of the constraints $g_i(\delta) \leq 0$ and keep the remaining ones as hard constraints in the maximization problem. Different choices of "relaxed" versus "nonrelaxed" constraints lead to different Lagrange relaxations (and upper bounds).

A special case: The $\mathcal{S}$-procedure. The well-known $\mathcal{S}$-procedure is an application of Lagrange relaxation in the case when the g_i's are quadratic functions of $\delta \in \mathbf{R}^l$. In this case, checking (1.6) yields a simple LMI in the multipliers τ_i. The $\mathcal{S}$-procedure lemma, which we recall below, is widely used, not only in control theory [258, 140, 64] but also in connection with trust region methods in optimization [376, 381].

Lemma 1.1 ($\mathcal{S}$-procedure). *Let $g_0, \ldots, g_p$ be quadratic functions of the variable $\delta \in \mathbf{R}^m$:*

$$g_i(\delta) \triangleq \begin{bmatrix} \delta \\ 1 \end{bmatrix}^T F_i \begin{bmatrix} \delta \\ 1 \end{bmatrix}, \quad i = 0, \ldots, p, \tag{1.9}$$

where $F_i = F_i^T$. The following condition on $g_0, \ldots, g_p$

$$g_0(\delta) \geq 0 \text{ for all } \delta \text{ such that } g_i(\delta) \leq 0, \; i = 1, \ldots, p,$$

holds if there exist scalars $\tau_i \geq 0$, $i = 1, \ldots, p$, such that

$$F(\tau) = F_0 + \sum_{i=1}^{p} \tau_i F_i \geq 0. \tag{1.10}$$

When $p = 1$, the converse holds, provided that there is some δ_0 such that $g_1(\delta_0) > 0$.

1.3 Uncertainty models

A typical decision problem is defined via some data, which we collect for convenience in a matrix M. For example, if the problem is defined in terms of the transfer function of an LTI system $M(s) = D + C(sI - A)^{-1}B$, the matrix M contains the four matrices A, B, C, D.

When the data is uncertain, we need to describe as compactly as possible the way the perturbation Δ affects the data matrix and also the structure of the perturbation

set $\mathbf{\Delta}$. This will result in an *uncertainty model* for the data. Formally, an uncertainty model for the data is defined as a matrix set

$$\{\mathbf{M}(\Delta) \mid \Delta \in \mathbf{\Delta}\}, \tag{1.11}$$

where $\mathbf{M}$ is a (possibly nonlinear) matrix-valued function of the "perturbation matrix" Δ, and $\mathbf{\Delta}$ is a matrix set.

1.3.1 Linear-fractional models

Definition

A linear-fractional model is a set of the form (1.11), where the following three assumptions are made.

Linear-fractional assumption. We assume that the matrix-valued function can be written, for almost every Δ, as

$$\mathbf{M}(\Delta) = M + L\Delta(I - D\Delta)^{-1}R$$

for appropriate constant matrices M, L, R and $D \in \mathbf{R}^{n_q \times n_p}$.

Uncertainty set assumption. In addition, the uncertainty set $\mathbf{\Delta}$ is required to satisfy the following.

For every vector pair p, q, the condition

$$p = \Delta q \text{ for some } \Delta \in \mathbf{\Delta}$$

can be characterized as a *linear matrix inequality* on a rank-one matrix:

$$\Phi_{\mathbf{\Delta}}\left(\begin{bmatrix} q \\ p \end{bmatrix}\begin{bmatrix} q \\ p \end{bmatrix}^T\right) \geq 0, \tag{1.12}$$

where $\Phi_{\mathbf{\Delta}}$ is a given linear operator "characterizing" $\mathbf{\Delta}$.

Well-posedness assumption. To simplify our exposition, we make a last assumption from now on that the above model is *well posed* over $\mathbf{\Delta}$, meaning that

$$\det(I - D\Delta) \neq 0 \text{ for every } \Delta \in \mathbf{\Delta}.$$

Checking well-posedness is very difficult in general; using the theory developed in section 1.4, we can derive sufficient LMI conditions for well-posedness. Chapter 12 discusses the issue of well-posedness in more detail.

Motivations

The above uncertainty models for the data seem very specialized. However, they can cover a wide variety of uncertain matrices, as we now show.

First, the *linear-fractional assumption* made on the matrix function $\mathbf{M}(\Delta)$ is motivated by the following lemma, a constructive proof of which appears in, e.g., [56, 446].

Lemma 1.2. *Every (matrix-valued) rational function* $\mathbf{M}(\delta)$ *of* $\delta \in \mathbf{R}^p$ *that is well defined for* $\delta = 0$ *admits an LFR of the form*

$$\begin{aligned} &\mathbf{M}(\delta) = M + L\Delta(I - D\Delta)^{-1}R, \\ &\text{where } \Delta = \mathbf{diag}(\delta_1 I_{r_1}, \ldots, \delta_l I_{r_l}), \end{aligned} \tag{1.13}$$

valid for whenever $\det(I - D\Delta) \neq 0$ *for appropriate matrices* M, L, R, D *and integers* $r_1, \ldots, r_l$.

We note that, when $l = 1$ (that is, for a *monovariable* rational matrix function), the matrices M, L, R, D are simply a state-space realization of the transfer matrix $M + L(sI - D)^{-1}R$, where $s = 1/\delta$. An LFR of $\mathbf{M}$ can then always be constructed in a "minimal" way, such that the size of matrix Δ is exactly the order of $\mathbf{M}$ in s. In the multivariable case, the integers r_i are greater than the degrees of the corresponding variable δ_i in $\mathbf{M}$, and the issue of minimality is much more subtle. Finally, we note that when $\mathbf{M}$ is polynomial in its argument, an LFR can always be constructed such that D is strictly upper triangular, so that this LFR is everywhere well posed, as is $\mathbf{M}$.

The LFR generalizes the state-space representation known for (monovariable) transfer matrices to the multivariable case.[2] For other comments on the LFR formalism, see, for example, [108, 254].

The above lemma shows that we can handle almost arbitrary algebraic functions of perturbation parameters, provided we define the perturbation matrix Δ as a diagonal matrix, with repeated elements. (In the following, we pay special attention to such diagonal perturbation structures.)

The *uncertainty set assumption* made on the set $\mathbf{\Delta}$ can appear also very specialized. In fact, it can handle a wide array of uncertainty bounds. The following is a short list of examples.

Unstructured case. Assume

$$\mathbf{\Delta} = \{\Delta \in \mathbf{R}^{n_p \times n_q} \mid \|\Delta\| \leq 1\}. \tag{1.14}$$

This case is referred to as the "unstructured perturbations" case and is a classic since the development of $\mathbf{H}_\infty$ control. The corresponding characterization is

$$\Phi_{\mathbf{\Delta}}\left(\begin{bmatrix} q \\ p \end{bmatrix}\begin{bmatrix} q \\ p \end{bmatrix}^T\right) = q^T q - p^T p.$$

Euclidean-norm bounds. Assume

$$\mathbf{\Delta} = \left\{\mathbf{diag}(\delta_1 I_{r_1}, \ldots, \delta_l I_{r_l}) \mid \delta \in \mathbf{R}^l,\ \|\delta\|_2 \leq 1\right\}. \tag{1.15}$$

The corresponding characterization is

$$\Phi_{\mathbf{\Delta}}\left(\begin{bmatrix} q \\ p \end{bmatrix}\begin{bmatrix} q \\ p \end{bmatrix}^T\right) = \mathbf{diag}(q_1 q_1^T, \ldots, q_l q_l^T) - pp^T.$$

(In the above, q_i denotes the ith $r_i \times 1$ block in vector q.)

[2]The above representation is usually referred to as the "linear-fractional transformation" (LFT) [446]. We believe the term representation is more appropriate here.

Maximum-norm bounds. Assume

$$\mathbf{\Delta} = \left\{ \mathbf{diag}(\delta_1 I_{r_1}, \ldots, \delta_l I_{r_l}) \mid \delta \in \mathbf{R}^l,\ \|\delta\|_\infty \leq 1 \right\}. \tag{1.16}$$

The corresponding characterization is

$$\Phi_{\mathbf{\Delta}} \left(\begin{bmatrix} q \\ p \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}^T \right) = \mathbf{diag}(q_1 q_1^T - p_1 p_1^T, \ldots, q_l q_l^T - p_l p_l^T).$$

(Again, p_i (resp., q_i) denotes the ith $r_i \times 1$ block in vector q (resp., p).)

Sector bounds. Assume $\mathbf{\Delta} = \{s \cdot I \mid s \in \mathbf{C},\ \ s + s^* \geq 0\}$. The corresponding characterization is

$$\Phi_{\mathbf{\Delta}} \left(\begin{bmatrix} q \\ p \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}^H \right) = pq^H + qp^H.$$

Of course, it is possible to mix different bounds to cover more complicated cases.

1.3.2 Operator point of view

The above uncertainty models for data matrices can be interpreted as input–output maps, as follows. Given $\Delta \in \mathbf{\Delta}$, the input–output relationship between two vectors u and y,

$$y = \mathbf{M}(\Delta)u,$$

can be rewritten, if $\det(I - D\Delta) \neq 0$, as a *quadratic* one:

$$\begin{array}{rcl} y & = & Mu + Lp, \\ q & = & Ru + Dp, \\ p & = & \Delta q. \end{array}$$

The above representation is not surprising: It shows that a nonlinear (algebraic) constraint can always be seen as a quadratic one, provided we introduce enough "dummy" variables (here, the vectors p, q).

In view of our assumption on the uncertainty set $\mathbf{\Delta}$, the "uncertain constraint"

$$y = \mathbf{M}(\Delta)u \text{ for some } \Delta \in \mathbf{\Delta}$$

can be rewritten in a quadratic matrix inequality form

$$\begin{bmatrix} y \\ q \end{bmatrix} = \begin{bmatrix} M & L \\ R & D \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}, \quad \Phi_{\mathbf{\Delta}} \left(\begin{bmatrix} q \\ p \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}^T \right) \geq 0. \tag{1.17}$$

The operator point of view allows us to consider (linear) dynamical relationships. For example, the LTI system

$$\dot{p} = Ap,$$

where A is a constant matrix, can be represented in the frequency domain as

$$q = Ap, \quad q = s \cdot p.$$

When we need to analyze stability, we restrict ourselves to all complex s such that $\Re s \geq 0$. In this case, the uncertain system to be considered is

$$q = Ap, \quad pq^H + qp^H \geq 0.$$

In section 1.4.6, we follow up this idea in connection with Lyapunov theory.

1.3.3 Examples

Let us now show how the above tools can be used to describe compactly a wide array of uncertain matrices and dynamical systems. To simplify our description, we take the operator point of view mentioned previously.

Matrices

The above framework covers the case when some (matrix) data occurring in the robust decision problem is affected by a perturbation vector δ, in an algebraic manner, and the vector δ is unknown-but-bounded.

Consider, for example, an "uncertain matrix" $\mathbf{M}(\delta)$, where $\mathbf{M}$ is an algebraic function of vector δ, and δ is unknown-but-bounded in the maximum-norm sense. This matrix can be represented as an input–output operator $u \longrightarrow y := \mathbf{M}(\delta)u$ in the form

$$\text{(1.18)} \qquad \begin{bmatrix} y \\ q \end{bmatrix} = \begin{bmatrix} M & L \\ R & D \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}, \quad \Delta = \mathbf{diag}(\delta_1 I_{r_1}, \ldots \delta_l I_{r_l}), \quad \|\Delta\| \leq 1.$$

The above form consitutes a *linear-fractional model* for the uncertain linear constraint $y = \mathbf{M}(\delta)u$, $\|\delta\|_\infty \leq 1$.

Model (1.18) can also be represented in the quadratic matrix inequality form (1.17), where $\Phi_{\mathbf{\Delta}}$ is defined in (1.16).

Dynamical systems

The input–output relationship $y = \mathbf{M}(\Delta)u$ may involve a dynamical system; in this case also, LFR models can be used.

For example, the uncertain dynamical system

$$\dot{y} = \mathbf{M}(\Delta)y, \quad \Delta \in \mathbf{\Delta}$$

can be represented, with the assumptions made previously in force, as

$$\begin{bmatrix} \dot{y} \\ q \end{bmatrix} = \begin{bmatrix} M & L \\ R & D \end{bmatrix} \begin{bmatrix} y \\ p \end{bmatrix}, \quad \Phi_{\mathbf{\Delta}} \left(\begin{bmatrix} q \\ p \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}^T \right) \geq 0.$$

It is possible to consider nonlinear systems with the above representations, using an "embedding" technique. Roughly speaking, if we know bounds on the trajectories of the above uncertain system, then these bounds will also hold for the nonlinear system

$$\dot{y} = \mathbf{M}(\Delta(y))y$$

if $\Delta(y) \in \mathbf{\Delta}$ for every y. (This technique, which is more powerful than the classical linearization technique, is related to the *comparison principle* in differential equations; see [238] for more details.)

We take another example arising in the control of linear systems with delayed input, where we assume only time-delay *interval* uncertainty. Consider the following (frequency-domain) constraint:

$$\text{(1.19)} \qquad y(s) = H(s)u(s), \text{ where } H(s) = H_0(s)e^{-\delta s},$$

where H_0 is a transfer function that represents the nominal system of the form

$$H_0(s) = H_{nd}(s)e^{-\delta_n s},$$

with H_{nd} free of delay and δ_n as the nominal delay. The uncertainty δ is of the form

$$\delta = \rho\Delta,$$

where Δ is an unknown-but-bounded real scalar ($-1 \leq \Delta \leq 1$). A way to handle this class of systems is to aproximate the delay uncertainty term using a first-order real Padé approximant. An LFR of the approximation is

$$e^{-\rho\Delta s} \approx \frac{1 - \frac{\rho}{2}s\Delta}{1 + \frac{\rho}{2}s\Delta} = 1 - \left(1 + \frac{\rho}{2}s\Delta\right)^{-1} \rho s\Delta, \ \Delta \in [-1, 1].$$

Using this LFR, we may obtain an LFR for the delayed system. (An alternate representation of this kind of uncertainty is given below.) Analysis and comparisons between several approximations of the delay uncertainty can be found in [415].

1.3.4 Toward other models

We now briefly mention other possible models. (For other remarks and comments on uncertainty modeling, we refer to [378].)

Integral quadratic constraint models

An alternative to the class of the above parameter-dependent models is one where we assume that some signals in a known linear time-invariant (LTI) system are subject to integral quadratic constraints (IQCs) that are usually written in the frequency domain. Thus, the unknown part of the system is only subject to energy-type constraints. For other comments on this formalism, see, for example, [274, 341].

This class of systems is perhaps more versatile than the class of parameter-dependent systems seen above. We can use this models to "embed" a large class of complicated systems, such as systems with delays, parameter-dependent systems with bounds on the parameter's rate of variation, infinite-dimensional systems, etc.

Consider for example the following system:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bp(t), \\ q(t) &= Cx(t) + Dp(t), \\ p(t) &= \Delta(t)q(t), \end{aligned} \tag{1.20}$$

where the uncertainty block $\Delta(t)$ satisfies an IQC. This IQC is defined as follows: let $\Pi : j\mathbf{R} \mapsto \mathbb{C}^{r\times r}$ be a bounded measurable function, with appropriate r. The corresponding IQC is

$$\int_{-\infty}^{+\infty} \begin{bmatrix} \hat{p}(j\omega) \\ \hat{q}(j\omega) \end{bmatrix}^* \Pi(j\omega) \begin{bmatrix} \hat{p}(j\omega) \\ \hat{q}(j\omega) \end{bmatrix} d\omega \geq 0, \tag{1.21}$$

where $\hat{p}$, $\hat{q}$ are the Fourier transforms of $p, q \in \mathbf{L}_2[0, \infty)$.

As an example, we consider again the delayed system (1.19) with $\delta_n = 0$, and the uncertainty delay interval of the form $[0, \rho]$. A different way to handle the uncertainty on the delay is to suppose that the uncertain "quantity" is $\Delta u(t) = u(t - h) - u(t)$, $h \leq \rho$ (here, $u(\cdot)$ is the input signal). This signal satisfies the IQC above, with (see [341]):

$$\Pi(j\omega) = \begin{bmatrix} 0 & \alpha(j\omega) + j\beta(j\omega) \\ \alpha(j\omega) - j\beta(j\omega) & -\alpha(j\omega) - j\beta(j\omega)cot\left(\frac{\rho\omega}{2}\right) \end{bmatrix},$$

where α and β are bounded real-valued functions on the imaginary axis such that

$$\begin{cases} \omega\beta(j\omega) \geq 0 \ \text{ for } \ |\omega| \leq \frac{\pi}{\rho}, \\ \beta(j\omega) = 0 \ \text{ otherwise.} \end{cases}$$

IQCs are examined in more detail in Chapter 6 of this book.

Polytopic models

In a polytopic model [64], we assume that the state-space matrices of the system are only known to lie in a given polytope. For example,

$$\dot{x} = \mathbf{A}(t)x, \text{ where } \mathbf{A}(t) \in \mathbf{Co}\{A_1, \ldots, A_L\} \text{ for every } t \geq 0. \tag{1.22}$$

Here, the set of admissible perturbations of $\mathcal{D}$ is the polytope $\{\lambda \mid \lambda_i \geq 0, \sum \lambda_i = 1\}$, and the mapping ϕ assigns the linear combination $\sum \lambda_i A_i$ to the vector λ. Polytopic systems are considered in Chapter 9 of this book.

Refined perturbation sets

The above models are very rough in that they impose no restriction on the rate of change of the perturbed elements. For example, the polytopic model (1.22) could be used for a system subject to possible failures; each matrix A_i in (1.22) represents a specific operating mode for the system. In practice, this model will be very conservative, as it does not prevent the system from jumping arbitrarily fast from a mode to any other. Some refined models simply impose a bound on the rate of variation of the perturbation parameters and make use of LMI analysis, which takes into account this bound.

Stochastic models

These models are often used in engineering to describe uncertainty and could be interpreted as more refined than deterministic ones. For example, for a system with several failure modes, a model that is more refined than the polytopic model could be based on systems with Markovian jumps. With the matrices A_i of the polytopic model, we associate a "transition probability matrix" that describes the probability of change from one mode to the other. Similarly, it is possible to consider a linear-fractional model, where the uncertain matrix Δ obeys a known white-noise-type distribution; see [118] for an example.

The reader should not make a conflict out of the distinction deterministic/stochastic: it is possible to consider stochastic models with deterministic uncertainty. For example, we may consider a linear system with Markovian jumps, where the transition probability matrix is unknown-but-bounded (see [121]).

1.4 Robust SDP

Robust decision problems are defined in terms of (a) a "nominal" problem of the form (1.3) and (b) an uncertainty model for the data. Equipped with (matrix-based) uncertainty models for the data described previously, we need to define more precisely the form of the nominal problem. It turns out that a class of problems very rich in applications is precisely an SDP. This motivates the study of *robust SDP* problems, which correspond to an SDP with uncertain data.

1.4.1 Definition

Consider the case when the constraints $f_i(x, \delta) \leq 0$, $i = 1, \ldots, p$, can be written as

$$\mathbf{F}(x, \Delta) \geq 0,$$

where $\mathbf{F}$ is affine in the decision variable x and takes its values in the set of symmetric matrices.

We make the assumption that $\mathbf{F}$ is a linear-fractional function of Δ, with LFR

$$\mathbf{F}(x,\Delta) = F(x) + L(x)\Delta(I - D\Delta)^{-1}R + \left(L(x)\Delta(I - D\Delta)^{-1}R\right)^T,$$

where $F(x) = F(x)^T$ and $L(x)$ are affine matrix-valued functions of x, and R, D are given matrices, while the uncertainty matrix Δ is restricted to a set $\mathbf{\Delta}$. We assume that the above LFR is well posed over $\mathbf{\Delta}$, and, in addition, we assume that the set $\mathbf{\Delta}$ satisfies the *uncertainty set assumption* defined in section 1.3.1.

The problem

$$\text{minimize } c^T x \text{ subject to } \mathbf{F}(x,\Delta) \geq 0 \text{ for every } \Delta \in \mathbf{\Delta}$$

is called a *robust SDP* problem. Note that we assumed that the "objective vector" c is independent of perturbation; this is done without loss of generality.

The above problem is convex; however, it is in general very difficult to solve (NP-hard; see [154, 323]). Checking if a given candidate solution x is robustly feasible is already very difficult in general. Our objective is to find lower bounds on this problem in the form of SDP.

1.4.2 Lagrange relaxation of a robust SDP

Let us fix the decision variable x and seek a sufficient condition for

$$\xi^T \mathbf{F}(x,\Delta)\xi \geq 0 \text{ for every } \xi, \ \ \Delta \in \mathbf{\Delta}.$$

Using the LFR, and using the well-posedness assumption, we rewrite the above as

$$\begin{gathered}\xi^T\left(F(x)\xi + 2L(x)p\right) \geq 0 \text{ for every } \xi, p, q, \Delta \text{ such that} \\ q = R\xi + Dp, \ \ p = \Delta q, \ \ \Delta \in \mathbf{\Delta}.\end{gathered}$$

With the previous characterization $p = \Delta q$, the above is seen to be equivalent to

$$\begin{gathered}\mathbf{Tr}\begin{bmatrix} F(x) & L(x) \\ L(x)^T & 0 \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix}^T \geq 0 \text{ for every } \xi, p \\ \text{such that } \Phi_{\mathbf{\Delta}}\left(\begin{bmatrix} R & D \\ 0 & I \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix}^T\begin{bmatrix} R & D \\ 0 & I \end{bmatrix}\right) \geq 0.\end{gathered} \tag{1.23}$$

We are ready now to use Lagrange relaxation, following the general ideas evoked in section 1.2.4. The previous condition holds if there exists a positive semidefinite matrix S such that $\forall\ \xi, p$,

$$\begin{bmatrix} \xi \\ p \end{bmatrix}^T\begin{bmatrix} F(x) & L(x) \\ L(x)^T & 0 \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix} \geq \mathbf{Tr} S\Phi_{\mathbf{\Delta}}\left(\begin{bmatrix} R & D \\ 0 & I \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix}\begin{bmatrix} \xi \\ p \end{bmatrix}^T\begin{bmatrix} R & D \\ 0 & I \end{bmatrix}\right)$$

or, equivalently,

$$\begin{bmatrix} F(x) & L(x) \\ L(x)^T & 0 \end{bmatrix} \geq \begin{bmatrix} R & D \\ 0 & I \end{bmatrix}^T \Phi_{\mathbf{\Delta}}^*(S)\begin{bmatrix} R & D \\ 0 & I \end{bmatrix}, \tag{1.24}$$

where $\Phi_{\mathbf{\Delta}}^*$ is the dual map of $\Phi_{\mathbf{\Delta}}$.

Hence, the Lagrange relaxation of robust SDP is an SDP of the form

$$\min\ c^T x \ \text{ subject to } S \geq 0 \text{ and (1.24)}.$$

The variables in this SDP are x (the original decision variable) and S (the "multiplier" matrix). The SDP provides an approximation of the robust SDP in the following senses: (a) it yields an upper bound on the robust SDP; (b) any optimal x is robust (i.e., it is feasible $\forall\ \Delta \in \mathbf{\Delta}$); (c) in the unstructured case ($\mathbf{\Delta}$ defined by (1.14)), the approximation is exact.

A special case: Robust linear programming (LP)

Consider the LP

$$\text{minimize } c^T x \text{ subject to } a_i^T x \geq b_i,\ i = 1, \ldots, L. \tag{1.25}$$

Assume that the a_i's and b_i's are subject to unstructured perturbations. That is, the perturbed value of $[a_i^T\ b_i]^T$ is $[a_i^T\ b_i]^T + \delta_i$, where $\|\delta_i\|_2 \leq \rho$, $i = 1, \ldots, L$, and $\rho > 0$ is a given measure of the perturbation level. We seek a robust solution to our problem; that is, we seek x that minimizes $c^T x$ subject to the robustness constraints

$$\left(\begin{bmatrix} a_i \\ b_i \end{bmatrix} + \delta_i \right)^T \begin{bmatrix} x \\ -1 \end{bmatrix} \leq 0 \text{ for every } \delta_i, \quad \|\delta_i\|_2 \leq \rho, \quad i = 1, \ldots, L.$$

This is a special case of a robust optimality problem, as defined in section 1.2.3.

Using the previous theory, it is straightforward to show that the robust LP is *equivalent* to the problem

$$\text{minimize } c^T x \text{ subject to } a_i^T x - \rho \sqrt{\|x\|_2^2 + 1} \geq b_i,\ i = 1, \ldots, L. \tag{1.26}$$

The above problem is a second-order cone program (SOCP) for which efficient special-purpose interior-point methods are available (see section 1.5.1).

We note that for every $\rho > 0$, the above problem yields a unique solution, that is, a regular (in particular, continuous) function of the problem's data (the a_i's and the b_i's). Thus, the approach provides a regularization procedure for the LP (1.25).

A simple variation of the robust LP is *LP with implementation constraints*, where we seek a solution x such that for every δx, $\|\delta x\| \leq \rho$, the modified vector $x + \delta x$ still satisfies the constraint of the original LP. This problem arises for example in the area of optimal filter design, where the filter's coefficients can be implemented only with finite precision.

1.4.3 A dual view: Rank relaxation

Lagrange relaxation can be interpreted in a dual way, in terms of relaxation of a rank constraint of the rank-one, positive semidefinite matrix

$$X = \begin{bmatrix} \xi \\ p \end{bmatrix} \begin{bmatrix} \xi \\ p \end{bmatrix}^T,$$

which appears in (1.23). Define

$$N := \begin{bmatrix} R & D \\ 0 & I \end{bmatrix}, \quad M(x) := \begin{bmatrix} F(x) & L(x) \\ L(x)^T & 0 \end{bmatrix} = M_0 + \sum_{i=1}^{m} x_i M_i.$$

Define also the convex set

$$\mathcal{X} := \left\{ X \mid \Phi_{\Delta}\left(NXN^T\right) \geq 0,\ X \geq 0 \right\}.$$

Let us fix the decision variable x again. Condition (1.23) holds if and only if

$$0 \geq \max \left\{ -\mathbf{Tr} M(x) X \mid X \in \mathcal{X},\ \mathbf{rank} X = 1 \right\}.$$

The robust SDP problem writes as the max-min problem

$$\min_x \max_X \left\{ c^T x - \mathbf{Tr} M(x) X \mid X \in \mathcal{X},\ \mathbf{rank} X = 1 \right\}.$$

Relaxing the rank constraint yields *lower* bound

$$\min_x \max_{X \in \mathcal{X}} c^T x - \mathbf{Tr} M(x) X,$$

which, by virtue of the saddle-point theorem (recall $\mathcal{X}$ is convex), is equivalent to the SDP in variable X

$$\max -\mathbf{Tr} M(0) X \quad : \quad \begin{array}{l} \mathbf{Tr} M_i X = c_i, \ 1 \le i \le m, \\ \Phi_{\mathbf{\Delta}} \left(N X N^T \right) \ge 0, \ \ X \ge 0. \end{array}$$

The above is the dual of the SDP obtained via Lagrange relaxation. We will see in section 1.7.1 how this connection between Lagrange and rank relaxation has been recognized earlier in the special case of combinatorial optimization problems. For a related point of view on the geometrical properties of rank-constrained symmetric matrices, see [192] and references therein.

1.4.4 Extension to nonconvex nominal problems

In many cases, notably in control synthesis, we encounter problems where the nominal problem is nonconvex. Most of these problems can be turned into *cone complementarity* problems (see section 1.5.2 for more details).

We have seen how to handle uncertainty in (convex) SDPs, which are *symmetric* eigenvalue problems. To illustrate the ideas, let us consider a (nonconvex) *unsymmetric* eigenvalue problem. For example, consider the following robust synthesis problem:

$$\min_x \max_{\Delta \in \mathbf{\Delta}} \rho \left(\mathbf{M}(x, \Delta) \right),$$

where $\rho(\cdot)$ denotes the spectral radius, and $\mathbf{M}(x, \Delta)$ is affine in x and rational in $\Delta \in \mathbf{\Delta}$, given in an LFR

$$\mathbf{M}(x, \Delta) = M(x) + L(x) \Delta (I - D\Delta)^{-1} R.$$

(We note that the fact that $\mathbf{M}$ is affine in x is done without much loss of generality.)

Robustness analysis

Let us fix λ, x, and seek sufficient condition for

$$\rho \left(\mathbf{M}(x, \Delta) \right) \le \lambda \text{ for every } \Delta \in \mathbf{\Delta}.$$

Let us first "forget" uncertainty and apply Lagrange relaxation to the problem of checking if

$$(zI - M)^H (zI - M) \le \lambda^2 I \text{ for every } z, \ |z| \le 1.$$

We obtain that λ is an upper bound on spectral radius if and only if there exist $X > 0$ such that

$$\lambda^2 X \ge M^T X M.$$

The above condition is an LMI in X, which implies that we can apply the robust SDP methodology to handle the case when M is uncertain. We obtain that λ is an upper bound on maximal spectral radius if there exist $P > 0$ and S such that

$$\begin{bmatrix} \lambda^2 P & 0 \\ 0 & 0 \end{bmatrix} \ge \begin{bmatrix} M^T(x) \\ L^T(x) \end{bmatrix} P \begin{bmatrix} M^T(x) \\ L^T(x) \end{bmatrix}^T + \begin{bmatrix} R & D \\ 0 & I \end{bmatrix}^T \Phi_{\mathbf{\Delta}}^*(S) \begin{bmatrix} R & D \\ 0 & I \end{bmatrix}.$$

For fixed x, minimizing λ a (quasi-) convex problem (over S and P) results in an *upper* bound on the worst-case spectral radius.

Robust synthesis

We now seek a sufficient condition, λ being fixed, so that

$$\exists\, x \text{ such that } \rho(\mathrm{M}(x,\Delta)) \leq \lambda \text{ for every } \Delta \in \mathbf{\Delta} \tag{1.27}$$

holds. Now x becomes a variable. Due to possible cross products between x and P in the previous sufficient condition, the problem is not convex, contrary to what happens in the robust SDP case.

Let $X = \lambda^2 P$, $Y = P^{-1}$, and write problem as a *conic complementarity problem*

$$\min \mathbf{Tr}XY \text{ subject to } \begin{bmatrix} X & \lambda I \\ \lambda I & Y \end{bmatrix} \geq 0,$$

$$\begin{bmatrix} X & 0 & M(x)^T \\ 0 & 0 & L(x)^T \\ M(x) & L(x) & Y \end{bmatrix} \geq \begin{bmatrix} R & D & 0 \\ 0 & I & 0 \end{bmatrix}^T \Phi_{\mathbf{\Delta}}^*(S) \begin{bmatrix} R & D & 0 \\ 0 & I & 0 \end{bmatrix}.$$

Condition (1.27) is true if and only if the optimum is $\lambda^2 n$.

The problem we obtain is a generalization of the SDP, in the sense that the SDP is a special complementarity problem (see section 1.5). In general, the above problem is not convex; however, several efficient techniques can be used (see Part V). One of these techniques, described in Chapters 16 and 17, relies on a simple linearization of the quadratic objective and leads to a sequence of SDPs. In some cases, this approximation technique is guaranteed to yield the global optimum of the original problem (see [128]).

1.4.5 On quality of relaxations

Associated with the relaxation methods comes the need to evaluate the quality of the approximations involved. In the context of a robust SDP, the first results are due to Nemirovski [39].

To illustrate the ideas involved in this analysis, consider the following robustness analysis problem: Given l symmetric $n \times n$ $F_1, \ldots, F_l$, and a scalar ρ, check if

$$\mathcal{P}_\rho \;:\; I \geq \sum_{i=1}^{l} \delta_i F_i \text{ for every } \delta,\ \|\delta\|_2 \leq \rho. \tag{1.28}$$

This seemingly simple problem is NP-hard. If we apply Lagrange relaxation, we obtain that a sufficient condition for property $\mathcal{P}_1$ (i.e., $\rho = 1$ in (1.28)) to hold is

$$\exists S_i > 0,\ \ i = 1, \ldots, l,\ \ 2 \cdot I \geq \sum_{i=1}^{l} (F_i S_i^{-1} F_i + S_i). \tag{1.29}$$

To evaluate the quality of the above approximation, we seek the smallest ρ such that condition (1.29) *implies* $\mathcal{P}_\rho$ holds. Clearly, such a ρ satisfies $\rho \geq 1$; the closer to 1, the better the approximation is.

Nemirovski has proved that $\rho \leq \min(\sqrt{n}, \sqrt{l})$. In terms of control, we interpret this result as a bound on the quality of the approximation to a kind of "μ-analysis" problem [108], with affine dependences and Euclidean-norm bounds. More refined results, including for other norm bounds, are underway [290].

1.4.6 Link with Lyapunov functions

There is more than one connection between the approach proposed here and Lyapunov theory; we will briefly hint at these connections here.

Lyapunov functions and Lagrange multipliers

Lyapunov functions can be interpreted in the context of Lagrange relaxations, which establish an interesting link between two "central" methods in control and optimization.

For the sake of simplicity, we first consider the problem of establishing stability of the LTI system

$$\dot{x} = Ax,$$

where $x \in \mathbf{R}^n$ is the state, and $A \in \mathbf{R}^{n\times n}$ is a (constant) square matrix. It is well known that the system is stable if and only if

$$(sI - A)^H(sI - A) > 0 \text{ for every } s, \;\; s + s^* \geq 0.$$

This shows that the stability problem is a robustness analysis problem (with respect to "uncertain" parameter s).

To attack it using Lagrange relaxation, we first use an LFR of the problem. Introduce $p = sx$; our robustness analysis problem is equivalent to checking if

$$\|p - Ax\|^2 > 0 \text{ for every } (p, x) \neq (0, 0), \;\; p = sx \text{ for some } s, \;\; s + s^* \geq 0.$$

Let us eliminate the "uncertain parameter" s by noting that

$$p = sx \text{ for some } s, \;\; s + s^* \geq 0 \Longleftrightarrow px^H + xp^H \geq 0.$$

Let us apply a Lagrange relaxation idea now. A sufficient condition for stability is that there exists a "multiplier" matrix $S > 0$ such that

$$\forall (p, x) \neq (0, 0), \;\; \|p - Ax\|^2 > \mathbf{Tr} S(px^H + xp^H).$$

A simple computation leads to the classical Lyapunov condition

$$\exists \, S \text{ such that } A^T S + SA < 0, \;\; S > 0,$$

which corresponds to the Lyapunov function candidate $V(x) = x^T Sx$.

Let us give another example of the "Lagrange relaxation" method for dynamical systems. Consider a system with delayed state

$$\dot{x}(t) = Ax(t) + A_d x(t - h). \tag{1.30}$$

The system is stable *independent of the delay* $\tau \geq 0$ if

$$\det(sI - A - zA_d) \neq 0 \text{ for every } s, z, \;\; s + s^* \geq 0 \text{ and } |z| \leq 1.$$

An equivalent stability condition is that

$$\|p - Aq - A_d p_d\|^2 > 0 \text{ for every } (p, p_d, q) \neq (0, 0, 0) \text{ such that}$$
$$pq^H + qp^H \geq 0, \;\; p_d p_d^H \leq qq^H.$$

This time, we may relax the *two* matrix inequality constraints, which after some matrix calculations yields the *sufficient* (possibly conservative) condition for stability: There exists symmetric matrices S, Z such that $S > 0$, $Z > 0$, and

$$\begin{bmatrix} A^T S + SA + Z & SA_d \\ A_d^T S & -Z \end{bmatrix} < 0.$$

To this condition, we can associate a "Lyapunov function candidate," termed as a Lyapunov–Krasovskii functional, defined as

$$V = x(t)^T Sx(t) + \int_{-\tau}^{0} x(t + \theta)^T Zx(t + \theta) \, d\theta.$$

Quadratic stability

The robust decision methodology can also be applied in the quadratic stability analysis for an uncertain system subject to unknown-but-bounded uncertainty:

$$\dot{x} = \mathbf{A}(\Delta)x, \quad \Delta \in \mathbf{\Delta},$$

where $\mathbf{\Delta}$ satisfies the assumptions of section 1.3.1 and $\mathbf{A}$ is defined via the LFR

$$\mathbf{A}(\Delta) = A + B_p\Delta(I - D_{qp}\Delta)^{-1}C_q. \tag{1.31}$$

(Recall that we assume that this LFR is well posed over $\mathbf{\Delta}$.)

To analyze stability, the idea is to apply Lagrange relaxation twice. In a first step, we "forget" uncertainty and apply Lagrange relaxation to the "frozen" system. This results in a sufficient condition for stability: There exists a matrix S such that

$$\mathbf{A}(\Delta)^T S + S\mathbf{A}(\Delta) < 0, \quad S > 0 \text{ for every } \Delta \in \mathbf{\Delta}.$$

This condition is known as a quadratic stability condition, since it requires the existence of a quadratic Lyapunov function proving stability of the uncertain system.

In general, the above condition is still not tractable, despite the fact it is a convex condition on S. However, the relaxed problem above now belongs to the realm of robust SDP. We can apply Lagrange relaxation again. We obtain that if there exist matrices S and W such that

$$\begin{bmatrix} A^TS + SA & B_p \\ B_p^T & 0 \end{bmatrix} + \begin{bmatrix} C_q & D_{qp} \\ 0 & I \end{bmatrix}^T \Phi_{\mathbf{\Delta}}^*(W) \begin{bmatrix} C_q & D_{qp} \\ 0 & I \end{bmatrix} < 0, \tag{1.32}$$
$$W > 0,$$

holds, where $\Phi_{\mathbf{\Delta}}$ is the linear operator that characterizes the uncertainty set $\mathbf{\Delta}$, then the system is quadratically stable.

The above problem is an SDP, with decision variable S (linked to the quadratic Lyapunov function proving quadratic stability), and W. When $\mathbf{\Delta}$ is the set defined by (1.14), the above condition is the well-known small gain theorem (see [64] and references therein).

The above idea can be extended in various ways, as illustrated in some other chapters (Parts III and IV) of this book. For example, we may consider parameter-dependent, or frequency-dependent Lyapunov functions, based on so-called IQCs.

Stochastic Lyapunov functions

Stochastic Lyapunov functions have been introduced by Kushner [237], and their use in the context of LMI optimization has been introduced in [64, 120, 121]. Consider a stochastic system of the form

$$dx = (Adt + Gdw)\,x,$$

where w is a scalar standard Wiener process (the "derivative" of this process can be interpreted as a random perturbation on the state-space matrix, hence the name multiplicative noise given to such perturbations). The above systems are useful models in a number of areas, most notably finance [256].

The second moment $X = \mathbf{E}(xx^T)$ obeys the deterministic equation

$$\dot{X} = AX + XA^T + GXG^T. \tag{1.33}$$

The stability of this system hinges on the existence of a *linear* Lyapunov function

$$V(X) = \mathbf{Tr}(XS) = \mathbf{E}(x^T S x), \tag{1.34}$$

where $S > 0$, which proves stability of the deterministic system (1.33) (evolving over the positive semidefinite cone). It can be shown that the (second-moment) stability is guaranteed if and only if

$$S > 0, \quad A^T S + SA + G^T S G < 0.$$

A stochastic Lyapunov function (in the sense of Kushner), namely, $v(x) = \mathbf{E}(x^T S x)$, corresponds to this Lyapunov function.

One of the advantages of LMI formulations to stochastic control problems is that once an LMI formulation is known, it is possible to apply Lagrange relaxation in the case when the "data" (for example, the "nominal" matrix A in the above) is uncertain.

1.5 Algorithms and software for SDP

Algorithms for SDPs are now the subject of intense research in the field of optimization, especially since 1995. This progress should have a great impact on LMI-based control as algorithms become more efficient. Recent methods to solve a general SDP can be roughly divided in two classes: interior-point methods and nondifferentiable optimization methods. These algorithms have been implemented, and several interior-point software packages are now generally available.

1.5.1 Facts about SDP and related problems

We briefly recall some important results on SDP and related problems.

Duality

The problem dual to problem (1.37) is

$$\begin{array}{ll} \text{maximize} & -\mathbf{Tr} F_0 Z \\ \text{subject to} & Z \geq 0, \ \mathbf{Tr} F_i Z = c_i, \ i = 1, \ldots, m, \end{array} \tag{1.35}$$

where Z is a symmetric $N \times N$ matrix and c_i is the ith coordinate of vector c. When both problems are strictly feasible (that is, when there exist x, Z, which satisfy the constraints strictly), the existence of optimal points is guaranteed [296, Theorem 4.2.1], and both problems have equal optimal objectives.

Related convex problems

We list several problems that are related to SDPs.

Second-order cone programming problems are special cases of SDPs. They take the form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \|C_i x + d_i\| \leq e_i^T x + f_i, \ \ i = 1, \ldots, L, \end{array} \tag{1.36}$$

where $C_i \in \mathbf{R}^{n_i \times m}$, $d_i \in \mathbf{R}^{n_i}$, $e_i \in \mathbf{R}^m$, $f_i \in \mathbf{R}$, $i = 1, \ldots, L$. The special structure of SOCPs can be exploited in algorithms to yield much faster methods than using general-purpose SDP methods; see, e.g., [296, 240, 9], as well as Chapter 3 of this book.

Determinant maximization (max-det) problems are of the form

$$\text{minimize } c^T x - \log\det G(x) \text{ subject to } F(x) \geq 0, \tag{1.37}$$

where G is an affine function taking values in the set of symmetric matrices of order p. Max-det problems are convex optimization problems and can be solved in polynomial time with interior-point methods [296, 402]. Chapter 2 of this book provides more details on max-det problems. Other interesting applications of max-det problems are reviewed in [406].

Generalized eigenvalue problems (GEVPs) are generalizations of SDPs, of the form

$$\text{minimize } \lambda \text{ subject to } \lambda B(x) - A(x) \geq 0, \quad B(x) \geq 0, \quad C(x) \geq 0.$$

Here, A, B, and C are symmetric matrices that depend affinely on $x \in \mathbf{R}^m$. GEVPs are not convex but are quasi convex and amenable to interior-point methods similar to those used for SDPs; see [63, 295].

1.5.2 Interior-point methods

One way to view interior-point methods is as a reduction of the original problem to a sequence of differentiable, unconstrained minimization subproblems. Each subproblem can then be efficiently solved, perhaps approximately. These methods can be described in the context of self-concordant barrier and potential functions; this is the point of view taken in [296, 404].

Primal-dual interior-point methods

Without doubt, the most efficient interior-point methods for LP are the primal-dual interior-point methods [423]. When a primal-dual interior-point method is used, each subproblem can be described by a linear least squares problem or, equivalently, a system of linear equations. These methods are very efficient for four reasons: (1) they enjoy robust convergence from infeasible starting points, with guaranteed complexity when implemented accordingly (though typically this is not done in practice); (2) they converge rapidly, even if the solutions are not unique; (3) the linear systems to be solved are sparse (or can be made sparse by, for example, using standard techniques to handle dense columns in the constraint matrix); consequently, very large problems can be solved; (4) the inherent ill conditioning in the linear systems to be solved does not, in practice, greatly limit achievable accuracy. However, this ill-conditioning does mean that iterative methods (e.g., conjugate gradient) are not generally practical to solve the linear systems: the standard workhorse is sparse Cholesky.

Since 1995, there has been a flurry of papers on primal-dual interior-point methods for SDPs; a complete bibliography on this subject is outside the scope of this chapter. Relevant references include [297, 8, 233, 286, 338, 391, 190, 404, 5, 219]. In these methods, the subproblem to be solved is again simply a system of linear equations, although this is not always equivalent to a least-squares problem. As in LP, these methods converge very rapidly; various complexity results are now available; software packages have also recently become available. However, primal-dual methods are not necessarily the best at exploiting sparsity in the data. The linear system that must be solved at every iteration invariably has dimension m, the number of variables in the primal form of the SDP. Therefore, with present technology, primal-dual methods are highly efficient if m is not too large (say, less than 1000), but if it is much larger, they are not practical unless sparsity can be exploited. See [38] for a primal alternative to primal-dual methods that

quite effectively exploits sparsity in SDPs arising in applications to graph theory. Even this approach, however, is limited by the size of linear systems that can be solved.

In what follows, we briefly describe primal-dual path-following methods, following Kojima et al. [233].

Optimality conditions and central path

When the primal problem (1.2) and its dual (1.35) are both strictly feasible, then we may write the optimality conditions

$$ZF = 0, \quad F \succeq 0, \quad Z \succeq 0, \quad (F, Z) \in \mathcal{L}, \tag{1.38}$$

where $\mathcal{L}$ is an affine subspace:

$$\mathcal{L} = \left\{ (F, Z) \;\middle|\; \begin{array}{l} F = F(x) \text{ for some } x \in \mathbf{R}^m \\ c_i = \mathbf{Tr} Z F_i, \quad i = 1, \ldots, m \end{array} \right\}. \tag{1.39}$$

We can interpret the above conditions as complementarity conditions over the positive semidefinite cone, similar to those arising in LP [233]. The convexity of the original SDP is transported here into a property of monotonicity of $\mathcal{L}$, which is crucial for the algorithm outlined below to converge globally.

We introduce a parameter μ that modifies the above conditions, as follows:

$$ZF = \mu I, \quad F \succeq 0, \quad Z \succeq 0, \quad (F, Z) \in \mathcal{L}. \tag{1.40}$$

Under strict feasibility conditions, the above system defines a unique primal-dual pair (F_μ, Z_μ). Solving (1.40) for decreasing values of the parameter μ leads to an optimal solution. The path followed by (F_μ, Z_μ) as $\mu \to 0$ is referred to as the central path, which explains the qualification "path following."

For each value of μ, the conditions (1.40) can be interpreted as the optimality conditions for a differentiable problem involving a barrier function. Thus, the above description of interior-point methods can be interpreted in the context of barrier functions [296, 404].

Predictor and corrector steps

Assume that we have found a pair (F_k, Z_k) that is close to the central path (i.e., approximately solve (1.40)). To find the new iterate, two steps are taken. In the *predictor* step, we seek to approach the optimum, that is, to satisfy $ZF = 0$. In the *corrector* step, we seek to return close to the central path by ensuring $ZF = \mu I$.

The search directions $\delta x, \delta Z$ for each step (predictor and corrector) are computed by linearization of the constraint $ZF = 0$ (predictor) or $ZF = \mu I$ (corrector) around the current value of (x, Z). Each step thus gives rise to a linear system in the elements of δx, δZ. Note that ZF can be linearized in a number of ways, depending on the specific method used.

The step lengths may be chosen small enough so that the new iterates will be guaranteed to be close to the central path again (and thus, strictly feasible). Here, closeness is measured using a proximity function such as

$$\| Z^{1/2} F Z^{1/2} - \mu I \|.$$

(Many alternatives are possible; see [233].) Depending on the proximity function used, we may define long-step and short-step strategies (see, for example, [286]).

There are two different approaches to initialization that have become standard. One approach uses an "infeasible start," i.e., points that satisfy the cone inequality constraints

but not the linear equality constraints. If the original problem turns out to be either primal or dual infeasible, iterates with large norms will typically be generated, alerting the user that the problem is most likely either primal or dual infeasible. The other approach is to construct a modified problem for which feasible starting points are known, which then guarantees that either the original problem will be solved or a certificate of infeasibility will be found [298]. Although this approach is very attractive theoretically, it remains unclear whether it is better than the infeasible-start approach in practice.

In fact, the whole issue of feasibility is much more subtle for SDP than it is for LP. An interesting discussion on this subject can be found in [257].

1.5.3 Nondifferentiable optimization methods

When the number of variables is sufficiently large, and sparsity cannot be effectively exploited, it may be advantageous to consider alternative methods to solve SDPs. For simplicity, we only discuss an LMI feasibility problem, where a solution to a set of LMIs is sought; such problems appear routinely in control, e.g., in stability analysis.

An LMI feasibility problem is easily formulated as an eigenvalue minimization problem of the form

$$\text{minimize } \lambda_{\max}(F(x)), \tag{1.41}$$

where $F(x)$ is an affine function taking values in the set of real symmetric matrices and $\lambda_{\max}$ denotes largest eigenvalue. (To see this, note that there exists an x such that $F(x) \leq 0$ if and only if the minimum of $\lambda_{\max}(F(x))$ is below zero.) The function $f(x) = \lambda_{\max}(F(x))$ is a nondifferentiable function; yet it is convex and all theoretical tools from convex analysis apply to this function: In particular, first-order and second-order generalized derivatives can be explicitly quantified and numerically implemented for this class of functions [305, 397, 373, 310, 137].

In Chapter 3 of this book, an extensive review of existing nondifferentiable optimization methods (NDOMs) is proposed ranging from early *subgradient methods* to the modern *bundle methods*. Chapter 3 also includes some very recent developments on *second-order bundle methods* as well as numerical results of a new type: a mixed polyhedral-semidefinite bundle method enables us to check the stability of polytopic linear differential inclusions with 1000 state variables (which implies more than $m = 500{,}000$ variables). A second-order bundle method is shown to bring in practice an asymptotic superlinear rate of convergence for smaller problems. Some perspectives are sketched to see how the latter methods could be made efficient on large-scale problems.

1.5.4 Software

Software for solving SDPs is generally available, both freely on the Internet and commercially. (To our knowledge, most of these SDP algorithms use interior-point methods.) Much of this has become available quite recently. Here is a (probably incomplete) list:

- The Matlab control toolbox, developed by Gahinet and others [153] uses Nesterov and Nemirovski's projective algorithm [296, 293].

- The code **sdpsol**, described in Chapter 2 of this book, is a parser/solver for SDPs and max-det problems. **sdpsol** uses the code SP, developed by Vandenberghe and Boyd [402].
  `http://www.stanford.edu/~boyd/group_index.html`
  The toolbox `lmitool`, developed by El Ghaoui, Nikoukha, and Delebecque [126],

and later by Commeau, is an interface to the codes SP, SDPpack, and SDPHA.
`http://www.ensta.fr/~gropco/`

- The code **SDPpack**, outlined in Chapter 3 of this book, implements a primal-dual algorithm due to Alizadeh, Haeberly, and Overton [8] and Alizadeh et al. [6].
`http://www.cs.nyu.edu/cs/faculty/overton/sdppack/sdppack.html`
- The code **SDPA**: developed by Fujisawa, Kojima, and Nakata; the underlying algorithm is described in [233].
`ftp://ftp.is.titech.ac.jp/pub/OpRes/software/SDPA`
- The code **SDPHA**: developed by Brixius, Sheng, and Potra; for a description of the underlying algorithm, see the web page.
`http://www.cs.uiowa.edu/brixius/SDPHA`
- The code **SDPT3**: uses the algoritm of Todd, Toh, and Tütüncü [391].
`http://www.math.nus.sg/~mattohkc/SDPT3.tar.Z`
- The code **SeDuMi**: developed by Sturm.
`http://www.crl.mcmaster.ca/ASPC/people/sturm/software/`

1.6 Illustrations in control

This book abounds with examples of applications of the LMI framework to control problems. In this section, we seek to provide complementary examples.

1.6.1 Optimal control and SDP duality

Notions central to SDP theory, such as duality, optimality conditions, sensitivity analysis, etc., have usually interesting interpretations for the analysis of (optimal) control problems. For example, the perturbation theory for the general SDP recently developed by Shapiro [372, 373] and Bonnans, Cominetti, and Shapiro [55] can be used for the analysis of Lyapunov and Riccati inequalities and equations encountered in optimal control.

The following example shows how SDP duality and optimality theory can be used in an optimal control problem. In a variety of stochastic control problems, a Riccati equation of the following form [422] arises:

$$A^TP + PA - PB^TR^{-1}BP + G^TPG + Q = 0, \tag{1.42}$$

where $Q = Q^T \geq 0$, $R > 0$, and A, B, G are matrices of appropriate sizes. When $G = 0$, we recover a standard Riccati equation (which can be solved via the eigenvectors of a Hamiltonian matrix). In the general case $G \neq 0$, the problem of computing the solution is nontrivial.

Under hypotheses that are physically reasonable, detailed below, we can solve the above equation using an SDP as follows. Consider the SDP

$$\begin{array}{c} \text{maximize } \mathbf{Tr}P \text{ subject to} \\ \left[\begin{array}{cc} A^TP + PA + Q + G^TPG & PB \\ B^TP & R \end{array}\right] \geq 0, \quad P = P^T. \end{array} \tag{1.43}$$

We form the problem dual to (1.43) (see section 1.5):

$$\begin{array}{c} \text{minimize } \mathbf{Tr}(XU + SR) \text{ subject to} \\ \left[\begin{array}{cc} X & U^T \\ U & S \end{array}\right] \geq 0, \quad AX + BU + (AX + BU)^T + GXG^T + I = 0. \end{array} \tag{1.44}$$

The above problem is strictly feasible if and only if the underlying stochastic system is stabilizable in a stochastic sense (with static state-feedback control law $u = Kx$, $K = UX^{-1}$). Likewise, it is easy to show that the primal problem (1.43) is strictly feasible when this system is detectable. Let us make the hypothesis of primal and dual strict feasibility.

Since both primal and dual problems are strictly feasible, it follows that a set of feasible primal-dual variables (P, X, U, S) are optimal if and only if

$$\begin{bmatrix} X & U^T \\ U & S \end{bmatrix} \begin{bmatrix} A^TP + PA + Q + G^TPG & PB \\ B^TP & R \end{bmatrix} = 0,$$

from which we get

$$\begin{aligned} X(A^TP + PA + Q + G^TPG) + U^TB^TP &= 0, \\ XPB + U^TR &= 0. \end{aligned} \tag{1.45}$$

With $R > 0$, we obtain $U = KX$, where $K = -R^{-1}B^TP$. The first equation of (1.45) then writes

$$X(A^TP + PA - PB^TR^{-1}BP + Q + G^TPG) = 0.$$

Finally, we observe that $X > 0$: if $Xz = 0$ for some z, then $Uz = 0$ (from $U = KX$). Since U, X are feasible for (1.44), we obtain

$$0 = z^T(AX + BU + (AX + BU)^T + GXG^T + I)z \geq z^Tz,$$

which implies $z = 0$. The conclusion is that P satisfies the Riccati equation; furthermore, the control law $u = Kx$ is stabilizing, since $X > 0$, and

$$\begin{aligned} &(A + BK)X + X(A + BK)^T + GXG^T \\ &= AX + BU + (AX + BU)^T + GXG^T = -I < 0. \end{aligned}$$

For other references where duality theory and control links are exploited, see, for example, [413].

The above approach can be extended to some other classes of nonstandard Riccati equations, involving, for example, jump linear systems [3], problems with indefinite control weighting matrix R [324], etc.

1.6.2 Impulse differential systems

Impulse differential systems are systems subject to state discontinuities (or jumps). These systems are *hybrid*, in the sense that they obey to a mixture of discrete- and continuous-time behaviors. See [27, 358] for references and also [439] for an application in the context of telecommunications. In classical control, hybrid systems are very common: they arise when one controls a (continuous-time) system with a discrete-time controller.

A typical impulse linear differential system is of the form

$$\begin{aligned} \dot{x} &= Ax, \quad t \neq \tau_i, \\ x(\tau_i^+) &= Mx(\tau_i^-), \end{aligned}$$

where A, M are given square $n \times n$ matrices, and τ_i, $i = 0, 1, \ldots$, is a strictly increasing infinite sequence of time instants, which determine when the (otherwise smooth) trajectory suddenly jumps from $x(\tau_i^-)$ to $x(\tau_i^+)$. By proper choice of the impulse matrix M, one seeks to guarantee stability, using small impulses at properly chosen time instants. (In the following, we assume the jumps are separated by a fixed interval T.)

The stability analysis of impulse systems is nontrivial in general. In [439], Yang and Chua have devised (based on an earlier result by [358]) a stability criterion, using quadratic Lyapunov functions. It turns out that it is easy to rewrite this criterion in the form of an LMI condition, as follows. If there exists $S > 0$ such that

$$A^T S + SA + 2\alpha S \leq 0, \quad \alpha \geq 0, \quad 2\alpha T > \log \lambda,$$
$$\lambda S \geq M^T S M,$$

then the system is asymptotically stable.

The above conditions can be interpreted in terms of the ellipsoid $\mathcal{E} = \{\xi \mid \xi^T S \xi \leq 1\}$: the first LMI says that, during the "continuous-time behavior," the trajectory never escapes the (possibly bigger) ellipsoid $\{\xi \mid \xi^T S \xi \leq e^{-2\alpha T}\}$; the last two conditions ensure that the jump brings the state back to $\mathcal{E}$, which implies some kind of invariance for $\mathcal{E}$. Note that we do not assume the continuous system to be stable, since the decay rate α may be negative; in some sense, we are requiring that the "degree of stability" of the discrete-time behavior dominates that of the continuous-time one.

The above condition is an LMI in S for fixed λ, α, which shows that the problem of checking stability (based on quadratic stability) is tractable, provided a plane search (on λ, α) is used. The main advantage of the LMI formulation is that it allows for *designing* an appropriate impulse M: with the change of variables $U = SM$, the above conditions become convex in *both* S, U.

With an LMI (sufficient) condition stability in hand, we can address a variety of control problems.

For example, we can extend the results to nonlinear and/or uncertain systems; for example, we may allow the matrix A to be unknown-but-bounded, or consider the case when the system without impulses is LTI with a sector-bounded nonlinearity, and apply Lur'e stability conditions [64]. Likewise, it is possible to deal with optimal (state-feedback) control problems using the framework used in section 1.6.1.

1.6.3 Delay differential systems

Delay systems arise in many applications, especially when a physical or biological transport phenomenon occurs (see, e.g., Murray [288] for interesting examples in biology).

A typical problem in control of delay system is to analyze stability as a function of delay, where the delay is viewed as a *parameter* of the system. The problem is still largely open [100]; even in the linear case, and with multiple (noncommensurable) delays, the problem is NP-hard (see Toker and Ozbay [395] for a precise statement and results).

We may distinguish *delay-independent* stability analysis (i.e., no information on the delay size is used) and *delay-dependent* stability analysis (based on this information). Sufficient stability conditions, both delay independent or not, have been devised using LMIs; see, e.g., [428, 248, 143]. Important tools in stability analysis are based on extensions of classical Lyapunov theory, leading to Krasovskii functionals or Razumikhin functions [186]. The Razumikhin theorem has been linked to the $\mathcal{S}$-procedure of section 1.2.4 in [301]. A survey of results and further comments are given in [301].

In section 1.4.6, we have given a delay-independent stability result, based on Lagrange relaxations, which is related to a particular choice of Lyapunov–Krasovskii functional candidate. This approach can be extended to a delay-dependent result, using an embedding technique (comparison principle) as follows.

Consider the stability analysis problem for the delay-differential system (1.30). We introduce the distributed delay system

$$\dot{x} = (A + A_d)x(t) - \int_{-h}^{0} \left(A_d A x(t+\theta) + A_d^2 x(t+\theta-h)\right) d\theta$$

and argue that the above system is an embedding of the original system (1.30), in the sense that stability of the above guarantees that of (1.30) (in fact, the above is an integration of the original equations over one delay interval). Using Laplace transforms and straightforward computations, we obtain a sufficient condition for stability:

$$\begin{gathered}\det\left(sI-(A+A_d)-z_1A_dA-z_2A_d^2\right)\neq 0\\ \text{for every } s,z_1,z_2\ \ s+s^*\geq 0,\ \text{and } |z_i|\leq h,\ i=1,2.\end{gathered}$$

It is then straightforward to apply the Lagrange relaxation technique (see section 1.4.6) and obtain the following delay-dependent relaxed LMI condition:

$$\begin{bmatrix} \begin{pmatrix} h^{-1}\left[(A+A_d)^TP+P(A+A_d)\right] \\ +S_1+S_2 \\ A^TA_d^TP \\ (A_d^T)^2P \end{pmatrix} & \begin{matrix} PA_dA & P(A_d)^2 \\ -S_1 & 0 \\ 0 & -S_2 \end{matrix} \end{bmatrix} < 0.$$

Computing a lower bound on the largest allowable delay is a generalized eigenvalue optimization problem (GEVP). Refinements of these results can be found in [300].

To the above LMI condition corresponds a Lyapunov–Krasovskii functional candidate of the form

$$V(x_t)=x(t)^TPx(t)+\int_{-h}^{0}\int_{t+\theta}^{t}x(v)^TS_1x(v)\,dvd\theta+\int_{-2h}^{-h}\int_{t+\theta}^{t}x(v)^TS_2x(v)\,dvd\theta.$$

The above ideas can be extended to multiple delays (seen as independent uncertain parameters). A different idea, leading to similar results, is based on the *IQC* theorem, combined with the Kalman–Yakubovich–Popov lemma [143] (see Chapter 6 for more on IQCs). The basic idea here is to view the delay as an uncertainty that satisfies a given IQC (see section 1.3.1) and then apply IQC analysis.

It is possible to refine the results using a more computationally intensive LMI condition, using an idea similar in some sense to the one exploited in the context of robustness analysis with multipliers (see Chapters 6 and 11, for example). Indeed, for the system (1.30), we may exhibit a Lyapunov–Krasovskii functional that yields a necessary and sufficient condition for stability [207]. Unfortunately, this functional cannot be parametrized using a finite-dimensional function (as is the case when working with LTI systems and quadratic Lyapunov functions); rather, it depends on several matrix functions (defined over an interval $[-\tau\ 0]$). It is possible to discretize this parametrization [179] and obtain LMIs whose degree of conservativeness is arbitrarily low, with an inversely proportional computational effort.

1.6.4 Open-loop and predictive robust control

Robust optimization opens up the possibility of designing robust open-loop controllers for uncertain systems. This idea is suggested in [66], and we briefly elaborate on it now. Consider, for example, a dynamical system

$$\begin{aligned} x_{k+1} &= \mathbf{A}_kx_k+\mathbf{B}_ku_k,\\ y_k &= \mathbf{C}_kx_k+\mathbf{D}_ku_k,\end{aligned}$$

where $\mathbf{A}_k$, $\mathbf{B}_k\in\mathbf{R}^{n\times n_u}$, $\mathbf{C}_k\in\mathbf{R}^{n_y\times n}$, and $\mathbf{D}_k$ are unknown-but-bounded time-varying matrices. We are given a time horizon $T\geq 0$ and denote by u and y the vectors $(u_0,\ldots,u_{T-1})\in\mathbf{R}^{Tn_u}$ and $(y_0,\ldots,y_{T-1})\in\mathbf{R}^{Tn_y}$. We also define three polytopes (or ellipsoids) $\mathcal{U}\subseteq\mathbf{R}^{Tn_u}$, $\mathcal{X}_0$, and $\mathcal{Y}\in\mathbf{R}^{Tn_y}$, sets of admissible controls, initial conditions, and outputs constraints, respectively.

The problem is to find a $u \in \mathcal{U}$, if any, such that for every $x_0 \in \mathcal{X}_0$, the resulting output satisfies $y \in \mathcal{Y}$.

If the state-space matrices are exactly known, the problem is equivalent to a simple linear program. In the case when the state-space matrices are uncertain, the robust counterpart can be approximated using the robust decision methodology. In terms of optimization, this versatile framework gives rise to challenging large-scale problems.

One way to avoid the curse of dimensionality is to construct ellipsoids of confidence for the future state and optimize (over a few time steps) the control vector. This idea is similar to the one used in model predictive control. The construction of the corresponding ellipsoids of confidence can be done using the techniques proposed shortly in section 1.7.2, in a recursive manner.

1.7 Robustness out of control

The title of this section is borrowed from that of a talk by J. C. Doyle [105].

The LMI formalism provides a unitary framework, not only for various control problems, but also in other fields, where LMIs have recently emerged as a useful tool. We give some examples of this interplay, concentrating on areas where control-related notions (e.g., robustness) seem to play a role that goes sometimes unrecognized yet, such as interval linear algebra.

1.7.1 LMIs in combinatorial optimization

SDPs are now recognized as yielding very efficient relaxations for (hard) combinatorial problems [161]. It turns out that these SDP relaxations for combinatorial problems can be obtained as special cases of the methodology outlined in section 1.4. (The connection between $\mathcal{S}$-procedure and such relaxations is presented in [66].) In turn, the SDP relaxations open up interesting perspectives for handling *combinatorial problems with uncertain data*, in view of the tools developed for the robust SDP.

We illustrate this via a simple example. Consider the NP-hard problem

$$\max_{\delta \in \mathbf{R}^l} \delta^T W \delta \text{ subject to } \delta_i^2 = 1, \quad i = 1, \ldots, l, \tag{1.46}$$

where W is a given symmetric matrix. When W assumes some special structure, the above problem is known in the combinatorial optimization literature as "the maximum cut" (MAX-CUT) problem [161, 154].

This problem is a robustness analysis problem, as defined in section 1.2.3. First we note that, without loss of generality, we may assume $W > 0$ and rewrite the problem as a robust synthesis problem:

$$\text{minimize } x \text{ subject to } \delta^T W \delta \leq x \text{ for every } \delta, \ \|\delta\|_\infty \leq 1.$$

Let us now apply our results, with

$$\mathbf{F}(x, \Delta) = \begin{bmatrix} x & \delta^T \\ \delta & W^{-1} \end{bmatrix} = F + L\Delta R + (L\Delta R)^T,$$

for appropriate matrices F, L, R, and with $\Delta = \mathbf{diag}(\delta_1, \ldots, \delta_l)$. The uncertainty set is

$$\mathbf{\Delta} = \{\Delta \text{ diagonal}, \ \|\Delta\| \leq 1\}.$$

The above set satisfies the assumptions made in section 1.3.1.

We obtain easily the following SDP approximation, yielding an upper bound on the original problem:

$$\min \mathbf{Tr} S \text{ subject to } W \leq S,\ S \text{ diagonal.} \tag{1.47}$$

In [161], Goemans and Williamson have also proposed an SDP relaxation (i.e., an upper bound) of the problem. The technique is actually a special case of the general technique described in section 1.4.3. Introduce the variable $X = \delta\delta^T$ and rewrite problem (1.46) in the equivalent form

$$\max_X \mathbf{Tr} WX \text{ subject to } \mathbf{rank} X = 1,\ X_{ii} = 1,\ X \geq 0.$$

Relaxing the nonconvex constraint $\mathbf{rank} X = 1$, we obtain an SDP that is exactly the "dual" (in the sense of SDP; see section 1.5.1) of (1.47). As shown in [161], the above relaxation is the most efficient currently available (in terms of closeness to the actual optimum in a certain stochastic sense).

Once an SDP formulation of the original problem is known, we can consider problems in which the "data matrix" W is uncertain, starting from the "nominal problem" (1.47). (In practice, such problems often arise in telecommunications, and the matrix W is estimated from statistics.)

1.7.2 Interval calculus

A basic problem in interval computations is the following. We are given a function $\mathbf{f}$ from $\mathbf{R}^l$ to $\mathbf{R}^m$ and a set confidence $\mathcal{D}$ for $\delta \in \mathbf{R}^l$ in the form of a product of intervals. We seek to estimate intervals of confidence for the components of $x = \mathbf{f}(\delta)$ when δ ranges to $\mathcal{D}$. (Sometimes, $\mathbf{f}$ is given in implicit form, as in the interval linear algebra problem; we return to this case in the next section.) Obtaining exact estimates for intervals of confidence for the elements of solutions x is in general NP-hard (see [345, 346] for the interval linear algebra case).

One classical approach to this problem resorts to interval calculus, introduced by Moore [287], where each one of the basic operations $(+, -, \times, /)$ is replaced by an "interval counterpart," and standard (e.g., LU) linear algebra algorithms are adapted to this new "algebra." Many refinements of this basic idea have been proposed, but the algorithms based on this idea have in general exponential complexity. (For control applications of interval calculus, see, e.g., [99].)

Robust SDP can be used for this problem as follows. Assume we can describe $\mathbf{f}$ explicitely as a rational function of its arguments; from Lemma 1.2, we can construct (in polynomial time) an LFR of $\mathbf{f}$ in the form

$$\mathbf{f}(\delta) = f + L\Delta\,(I - D\Delta)^{-1} r, \text{ where } \Delta = \mathbf{diag}\,(\delta_1 I_{r_1}, \ldots, \delta_l I_{r_l})\,.$$

Assume first that we seek an ellipsoid of confidence for the solution in the form

$$\mathcal{E} = \left\{ x \mid (x - x_0)(x - x_0)^T \leq P \right\}, \tag{1.48}$$

where $x_0 \in \mathbf{R}^n$ is the center and $P \geq 0$ defines the "shape" (our parametrization allows for degenerate, "flat" ellipsoids to handle cases when some components of the solution are certain). We seek to minimize the "size" of $\mathcal{E}$ subject to $\mathbf{f}(\delta) \in \mathcal{E}$ for every $\delta \in \mathcal{D}$. Measuring the size of $\mathcal{E}$ by $\mathbf{Tr} P$ (other measures are possible), we obtain the following equivalent formulation of the problem:

$$\text{minimize } \mathbf{Tr} P \text{ subject to}$$
$$\begin{bmatrix} P & (\mathbf{f}(\delta) - x_0) \\ (\mathbf{f}(\delta) - x_0)^T & 1 \end{bmatrix} \geq 0 \text{ for every } \delta \in \mathcal{D}. \tag{1.49}$$

The above is obviously a robust SDP problem (in variables x_0, P) for which an explicit SDP counterpart (approximation) can be devised, provided $\mathcal{D}$ takes the form of a (general) ellipsoidal set. (A typical set is a product of intervals $\Pi[\underline{\delta}_i \ \overline{\delta}_i]$, where $\underline{\delta}_i$, $\overline{\delta}_i$ are given.)

The above method finds ellipsoids of confidence, but it is also possible to find intervals of confidence for the components of $\mathbf{f}(\delta)$ by modifying the objective of the above robust SDP suitably (for example, if we minimize the $(1,1)$ component of the matrix variable P instead of its trace, we will obtain an interval of confidence for the first component of $\mathbf{f}(\delta)$ when δ ranges to $\mathcal{D}$).

1.7.3 Structured linear algebra

Many linear algebra problems involve matrices with a given structure (in identification, for instance, we obtain least squares problems in which the coefficient matrix has a Toeplitz structure). One of the most active research areas in numerical linear algebra pertains to exploiting perturbation structure, both for error analysis and for computing the solution.

Consider, for example, a *polynomial interpolation problem* leading to a linear system of the form $Ax = b$, where x contains the polynomial coefficients and A is a matrix with a Vandermonde structure. In some applications, the interpolation points are not precisely known. This results in an "uncertain Vandermonde system" of the form $\mathbf{A}(\delta)x = \mathbf{b}(\delta)$, where $\mathbf{A}(\delta)$, $\mathbf{b}(\delta)$ are matrix-valued (resp., vector-valued) polynomial functions of a perturbation vector δ.

A structured linear equation is a model of the form

$$\mathbf{A}(\delta)x = \mathbf{b}(\delta), \tag{1.50}$$

where $\mathbf{A}(\delta)$, $\mathbf{b}(\delta)$ are (affine) functions of a parameter vector $\delta \in \mathbf{R}^l$, and x is the variable. We assume that δ is unknown-but-bounded; for example, $\|\delta\|_\infty \leq 1$. (Note that (1.50) is not an equation but a model of the way an equation is subject to perturbations.) The above model can be represented in the linear-fractional form:

$$\begin{aligned} &Ax + Lp = b, \\ &q = R_A x + R_b + Dp, \\ &p = \Delta(\delta)q, \quad \Delta(\delta) = \mathbf{diag}(\delta_1 I_{r_1}, \ldots, \delta_l I_{r_l}). \end{aligned}$$

A number of problems pertain to the structured equation (1.50).

Robust least squares

One such problem is robust least squares, where we seek a vector x that minimizes the worst-case residual

$$r_S(\mathbf{A}, \mathbf{b}, x) \triangleq \max_{\|\delta\|_\infty \leq 1} \|\mathbf{A}(\delta)x - \mathbf{b}(\delta)\|. \tag{1.51}$$

Such a vector can be termed robust in the sense that it minimizes the effect of perturbations on the "performance" (measured by the residual). The above problem has many implications in control, notably in robust identification.

The robust least-squares problem is a robust SDP problem, since the nominal problem, i.e., the classical least-squares problem, is a special case of an SDP. Details on this approach are given in [125].

Condition estimates and accurate solutions

Another problem is to *measure* the sensitivity of a candidate solution x. A way to quantify this sensitivity (or backward error) is to compute

$$\max_{\delta,y} \|y - x\| \text{ subject to (1.50)}, \quad \|\delta\|_\infty \leq 1.$$

It is also interesting to compute a vector x that minimizes (an upper bound on) the above quantity. Such a vector is a solution for (1.50) with minimum sensitivity with respect to the perturbation vector δ; in this sense, it is accurate (with respect to the given perturbation structure and bounds).

As seen in the next paragraph, the above problems are robust SDP problems, which can be attacked using the methodology described previously.

Interval linear algebra

The basic problem of interval linear algebra is a slight variation on the problem considered in section 1.7.2 and is as follows. We are given matrices $\mathbf{A} \in \mathbf{R}^{n\times n}$, $\mathbf{b} \in \mathbf{R}^n$, the elements of which are only known within intervals; in other words, $[\mathbf{A}\ \mathbf{b}]$ is only known to belong to an "interval matrix set" $\mathcal{U}$. we seek to compute intervals of confidence for the set of solutions, if any, to the equation $\mathbf{A}x = \mathbf{b}$.

To attack this problem we seek an ellipsoid of confidence in the form (1.48), where $P > 0$. For simplicity, we assume first that A, b are given; using the $\mathcal{S}$-procedure, we find that a necessary and sufficient condition for $\mathcal{E}$ to be an ellipsoid of confidence is that there exists a scalar multiplier τ such that

$$\text{for every } y,\ (y - x_0)^T P^{-1}(y - x_0) + \tau\|Ay - b\|^2 \leq 1.$$

This condition is easily converted in an LMI in x_0, P. From then on, we can use the robust SDP methodology to handle the case when A, b are uncertain.

For further reference, we detail the result in the special case when $[\mathbf{A}\ \mathbf{b}]$ is an uncertain matrix subject to "unstructured additive perturbations." Assume

$$[\mathbf{A}\ \mathbf{b}] \in \mathcal{U} = \{[A + \Delta A\ b + \Delta b] \mid \|[\Delta A\ \Delta b]\| \leq \rho\},$$

where $[A\ b] \in \mathbf{R}^{n\times(m+1)}$ and $\rho \geq 0$ are given. In this case, our results are exact and yield an ellipsoid center of the form

$$x_0 = (A^T A - \rho^2 I)^{-1} A^T b. \tag{1.52}$$

(We assume that $\sigma_{\min}([A\ b]) \geq \rho$; otherwise the ellipsoid of confidence is unbounded. Except in degenerate cases, this guarantees the existence of the inverse in the above.)

Structured total least squares

The so-called structured total least squares (STLS) problem, which arises in many areas, e.g., in identification and other inverse problems in engineering [96], is as follows. We are given $\mathbf{A}(\delta) \in \mathbf{R}^{m\times n}$, $\mathbf{b}(\delta) \in \mathbf{R}^m$, two (algebraic) functions of a parameter $\delta \in \mathbf{R}^l$. The STLS problem is

$$\text{minimize } \|\delta\| \text{ subject to } \mathbf{A}(\delta)x = \mathbf{b}(\delta). \tag{1.53}$$

Here, we are interested in finding one value of the "perturbation" vector δ that makes the vector x feasible. In this sense, the problem belongs to the class of "backward error

analysis" problems: A perturbation is sought such that the linear equation above becomes consistent.

The STLS problem can be solved (at least approximately) by resorting to robust optimization as follows. A scalar ρ is an upper bound on the objective in (1.53) if there exists $x \in \mathbf{R}^m$ such that

$$\mathbf{A}(\delta)x = \mathrm{b}(\delta) \text{ for some } \delta, \quad \|\delta\| \leq \rho.$$

Assume that we have computed an ellipsoid of confidence of center x_0 and shape matrix P, denoted by $\mathcal{E}_\rho$, for the set of solutions of the above constraint. The quantity

$$\min \rho \text{ subject to } \mathcal{E}_\rho \text{ is not empty}$$

is an upper bound on the STLS problem. The above problem turns out to be a GEVP, with variables ρ, x_0, P.

In the unstructured (additive) perturbations case, we recover the classical notion of the total least-squares solution developed by Golub and Van Loan [171]. The upper bound computed via GEVP is exact; the ellipsoid of confidence can be shown to be reduced by the singleton, and the optimal center is given by (1.52), where $\rho = \sigma_{\min}([A\ b])$.

1.7.4 Robust portfolio optimization

A portfolio is a collection of financial assets, say, $p_1, \ldots, p_n$. Each asset p_i has a (random) return over a given horizon, denoted r_i. If x_i is the proportion of the total value of an asset p invested in the ith asset, then the return of the portfolio over the given horizon is $\rho = x_1 r_1 + \cdots + x_n r_n = r^T x$.

The return of each asset is, of course, unknown a priori. However, it is possible to deduce from past measurements the values of the mean $\mu = \mathbf{E}(r)$ and covariance matrix $V = \mathbf{E}(rr^T)$. The standard mean-variance portfolio choice problem, introduced in the fifties by Markowitz [263], is to minimize the variance of the asset's return, $x^T V x$, subject to a given level α of mean return, $\alpha = \mu^T x$.

This problem can be written

$$\text{minimize } x^T V x \text{ subject to } x_i \geq 0, \quad e^T x = 1, \quad \mu^T x = \alpha.$$

(In the above, e stands for the n-vector with every component equal to one.) The problem above belongs to the class of a convex quadratic problem (a subset of SDP problems), and, as such, can be very efficiently solved using interior-point methods.

One may then consider variations of this problem and solve them using the LMI framework. We obtain one interesting example when we assume that the mean and covariance matrices are unknown-but-bounded. This leads to portfolio choices that are robust with respect to uncertainty in mean and covariance. Another interesting perspective is to formulate the problem in a dynamic (multistage) setting and apply robust control techniques to devise a robust portfolio update.

1.8 Perspectives and challenges

1.8.1 Algorithms and software

Efficient algorithms and software should exploit the structure of the given problem in the most systematic way. For example, many problems in control lead to SDPs of the form (1.2), where the coefficient matrices are sparse. A sparse SDP solver would thus be

extremely useful. Interior-point methods seem difficult to adapt directly to sparse LMI problems. To understand why, recall from section 1.5.2 that at each step we have to find (approximate) solutions to a complementarity equation of the form

$$FZ = \mu I,$$

where μ is a parameter, and F (resp., Z) is the primal (resp., dual) variable. Sparsity of F does not imply that of Z, and this makes the use of sparse computations uneasy to propagate throughout the iterations.

In contrast, the nondifferentiable methods, described in section 1.5.3, and in more detail in Chapter 3, seem promising for sparse SDPs. In particular, first-order (bundle) methods only require the computation of a (few) subgradient(s), and this can be done with sparse calculus. One problem with such methods is slow convergence, but we believe that using second-order information (such as U-Lagrangians), it will be possible to increase convergence speed at a controlled increased cost per iteration (recall that with the U-Lagrangian method, we work on a subspace of dimension equal to the multiplicity at the optimum, and this number is often much smaller than the size of the problem). More theoretical studies are needed to make this statement anything else than a belief. In particular, more work is needed on generic multiplicity at the optimum for SDPs, along the lines of [310, 326, 7].

Sparsity is not the only structure of which we may take advantage. SDPs that pertain to matrices with Toeplitz or Lyapunov, etc., structure can often be solved much faster than general SDPs. (Structure is also crucial information to use for solving linear equations.) This issue is raised in detail in Chapter 4 of this book.

1.8.2 Nonconvex LMI-constrained problems

A number of problems arising in control and other fields are not convex. We encounter the following classes. (We only discuss feasibility problems, to simplify; see Part V of this book for more on these problems.)

Bilinear matrix inequality (BMI) problems are of the following form:

$$\text{Find } x \text{ such that } F(x) = F_0 + \sum_{i=1}^{m} x_i F_i + \sum_{i,j} x_i x_j F_{ij} \geq 0, \tag{1.54}$$

where F_i, G_{ij} are given symmetric matrices. This class contains much of combinatorial optimization; a typical example of such a problem is the MAX-CUT problem mentioned in section 1.7.1.

Rank-constrained problems (RCPs) are of the following form:

$$\text{Find } x \text{ such that } F(x) \geq 0, \ \mathbf{Rank}\ G(x) \leq k,$$

where $F(\cdot), G(\cdot)$ are two affine matrix functions (with $F(\cdot)$ symmetric), and k is a given integer. Rank-constrained problems arise in, e.g., reduced-order feedback control (see [108, 124, 128]).

Conic complementarity problems (CCPs) consist of solving

$$\text{minimize } \mathbf{Tr} ZF, \quad (F, Z) \in \mathcal{K}, \tag{1.55}$$

where $\mathcal{K}$ is a convex subset of the cone of semidefinite matrices, and F, Z are variables. When the problem is "monotone" (that is, when the set $\mathcal{K}$ is a "self-dual" cone), then the above problem can be solved using primal-dual interior-point methods for SDPs. For

general cones $\mathcal{K}$, however, the above problem is hard. Conic complementarity problems are generalizations of the famous linear complementarity problems and arise in many situations, including robust control (see [128]); see section 1.4.4 for an example.

The above problems are in general NP-hard. As we will see, they are polynomially equivalent, in the sense that we may transform one into the other in polynomial time. This equivalence can be used to devise efficient relaxation algorithms.

First, we may rewrite any BMI in the form of an RCP, as follows. Define $X = xx^T$, and rewrite the constraint in (1.54) as

$$\begin{array}{c} F_0 + \sum_{i=1}^m x_i F_i + \sum_{i,j} X_{ij} F_{ij} \geq 0, \quad \left[\begin{array}{cc} X & x \\ x^T & 1 \end{array} \right] \geq 0, \\ \mathbf{Rank} \left[\begin{array}{cc} X & x \\ x^T & 1 \end{array} \right] = 1. \end{array} \tag{1.56}$$

We obtain a rank-constrained problem. Dropping the rank constraint yields an LMI condition, which provides us with a relaxation of the original BMI problems, very similar to the one used in combinatorial optimization (see section 1.7.1).

Conversely, any rank-constrained problem is a BMI and a CCP. First we may assume without loss of generality that the matrix $G(x)$ is square ($n_G \times n_G$) and symmetric. Next, the constraint $\mathbf{Rank} G(x) \leq k$ is equivalent to saying that $\mathbf{Tr} ZF = 0$, $F \geq 0$, $Z \geq 0$, where

$$F = \left[\begin{array}{cc} G(x) & M \\ M^T & I \end{array} \right], \quad Z = \left[\begin{array}{cc} S & U \\ U^T & V \end{array} \right] \tag{1.57}$$

for some $M \in \mathbf{R}^{n_G \times k}$, $S \in \mathbf{R}^{n_G \times n_G}$, $S \geq I$, U, V of appropriate size. Note that the constraint $\mathbf{Tr} ZF = 0$ can be written as a BMI, which shows the polynomial equivalence between BMIs and RCPs.

By solving the CCP

$$\text{minimize } \mathbf{Tr} ZF \text{ subject to } (Z, F) \in \mathcal{K},$$

where $\mathcal{K}$ denotes the cone of positive semidefinite pairs (F, Z) of the form (1.57), with $S \geq I$, we solve the original RCP.

The conic complementarity formulation seems very similar to the problem we encounter in primal-dual interior-point methods for solving SDPs. This suggests an algorithmic approach to such problems, where we directly apply a primal-dual interior-point method to the problem. (Of course, this method will converge globally when the complementarity problem is monotone.) For nonmonotone problems, the method will at least provide a local minimum.

1.8.3 Dynamical systems over cones

The developments made in section 1.4.6 seem to indicate that to analyze a (possibly stochastic) dynamical system with a vector state $x \in \mathbf{R}^n$ in the context of LMIs, it is fruitful to consider the (equivalent) system describing the evolution of the matrix $X = xx^T$ (or its expected value, if a stochastic system is at hand).

By definition, the resulting (deterministic) dynamical system leaves the positive semidefinite cone invariant. This motivates a closer study of dynamical systems of the form

$$\dot{X} = \phi(X), \tag{1.58}$$

where ϕ is a deterministic map, with the property that the positive semidefinite cone is invariant for (1.58). There seem to be a number of (apparently unexplored) interesting

connections between several optimization and control notions, all related to systems of the form above.

For example, for the LTI system $\dot{x} = Ax$, the resulting cone-invariant system takes the form (1.58), with $\phi(X) = A^T X + XA^T$. A (linear) Lyapunov candidate of the form $V(X) = \mathbf{Tr}XS$ (with $S > 0$ given) proving stability of system (1.58) can be interpreted as a *primal-dual* gap (X is the "primal" variable, S the "dual" one) and is related to a quadratic Lyapunov function proving stability of the original (state-vector) system. The complementarity condition of the form (1.40) encountered in SDP, where a product of primal and dual variables are inverse of each other, can be related to the notion of adjoint system. Indeed, we note that if $X(t)$ is a trajectory of (1.58), with $X(t)$ invertible for every $t \geq 0$, then the inverse matrix $S(t) = X(t)^{-1}$ satifies the adjoint system

$$-\dot{S} = \phi^*(S) = A^T S + SA.$$

It would be interesting to see how the above connections can be exploited in control and optimization problems and if the study of cone-invariant systems (1.58) can be made for cones other than the positive semidefinite cone.

1.8.4 Quality of relaxations for uncertain systems

As seen in section 1.4.5, the problem of estimating the quality of relaxations is crucial in robust SDP, and this opens several challenging new problems. Good quality estimates would be of great importance for robust control, as they would enable us to bound the conservativeness of well-known tools such as μ-analysis.

Consider for example the uncertain dynamical system

$$\dot{x} = \mathbf{A}(\delta(t))x, \quad \|\delta(t)\|_2 \leq 1 \text{ for every } t \geq 0, \tag{1.59}$$

where the $n \times n$ matrix $\mathbf{A}$ is an algebraic function of its argument $\delta \in \mathbf{R}^l$.

A sufficient condition for stability is quadratic stability (see section 1.4.6):

$$\exists\, P > 0, \quad \mathbf{A}(\delta)^T P + P\mathbf{A}(\delta) < 0 \text{ for every } \delta, \quad \|\delta\|_2 \leq 1.$$

In turn, the above condition can be relaxed using robust SDP techniques, leading to a (tractable, sufficient) LMI condition of the form (1.32).

In some cases, it is possible to assess the possible conservativeness of this LMI condition over the quadratic stability one. For example, if $\mathbf{A}$ is affine in δ, then Nemirovski's results apply (see section 1.4.5), leading to a quality bound (in the sense defined in section 1.4.5) of $\min(\sqrt{n}, \sqrt{l})$.

The more general case when the matrix $\mathbf{A}$ is nonlinear (algebraic) in the parameter vector δ can perhaps be attacked using a singular perturbation technique, as follows. Represent $\mathbf{A}$ in the LFR format

$$\mathbf{A}(\delta) = A + B_p\Delta(I - D_{qp}\Delta)^{-1}C_q, \quad \Delta = \mathbf{diag}(\delta_1 I_{r_1}, \ldots, \delta_l I_{r_l}).$$

As usual, we assume this LFR to be well posed over $\mathbf{\Delta}$. Now let $\epsilon > 0$, and associate to system (1.59) the following "augmented" system:

$$\begin{array}{rcl} \dot{x} & = & Ax + B_p p, \\ \epsilon\dot{p} & = & \Delta(t)C_q x + (\Delta(t)D_{qp} - I)p, \quad \Delta(t) \in \mathbf{\Delta} \text{ for every } t \geq 0, \end{array} \tag{1.60}$$

where $\mathbf{\Delta}$ is the set defined in (1.15).

Under certain conditions classical in singular perturbations theory, involving (robust) stability of the matrix $D_{qp}\Delta - I$ for $\Delta \in \mathbf{\Delta}$,[3] the above system can serve as an approximation to system (1.59). For every $\epsilon > 0$, the above system is equivalent to an uncertain linear system, with *affine* dependence on the uncertain parameters δ. For this system, the conservativeness estimate for quadratic stability is $\min(\sqrt{n+N}, \sqrt{l})$, where $N = r_1 + \cdots + r_l$. Since by definition $r_i \geq 1$, we obtain the conservativeness estimate $\sqrt{l}$.

The challenge here is to see how this analysis can be extended to the case of arbitrarily small ϵ in order to give a rigorous conservativeness estimate.

1.9 Concluding remarks

In this chapter, we sought to give an overview of the LMI method. We proposed a unitary framework based on optimization ideas for addressing a large class of decision problems with uncertainty. We have indicated some possible research directions where this method could be applied.

In control theory, to borrow words from Doyle, Packard, and Zhou [108],

> *LMIs play the same central role in the postmodern theory as Lyapunov function and Riccati equations played in the modern, and in turn various graphical techniques such as Bode, Nyquist and Nichols plots played in the classical.*

Thus, the LMI approach constitutes a basis for a "postmodern control theory" [108], which allows robust, multicriteria synthesis. Benefits of both classical (PID) or modern (LQG, $\mathbf{H}_\infty$) approaches are combined: reduced number and clear physical interpretation of design parameters and simplicity of numerical solution, even for multivariable problems. In addition, this approach provides guarantees for the design and allows (possibly competing) specifications.

In our opinion, LMI-based control methods have reached a certain degree of "academic" maturity, even though many control areas are yet unexplored. Both theoretical grounds and efficient algorithms are now available, and this should lead to more and more industrial applications, some of which are in progress.

We strongly believe that the LMI approach could be applied to many other engineering problems, especially design problems with robustness and multicriteria issues. We presented some of these problems with an optimization point of view. The reason is that the language of optimization seems the best suited as a common framework for these problems.

In some ways, we could widen the scope of the above quote from [108] by saying that LMIs (and convex optimization ideas) should play the same central role in "postmodern" engineering as numerical linear algebra (least squares, eigenvalue problems, etc.) played in the past 20 years.

Acknowledgments

We are grateful to Michael Overton and François Oustry, who completely rewrote the section on algorithms (section 1.5). The first author is greatly indebted to Arkadi Nemirovski for many stimulating discussions on the subject of robust optimization.

[3] If $\mathbf{A}$ is polynomial in δ, then D_{qp} can be constructed as a strictly upper triangular matrix, and $D_{qp}\Delta - I$ is stable for every $\Delta \in \mathbf{\Delta}$.

This page intentionally left blank

Part II

Algorithms and Software

This page intentionally left blank

Chapter 2

Mixed Semidefinite–Quadratic–Linear Programs

Jean-Pierre A. Haeberly, Madhu V. Nayakkankuppam, and Michael L. Overton

2.1 Introduction

In this chapter, we consider mixed semidefinite–quadratic–linear programs (SQLPs). These are linear optimization problems with three kinds of cone constraints, namely, the semidefinite cone, the quadratic cone, and the nonnegative orthant. This chapter outlines a primal-dual path-following method to solve these problems and highlights the main features of SDPpack, a Matlab package that solves such programs. Furthermore, we give some examples where such mixed programs arise and provide numerical results on benchmark problems.

A mixed SQLP has the form

$$\begin{aligned}
&\max && b^T y && && (2.1)\\
&\text{s.t.} && && &\\
&F_0^{(k)} + \textstyle\sum_{i=1}^m y_i F_i^{(k)} \succeq 0, && && k = 1,\ldots,L, && (2.2)\\
&\|(c^{(k)} - A^{(k)})^T y\| \le \gamma^{(k)} - (g^{(k)})^T y, && && k = 1,\ldots,M, && (2.3)\\
&(A^{(0)})^T y \le c^{(0)}, && && && (2.4)
\end{aligned}$$

where $y \in \mathbf{R}^m$ and

$$\begin{gathered}
F_i^{(k)} \in \mathcal{S}^{n_k}, \quad i = 0,\ldots,m,\ k = 1,\ldots,L,\\
A^{(k)} \in \mathbf{R}^{m\times p_k},\ c^{(k)} \in \mathbf{R}^{p_k},\ g^{(k)} \in \mathbf{R}^m,\ \gamma^{(k)} \in \mathbf{R}, \quad k = 1,\ldots,M,\\
A^{(0)} \in \mathbf{R}^{m\times p_0},\ c^{(0)} \in \mathbf{R}^{p_0}.
\end{gathered}$$

The first set of constraints (2.2) consists of L semidefinite cone constraints of size $n_1,\ldots,n_L$, respectively. The second set of constraints (2.3) consists of M quadratic

cone constraints of size $p_1, \ldots, p_M$, respectively. The final constraint (2.4) consists of p_0 ordinary linear inequality constraints. We use y instead of x as the optimization variable, with maximization objective $b^T y$ instead of minimization objective $c^T x$ as used elsewhere in this book in order to make the notation used here more compatible with the SDPpack software to be described later in this chapter.

It is convenient to rewrite the problem as follows:

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & \end{array}$$

$$F^{(k)} = F_0^{(k)} + \textstyle\sum_{i=1}^m y_i F_i^{(k)}, \quad k = 1, \ldots, L, \tag{2.5}$$

$$f^{(k)} = c^{(k)} - (A^{(k)})^T y, \quad \phi^{(k)} = \gamma^{(k)} - (g^{(k)})^T y, \quad k = 1, \ldots, M, \tag{2.6}$$

$$f^{(0)} = c^{(0)} - (A^{(0)})^T y, \tag{2.7}$$

$$F^{(k)} \geq 0, \ k = 1, \ldots, L, \quad \|f^{(k)}\| \leq \phi^{(k)}, \ k = 1, \ldots, M, \quad f^{(0)} \geq 0, \tag{2.8}$$

where $F^{(k)} \in \mathcal{S}^{n_k}$ $(k = 1, \ldots, L)$, $f^{(k)} \in \mathbf{R}^{p_k}$ and $\phi^{(k)} \in \mathbf{R}$ $(k = 1, \ldots, M)$, $f^{(0)} \in \mathbf{R}^{p_0}$.

We write

$$\begin{aligned} F_i &\triangleq \mathbf{diag}(F_i^{(1)}, \ldots, F_i^{(L)}) \quad (i = 0, \ldots, m), \\ F &\triangleq \mathbf{diag}(F^{(1)}, \ldots, F^{(L)}), \end{aligned}$$

and

$$f \triangleq \left[\phi^{(1)}, (f^{(1)})^T, \ldots, \phi^{(M)}, (f^{(M)})^T\right]^T.$$

The dual optimization problem is

$$\min \ \sum_{k=1}^{L} \mathbf{Tr} F_0^{(k)} Z^{(k)} + \sum_{k=1}^{M} \left[(c^{(k)})^T z^{(k)} + \gamma^{(k)} \zeta^{(k)}\right] + (c^{(0)})^T z^{(0)} \tag{2.9}$$

$$\begin{array}{ll} \text{s.t.} & \\ & \begin{bmatrix} -\sum_{k=1}^{L} \mathbf{Tr} F_1^{(k)} Z^{(k)} \\ \vdots \\ -\sum_{k=1}^{L} \mathbf{Tr} F_m^{(k)} Z^{(k)} \end{bmatrix} + \sum_{k=1}^{M} \left[A^{(k)} z^{(k)} + \zeta^{(k)} g^{(k)}\right] + A^{(0)} z^{(0)} = b, \end{array} \tag{2.10}$$

$$Z^{(k)} \geq 0, \ k = 1, \ldots, L, \quad \|z^{(k)}\| \leq \zeta^{(k)}, \ k = 1, \ldots, M, \quad z^{(0)} \geq 0, \tag{2.11}$$

where the dual variables are $Z^{(k)} \in \mathcal{S}^{n_k}$, $k = 1, \ldots, L$, $z^{(k)} \in \mathbf{R}^{p_k}$, and $\zeta^{(k)} \in \mathbf{R}$, $k = 1, \ldots, M$, and $z^{(0)} \in \mathbf{R}^{p_0}$. Note that the constraints $Z^{(k)} \geq 0$ are semidefinite constraints. We write

$$Z = \mathbf{diag}(Z^{(1)}, \ldots, Z^{(L)})$$

and

$$z = \left[\zeta^{(1)}, (z^{(1)})^T, \ldots, \zeta^{(M)}, (z^{(M)})^T\right]^T.$$

Let us denote the primal objective (2.1) by $\mathcal{P}(y)$ and the dual objective (2.9) by $\mathcal{D}(Z, z, z^{(0)})$. If y and $(Z, z, z^{(0)})$, respectively, satisfy the primal and dual feasibility constraints, it immediately follows that

$$\mathcal{P}(y) = \mathcal{D}(Z, z, z^{(0)}) - \mathbf{Tr} ZF - z^T f - (z^{(0)})^T f^{(0)}.$$

Since Z and F are both positive semidefinite, $\mathbf{Tr} ZF \geq 0$; this follows using the Cholesky factorizations $Z = R^T R$ and $F = S^T S$, as

$$\mathbf{Tr} ZF = \mathbf{Tr} R^T R S^T S = \mathbf{Tr} R S^T S R^T = \mathbf{Tr} (SR^T)^T (SR^T) = \|SR^T\|_F^2,$$

where $\|\cdot\|_F$ stands for the Frobenius norm. Likewise, Cauchy–Schwarz shows that $z^T f \geq 0$, since

$$z^T f = \sum_{k=1}^{L} \zeta^{(k)}\phi^{(k)} + (z^{(k)})^T f^{(k)} \geq \sum_{k=1}^{L} \zeta^{(k)}\phi^{(k)} - \|z^{(k)}\|, \|f^{(k)}\| \geq 0.$$

Also, of course, $(z^{(0)})^T f^{(0)} \geq 0$ is implied by $z^{(0)} \geq 0$, $f^{(0)} \geq 0$. Thus we have

$$\mathcal{P}(y) \leq \mathcal{D}(Z, z, z^{(0)})$$

with equality (a zero duality gap) if and only if $\mathbf{Tr}ZF = 0$, $z^T f = 0$, and $(z^{(0)})^T f^{(0)} = 0$. These three conditions are called complementarity conditions and may be summarized as

$$\mathbf{Tr}ZF + z^T f + (z^{(0)})^T f^{(0)} = 0. \tag{2.12}$$

Furthermore, again using Cholesky, we see that if $\mathbf{Tr}ZF = 0$, then the matrix product ZF must itself be zero (since $SR^T = 0$), and therefore (since Z and F are symmetric) that $ZF + FZ = 0$. (Recall that this is a block-diagonal matrix equation.)

Likewise, we can also elaborate on the second complementarity condition. The condition for equality in Cauchy–Schwarz shows that if $z^T f = 0$, then

$$(z^{(k)})^T f^{(k)} + \zeta^{(k)}\phi^{(k)} = 0, \text{ with } \phi^{(k)} z^{(k)} = -\zeta^{(k)} f^{(k)}, \quad k = 1, \ldots, M.$$

Following [2], this condition may be conveniently expressed as

$$\mathbf{Arw}(z)\,\mathbf{Arw}(f)\,\mathbf{e}_q = 0,$$

where $\mathbf{e}_q$ is a block vector whose kth block is $\mathbf{e}_q^{(k)} = [1, 0, \ldots, 0]^T \in \mathbf{R}^{p_k+1}$, $k = 1, \ldots, M$, and the block-diagonal matrix

$$\mathbf{Arw}(z) \triangleq \mathbf{diag}(\mathbf{arw}(\zeta^{(1)}, z^{(1)}), \ldots, \mathbf{arw}(\zeta^{(M)}, z^{(M)}))$$

has blocks defined, for $\alpha \in \mathbf{R}$, $a \in \mathbf{R}^p$, by the "arrow matrix"

$$\mathbf{arw}(\alpha, a) \triangleq \begin{bmatrix} \alpha & a_1 & a_2 & \ldots & a_p \\ a_1 & \alpha & 0 & \ldots & 0 \\ a_2 & 0 & \alpha & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_p & 0 & 0 & 0 & \alpha \end{bmatrix}.$$

Finally, the third complementarity condition $(z^{(0)})^T f^{(0)} = 0$ implies the componentwise condition $(z^{(0)})_k (f^{(0)})_k = 0$, $k = 1, \ldots, p_0$, since $z^{(0)} \geq 0$ and $f^{(0)} \geq 0$. Letting $\mathbf{e}_\ell = [1, \ldots, 1]^T$, we write this as

$$\mathbf{diag}(z^{(0)})\,\mathbf{diag}(f^{(0)})\,\mathbf{e}_\ell = 0.$$

This much is just linear algebra. To go further we make use of a well-known result from optimization theory [296] stating that if strictly feasible primal and dual variables exist (points that satisfy all the inequalities strictly), then the optimal primal and dual objective values are equal and that there exist primal and dual variables achieving these optimal values. Thus, the complementarity conditions must hold at the (optimal) solutions.

In what follows we make this strict feasibility assumption and we also assume, mainly for convenience, that there are no redundant equality constraints in the dual formulation of the problem.

2.2 A primal-dual interior-point method

The SQLP described in section 1 can be efficiently solved with interior-point methods. Of the many flavors of interior-point methods [297, 423], we briefly describe the algorithm implemented in the public domain package SDPpack [6]. This is a primal-dual path-following algorithm based on the "XZ+ZX" direction of [8] (to be translated here as "ZF+FZ" given the notational differences), using a Mehrotra predictor-corrector scheme.

The optimality conditions are completely determined by the three complementarity conditions together with the primal and the dual constraints. Let **svec** denote a map from symmetric, block diagonal matrices (with block sizes $n_1, \ldots, n_L$) to column vectors, so that $\mathbf{svec}(F) \in \mathbf{R}^N$, where

$$N = \sum_{k=1}^{L} n_k(n_k+1)/2,$$

and such that $\mathbf{Tr}ZF = \mathbf{svec}(Z)^T\mathbf{svec}(F)$ for all symmetric block-diagonal matrices Z, F. Then we may abbreviate the primal slack linear constraints (2.5)–(2.7) by

$$\begin{bmatrix} \mathbf{A}_s^T \\ \mathbf{A}_q^T \\ \mathbf{A}_\ell^T \end{bmatrix} y + \begin{bmatrix} \mathbf{svec}(F) \\ f \\ f^{(0)} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_s \\ \mathbf{c}_q \\ \mathbf{c}_\ell \end{bmatrix},$$

where

$$\mathbf{A}_s = \begin{bmatrix} -\mathbf{svec}\left(\mathbf{diag}(F_1^{(1)}, \ldots, F_1^{(L)})\right)^T \\ \vdots \\ -\mathbf{svec}\left(\mathbf{diag}(F_m^{(1)}, \ldots, F_m^{(L)})\right)^T \end{bmatrix}, \quad \mathbf{c}_s = \mathbf{svec}\left(\mathbf{diag}(F_0^{(1)}, \ldots, F_0^{(L)})\right),$$

$$\mathbf{A}_q = [\, g^{(1)} \quad A^{(1)} \quad \cdots \quad g^{(M)} \quad A^{(M)} \,], \quad \mathbf{c}_q = \left[\gamma^{(1)}, (c^{(1)})^T, \cdots, \gamma^{(M)}, (c^{(M)})^T\right]^T,$$

and

$$\mathbf{A}_\ell = A^{(0)}, \quad \mathbf{c}_\ell = c^{(0)}.$$

The dimensions of $\mathbf{A}_s$, $\mathbf{A}_q$, and $\mathbf{A}_\ell$ are, respectively, $m \times N$, $m \times \sum_{k=1}^{M}(p_k+1)$, and $m \times p_0$. The dual equality constraints (2.10) are then written

$$[\, \mathbf{A}_s \quad \mathbf{A}_q \quad \mathbf{A}_\ell \,] \begin{bmatrix} \mathbf{svec}(Z) \\ z \\ z^{(0)} \end{bmatrix} = b.$$

Furthermore, the complementarity conditions can be written

$$\begin{bmatrix} \mathbf{svec}(\frac{1}{2}(ZF+FZ)) \\ \mathbf{Arw}(z)\mathbf{Arw}(f)\mathbf{e}_q \\ \mathbf{diag}(z^{(0)})\mathbf{diag}(f^{(0)})\mathbf{e}_\ell \end{bmatrix} = 0,$$

where $\mathbf{e}_q$ and $\mathbf{e}_\ell$ were defined in section 2.1. We collect the primal and dual equality constraints and a relaxed form of the complementarity condition, called a "centering"

condition, in the nonlinear system of equations

$$H(y,F,f,f^{(0)},Z,z,z^{(0)}) = \begin{bmatrix} \begin{bmatrix} \mathbf{A}_s^T \\ \mathbf{A}_q^T \\ \mathbf{A}_\ell^T \end{bmatrix} y + \begin{bmatrix} \mathbf{svec}(F) \\ f \\ f^{(0)} \end{bmatrix} - \begin{bmatrix} \mathbf{c}_s \\ \mathbf{c}_q \\ \mathbf{c}_\ell \end{bmatrix} \\ [\mathbf{A}_s \quad \mathbf{A}_q \quad \mathbf{A}_\ell] \begin{bmatrix} \mathbf{svec}(Z) \\ z \\ z^{(0)} \end{bmatrix} - b \\ \mathbf{svec}(\tfrac{1}{2}(ZF+FZ) - \mu I) \\ \mathbf{Arw}(z)\mathbf{Arw}(f)\mathbf{e}_q - \mu \mathbf{e}_q \\ \mathbf{diag}(z^{(0)})\mathbf{diag}(f^{(0)})\mathbf{e}_\ell - \mu \mathbf{e}_\ell \end{bmatrix} = 0,$$

where $\mu > 0$ is a parameter. It is well known that, for all $\mu > 0$, this nonlinear system of equations has a unique solution satisfying the cone inequality constraints, and the set of such solutions is called the *central path* [296]. As $\mu \to 0$, this central solution converges to a (not necessarily unique) primal-dual solution of the SQLP.

The algorithm we now outline uses Newton's method to approximately follow the central path as $\mu \to 0$. For a given approximate primal-dual solution $(y,F,f,f^{(0)},Z,z,z^{(0)})$ and central path parameter μ, Newton's method defines a "search direction" by the linear system of equations

$$\mathbf{J} \begin{bmatrix} \Delta y \\ \mathbf{svec}(\Delta F) \\ \Delta f \\ \Delta f^{(0)} \\ \mathbf{svec}(\Delta Z) \\ \Delta z \\ \Delta z^{(0)} \end{bmatrix} = -H(y,F,f,f^{(0)},Z,z,z^{(0)}), \tag{2.13}$$

where the Jacobian $\mathbf{J}$ of H is the matrix

$$\begin{bmatrix} \mathbf{A}_s^T & I & 0 & 0 & 0 & 0 & 0 \\ \mathbf{A}_q^T & 0 & I & 0 & 0 & 0 & 0 \\ \mathbf{A}_\ell^T & 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{A}_s & \mathbf{A}_q & \mathbf{A}_\ell \\ 0 & I*Z & 0 & 0 & I*F & 0 & 0 \\ 0 & 0 & \mathbf{Arw}(z) & 0 & 0 & \mathbf{Arw}(f) & 0 \\ 0 & 0 & 0 & \mathbf{diag}(z^{(0)}) & 0 & 0 & \mathbf{diag}(f^{(0)}) \end{bmatrix}.$$

Here the symbol $*$ is the "symmetrized Kronecker product" [8]: For symmetric block-diagonal matrices A, B, X (with block sizes $n_1, \ldots, n_L$),

$$(A*B)\,\mathbf{svec}X = \frac{1}{2}\mathbf{svec}\,(AXB + BXA)\,.$$

This linear system is not solved directly but via an efficient method exploiting block Gaussian elimination that is outlined in the next section.

Once primal and dual "search directions" are computed, it is necessary to compute steps along these to the relevant cone boundaries, so that subsequent iterates will respect the inequality constraints. This requires an eigenvalue computation of a block-diagonal symmetric matrix to determine the step to the boundary of the semidefinite cone, the solution of M quadratic equations to determine the step to the boundary of the quadratic cone, and an ordinary ratio test to determine the step to the boundary of the positive

orthant. The minimum of these step lengths determines the maximum step that can be taken. This procedure is used to determine one step length for the primal variables and another for the dual variables.

At a more detailed level, the algorithm uses a predictor-corrector scheme. This requires the following steps, which constitute one interior-point iteration:

- Given a current iterate $(y, F, f, f^{(0)}, Z, z, z^{(0)})$, compute a *predictor* search direction by solving the linear system (2.13), with $\mu = 0$, for $(\Delta y^{\text{pred}}, \Delta F^{\text{pred}}, \Delta f^{\text{pred}}, (\Delta f^{(0)})^{\text{pred}}, \Delta Z^{\text{pred}}, \Delta z^{\text{pred}}, (\Delta z^{(0)})^{\text{pred}})$.

- Compute the primal and dual step lengths to the cone boundaries along this direction, α^{pred} and β^{pred}, respectively, so that

$$\begin{pmatrix} F^{\text{pred}} = F + \alpha^{\text{pred}} \Delta F^{\text{pred}} \\ f^{\text{pred}} = f + \alpha^{\text{pred}} \Delta f^{\text{pred}} \\ (f^{(0)})^{\text{pred}} = f^{(0)} + \alpha^{\text{pred}} (\Delta f^{(0)})^{\text{pred}} \end{pmatrix}$$

  and

$$\begin{pmatrix} Z^{\text{pred}} = Z + \beta^{\text{pred}} \Delta Z^{\text{pred}} \\ z^{\text{pred}} = z + \beta^{\text{pred}} \Delta z^{\text{pred}} \\ (z^{(0)})^{\text{pred}} = z^{(0)} + \beta^{\text{pred}} (\Delta z^{(0)})^{\text{pred}} \end{pmatrix}$$

  are each on the boundary of the product of the semidefinite cone, the quadratic cone, and the positive orthant.

- Compute

$$\mu = \frac{\left(\mathbf{Tr} Z^{\text{pred}} F^{\text{pred}} + (z^{\text{pred}})^T f^{\text{pred}} + ((z^{(0)})^{\text{pred}})^T (f^{(0)})^{\text{pred}}\right)^3}{\nu \left(\mathbf{Tr} ZF + z^T f + (z^{(0)})^T f^{(0)}\right)^2},$$

  where $\nu = \sum_{k=1}^{M} n_k + \sum_{k=1}^{L} (p_k + 1) + p_0$. This expression for μ is known as Mehrotra's formula.

- Recompute the relaxed complementarity conditions in the right-hand side of (2.13) using the Mehrotra formula for μ and including second-order corrections, by substituting

$$\begin{bmatrix} \mathbf{svec}(\mu I - \frac{1}{2}(ZF + FZ + \Delta Z^{\text{pred}} \Delta F^{\text{pred}} + \Delta F^{\text{pred}} \Delta Z^{\text{pred}})) \\ \mu \mathbf{e}_q - (\mathbf{Arw}(z)\mathbf{Arw}(f) + \mathbf{Arw}(z^{\text{pred}})\mathbf{Arw}(f^{\text{pred}}))\mathbf{e}_q \\ \mu \mathbf{e}_\ell - (\mathbf{diag}(z^{(0)})\mathbf{diag}(f^{(0)}) + \mathbf{diag}((z^{(0)})^{\text{pred}})\mathbf{diag}((f^{(0)})^{\text{pred}})\mathbf{e}_\ell \end{bmatrix}$$

  for the last three components in the right-hand side of (2.13). Solve the linear system again for a *corrector* search direction.

- Compute primal and dual step lengths α and β along the corrector search direction and update the primal and dual variables accordingly. The step lengths are chosen so that the new iterates satisfy the inequality constraints. Furthermore, they are never set to a value larger than unity. This can be done by taking the step length to be the smaller of unity and a fixed fraction (say, 0.999) times the step to the boundary of the cone. If a unit primal (dual) step is taken, the new iterate also satisfies the primal (dual) equality constraints.

The choice of the sequence of values for μ greatly influences the performance of the algorithm. The method described above is Mehrotra's predictor-corrector scheme [276], originally proposed for linear programming, which significantly reduces the number of

interior-point iterations needed to compute a solution of given accuracy. The predictor step helps calculate a target value for μ, while the corrector step includes some second-order terms lost in the linearization of H in the predictor step. For a detailed interpretation of Mehrotra's predictor-corrector scheme, see [423, pp. 193–198].

2.3 Software

In this section, we first discuss some implementation details involved in solving the linear system arising in each interior-point iteration. Then we highlight the main features of SDPpack (Version 0.9 Beta),[4] a package (based on Matlab 5.0) for solving SQLP, written by the authors together with F. Alizadeh and S. Schmieta.

2.3.1 Some implementation issues

The Newton system (2.13) admits a reduction to a compact Schur complement form via block Gauss elimination. Let

$$\begin{aligned} \mathbf{U} &= \mathbf{diag}(I * F, \mathbf{Arw}(f), f^0), \\ \mathbf{V} &= \mathbf{diag}(I * Z, \mathbf{Arw}(z), z^0), \\ \mathbf{A} &= [\mathbf{A}_s \ \mathbf{A}_q \ \mathbf{A}_\ell], \end{aligned} \tag{2.14}$$

so that

$$\mathbf{J} = \begin{bmatrix} \mathbf{A}^T & I & 0 \\ 0 & 0 & \mathbf{A} \\ 0 & \mathbf{V} & \mathbf{U} \end{bmatrix}.$$

Eliminating $(\mathbf{svec}(\Delta F), \Delta f, \Delta f^0)$ in (2.13), the matrix $\mathbf{J}$ transforms to

$$\begin{bmatrix} \mathbf{A}^T & -\mathbf{V}^{-1}\mathbf{U} \\ 0 & \mathbf{A} \end{bmatrix},$$

and on further elimination of $(\mathbf{svec}(\Delta Z), \Delta z, \Delta z^0)$, we arrive at the $m \times m$ Schur complement matrix

$$M \triangleq \mathbf{A}\mathbf{U}^{-1}\mathbf{V}\mathbf{A}^T$$

in a linear system involving only the variable Δy. The Schur complement matrix M can be computed as the sum of the individual Schur complements for the semidefinite (M_s), the quadratic (M_q), and the linear (M_ℓ) parts. Moreover, M_s and M_q can be computed *blockwise*. Writing $\mathbf{A}_s^{(k)}$ (likewise $\mathbf{A}_q^{(k)}$) to mean the columns of $\mathbf{A}_s$ (likewise $\mathbf{A}_q$) corresponding to the kth block, we can verify that $M = M_s + M_q + M_\ell$, where

$$\begin{aligned} M_s &\triangleq \sum_{k=1}^{L} \mathbf{A}_s^{(k)} (I * F^{(k)})^{-1} (I * Z^{(k)}) (\mathbf{A}_s^{(k)})^T, \\ M_q &\triangleq \sum_{k=1}^{M} \mathbf{A}_q^{(k)} \mathbf{arw}(\phi^{(k)}, f^{(k)})^{-1} \mathbf{arw}(\zeta^{(k)}, z^{(k)}) (\mathbf{A}_q^{(k)})^T, \\ M_\ell &\triangleq \mathbf{A}_\ell \, \mathbf{diag}(f^{(0)})^{-1} \, \mathbf{diag}(z^{(0)}) \mathbf{A}_\ell^T. \end{aligned} \tag{2.15}$$

The contribution to the (i,j) entry of M_s from the kth semidefinite block can be computed as $F_i^{(k)} \bullet V_j^{(k)}$, where the $n_k \times n_k$ symmetric matrix $V_j^{(k)}$ is the solution to the

[4] Available from the SDPpack web page: http://www.cs.nyu.edu/overton/sdppack.

Lyapunov equation

$$F^{(k)}V_j^{(k)} + V_j^{(k)}F^{(k)} = R_j^{(k)}, \tag{2.16}$$

where $R_j^{(k)} = Z^{(k)}F_j^{(k)} + F_j^{(k)}Z^{(k)}$. The standard method to solve this equation is via the spectral factorization $F^{(k)} = P\,\mathbf{diag}(\lambda)\,P^T$, where P is an orthonormal matrix of eigenvectors and $\lambda = [\lambda_1, \ldots, \lambda_{n_k}]^T$, a vector of eigenvalues. The Lyapunov equation is solved by computing

$$S = (P^T R_j^{(k)} P) \circ L, \text{ where } L(s,t) = 1/(\lambda_s + \lambda_t), \text{ and} \tag{2.17}$$

$$V_j^{(k)} = PSP^T, \tag{2.18}$$

where the symbol $\circ$ in (2.17) denotes the Hadamard (or componentwise) product. The contribution from the kth quadratic block to M_q can be computed efficiently by observing that the arrow matrix $\mathbf{arw}(\zeta^{(k)}, z^{(k)})$ is the sum of a diagonal matrix and a matrix of rank two.

2.3.2 Using SDPpack

We now describe how to use the main routines of SDPpack (Version 0.9 beta); in what follows, >> denotes the Matlab prompt.

The problem data and an initial starting point are stored using structures:

Matlab variables	Data
`A.s`, `A.q`, `A.l`	$\mathbf{A}_s$, $\mathbf{A}_q$, $\mathbf{A}_\ell$
`b`	b
`C.s`, `C.q`, `C.l`	F_0, $\mathbf{c}_q$, $\mathbf{c}_\ell$
`blk.s`, `blk.q`, `blk.l`	$[n_1, \ldots, n_L]^T$, $[p_1 + 1, \ldots, p_M + 1]^T$, p_0
`X.s`, `X.q`, `X.l`	Z, z, $z^{(0)}$
`y`	y
`Z.s`, `Z.q`, `Z.l`	F, f, $f^{(0)}$

For example, the following lines of Matlab code set up a randomly generated SQLP:

```
>> setpath;   % add SDPpack subdirectories to Matlab path
>> blk.s = [5 5 5];  blk.q = [10 10];  blk.l = 30;
>> m = 10;
>> sqlrnd;    % set up a random, strictly feasible SQLP
```

The constraint matrix `A.s` may be set up using the routine `makeA`, which assumes that the matrices F_i are stored using Matlab's cell arrays. Here is an example with two semidefinite blocks and one quadratic block:

```
>> clear
>> blk.s = [3 3]; blk.q = 3; m = 2;
>> b = [1 2]';
>> negF{1} = speye(6);
>> negF{2}(1:3,1:3) = sparse([1 2 3; 2 4 5; 3 5 6]);
>> negF{2}(4:6,4:6) = sparse([1 0 0; 0 2 2; 0 2 3]);
>> A.s = makeA(negF, blk.s);  % creates the 2 x 12 matrix A.s
>> C.s(1:3,1:3) = sparse(ones(3,3));
>> C.s(4:6,4:6) = sparse(zeros(3,3));
>> A.q = [1 2 3; 4 5 6];
>> C.q = [1 -1 -1]';
```

Note that when there is more than one semidefinite block (as in the example above with two blocks), the constraint matrices `negF{`i`}` containing $-F_i$ $(i = 1, \ldots, m)$ and the cost matrix `C.s` containing F_0 must be stored in sparse format. The routines `import` (`export`) can be used to load (save) ASCII data from (to) a file.

Once the problem data are set up, the optimization parameters, which are stored in the structure `opt`, must be assigned appropriate values; these may be initialized to their default values with the routine

```
>> setopt;
```

The next step is to provide a starting point, i.e., the variables `X`, `y`, and `Z`. Generally, it suffices to provide an infeasible starting point, i.e., points for which (2.5)–(2.7) and (2.10) need not necessarily hold, although the cone inequalities (2.8) and (2.11) must be satisfied. A default starting point (usually infeasible) may be generated with the routine

```
>> init;
```

which initializes the primal variable y to zero, the primal slack variables `Z.s`, `Z.q`, `Z.l` to `scalefac` $*$ $(I, \mathbf{e}_q, \mathbf{e}_\ell)$, and the dual variables `X.s`, `X.q`, `X.l` to `scalefac` $*$ $(I, \mathbf{e}_q, \mathbf{e}_\ell)$, where `scalefac` is a user-defined scale factor set to a default value by `setopt`. Again, if there is more than one semidefinite block, the matrices `X.s` and `Z.s` must be stored in the sparse format.

The solver can then be invoked with the Matlab function call

```
>> [X, y, Z, iter, compval, feasval, objval, termflag] = ...
      fsql(A, b, C, blk, X, y, Z, opt);
```

or by the driver script

```
>> sql;
```

which optionally prints out additional summary information[5] about the accuracy to which the problem was solved and the CPU time consumed. In the later iterations, it is common to see Matlab warnings about ill-conditioned systems of linear systems. This ill conditioning is a common phenomenon in interior-point methods. The warnings may be suppressed by the Matlab command `warning off`.

When the algorithm terminates, a termination flag indicates the cause for termination, which is usually one of the following:

- A solution of the requested accuracy has been found. In this case, the sum of the primal infeasibility, the dual infeasibility and complementarity at the computed solution satisfies both an absolute and a relative tolerance, specified in the `opt` structure.
- An iterate $(F, f, f^{(0)})$ or $(Z, z, z^{(0)})$ violates a cone constraint—a normal phenomenon that usually happens due to numerical rounding errors in the last stages of the algorithm. In this case, the previous iterate, which *does* satisfy the cone constraints, is returned as the computed solution.
- The Schur complement matrix is numerically singular. If this occurs in the early stages of the algorithm, the most likely cause is linear dependence in the rows of the constraint matrix **A**. Preprocessing can be done (cf. section 2.3.3) to detect and rectify this situation.

[5]References to "primal" and "dual" in the output (and in the User Guide) actually refer to the "dual" and "primal" problems in the notation used here.

- The algorithm fails to make sufficient progress in an iteration. The progress made in an iteration is deemed insufficient (based on parameters in the opt structure) if the step causes an inordinate increase in the primal or the dual infeasibilities without a commensurate decrease in complementarity. Again, the previous iterate is returned as the computed solution.

- The norm of $(F, f, f^{(0)})$ or of $(Z, z, z^{(0)})$ exceeds a threshold (defined by the user in the opt structure). Primal (dual) unboundedness is an indication of dual (primal) infeasibility.

For full details, we refer the reader to the SDPpack User Manual [6]; see also Table 2.1.

Due to the loop overhead in Matlab, each block diagonal matrix is stored as one large sparse matrix, instead of as individual blocks. This has the advantage that a sparse block in such a matrix is efficiently handled using Matlab's sparse matrix operations. On the other hand, a dense block suffers the penalty involved in sparse matrix access. The loop overhead in Matlab also hinders the effective use of sparsity in the formation of the Schur complement matrix. For instance, in the truss topology design problems, most of the blocks in every constraint matrix are completely zero, and hence do not contribute to the Schur complement at all (see (2.15)), but this could not be exploited effectively. A stand-alone C code currently under development incorporates some techniques to improve performance on sparse problems.

2.3.3 Miscellaneous features

The package provides several support routines that include the following:

- The routines `scomp`, `pcond`, and `dcond` to examine complementarity and degeneracy conditions [7, 132]. These may be used, in particular, to check whether the computed primal and dual solutions are unique.

- The routine `makesql` to generate random SQLPs with solutions having prescribed "ranks." This routine lets the user specify

  - for each semidefinite block k, the rank of $F^{(k)}$ (likewise $Z^{(k)}$);
  - for each quadratic block k, whether $f^{(k)}$ (likewise $z^{(k)}$) lies in the interior of the quadratic cone, or on its boundary, or is identically zero; and
  - for the linear block, the number of components of $f^{(0)}$ (likewise $z^{(0)}$) that are positive.

  Since nondegeneracy automatically implies bounds on ranks of the solution [7], this routine is particularly useful in generating test problems whose solutions have specified degeneracy and complementarity properties.

- The routine `preproc` to examine the consistency of the constraints. If they are consistent, the redundant constraints, if any, are eliminated via a QR factorization of the constraint matrix **A** (see (2.14)).

- The specialized solvers `dsdp` and `lsdp` (based on the "XZ method" [190, 235]) that exploit the special structure of problems arising in combinatorial optimization (for example, MAX-CUT relaxations and the Lovasz θ-function). In these applications, the number of semidefinite blocks $L = 1$ and the matrices $F_1^{(1)}, \ldots, F_m^{(1)}$ are all of the form $e_i e_i^T$ or $e_i e_j^T + e_j e_i^T$. Here e_i stands for the unit vector with 1 in the ith location and zeros everywhere else.

More information about any routine is available by typing `help <routine name>` from the Matlab prompt.

2.4 Applications

We consider two examples of SQLP.

Robust least squares. This problem is taken from [125]. Given $B, \Delta B \in \mathbf{R}^{r\times s}$ $(r > s)$ and $d, \Delta d \in \mathbf{R}^r$, we want to compute the *unstructured robust least squares* solution $x \in \mathbf{R}^s$ that minimizes the worst-case residual

$$\min_x \max_{\|\,[\Delta B \;\; \Delta d]\,\|_F \le \rho} \|(B+\Delta B)x - (d+\Delta d)\|\,,$$

where the perturbation $[\Delta B \;\; \Delta d]$ is unknown, but the bound ρ on its norm is known. It is shown in [125] that this problem can be formulated using quadratic cone constraints. Adding componentwise bound constraints, say, $x_{\min} \le x \le x_{\max}$, converts this into an SQLP with a quadratic and a linear component.

Clock mesh design. This circuit design problem, taken from [405], deals with the propagation of a clock signal in integrated circuits. A rectangular mesh with fixed grid points is given. The interconnecting wire segment between grid points i and j is assumed to be a rectangular strip of a fixed length, but variable width d_{ij}. The electrical characteristics of each wire segment are modeled as a π-segment, i.e., a grounded input capacitance, a grounded output capacitance, and a connecting conductance, all of which are assumed to depend affinely on the design parameters d_{ij}. A synchronous clock signal (of fixed frequency) applied to a fixed subset of the grid points has to be propagated to the entire mesh with minimal delay. There is an inherent trade-off between the *dominant delay* (see [405] for the definition) and the power consumption of the circuit. The goal is to optimally choose the widths d_{ij} of the rectangular wire segments so that the power consumption of the circuit is minimized for a fixed value of the dominant delay that we are willing to tolerate. It is shown in [405] that this can be formulated as a semidefinite program (SDP). Physical limitations on the width of the segments further impose bound constraints of the form $d_{\min} \le d_{ij} \le d_{\max}$, which results in an SQLP with a semidefinite and a linear block. Now consider a slight variation of this problem: The objective is the same as before, but now we have a rectangular mesh whose grid spacing is not fixed. Here the wire widths are fixed, but the spacing l_i between successive columns and r_j between successive rows of grid points are variables with $l_{\min} \le l_i \le l_{\max}$ and $r_{\min} \le r_i \le r_{\max}$. Let us further impose a restriction on the maximum allowable size of the circuit by imposing a bound on the length of the diagonal of the rectangular mesh. This last bound constraint is a convex quadratic constraint, thus resulting in an SQLP with semidefinite, quadratic, and linear components.

2.5 Numerical results

In this section, we present numerical results on a variety of test problems, both randomly generated and from applications. The latter come from control theory (LMI problems) and truss topology design. The randomly generated problems include general dense problems, the unstructured robust least squares problem, and problems with specific sparsity structure such as those arising from the clock mesh design application.

Table 2.1: Legend for termination flag `tf`.

`tf`	Description
−2	$\|\|(Z, z, z^{(0)})\|\|$ exceeds user-defined limit.
−1	$\|\|(F, f, f^{(0)})\|\|$ exceeds user-defined limit.
0	A solution of requested accuracy has been found.
1	$(Z, z, z^{(0)})$ or $(F, f, f^{(0)})$ is outside the cone or on its boundary.
2	F has a nonpositive eigenvalue.
3	Schur complement matrix is numerically singular.
4	Insufficient progress.
5	Primal or dual step length is too small.
6	Exceeded maximum number of iterations allowed.
7	Failed data validation checks.

In the tables, the first column (#) is the problem number, the second column is the dimension of the primal slack variable

$$\dim \triangleq \sum_{k=1}^{L} n_k(n_k+1)/2 + \sum_{k=0}^{M}(p_k+1) + p_0,$$

m is the dimension of y, `iter` is the number of interior-point iterations, r_d, r_p, and r_c are, respectively, the dual infeasibility norm (see (2.10)), the primal infeasibility norm (see (2.5)–(2.7)), and the complementarity $\mathbf{Tr}(ZF) + z^T f + (z^{(0)})^T f^{(0)}$ evaluated at the computed solution. The quantity `scalefac` is the factor by which the default starting point $((I, \mathbf{e}_q, \mathbf{e}_\ell), 0, (I, \mathbf{e}_q, \mathbf{e}_\ell))$ is scaled. When `scalefac` is omitted, it means that the default value of 100 was used. The quantities r_p, r_d, and r_c are all shown on a log (base 10) scale. The last column (`tf`) is the termination flag (denoted `termflag` in the *SDPpack User Guide*). A brief description of this flag is given in Table 2.1; for a more detailed explanation, consult the *SDPpack User Guide* [6]. The asterisks, if any, next to a termination flag indicate the number of *restarts* that were necessary: If the primal or the dual step lengths get very small, the starting point is scaled with a larger value of `scalefac`, the fraction of the step taken toward the boundaries of the cones is made more conservative (i.e., reduced), and the algorithm is restarted. The primal and the dual objective values, denoted $\mathcal{P}(y)$ and $\mathcal{D}(Z, z, z^{(0)})$ in the tables, are provided only for the LMI and the truss topology design problems,[6] and not for any of the randomly generated ones.

All the runs were performed on a Sun Sparc Ultra II workstation (200 MHz CPU with 192 MB of memory). Unless explicitly noted otherwise in the tables, all the problems were solved with SDPpack's options set to their default values.

Random dense problems. Table 2.2 shows results for problems with only semidefinite constraints. Table 2.3 shows results for problems with only quadratic cone constraints. In Table 2.2, each problem has three blocks of equal size; the five runs tabulated there correspond to block sizes of 20, 40, 60, 80, and 100. Each problem in Table 2.3 also has three blocks of equal size; the block sizes for the five runs are 50, 100, 150, 200, and 250.

Clock mesh design. This problem from [405] was described in section 2.4. These are problems with one semidefinite block and a linear block. The constraint matrices

[6] Available from the SDPpack web page.

Table 2.2: Randomly generated with semidefinite blocks only.

#	dim	m	iter	r_d	r_p	r_c	CPU secs	tf
1	630	60	14	-13	-14	-14	1.88e+01	0
2	2460	120	13	-13	-13	-12	1.57e+02	0
3	5490	180	15	-12	-13	-11	8.45e+02	0
4	9720	240	13	-12	-13	-12	2.13e+03	0
5	15150	300	14	-10	-13	-07	5.72e+03	4

Table 2.3: Randomly generated with quadratic blocks only.

#	dim	m	iter	r_d	r_p	r_c	CPU secs	tf
1	150	50	8	-14	-15	-10	5.40e-01	0
2	300	100	8	-13	-14	-13	1.29e+00	0
3	450	150	8	-13	-13	-10	3.48e+00	0
4	600	200	8	-12	-14	-11	7.00e+00	0
5	750	250	8	-13	-13	-10	1.28e+01	0

Table 2.4: Clock mesh design problems (`scalefac` = 1000): #1 is an 8×8 mesh, and #2 a 12×12 mesh.

#	dim	m	iter	r_d	r_p	r_c	CPU secs	tf
1	3609	144	16	-13	-16	-09	1.92e+02	0
2	14989	312	21	-12	-16	-09	2.84e+03	0

are extremely sparse, and the spatial mesh adjacency relation they describe gives them a banded structure; the results are shown in Table 2.4.

Robust least squares. This problem from [125] was described in section 2.4. Numerical results for this problem class are shown in Table 2.5.

Truss topology design. These problems are pure SDPs that arise in truss topology design. The constraint matrices here are extremely sparse even for the moderately sized problems. The results are summarized in Table 2.6.

LMI problems. These problems are pure SDPs that arise in control applications. On the problems where the final complementarity residual is unsatisfactory (see Table 2.7), the dual iterates indeed get large, indicating that the primal problem may be infeasible or weakly infeasible [257].

2.6 Other computational issues

Several enhancements to the basic interior-point iteration of section 2.2 have been proposed, and some used with success, in linear programming. We are concerned with extending some of these techniques to SQLP, namely,

- Gondzio's multiple centrality corrections [172], and
- Mehrotra's higher-order corrections [275].

Table 2.5: Unstructured robust-least squares problems (`scalefac` = 1000).

#	dim	m	`iter`	r_d	r_p	r_c	CPU secs	`tf`
1	1103	102	7	–13	–13	–09	3.32e+00	0
2	2203	202	7	–12	–13	–09	1.74e+01	0
3	3303	302	7	–12	–13	–09	4.93e+01	0
4	4403	402	7	–12	–12	–09	1.02e+02	0
5	5503	502	7	–12	–12	–09	1.86e+02	0

Table 2.6: Truss topology design.

#	dim	m	`iter`	r_d	r_p	r_c	$\mathcal{D}(Z,z,z^{(0)})$	$\mathcal{P}(y)$	CPU secs	`tf`
1	19	6	10	–14	–15	–11	9.00e+00	9.00e+00	4.20e–01	0
2	331	58	12	–13	–14	–11	1.23e+02	1.23e+02	4.81e+00	0
3	91	27	15	–14	–15	–08	9.11e+00	9.11e+00	1.86e+00	4
4	37	12	11	–13	–16	–11	9.01e+00	9.01e+00	6.10e–01	0
5	1816	208	15	–10	–14	–09	1.33e+02	1.33e+02	7.28e+01	0
6	901	172	28	–07	–13	–07	9.01e+02	9.01e+02	5.85e+01	4
7	451	86	29	–06	–13	–08	9.00e+02	9.00e+02	2.40e+01	4
8	6271	496	18	–10	–14	–11	1.33e+02	1.33e+02	1.13e+03	0

The main motivation behind considering these enhancements for SQLP is the ratio of the cost of forming and factorizing the Schur complement matrix relative to the cost of a backsolve. For SQLP, this ratio is typically much larger than for LP. Thus, having expended the effort involved in forming and factorizing the Schur complement matrix, it is worthwhile to improve the quality of the search direction by solving multiple linear systems with the same coefficient matrix but different right-hand sides. Both the above schemes can be extended to SQLP and are described in detail elsewhere [184]. In our preliminary experiments, adapting Gondzio's multiple centrality corrections to SQLP reduces iteration count and CPU time roughly to the same extent as reported for LP. On the other hand, Mehrotra's higher-order corrections reduce the number of iterations and the overall CPU time significantly; this observation is different from LP experience, where the overall CPU time is typically not reduced.

2.7 Conclusions

In this chapter, we have described mixed SQLPs. These are linear optimization problems with mixed cone constraints, namely, the semidefinite cone, the quadratic cone, and the nonnegative orthant. This problem class includes pure semidefinite programming, convex quadratically constrained quadratic programming, and linear programming as special cases. We have presented a primal-dual interior-point algorithm for such problems and have described the salient features of SDPpack (Version 0.9 beta), a software package that implements this algorithm. A couple of examples have been given to illustrate how such mixed programs arise in practice, and the numerical results shown here demonstrate the performance of SDPpack on several classes of test problems. Finally, we have briefly listed some enhancements of interior-point algorithms for LP; their extensions to SQLP have been implemented in a new stand-alone C code currently under development.

Table 2.7: LMI problems from $\mathbf{H}_\infty$ control (`scalefac` = 1000).

#	d	m	`iter`	r_d	r_p	r_c	$\mathcal{D}(Z,z,z^{(0)})$	$\mathcal{P}(y)$	CPU secs	`tf`
1	41	13	12	–08	–12	–10	–2.03e+00	–2.03e+00	7.60e–01	2
2	51	13	14	–09	–11	–08	–1.10e+01	–1.10e+01	9.20e–01	4
3	51	13	16	–05	–12	–10	–5.70e+01	–5.69e+01	9.90e–01	1
4	51	13	15	–06	–10	–11	–2.75e+02	–2.75e+02	9.10e–01	2
5	51	13	17	–04	–12	–09	–3.63e+02	–3.63e+02	1.04e+00	1
6	51	13	33	–02	–11	–08	–4.49e+02	–4.49e+02	2.00e+00	2
7	51	13	11	–05	–11	–07	–3.91e+02	–3.91e+02	6.90e–01	1
8	51	13	15	–05	–11	–09	–1.16e+02	–1.16e+02	9.10e–01	1
9	51	13	17	–07	–13	–11	–2.36e+02	–2.36e+02	1.05e+00	2
10	66	21	27	–05	–07	–05	–1.09e+02	–1.09e+02	2.30e+00	–1
11	97	31	23	–07	–07	–04	–6.60e+01	–6.59e+01	3.23e+00	–1
12	120	43	25	–08	–07	–01	–2.00e–01	–1.78e–01	5.31e+00	–1
13	178	57	29	–04	–09	–02	–4.44e+01	–4.44e+01	8.73e+00	2
14	227	73	30	–07	–09	–03	–1.30e+01	–1.30e+01	1.39e+01	5
15	273	91	18	–06	–08	–02	–2.39e+01	–2.40e+01	2.23e+01	–1*
16	51	13	17	–07	–13	–11	–2.36e+02	–2.36e+02	1.08e+00	2

Acknowledgments

We thank Pascal Gahinet, Arkadii Nemirovskii, and Lieven Vandenberghe for supplying us with the LMI, the truss, and the clock mesh design problems, respectively.

This page intentionally left blank

Chapter 3

Nonsmooth Algorithms to Solve Semidefinite Programs

Claude Lemaréchal and François Oustry

3.1 Introduction

Many practical applications of LMI methods require the solutions of some large-scale semidefinite programs (SDPs). In this chapter, we describe nonsmooth algorithms to solve SDPs, with two motivations in mind:

- to increase the size of tractable eigenvalue optimization problems,
- to improve asymptotic properties of the algorithms.

Both properties cannot be achieved simultaneously. Yet a good trade-off must be identified between problem size and speed of convergence; and this happens to be possible with algorithms for nonsmooth optimization, which in addition offer a lot of flexibility.

After a short introduction of the objects necessary for later understanding, we outline in section 3.2 the existing nonsmooth algorithms that can be used "off the shelf" in the present framework. Among them are bundle methods, which have recently been specialized to exploit the particular structure of the problem; section 3.3 is devoted to them, while their specialized versions form section 3.4. Finally, even though the problem belongs to the realm of nonsmooth optimization, it lends itself to second-order analysis. Accordingly, second-order algorithms can be applied, which bring good asymptotic properties. This is the subject of section 3.5, which contains a condensed introduction to the necessary theory, together with corresponding implementations. Comparative numerical illustrations are given in section 3.6, and we conclude with a section on possible future developments.

Original structure: Convexity. Let us write an SDP as

$$\text{minimize } c^T x \quad \text{subject to} \quad \lambda_{\max}(F(x)) \leq 0\,, \tag{3.1}$$

where $c \in \mathbf{R}^m$ is the cost vector and F is an affine mapping from $\mathbf{R}^m$ to $\mathcal{S}_n$: $F(x) = F_0 + \mathcal{F}x$, with $\mathcal{F}x = \sum_{i=1}^m x_i F_i$.

The key characteristic of the problem is the constraint-function $f := \lambda_{\max} \circ F$, more precisely, the function $X \mapsto \lambda_{\max}(X)$. This latter function is convex: it is the support function of the compact convex set

$$\mathcal{C}_n := \{V \in \mathcal{S}_n : V \succeq 0,\ \mathbf{Tr}\, V = 1\}.$$

This can be seen from the Rayleigh variational formulation

$$\lambda_{\max}(X) = \max_{v \in \mathbf{R}^n,\, |v|=1} v^T X v = \max_{V \in \mathcal{C}_n} V \bullet X, \tag{3.2}$$

where $\bullet$ is the standard scalar product in $\mathcal{S}_n$.

Interior-point methods transform the original problem (3.1) to a smooth one by adding some structure, thereby obtaining polynomiality; but some information is then lost, which is useful for local convergence. By contrast, nonsmooth analysis directly handles the problem with its "pure structure."

The role of convex analysis was first emphasized by R. Bellman and K. Fan in [37] and developed further in [195, 247], among others. A first step is to express the subdifferential $\partial\lambda_{\max}(X)$, and this comes easily from elementary convex analysis: indeed (see, for example, Proposition VI.2.1.5 in [194]) (3.2) shows that this subdifferential is the face of $\mathcal{C}_n$ exposed by X. In other words, let r be the multiplicity of $\lambda_{\max}(X)$ and let Q be an $n \times r$ matrix whose columns form an orthonormal basis of the corresponding eigenspace. Then

$$\partial\lambda_{\max}(X) = \{QZQ^T : Z \in \mathcal{C}_r\}. \tag{3.3}$$

This formulation will be used in the so-called rich-oracle methods. For *poor-oracle* methods we will rather refer to an equivalent form:

$$\partial\lambda_{\max}(X) = \mathbf{Co}\{vv^T :\ Xv = \lambda_{\max}(X)v,\ v^T v = 1\}. \tag{3.4}$$

Once $\partial\lambda_{\max}$ is thus characterized, ∂f is derived by simple chain rules. In fact, denote by $\mathcal{F}^* : \mathcal{S}_n \to \mathbf{R}^m$ the adjoint of $\mathcal{F}$:

$$\mathcal{S}_n \ni V \mapsto \mathcal{F}^* V := [F_1 \bullet V, \ldots, F_m \bullet V]^T.$$

Then the subdifferential of f at x is just the image by $\mathcal{F}^*$ of the subdifferential of $\lambda_{\max}$ at $X = F(x)$:

$$\partial f(x) = \mathcal{F}^* \partial\lambda_{\max}(F(x)). \tag{3.5}$$

Remark 3.1. *A further advantage of this approach via nonsmooth analysis is that it does not rely on affinity of the mapping A:* smoothness *suffices. When A is not affine, f is no longer convex, but its essential* composite *structure is preserved. Namely,* (3.5) *can be applied by replacing the linear operator $\mathcal{F}^*$ by $\mathcal{F}_x^*$:*

$$\mathcal{S}_n \ni V \mapsto \mathcal{F}_x^* V := \left[\frac{\partial F(x)}{\partial x_1} \bullet V, \ldots, \frac{\partial F(x)}{\partial x_m} \bullet V\right]^T.$$

Note that the subdifferential (of $\lambda_{\max}$, and hence of f) is highly unstable: it changes drastically when the multiplicity of $\lambda_{\max}$ changes. As will be seen in section 3.3, the whole idea of so-called bundle methods can be seen as constructing good approximations of a much more stable object than the subdifferential of a convex function, the approximate subdifferential. For $\varepsilon \geq 0$, the ε-subdifferential $\partial_\varepsilon f(x)$ is defined by

$$\partial_\varepsilon f(x) := \{g \in \mathbf{R}^m : f(y) - f(x) \geq g^T(y - x) - \varepsilon\}.$$

It is not hard to see that, for the maximum eigenvalue function,

$$\partial_\varepsilon \lambda_{\max}(X) = \{V \in \mathcal{C}_n : V \bullet X \geq \lambda_{\max}(X) - \varepsilon\}, \tag{3.6}$$

and for $f = \lambda_{\max} \circ A$ the same chain rule still applies:

$$\partial_\varepsilon f(x) = \mathcal{F}^* \partial_\varepsilon \lambda_{\max}(F(x)). \tag{3.7}$$

From (3.6) we see that $\partial_\varepsilon \lambda_{\max}(X)$ is no longer a face of $\mathcal{C}_n$: it is the intersection of $\mathcal{C}_n$ with the half-space $\{V \in \mathcal{S}_n : V \bullet X \geq \lambda_{\max}(X) - \varepsilon\}$. In particular, almost all matrices in $\partial_\varepsilon \lambda_{\max}(X)$ have rank n (for $\varepsilon > 0$), whereas the rank of matrices in $\partial \lambda_{\max}(X)$ is at most the multiplicity r, which is 1 for almost all X; this gives an idea of the big gap existing between these two convex sets.

SDP and maximum eigenvalue minimization. For the sake of simplicity, most of the algorithms we present here are designed to solve the unconstrained problem

$$\min_{x \in \mathbf{R}^m} f(x) = \lambda_{\max}(F(x)). \tag{3.8}$$

Note that nonsmooth algorithms can be used to solve (3.1) as well, using exact penalty or more sophisticated techniques as in [244].

Remark 3.2. *In some cases, for example, when the diagonal of $F(x)$ is fixed* [189] *or free* [307], *transforming* (3.1) *to* (3.8) *can be done efficiently because an efficient value for the penalty coefficient is known. Such is often the case when* (3.1) *is the relaxation of a graph problem; see* [178] *or* [160] *for more on the origins of these problems.*

We also mention that algorithms for nonsmooth optimization extend in a straightforward way to the more general problem

$$\text{minimize} \sum_{i=1}^{r} \lambda_i(F(x)),$$

where $\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues in decreasing order and $r < n$ is a given integer. In fact, the above function is convex as well.

Poor, rich, and intermediate oracles. For a resolution algorithm to work, it is necessary to compute information regarding (3.1) or (3.8) for given matrices $X = F(x)$: Typically, it can be the spectrum of X, or its QR decomposition, etc. We find it convenient to consider this computation as done by an *oracle* that, having X as input, answers the required information. A distinguished feature of nonsmooth optimization is that it can be satisfied with "poor oracles," which compute only the maximal eigenvalue, together with just one associated eigenvector. In view of (3.5) and (3.4), this provides a subgradient of $\lambda_{\max}$ or of f.

Needless to say, poor oracles are advantageous for problems with large n. The price to pay is that the corresponding algorithms usually have modest speed of convergence. On the other hand, nonsmooth optimization accommodates "intermediate" oracles, computing a number $r \in \{1, \ldots, n\}$ of largest eigenvalues and associated eigenvectors (with r possibly varying along the iterations). This supposedly improves performances; it is thus possible to balance sophistication of the oracle with convergence speed of the minimization algorithm.

3.2 Classical methods

The nonsmooth optimization problem (3.8) can be solved by any existing algorithm for nonsmooth optimization, ignoring the particular structure enjoyed by the function $\lambda_{\max}$. Such methods use a poor oracle and are fairly classical. They can be divided into two classes, based on totally different motivations.

Subgradient methods. The methods of the first class have developed in the 1960s, mainly in Russia and Ukraine (see the review [335]). Let the oracle have computed at x_k a subgradient g_k of the form $\mathcal{F}^* vv^T$ for some eigenvector v; see (3.5) and (3.4). Then the basic subgradient method computes

$$x_{k+1} = x_k - t_k W_k g_k \,. \tag{3.9}$$

For the moment, W_k is the identity I_m in $\mathbf{R}^m$. As for the step size $t_k > 0$, known formulas are

$$t_k = \frac{1}{k|g_k|} \quad \text{or} \quad t_k = \frac{\lambda_{\max}(F(x_k)) - \bar{f}}{|g_k|^2}$$

(the second formula is valid only if the optimal value $\bar{f}$ of (3.8) is known).

The above method is purely "Markovian," in the sense that iteration k depends only on the differential information of f at x_k. In the 1970s, N. Z. Shor inserted a memory mechanism to update the preconditioner W_k in (3.9). As explained in his monograph [375], the basic object for this is *space dilation* along a vector $\xi = \xi_k \in \mathbf{R}^m$, i.e., the operator that multiplies by $\alpha = \alpha_k > 0$ the component along ξ of each vector in $\mathbf{R}^m$. The well-known *ellipsoid algorithm* is obtained with $\xi_k = g_k$ and specific values for α_k, t_k. On the other hand, taking $\xi_k = g_k - g_{k-1}$ results in the so-called r-algorithm, lesser known but rather efficient in many cases.

Pure cutting-plane method. The methods of the second class are all based on approximating f by the polyhedral (or cutting-plane) "model"

$$\check{f}_k(x) := \max_{i=1,\dots,k} f(x_i) + g_i^T(x - x_i) \,. \tag{3.10}$$

Here, $f(x_i) = \lambda_{\max}(F(x_i))$ and $g_i = \mathcal{F}^* v_i v_i^T$ (see (3.4) and (3.5)) make up the information computed by the poor oracle at each iterate x_i, prior to the current iteration k. Incidentally, note the heavily "non-Markovian" character of this object: it involves explicitly the past information, stored in a "bundle" $\{f(x_i), g_i\}_{i=1}^k$. The next iterate x_{k+1} is computed with the help of this information, and the whole success of the method will rely on the optimizer's ability to appropriately weight the elements of the bundle.

Now, the pure cutting-plane method uses the simplest strategy: take for x_{k+1} a minimizer of $\check{f}_k$; this amounts to solving the linear program with an additional variable r:

$$\begin{cases} \text{minimize } r\,, \quad (x,r) \in \mathbf{R}^m \times \mathbf{R} \\ \text{subject to } f(x_i) + g_i^T(x - x_i) \le r \ \text{ for } i = 1,\dots,k\,. \end{cases} \tag{3.11}$$

A simple example, due to A. S. Nemirovski and reported in [194, section XV.1.1], illustrates the instability of this scheme: in some situations, billions of iterations may be needed to obtain just one-digit accuracy.

Method of analytic centers. A recent strategy to cope with the above instability is as follows. In the graph-space $\mathbf{R}^m \times \mathbf{R}$, consider the set P_k defined by

$$P_k := \{z = (x, r) : \check{f}_k(x) \leq r \leq \ell_k\};$$

here $\ell_k := \min_i \check{f}_i(x_i)$ is the best function value returned by the oracle up to the kth iteration. Because $\check{f}_k$ is polyhedral, P_k is a convex polyhedron; write it in terms of abstract affine inequalities as

$$P_k = \{z \in \mathbf{R}^m \times \mathbf{R} : c_i(z) \geq 0 \text{ for } i \in I\}.$$

In the pure cutting-plane method, $z_{k+1} = (x_{k+1}, \check{f}_k(x_{k+1}))$ is the lowest point in P_k. A more stable point was defined in [379]: the *analytic center* of P_k, i.e., is the unique solution of the optimization problem

$$\max_{z \in P_k} \prod_{i \in I} c_i(z) \qquad \text{or equivalently} \qquad \max_{z \in P_k} \sum_{i \in I} \log c_i(z).$$

Here, we recognize the usual barrier associated with a convex polyhedron. Naturally, the above optimization problem cannot be solved exactly (this would imply an infinite subalgorithm at each iteration k of the outer nonsmooth optimization algorithm). On the other hand, the numerous works in interior-point methods and complexity theory can be used as a guide to control the number of Newton iterations to be performed for each k, while preserving the main convergence properties of the outer algorithm. An algorithm based on these ideas, ACCPM, is described in [162]. Appropriate implementations of this algorithm have been used successfully for some applications; as for its theoretical properties, see [294], [163] for some first complexity results.

Bundle methods. Another stabilization process is achieved by bundle methods. In the framework or eigenvalue optimization, this was done, for example, in [231, 278, 366] (the method of [231] is best explained in [232]). Actually, these works deal with the nonconvex case ($F(\cdot)$ nonaffine). Numerical experiments are reported in [366] for relaxations of combinatorial problems with $n \simeq 100$ and $m \simeq 1000$.

Actually, these methods have recently received increased attention and rather sophisticated variants, exploiting the structure of the present problem, have been developed. For future reference, we find it convenient to study this class of methods more deeply.

3.3 A short description of standard bundle methods

For a study of bundle methods, we recommend [194, Chapter XV]; see also [230] for their application to nonconvex problems, and the direct resolution of constrained problems such as (3.1), without a detour via (3.8).

We now demonstrate the principles underlying these methods, to minimize a general convex function. At the kth iteration, we have a *stability center*, call it x_k, and we perform the following operations:

(i) we choose a *model*, call it φ_k, supposed to represent f;

(ii) we choose a *norming*, call it $\|\cdot\|_k$;

(iii) we compute a *candidate* y_{k+1} realizing a compromise between diminishing the model

$$\varphi_k(y_{k+1}) < \varphi_k(x_k),$$

and keeping close to the stability center

$$\|y_{k+1} - x_k\|_k \text{ small};$$

(iv) the candidate is then *assessed* in terms of the true function f, and the stability center x_k is
- moved to y_{k+1} if it is deemed good enough (descent step),
- or left as it is (null step).

Thus, two sequences are involved in this description, rather than one: The sequence of iterates is now denoted by $\{y_k\}$, while $\{x_k\}$ is rather a subsequence of selected iterates.

Depending on the specific choices made for the above operations, a whole family of bundle methods can be constructed. We now describe probably the most efficient of them: *cutting planes with stabilization by penalty.*

Such a method has the following features: (i) The model taken for the kth iteration is still the cutting-plane function $\varphi_k := \check{f}_k$ of (3.10). (ii) The norm is split into two parts: first, we choose a symmetric positive definite matrix M_k and we set $\|x\|_k^2 := x^T M_k x$; second, we choose a spring strength $\mu_k > 0$. (iii) The candidate is then the minimizer of the function $\varphi_k(\cdot) + \frac{1}{2}\mu_k\|\cdot - x_k\|_k^2$.

Remark 3.3. *This last function is related to the Moreau–Yosida regularization of f:*

$$\mathbf{R}^m \ni x \mapsto \min_{y\in\mathbf{R}^m} f(y) + \frac{1}{2}\mu_k(y-x)^T M_k(y-x)\,.$$

Call y_{k+1} the unique solution of this last problem when $x = x_k$. Then, insofar as $\varphi_k \simeq f$, y_{k+1} approximates the so-called proximal point

$$p_{[\mu_k M_k]}(x_k) := \mathrm{Argmin}_{y\in\mathbf{R}^m} f(y) + \frac{1}{2}\mu_k(y-x_k)^T M_k(y-x_k)\,.$$

The present stabilization of cutting planes is therefore also called the proximal bundle method.

Then the resulting algorithm has the following form.

Algorithm 3.1 (cutting planes with stabilization by penalty). *The initial point x_1 is given, together with a stopping tolerance $\delta \geq 0$. Choose a spring strength $\mu_1 > 0$, a positive definite matrix $M_1 \in \mathbf{R}^{m\times m}$, and a descent coefficient $\omega \in]0,1[$. Initialize the iteration counter $k = 1$ and $y_1 = x_1$; compute $f(y_1)$ and $g_1 \in \partial f(y_1)$.*

- *STEP* 1. *Compute the solution y_{k+1} of*

$$\min_{y\in\mathbf{R}^m} \varphi_k(y) + \frac{1}{2}\mu_k\|y - x_k\|_k^2 \tag{3.12}$$

and set the decrement of the model

$$\delta_k := f(x_k) - \varphi_k(y_{k+1}) - \frac{1}{2}\mu_k\|y_{k+1} - x_k\|_k^2 \geq 0\,. \tag{3.13}$$

- *STEP* 2. *If $\delta_k \leq \delta$ stop.*

- *STEP* 3. *Call the oracle to obtain* $f(y_{k+1})$ *and* $g_{k+1} \in \partial f(y_{k+1})$. *If*

$$f(y_{k+1}) \leq f(x_k) - \omega\delta_k\,,$$

set $x_{k+1} = y_{k+1}$, *choose* $\mu_{k+1} > 0$ *and* $M_{k+1} \succ 0$. *Otherwise set* $x_{k+1} = x_k$ *(null step).*
Update the model $\varphi_{k+1} = \check{f}_{k+1}$.
Replace k *by* $k+1$ *and loop to Step* 1.

Naturally, the "master program" (3.12) is nothing more than a linear-quadratic program resembling (3.11):

$$\left\{\begin{array}{l}\text{minimize } r + \frac{1}{2}\mu_k\|y - x_k\|_k^2\,, \quad (y,r) \in \mathbf{R}^m \times \mathbf{R}\\ \text{subject to } \quad f(y_i) + g_i^T(y - y_i) \leq r \ \text{ for } i = 1,\ldots,k\,.\end{array}\right. \tag{3.14}$$

This form makes it easier to write the dual of (3.12), a crucial point to fully understand the different variants of bundle methods. Let us denote by $\Delta_k := \{\alpha \in \mathbf{R}_+^k : \sum_{i=1}^k \alpha_i = 1\}$ the unit simplex of $\mathbf{R}^k$ and by e_k^i the *linearization errors* between the y_i's and x_k:

$$\begin{array}{l} e_k^i := f(x_k) - f(y_i) - g_i^T(x_k - y_i) \ \text{ for } \ i = 1,\ldots,k\,,\\ e_k := [e_k^1,\ldots,e_k^k]^T\,.\end{array} \tag{3.15}$$

In the result below we also introduce $W_k := M_k^{-1}$ and the dual norm $\|\cdot\|_{*,k}^2 := (\cdot)^T W_k\,\cdot$.

Lemma 3.1 (see [194, Lemma XV.2.4.1]). *For* $\mu_k > 0$ *and* $M_k \succ 0$, *the unique solution of the penalized problem* (3.12) = (3.14) *is*

$$y_{k+1} := x_k - \frac{1}{\mu_k} W_k \sum_{i=1}^k \alpha_i g_i\,, \tag{3.16}$$

where $\bar{\alpha} \in \mathbf{R}^k$ *solves*

$$\min_{\alpha\in\Delta_k} \frac{1}{2}\left\|\sum_{i=1}^k \alpha_i g_i\right\|_{*,k}^2 + \mu_k \alpha^T e_k\,. \tag{3.17}$$

Furthermore, with δ_k *of* (3.13) *and* $\bar{g}_k := \sum_{i=1}^k \bar{\alpha}_i g_i$, *we have the primal-dual relation*

$$\varepsilon_k := \bar{\alpha}^T e_k = \delta_k - \frac{1}{2\mu_k}\|\bar{g}_k\|_{*,k}^2\,. \tag{3.18}$$

We also give the following result, relating the subdifferentials at previous iterates with the approximate subdifferential at the current stability center.

Proposition 3.1 (see [194, Proposition XIV.2.1.1]). *For all* $y \in \mathbf{R}^m$ *and* $\alpha \in \Delta_k$, *we have*

$$f(y) \geq f(x_k) + \left(\sum_{i=1}^k \alpha_i g_i\right)^T (y - x_k) - \alpha^T e_k\,,$$

where e_k *is the vector of linearization errors defined in* (3.15). *In particular, for* $\varepsilon \geq \min_i e_k^i$,

$$\emptyset \neq S_k(\varepsilon) := \left\{g = \sum_{i=1}^k \alpha_i g_i : \alpha \in \Delta_k,\ \alpha^T e_k \leq \varepsilon\right\} \subset \partial_\varepsilon f(x_k)\,. \tag{3.19}$$

Thus, for $\bar{\alpha}$ solving (3.17), consider ε_k and $\bar{g}_k$ of Lemma 3.1. We see that $\bar{g}_k \in \partial_{\varepsilon_k} f(x_k)$. Indeed we have

$$\bar{g}_k = \text{Argmin}\,\{\|g\|_{*,k} : g \in S_k(\varepsilon_k)\}\,. \tag{3.20}$$

A nice interpretation of Lemma 3.1 in terms of the subgradient algorithm (3.9) ensues; this interpretation will be important for our further development. If we compare (3.16) with (3.9), we see that $(1/\mu_k)$ can be interpreted as a step length and that the chosen subgradient is now the projection (for the current dual norm) of the origin onto the approximation $S_k(\varepsilon_k) \subset \partial_{\varepsilon_k} f(x_k)$.

In what follows, we denote by Algorithm 3.1* the (dual) version of Algorithm 3.1, where we solve (3.17) instead of (3.12) and y_{k+1} is updated by (3.16).

Remark 3.4. *The relation $\bar{g}_k \in \partial_{\varepsilon_k} f(x_k)$ implies*

$$f(y) \geq f(x_k) - \|\bar{g}_k\|\,\|y - x_k\| - \varepsilon_k \quad \text{for all } y \in \mathbf{R}^m.$$

As a result, x_k satisfies an approximate minimality condition if both $\|\bar{g}_k\|$ and ε_k are small. In view of (3.18), *this will be the case when δ_k is small, providing that $\mu_k \lambda_{\max}(M_k)$ is not large. This explains the stopping criterion in Step 2 of the algorithm.*

To conclude this section, we mention another important point: It is actually possible to "clean the bundle" so as to avoid an intolerably growing number of affine functions in the model φ_k. All known implementations of Algorithm 3.1 take $M_k \equiv I_m$ and differ by the management of μ_k (a crucial problem for efficiency). The most recent implementation is described in [246], where some comparative numerical experiments are reported.

3.4 Specialized bundle methods

It is observed in [366] that dramatic improvements can be obtained with a slightly enriched oracle, which computes at x_k not only $f(x_k)$ and some eigenvector v_k, but also a few more eigenvectors corresponding to "almost maximal" eigenvalues. This extra information can be inserted into the the definition of $\tilde{f}_k$, which will approximate f more accurately while staying polyhedral. This is the basis for the methods described in this section.

3.4.1 Rich oracle: Dual methods

The first instance of a nonsmooth algorithm to solve (3.8) is probably the pioneering work [87], which definitely adopts a *dual* approach. As mentioned above, the idea there is to realize that the eigenvectors associated with eigenvalues of $F(x)$ falling within ε of $\lambda_{\max}(F(x))$ yield an enlarged $\partial f(x)$ (denoted by $\delta_\varepsilon f(x)$ below), which turns out to produce good descent directions. A quantitative analysis of this enlargement is given in [307]. At $X \in \mathcal{S}_n$ and for $\varepsilon \geq 0$, we use the following notation:

$$\begin{aligned} &r_\varepsilon(X) := \max\{i \in 1,\ldots,n : \lambda_i(X) \geq \lambda_{\max}(X) - \varepsilon\}\,,\\ &Q_\varepsilon(X) \text{ is an } n \times r_\varepsilon(X) \text{ matrix whose columns form an}\\ &\text{orthonormal basis of the } r_\varepsilon(X)\text{-dimensional space}\\ &\text{spanned by the corresponding eigenvectors.} \end{aligned} \tag{3.21}$$

Then, by analogy with (3.3) and (3.5), we consider the following enlarged subdifferentials $\delta_\varepsilon \lambda_{\max}(X)$ and $\delta_\varepsilon f(x)$:

$$\begin{aligned} &\delta_\varepsilon \lambda_{\max}(X) := \{Q_\varepsilon(X) Z Q_\varepsilon(X)^T : Z \in \mathcal{C}_{r_\varepsilon(X)}\}\,,\\ &\delta_\varepsilon f(x) = \mathcal{F}^* \delta_\varepsilon \lambda_{\max}(F(x))\,. \end{aligned} \tag{3.22}$$

Since the columns of $Q_\varepsilon(X)$ span in particular the first eigenspace of X, it is clear that $\delta_\varepsilon f(x)$ contains the exact subdifferential. To see that this enlargement is included in the ε-subdifferential $\partial_\varepsilon f(x)$, it suffices to note that any $V = Q_\varepsilon(X) Z Q_\varepsilon(X)^T \in \delta_\varepsilon \lambda_{\max}(X)$, which obviously lies in $\mathcal{C}_n$, also lies in the half-space $\{V \in \mathcal{S}_n : V \bullet X \geq \lambda_{\max}(X) - \varepsilon\}$: In fact,

$$V \bullet X = Z \bullet Q_\varepsilon(X)^T X Q_\varepsilon(X) = \sum_{i=1}^{r_\varepsilon(X)} Z_{ii} \lambda_i(X) \geq \lambda_{\max}(X) - \varepsilon\,,$$

since $\lambda_i(X) \geq \lambda_{\max}(X) - \varepsilon$ for $i \leq r_\varepsilon(X)$, and $\sum_{i=1}^{r_\varepsilon(X)} Z_{ii} = \mathbf{Tr} Z = 1$. Together with (3.6) and (3.7), we do have

$$\partial f(x) \subset \delta_\varepsilon f(x) \subset \partial_\varepsilon f(x)\,. \tag{3.23}$$

The whole purpose of the first-order analysis in [307] then consists in quantifying the distance between $\delta_\varepsilon f(x)$ and $\partial_\varepsilon f(x)$ (knowing that we would like to approximate the latter set). Clearly, $\delta_\varepsilon \lambda_{\max}(X)$ does not contain any full-rank matrix if $r_\varepsilon(X) < n$; this seems to imply that this set is a poor approximation of the (full-dimensional) ε-subdifferential. Nevertheless, it is proved in [307] that the shortest element

$$g_\varepsilon := \mathrm{Argmin}\{\|g\|_*^2 : g \in \delta_\varepsilon f(x)\} \tag{3.24}$$

does provide a good descent direction $d_\varepsilon = -g_\varepsilon$. Specifically, f is guaranteed to decrease along d_ε by a quantity proportional to the gap defined at $X = F(x)$ by $\Delta_\varepsilon(X) := \lambda_{r_\varepsilon(X)}(X) - \lambda_{r_\varepsilon(X)+1}(X)$.

From there, a minimization algorithm can be constructed. At each iterate $x = x_k$, one chooses $\varepsilon = \varepsilon_k$, one constructs the inner approximation $\delta_\varepsilon f(x_k)$, one computes its corresponding shortest element to obtain a direction d_k, along which a line search is performed. With a choice of ε_k careful enough to appropriately control the gap $\Delta_{\varepsilon_k}(X_k)$, a globally convergent method is obtained [307].

3.4.2 Rich oracle: A primal approach

The algorithm outlined above implies a delicate choice of ε. In fact, this parameter plays an ambivalent role:

(i) It is the tolerance on the largest eigenvalue, driving the quality of the direction of search; as such, ε should be "large."

(ii) It is also the final accuracy driving the quality of the solution: The algorithm stops when $\delta_\varepsilon f(x_k)$ contains a small vector (see Remark 3.4); as such, ε should be "small."

In the present section, we combine the two subdifferential enlargements of sections 3.3, 3.4.1. This allows us to restrict the ε of the latter section to its role (ii) of a stopping tolerance. We therefore *fix* this ε to a *small* tolerance $\bar{\varepsilon}$. The resulting $\delta_{\bar{\varepsilon}} f$ becomes a slim enlargement of ∂f, but it is enriched with the bundle $S_k(\varepsilon_k)$ of Proposition 3.1. Alternatively, we can consider $\delta_{\bar{\varepsilon}} f$ as enriching the bundle $S_k(\varepsilon_k)$. In addition to this mutual enrichment, we also transform the dual formulation of section 3.4.1 to a primal one as in section 3.3.

Since we are dealing now with $\bar{\varepsilon}$-subgradients, in particular those belonging to $\delta_{\bar{\varepsilon}} f(y_k)$ at iteration k, we need to adapt Proposition 3.1. At iteration k, we have $k-1$ previous $\bar{\varepsilon}$-subgradients, $g_i \in \delta_{\bar{\varepsilon}} f(y_i)$ for $i = 1, \ldots, k-1$, and at the current point $g \in \delta_{\bar{\varepsilon}} f(y_k)$,

which will be considered as a local dual variable (together with the vector $\alpha \in \Delta_k$). We define now the $\bar{\varepsilon}$-linearization errors at iteration y_k:

$$
\begin{array}{l}
\tilde{e}_k^i := f(x_k) - f(y_i) - g_i^T(x_k - y_i) + \bar{\varepsilon} \ \text{ for } \ i = 1, \ldots, k-1\,, \\
\tilde{e}_k^k(g) = f(x_k) - f(y_k) - g^T(x_k - y_k) + \bar{\varepsilon} \ \text{ for } \ g \in \delta_{\bar{\varepsilon}} f(y_k)\,, \\
\tilde{e}_k(g) := [\tilde{e}_k^1, \ldots, \tilde{e}_k^k(g)]^T \ \text{ for } \ g \in \delta_{\bar{\varepsilon}} f(y_k)\,.
\end{array} \tag{3.25}
$$

We see here that the kth linearization error plays a particular role since it becomes a function of the dual variable g.

Proposition 3.2. *For all $y \in \mathbf{R}^m$, $\alpha \in \Delta_k$, and $g_k \in \delta_{\bar{\varepsilon}} f(y_k)$, we have*

$$
f(y) \geq f(x_k) + \left(\textstyle\sum_{i=1}^k \alpha_i g_i\right)^T (y - x_k) - \alpha^T \tilde{e}_k(g_k)\,.
$$

In particular, for $\varepsilon \geq \min\{\tilde{e}_1, \ldots, \tilde{e}_{k-1}, \tilde{e}_k(g) : g \in \delta_{\bar{\varepsilon}} f(y_k)\}$,

$$
S_{k,\bar{\varepsilon}}(\varepsilon) := \left\{ g = \textstyle\sum_{i=1}^k \alpha_i g_i : \alpha \in \Delta_k,\ \alpha^T \tilde{e}_k(g_k) \leq \varepsilon \right\} \subset \partial_\varepsilon f(x_k)\,. \tag{3.26}
$$

Proof. Follow line by line the proof of [194, Proposition XIV.2.1.1], starting with the $\bar{\varepsilon}$-subgradient inequalities

$$
f(y) \geq f(y_i) + g_i^T(y - y_i) - \bar{\varepsilon} = f(x_k) + g_i^T(y - x_k) - \tilde{e}_k^i \ \text{ for } i = 1, \ldots, k-1\,,
$$

$$
f(y) \geq f(y_k) + g^T(y - y_k) - \bar{\varepsilon} = f(x_k) + g^T(y - x_k) - \tilde{e}_k^k(g) \ \text{ for all } g \in \delta_{\bar{\varepsilon}} f(y_k)
$$

instead of the subgradient inequalities. □

Our (inner) approximation of $\partial_\varepsilon f(x_k)$ is now $S_{k,\bar{\varepsilon}}(\varepsilon)$ of (3.26). Then the dual program (3.17) is no longer quadratic and becomes

$$
\left\{
\begin{array}{ll}
\text{minimize } \tfrac{1}{2}\left\|\textstyle\sum_{i=1}^k \alpha_i g_i\right\|_{*,k}^2 & \text{subject to} \\
\alpha \in \Delta_k\,, & \\
\alpha^T \tilde{e}_k(g_k) \leq \varepsilon & (*)\,, \\
g_k = \mathcal{F}^*(Q_k Z Q_k^T),\ Z \in \mathcal{C}_{r_k}\,, &
\end{array}
\right. \tag{3.27}
$$

where Q_k and r_k stand, respectively, for $Q_{\bar{\varepsilon}}(F(y_k))$ and $r_{\bar{\varepsilon}}(F(y_k))$ of (3.21).

In section 3.3 we presented a primal formulation and interpreted it in the dual space. Here we do a reverse presentation by introducing the Lagrange multiplier μ_k associated with the constraint $(*)$ above: (3.27) can be written in a penalized form

$$
\left\{
\begin{array}{l}
\text{minimize } \tfrac{1}{2}\left\|\textstyle\sum_{i=1}^k \alpha_i g_i\right\|_{*,k}^2 + \mu_k \alpha^T \tilde{e}_k(g) \\
\quad \text{subject to} \\
\alpha \in \Delta_k \ \text{ and } \ g_k = \mathcal{F}^*(Q_k Z Q_k^T),\ Z \in \mathcal{C}_{r_k}\,.
\end{array}
\right. \tag{3.28}
$$

From now on, we assume $\mu_k > 0$. Going further into our primalization process, we construct the dual of (3.28) and obtain the following result.

Proposition 3.3. *Up to the multiplicative factor $\mu_k > 0$, the dual of* (3.28) *is*

$$
\left\{
\begin{array}{l}
\textit{minimize } t + \tfrac{1}{2}\mu_k \|y - x_k\|_k^2 \quad (y,t) \in \mathbf{R}^m \times \mathbf{R} \\
\quad \textit{subject to} \\
f(x_k) - \tilde{e}_i + g_i^T(y - x_k) \leq t \ \text{ for } \ i = 1, \ldots, k-1\,, \\
(f(y_k) - \bar{\varepsilon}) I_{r_k} + Q_k^T \mathcal{F}(y - y_k) Q_k \preceq t I_{r_k}\,.
\end{array}
\right. \tag{3.29}
$$

If (t_{k+1}, y_{k+1}) *and* $(\bar{\alpha}, \bar{Z})$ *solve, respectively,* (3.29) *and* (3.28), *we have the following primal-dual relations where* $g_k := \mathcal{F}^*(Q_k \bar{Z} Q_k^T)$:

$$y_{k+1} = x_k - \frac{1}{\mu_k} W_k \sum_{i=1}^{k} \bar{\alpha}_i g_i \quad \text{and} \quad t_{k+1} = \tilde{\varphi}_k(y_{k+1}), \tag{3.30}$$

where

$$\begin{array}{rcl} \tilde{\varphi}_k(y) & := & \max\{\tilde{l}_1(y), \ldots, \tilde{l}_k(y)\}, \\ \tilde{l}_i(y) & := & f(y_i) + g_i^T(y - y_i) - \bar{\varepsilon} = f(x_k) + g_i^T(y - x_k) - \tilde{e}_i \\ & & \quad \text{for } i = 1, \ldots, k-1, \text{ and} \\ \tilde{l}_k(y) & := & f(y_k) - \bar{\varepsilon} + \bar{Z} \bullet (Q_k^T \mathcal{F}(y - y_k) Q_k). \end{array} \tag{3.31}$$

Proof. Problems (3.28) and (3.29) are two convex optimization problems: a convex quadratic objective is minimized under LMI constraints; we apply duality theory (see [373], for example). Consider (3.29); for fixed $y \in \mathbf{R}^m$ and t large enough, the constraints are strictly satisfied; i.e., the Slater condition clearly holds. Then there is no duality gap and it suffices to prove that the dual of (3.29) is (3.28). For this let us introduce a $(k-1)$-uple $\alpha \in \mathbf{R}^{k-1}$ of nonnegative scalar multipliers (associated with the $k-1$ scalar constaints) and a positive semidefinite matrix multiplier $U \in \mathcal{S}_{r_k}$ (associated with the LMI constraint). The associated Lagrangian is

$$\begin{array}{rcl} L(t, y, \alpha, U) & = & t + \frac{1}{2}\mu_k \|y - x_k\|_k^2 \\ & & \quad + \sum_{i=1}^{k-1} \alpha_i [f(x_k) - \tilde{e}_i + g_i^T(y - x_k) - t] \\ & & \quad + U \bullet [(f(y_k) - t - \bar{\varepsilon}) I_{r_k} + Q_k^T \mathcal{F}(y - y_k) Q_k]. \end{array}$$

The dual of (3.29) is then

$$\max_{\substack{\alpha \in \mathbf{R}_+^{k-1}, \\ U \in \mathcal{S}_{r_k}, U \succeq 0}} \quad \min_{(y,t) \in \mathbf{R}^m \times \mathbf{R}} L(t, y, \alpha, U). \tag{3.32}$$

For the minimization in (3.32) to have a finite value we need the multipliers to satisfy the following hidden constraint:

$$\sum_{i=1}^{k-1} \alpha_i + \mathbf{Tr} U = 1.$$

On the other hand, the Lagrangian is strongly convex in y and the optimality condition for the minimization part gives us

$$\mu_k(y - x_k) = W_k \left(\sum_{i=1}^{k-1} \alpha_i g_i + \mathcal{F}^*(Q_k U Q_k^T) \right).$$

Plugging this relation into the Lagrangian, setting $Z = \frac{U}{\mathbf{Tr} U}$ (or $Z = 0$ if $U = 0$) and $\alpha_k = \mathbf{Tr} U$, we obtain (3.28) after multiplication by $\mu_k > 0$. □

3.4.3 A large class of bundle methods

Starting from the above development, several options can be made to obtain implementable bundle methods, which will combine the global convergence properties of a bundling mechanism with the fast termination allowed by a rich oracle.

Models. Consider the function $\bar{\varphi}_k$ defined in (3.31). From (3.23) and (3.6), we have for all $g_i \in \delta_{\bar{\varepsilon}} f(y_i), i = 1, \ldots, k-1$,

$$\exists V_i \in \mathcal{C}_n : V_i \bullet F(y_i) \geq f(y_i) - \bar{\varepsilon} \text{ and } g_i = \mathcal{F}^* V_i .$$

For these V_i's, we therefore have

$$\tilde{l}_i(y) = f(y_i) - \bar{\varepsilon} + g_i^T (y - y_i) \leq V_i \bullet F(y_i) + V_i \bullet \mathcal{F}(y - y_i) = V_i \bullet F(y) .$$

On the other hand,

$$\begin{aligned} \max_{Z \in \mathcal{C}_{r_k}} (Q_k Z Q_k^T) \bullet F(y) &= \max_{Z \in \mathcal{C}_{r_k}} \{(Q_k Z Q_k^T) \bullet F(y_k) \\ &\qquad + (Q_k Z Q_k^T) \bullet \mathcal{F}(y - y_k)\} \\ &\geq f(y_k) - \bar{\varepsilon} + \max_{Z \in \mathcal{C}_{r_k}} (Q_k Z Q_k^T) \bullet \mathcal{F}(y - y_k) \\ &\qquad [\text{since } Q_k Z Q_k^T \in \partial_{\bar{\varepsilon}} \lambda_{\max}(F(y_k))] \\ &= \tilde{l}_k(y) . \end{aligned}$$

Then introduce the set

$$\tilde{\mathcal{C}}_k = \mathbf{Co}\{V_1, \ldots, V_{k-1}, \{Q_k Z Q_k^T : Z \in \mathcal{C}_{r_k}\}\} .$$

Recall (3.2); the following inequalities hold:

$$\tilde{\varphi}_k(y) \leq \max_{V \in \tilde{\mathcal{C}}_k} V \bullet F(y) \leq f(y) \text{ for all } y \in \mathbf{R}^m . \tag{3.33}$$

We see that, if based on the intermediate model above, a bundle method can be seen as a process to update approximations $\tilde{\mathcal{C}}_k$ of the big convex set $\mathcal{C}_n$. Furthermore, observe that the first inequality becomes an equality when $\bar{\varepsilon} = 0$. Then such an approach resembles that of [189], which consists in approximating $\mathcal{C}_n$ by

$$\hat{\mathcal{C}}_k = \{\alpha \hat{V} + P_k Z P_k^T : \alpha \geq 0, Z \in \mathcal{S}_{\hat{k}},\ Z \geq 0,\ \alpha \hat{V} + \mathbf{Tr} Z = 1\} ;$$

here $\hat{V} \in \mathcal{C}_n$ is an aggregate matrix that contains information from the previous iterates, and the columns of P^k are orthonormal vectors storing past and current information.

Implementations. Consider now the effective resolution of the quadratic-sdp problem (3.29) or (3.28). In view of its nonlinear constraints, it cannot be solved exactly; and since it must be solved at each iteration of the outer minimization algorithm, a fast method is in order.

In [189] (where $\bar{\varepsilon} = 0$), this subproblem is solved using interior-point methods; the bundle scheme is thereby assigned the role of a master program controlling the size of small quadratic-sdp problems.

Here, we propose the approximate resolution of (3.28) in two steps:

(qp_1) compute an approximate solution g_k of

$$\min\{\|g\|_{k,*}^2 : g \in \delta_{\bar{\varepsilon}} f(y_k)\} ; \tag{3.34}$$

(qp_2) then solve the quadratic program

$$\min_{\alpha \in \Delta_k} \frac{1}{2} \left\| \sum_{i=1}^k \alpha_i g_i \right\|_{*,k}^2 + \mu_k \alpha^T \tilde{e}_k , \tag{3.35}$$

where

$$\begin{aligned} &\tilde{e}_k^i := f(x_k) - f(y_i) - g_i^T (x_k - y_i) + \bar{\varepsilon} \text{ for } i = 1, \ldots, k , \\ &\text{and } \tilde{e}_k := [\tilde{e}_k^1, \ldots, \tilde{e}_k^k]^T . \end{aligned} \tag{3.36}$$

Four main reasons lead us to this choice:

(i) The support function of the set $\delta_{\bar\varepsilon} f(y_k)$ is known explicitly; therefore, an efficient method, called the support black-box function in [307, section 3.1], can be used to obtain an approximation of (3.34).

(ii) Step (qp_1) can be viewed as a refinement of the oracle giving $f(y_k)$ and g_k: Now g_k coming from (qp_1) is a carefully selected element of $\delta_{\bar\varepsilon} f(y_k)$, possibly more efficient than any exact subgradient at y_k.

(iii) By decomposing (3.28) in two steps, local and global work are cleanly separated: While (qp_2) guarantees global convergence, the solution g_k of (3.34) is exactly the local information needed to form the so-called $\mathcal{U}$-Hessian of f and get asymptotic quadratic convergence (see section 3.5 below).

(iv) Step (qp_2) is similar to (3.17), with approximate instead of exact subgradients: Then the two steps can be naturally inserted in the general scheme of (3.3) (see Algorithm 3.2 below); the resulting method can be seen as a *polyhedral-semidefinite proximal bundle method*.

In view of this last point, the notation of (3.10) can still be used (even though the g_i's have a different origin), keeping in mind that it is now $\check f - \bar\varepsilon$ that underapproximates f. Indeed, we can see from (3.31), (3.33) that the models are ordered as follows: For all $y \in \mathbf{R}^m$,

$$\check f_k(y) - \bar\varepsilon \le \bar\varphi_k(y) \le \max_{V\in\bar{\mathcal{C}}_k} V \bullet F(y) \le f(y)\,. \tag{3.37}$$

In a way, the present approach consists in replacing the model $\bar\varphi_k$ of (3.31) by $\check f_k - \bar\varepsilon$. Thus, the analysis of (3.17) can be reproduced to obtain the analogue of Lemma 3.1.

Lemma 3.2. *For $\mu_k > 0$ and $M_k > 0$, the unique solution of*

$$\min_{y\in\mathbf{R}^m} \check f_k(y) + \frac{1}{2}\mu_k \|y - x_k\|_k^2 \tag{3.38}$$

is $y_{k+1} := x_k - \frac{1}{\mu_k} W_k \sum_{i=1}^k \bar\alpha_i g_i$, where $\bar\alpha \in \mathbf{R}^k$ solves (3.35). Furthermore, setting $\delta_k := f(x_k) - \check f_k(y_{k+1}) + \bar\varepsilon - \frac{1}{2}\mu_k\|y_{k+1} - x_k\|_k^2$ and $\bar g_k := \sum_{i=1}^k \bar\alpha_i g_i$, we have

$$0 \le \varepsilon_k := \bar\alpha^T \bar e_k = \delta_k - \frac{1}{2\mu_k}\|\bar g_k\|_{*,k}^2\,. \tag{3.39}$$

Algorithm 3.2 (polyhedral-sdp proximal bundle method). *The initial point x_1 is given, together with a stopping tolerance $\delta \ge 0$. Choose a spring strength $\mu_1 > 0$, the tolerance $\bar\varepsilon \ge 0$, a positive definite matrix M, and a descent coefficient $\omega \in]0,1[$. Initialize the iteration counter $k = 1$ and $y_1 = x_1$; compute $f(y_1)$ and $g_1 \in \partial f(y_1)$.*

- *STEP* 1. *Compute a solution $\bar\alpha$ of* (qp_2) *and set*

$$\begin{aligned} \bar g_k &:= \textstyle\sum_{i=1}^k \bar\alpha_i g_i\,, \\ y_{k+1} &:= x_k - \tfrac{1}{\mu_k} W_k \bar g_k\,, \\ \delta_k &:= f(x_k) - \check f_k(y_{k+1}) + \bar\varepsilon - \tfrac{1}{2}\mu_k\|y_{k+1} - x_k\|_k^2\,. \end{aligned} \tag{3.40}$$

- *STEP* 2. *If $\delta_k \le \delta$ stop.*

- *STEP* 3. *Call the oracle. Compute an approximate solution* $g_{k+1} \in \delta_{\bar{\varepsilon}} f(y_{k+1})$ *of* (3.34) *and* $\tilde{e}_k$ *according to* (3.36). *If*

$$f(y_{k+1}) \leq f(x_k) - \omega\delta_k\,, \tag{3.41}$$

set $x_{k+1} = y_{k+1}$, *and choose* $\mu_{k+1} > 0$ *and* $M_{k+1} \succ 0$. *Otherwise set* $x_{k+1} = x_k$ *(null step).*
Replace k *with* $k+1$ *and loop to Step* 1.

When $\bar{\varepsilon} = 0$, Algorithm 3.2 is exactly Algorithm 3.1*. Numerical experiments reported in section 3.6 will show that the introduction of $\bar{\varepsilon}$ together with (qp_1) significantly improves the speed of convergence.

As for theoretical properties, the global convergence of Algorithm 3.2 is analyzed just as in [194, section XV.3.2]. Via an appropriate control of the sequence $\{\mu_k \lambda_{\max}(M_k)\}$, and assuming $\delta > 0$, the algorithm stops at some iteration k; then the last iterate satisfies an approximate minimality condition, alluded to in Remark 3.4.

3.5 Second-order schemes

The rich structure enjoyed by $\lambda_{\max}$ allows an appropriate second-order analysis, even though $\nabla\lambda_{\max}$ does not exist. This can be performed with tools from differential geometry: [24, 310, 311, 374] or from convex analysis via the recent $\mathcal{U}$-Lagrangian theory of [245].

3.5.1 Local methods

With an appropriate second-order approximation of $\lambda_{\max}$ on hand, Newton-type methods can be developed for (3.1) or (3.8). This is done by M. L. Overton in [308], inspired by the work of R. Fletcher [137]. The approach is further developed in [309, 311, 310, 374, 130], and revisited in the light of $\mathcal{U}$-Lagrangians in [305].

These methods are local, in the sense that convergence can be guaranteed only in a neighborhood of a solution x^*. Furthermore the multiplicity r^* of $\lambda_{\max}(F(x^*))$ must be known. It is then natural to combine the globally convergent bundle methods of section 3.4 with the present fast local methods by inserting in the former the appropriate second-order information used by the latter. For methods of section 3.4.1, this is done in [307]. Without any assumption on (3.8), the resulting algorithm generates a minimizing sequence. Its quadratic convergence needs some *strict complementarity* and *nondegeneracy* assumptions.

Here, we proceed to demonstrate how the same metric can be inserted in a primal formulation like section 3.4.3. For this, we first need to explain what such a metric is; we do this via the $\mathcal{U}$-Lagrangian theory.

3.5.2 Decomposition of the space

Notationally, a hat on a matrix or vector (say, $\hat{x}$) will suggest some intermediate variable, rather than the current point of an algorithm. We denote by $\mathcal{M}_r$ the set of matrices whose maximum eigenvalue has a fixed multiplicity $r \in \{1, \ldots, n\}$. At a point $\hat{X} \in \mathcal{M}_r$, we decompose the space $\mathcal{S}_n$ as the direct sum of $\mathcal{U}_{\lambda_{\max}}(\hat{X})$ and $\mathcal{V}_{\lambda_{\max}}(\hat{X})$, the tangent and normal spaces to $\mathcal{M}_r$ at $\hat{X}$. These two subspaces have an explicit description:

$$\begin{aligned} \mathcal{V}_{\lambda_{\max}}(\hat{x}) &= \{\hat{Q} Z \hat{Q}^T : Z \in \mathcal{S}_r,\ \mathbf{Tr}\, Z = 0\}\,, \\ \mathcal{U}_{\lambda_{\max}}(\hat{x}) &= \{U \in \mathcal{S}_n : \hat{Q}^T U \hat{Q} - \tfrac{\mathbf{Tr}\hat{Q}^T U \hat{Q}}{r} I_r = 0\}\,, \end{aligned} \tag{3.42}$$

where r and $\hat{Q}$ are the $r_0(\hat{X})$ and $Q_0(\hat{X})$ of (3.21).

To this decomposition of $\mathcal{S}_n$, there corresponds the following decomposition of the space of variables $\mathbf{R}^m$: At a point $\hat{x} \in \mathbf{R}^m$ such that $F(\hat{x}) = \hat{X} \in \mathcal{M}_r$, i.e., $\hat{x} \in A^{-1}\mathcal{M}_r$, we write

$$\mathbf{R}^m = \mathcal{U}_f(\hat{x}) \oplus \mathcal{V}_f(\hat{x}),$$

where $\mathcal{U}_f(\hat{x}) := \mathcal{F}^{-1}\mathcal{U}_{\lambda_{\max}}(\hat{X})$ and $\mathcal{V}_f(\hat{x}) := \mathcal{U}_f(\hat{x})^\perp$.

It is shown in [305] that $\mathcal{U}_f(\hat{x})$ is the largest subspace where the directional derivative $f'(\hat{x}, \cdot)$ of f at $\hat{x}$ is linear, and $\mathcal{V}_f(\hat{x})$ is the linear subspace parallel to the affine hull of $\partial f(\hat{x})$.

3.5.3 Second-order developments

Take $\hat{x} \in A^{-1}\mathcal{M}_r$ and $\hat{G} \in \mathcal{C}_n$ such that $\mathcal{F}^*\hat{G} \in \partial f(\hat{x})$. At the primal-dual pair $(\hat{X}, \hat{G})$ we define the following symmetric operator $\mathcal{H}(\hat{X}, \hat{G})$:

$$\mathcal{S}_n \ni D \mapsto \mathcal{H}(\hat{X}, \hat{G}) \cdot D = \hat{G}DR(\hat{X})^\dagger + R(\hat{X})^\dagger D\hat{G}, \tag{3.43}$$

where $R(\hat{X}) = \lambda_{\max}(\hat{X})I_n - \hat{X}$.

The operator induced by $\mathcal{H}(\hat{X}, \hat{G})$ in the subspace $\mathcal{U}_{\lambda_{\max}}(\hat{X})$ is called the $\mathcal{U}$-Hessian of $\lambda_{\max}$ at $(\hat{X}, \hat{G})$. It collects the relevant second-order information on $\lambda_{\max}$ along $\mathcal{M}_r$ as follows.

Theorem 3.1. *For all $D \in \mathcal{S}_n$ such that $\hat{X} + D \in \mathcal{M}_r$, we have*

$$\lambda_{\max}(\hat{X} + D) = \lambda_{\max}(\hat{X}) + D \bullet \hat{G} + \frac{1}{2} D \bullet \mathcal{H}(\hat{X}, \hat{G}) \cdot D + o(\|D\|^2). \tag{3.44}$$

In the space of decision variables, (3.44) *is written as the following: For all $d \in \mathbf{R}^m$ such that $F(\hat{x}) + \mathcal{F}d \in \mathcal{M}_r$,*

$$f(\hat{x} + d) = f(\hat{x}) + \hat{g}^T d + \frac{1}{2} d^T [\mathcal{F}^* \circ \mathcal{H}(\hat{X}, \hat{G}) \circ \mathcal{F}] d + o(\|d\|^2). \tag{3.45}$$

Proof. The first development (3.44) is given in [305, Theorem 4.12, Corollary 4.13]. The second development is obtained by taking $D = \mathcal{F}d$ and by noticing that $o(\|\mathcal{F}d\|^2) \leq \|\mathcal{F}\|_\infty o(\|d\|^2)$ is also an $o(\|d\|^2)$. □

3.5.4 Quadratic step

Now we come back to our algorithmical motivations. So far, the viscosity parameter $\bar{\varepsilon}$ had the role (ii) of a stopping criterion (see the beginning of section 3.4.2). At this stage we assign it an additional role:

(iii) $\bar{\varepsilon}$ enables us to "stabilize" the multiplicity of $\lambda_{\max}$.

At iteration k, assume that $y_k = x_k$ (i.e., the previous step was a descent step). As in section 3.4.1, we denote by r_k and Q_k the approximate multiplicity $r_{\bar{\varepsilon}}(F(x_k))$ and the associated matrix $Q_{\bar{\varepsilon}}(F(x_k))$ of (3.21). The rationale for our quadratic step is to minimize the quadratic approximation of (3.45) (or an "$\bar{\varepsilon}$-regularization" of it) on the linearized constraint $F(x_k + d) \in \mathcal{M}_{r_k}$ (knowing that $\mathcal{M}_{r_k}$ is supposed to contain the optimal solution). For this we introduce a dual matrix $\hat{G}_k \in \mathcal{C}_n$ such that $\mathcal{F}^*\hat{G}_k = g_k$,

where $g_k \in \delta_{\bar{\varepsilon}}\lambda_{\max}(F(x_k))$ is the output of (qp_1) in section 3.4.3. We also introduce the symmetric operator

$$
\begin{aligned}
&H_k := \mathcal{F}^* \circ \mathcal{H}(\hat{G}_k, \hat{X}_k) \circ \mathcal{F}\,,\\
&\hat{X}_k := \hat{\lambda}_k Q_k Q_k^T + F_k - Q_k \mathbf{diag}(\lambda_1(F_k), \ldots, \lambda_{r_k}(F_k)) Q_k^T\,,\\
&\hat{\lambda}_k := \tfrac{1}{r_k} \textstyle\sum_{i=1}^{r_k} \lambda_i(F_k)\,,
\end{aligned}
\tag{3.46}
$$

where $F_k := F(x_k)$.

In [305, section 5.4], the constraint requiring $F(x_k+d) \in \mathcal{M}_{r_k}$ is shown to be linearized as $F(x_k+d) \in \mathcal{U}_{\lambda_{\max}}(\hat{X}_k)$. This is a linear equation, which can be written, using (3.42),

$$
\begin{aligned}
&[D\phi]_k\, d + L_k = 0\,, \text{where}\\
&\mathbf{R}^m \ni d \mapsto [D\phi]_k\, d = \hat{Q}^T(\mathcal{F}\, d)\hat{Q} - \tfrac{\mathbf{Tr}(\hat{Q}^T(\mathcal{F}\, d)\hat{Q})}{r} I_r\,,\\
&L_k = \mathbf{diag}(\lambda_1(F_k), \ldots, \lambda_{r_k}(F_k)) - \hat{\lambda}_k I_{r_k}\,.
\end{aligned}
$$

We therefore obtain the following quadratic step:

$$
\begin{cases}
\text{minimize } g_k^T d + \frac{1}{2} d^T H_k d\,,\\
[D\phi]_k\, d + L_k = 0\,.
\end{cases}
\tag{3.47}
$$

We define $\mathcal{U}_k$ and $\mathcal{V}_k$, the $\bar{\varepsilon}$-regularizations of $\mathcal{U}_f(x_k)$ and $\mathcal{V}_f(x_k)$:

$$
\mathcal{U}_k := \mathbf{Kernel}\ [D\phi]_k \quad \text{and} \quad \mathcal{V}_k := \mathcal{U}_k^{\perp} = \mathbf{Range}\ [D\phi]_k^*\,.
\tag{3.48}
$$

We assume that $\bar{\varepsilon}$-transversality holds at x_k: $[D\phi]_k$ is onto (or, equivalently, $[D\phi]_k^*$ is injective). If in addition the Hessian matrix H_k is positive definite on $\mathcal{U}_k$, then (3.47) has a unique solution d_k. Under these assumptions (3.47) can be solved in two steps.

Normal step. Since $[D\phi]_k$ is onto, it has a right inverse $[D\phi]_k^- = [D\phi]_k^* \circ ([D\phi]_k \circ [D\phi]_k^*)^{-1}$. Define

$$
v_k := [D\phi]_k^-\, L_k\,.
\tag{3.49}
$$

Then we have $v \in \mathcal{V}_k$ and $[D\phi]_k\, v_k + L_k = 0$.

Tangent step. The set of directions d satisfying the linear equation of (3.47) is $v_k + \mathcal{U}_k$. Then, setting $u = d - v_k$, (3.47) is the simple problem

$$
\text{minimize}_{u \in \mathcal{U}_k}\ (g_k + H_k\, v_k)^T u + \frac{1}{2} u^T H_k u\,,
\tag{3.50}
$$

whose unique solution is

$$
u_k := -P_{\mathcal{U}_k} \circ H^{\dagger} \circ P_{\mathcal{U}_k}\,(g_k + H_k\, v_k)\,.
\tag{3.51}
$$

Here $P_{\mathcal{U}_k}$ is the orthogonal projection onto $\mathcal{U}_k$: $P_{\mathcal{U}_k}$ is symmetric, $P_{\mathcal{U}_k}^2 = I_m$ and $\mathbf{Range}\, P_{\mathcal{U}_k} = \mathcal{U}_k$.

The unique solution of (3.47) is then $d_k = u_k + v_k$.

3.5.5 The dual metric

The previous section concerned a quadratic model of f yielding fast asymptotic convergence of an algorithm. Now we suggest a possibility to choose the metric in (qp_2). We look for a matrix W_k of the form $W_k = W_{\mathcal{U}_k} + W_{\mathcal{V}_k}$ such that

$$
\begin{aligned}
&W_{\mathcal{U}_k} \text{ and } W_{\mathcal{V}_k} \text{ are symmetric},\\
&W_{\mathcal{U}_k} + W_{\mathcal{V}_k} \text{ is positive definite},\\
&\mathbf{Range}\, W_{\mathcal{U}_k} = \mathcal{U}_k \text{ and } \mathbf{Range}\, W_{\mathcal{V}_k} = \mathcal{V}_k\,.
\end{aligned}
$$

From the previous paragraph, it is rather natural to take

$$W_{\mathcal{U}_k} := P_{\mathcal{U}_k} H_k^{\dagger} P_{\mathcal{U}_k} . \tag{3.52}$$

Now we want to choose $W_{\mathcal{V}_k}$ so that the dual problem (3.34) is well conditioned. Notice that the enlargement $\delta_\varepsilon f(x_k)$ can be described by

$$\delta_\varepsilon f(x_k) = \{[D\phi]_k^* Z + g_c : Z \in \mathcal{S}_{r_k}, Z \succeq 0\},$$

where $g_c := \mathcal{F}^* \frac{Q_k Q_k^T}{r_k}$. Then the metric induced by $\|\cdot\|_{k,*}$ in $\mathcal{S}_{r_k}$ is $[D\phi]_k \circ W_{\mathcal{V}_k} \circ [D\phi]_k^*$. Thus a natural choice for $W_{\mathcal{V}_k}$ is

$$W_{\mathcal{V}_k} := [D\phi]_k^* ([D\phi]_k \circ [D\phi]_k^*)^{-2} [D\phi]_k , \tag{3.53}$$

since in this case the induced metric in $\mathcal{S}_{r_k}$ is the standard Frobenius norm.

3.5.6 A second-order bundle method

Now we can state an algorithm piecing together the objects defined in the preceding sections. When Algorithm 3.2 has just performed a descent step, the model seems good and a Newton step can be tried. On the other hand, a null step seems to indicate that the bundle is not rich enough (at least locally), and a more conservative strategy is advisable to enrich the bundle.

Algorithm 3.3 (second-order proximal bundle method). *The initial point x_1 is given, together with a stopping tolerance $\delta \geq 0$. Choose a spring strength $\mu_1 > 0$, the tolerance $\bar{\varepsilon} \geq 0$, a positive definite matrix M_1, and a descent coefficient $m \in]0,1[$. Initialize the iteration counter $k = 1$ and $y_1 = x_1$; compute $f(y_1)$ and $g_1 \in \partial f(y_1)$.*

- *STEP* 1. *If $y_k = x_k$ goto Step* 1.1 *else goto Step* 1.2.

  STEP 1.1. *Compute $d_k := v_k + u_k$, where v_k and u_k, respectively, are given by* (3.49) *and* (3.51). *Set*

  $$\begin{aligned} y_{k+1} &:= x_k + \tfrac{1}{\mu_k} d_k , \\ \delta_k &:= g_k^T (y_{k+1} - x_k) + \tfrac{1}{2}\mu_k \|y_{k+1} - x_k\|_k^2 . \end{aligned}$$

  STEP 1.2. *Compute $\bar{\alpha}$ solving* (qp_2) *and set*

  $$\begin{aligned} \bar{g}_k &:= \textstyle\sum_{i=1}^k \bar{\alpha}_i g_i , \\ y_{k+1} &:= x_k - \tfrac{1}{\mu_k} W_k \bar{g}_k , \\ \delta_k &:= f(x_k) - \check{f}_k(y_{k+1}) + \bar{\varepsilon} - \tfrac{1}{2}\mu_k \|y_{k+1} - x_k\|_k^2 . \end{aligned}$$

- *STEP* 2. *If $\delta_k \leq \delta$ stop.*

- *STEP* 3. *Call the oracle. Compute an approximate solution $g_{k+1} \in \delta_{\bar{\varepsilon}} f(y_{k+1})$ of* (3.34) *and $\tilde{e}_k$ according to* (3.36). *If*

  $$f(y_{k+1}) \leq f(x_k) - \omega\delta_k ,$$

  set $x_{k+1} = y_{k+1}$, choose $\mu_{k+1} > 0$, and set W_{k+1} from (3.52), (3.53). *Otherwise set $x_{k+1} = x_k$ (null step).*
  Replace k with $k+1$ and loop to Step 1.

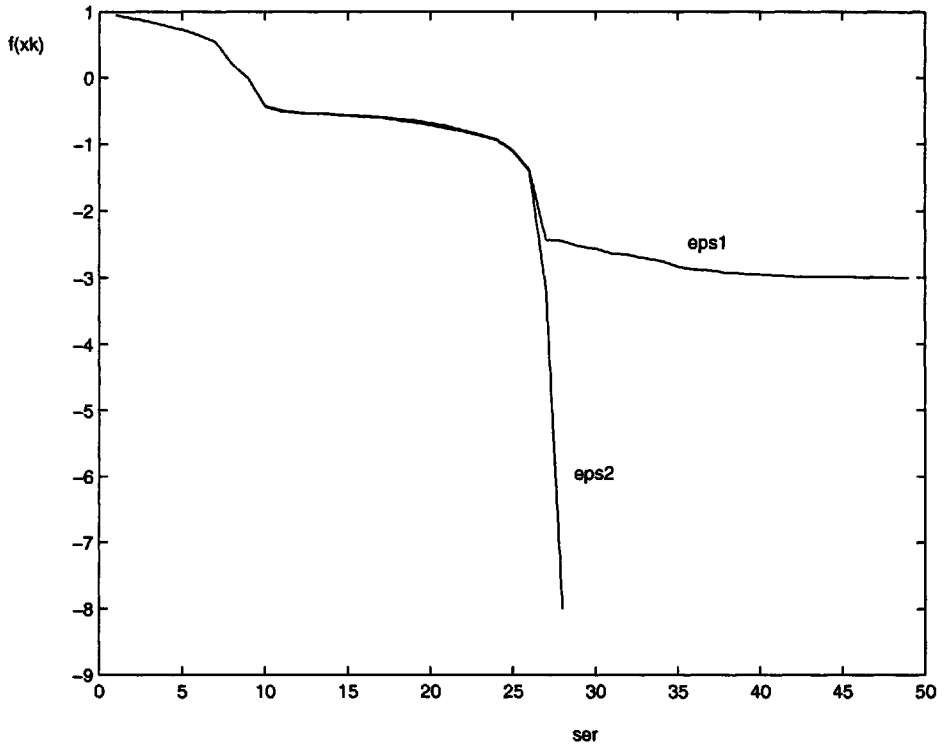


Figure 3.1: Influence of $\bar{\varepsilon}$.

As far as the bundling mechanism is concerned, this algorithm is formally the same as Algorithm 3.2. Its global convergence is therefore established likewise, via appropriate safeguards on the various metrics used (see the end of section 3.5.1). Its local convergence is also easy to analyze: When $\mu_k = 1$ is accepted for the descent test, the algorithm becomes the second-order bundle method of [307] and therefore enjoys the same asymptotic properties.

3.6 Numerical results

3.6.1 Influence of ε

This first example is aimed at illustrating an observation made by Schramm and Zowe and reported by Overton in [309]. We consider the Lovasz problem for a class of undirected graphs given in [366]: given integers $\alpha \geq 1$ and $\omega \geq 3$, set $n = \alpha\omega + 1$ and define the graph $C_n^{\omega-1}$ to have the property that vertices i and j (in $\{1,\ldots,n\}$) are adjacent if $j - i < \omega$ or $i + n - j < \omega$. This property defines the structure of the variables M of Lovasz problem:

$$\begin{array}{ll} & \text{minimize } \lambda_{\max}(ee^T + M) \\ (3.54) & \text{subject to } M \text{ is an adjacency matrix of } C_n^{\omega-1}, \end{array}$$

where e is the n-vector on all ones.

Figure 3.1 shows the influence of $\bar{\varepsilon}$ on a small instance of (3.54), where $\alpha = 3$ and $\omega = 4$: this gives $m = 39$ variables in the 13×13 symmetric matrix M. We observe here not only the improvement of the algorithm for $\bar{\varepsilon} > 0$ but also a superlinear behavior

Table 3.1: Lovasz problem.

α	ω	n	m	f^*	$\|\bar{g}\|$	# iter	CPU time
10	6	61	1891	10.120845	$6.1\,10^{-7}$	129	20 s
20	10	201	1809	20.106	$4.2\,10^{-3}$	481	16 min
20	50	1001	49049	99.2	$1.1\,10^{-1}$	96	5 h

without the use of second-order information. This might be explained by the peculiar structure of these problems: The multiplicity at the solution is very high ($r^* = 7$ to precision 10^{-8}) and $\mathcal{V}^*$, the affine hull of $\partial f(x^*)$, is therefore very large. Moreover, first-order information is enough to get a second-order approximation of f in $\mathcal{V}^*$ [245].

3.6.2 Large-scale problems

Lovasz problem. In order to be able to solve (3.54) when n is large, we exploit in the oracle the sparse structure of the adjacency matrices M. In Table 3.1, we report for several values of α and ω, the final value of the function, the norm of the corresponding aggregate subgradient, the number of total iterations (descent or null steps), and the total CPU time used by Algorithm 3.2 on a Sun Sparc Ultra 1 with 196 MB RAM.

Quadratic stability of polytopic linear differential inclusions (PLDIs). In this paragraph, along the lines of [64, Chapter 5], we analyze the stability of a PLDI

$$\dot{x} = A(t)\,x, \quad A(t) \in \mathbf{Co}\{A_1, \ldots, A_L\}, \tag{3.55}$$

where the A_i's are given stable $n \times n$ real matrices. Then we define the structured mapping:

$$\begin{array}{rcl} \mathcal{S}_n & \to & \oplus_{s=1}^{L} \mathcal{S}_n\,, \\ P & \mapsto & \mathcal{F}\cdot P := \oplus_{s=1}^{L} PA_i + A_i^T P\,; \end{array}$$

a sufficient condition for (3.55) to be quadratically stable is

$$\exists P > 0: \quad \lambda_{\max}(\mathcal{F}\cdot P) < 0\,. \tag{3.56}$$

Then an equivalent condition for (3.56) to hold is to require that the value of the eigenvalue optimization problem

$$\begin{array}{l} \inf \ \lambda_{\max}(\mathcal{F}\cdot P)\,, \\ P > 0\,, \\ \mathbf{Tr}P = n\,, \end{array} \tag{3.57}$$

is negative. Actually, using a generalization of Lyapunov's theorem [199, Theorem 2.4.10], we can drop the constraint $P > 0$; the equality constraint can also be removed by eliminating one variable. Yet for the implementation of the oracle presented in section 3.4.3, we do keep the matricial structure of the decision variable P instead of decomposing P on a basis of $\mathcal{S}_n$; the minimized functions have the form

$$f(P) := \lambda_{\max}(\mathcal{F}\cdot E + \mathcal{F}\cdot \tilde{P})\,,$$

where E is the symmetric matrix with zeros everywhere except $E_{nn} = n$ and $\tilde{P}$ equals P everywhere except $\tilde{P}_{nn} := -\sum_{i=1}^{n-1} P_{ii}$. The adjoints of $P \mapsto \mathcal{F}\cdot\tilde{P}$ have a similar matricial structure. This formulation eases the resolution of (3.57) when n and $m = \frac{n(n+1)}{2} - 1$ are large numbers. In Table 3.2, we take $L = 2$ and we generate stable random matrices

Table 3.2: PLDI, $n \geq 500$.

n	m	f^*	$\|\bar{g}\|$	# iter	CPU time
500	125249	10^{-2}	10^{-2}	80	7 h
1000	500499	$5.1\,10^{-2}$	$6.3\,10^{-2}$	47	20 h

A_1 and A_2 of various sizes. The most difficult situations are when the value of (3.57) is nonnegative. We report for such situations the final value of the function, the norm of the corresponding aggregate subgradient, the number of total iterations (descent or null steps), and the total CPU time. Algorithm 3.2 is run with the viscosity parameter $\bar{\varepsilon} = 10^{-3}$ and the stopping tolerance $\delta = 10^{-2}$. We note that the observed multiplicity to the precision $10 * \bar{\varepsilon}$ is, in these situations, $r^* = n$.

3.6.3 Superlinear convergence

In this paragraph we focus our attention on a small instance of (3.57): $n = 5$ and the matrices A_1 and A_2 are given by

$$A1 = \begin{bmatrix} -0.30514981684203 & -0.12717167399150 & -0.37080266389640 & -0.57112026933673 \\ -0.22727849390700 & -0.24318911206550 & -0.36426275722518 & -0.02224065695781 \\ -0.36873022242947 & -0.25327247833998 & -0.68348855951185 & -0.74119819525553 \\ -0.42560381552175 & -0.27639287914406 & -0.57459988147874 & -0.55530006009433 \end{bmatrix},$$

$$A2 = \begin{bmatrix} -0.70166259253876 & -0.05779033486378 & -0.10942688756431 & -0.48637313513570 \\ 0.10902133192757 & -0.14730314658672 & 0.01058304534831 & -0.25766505826345 \\ -0.26311036762919 & -0.10429623489082 & -0.04975331091086 & -0.10833871463463 \\ -0.56854215687151 & -0.10666376499017 & -0.30404398482014 & -0.60107828929368 \end{bmatrix}.$$

Although both A_1 and A_2 are stable, their respective spectral abscissa are $\alpha(A_1) = -2.93e-05$ and $\alpha(A_2) = -3.25e-06$; there is no quadratic Lyapunov function proving the stability of (3.55). Indeed, using the second-order proximal bundle method, we solve (3.57) to machine precision: At the final point P^* generated by the algorithm we have $\bar{g} \in \partial f(P^*)$ with $\|\bar{g}\| \leq 2e-15$. The final value is $f^* := f(P^*) = 0.01149788039455$ with multiplicity 4 to the machine precision; the next eigenvalue is $\lambda_5(\mathcal{F} \cdot E + \mathcal{F} \cdot \tilde{P}^*) = -0.00670989012524$. The corresponding solution is positive definite,

$$P^* = \begin{bmatrix} 1.40104588451064 & 0.47711672234709 & 0.47017792285932 & 1.14391695175544 \\ 0.47711672234709 & 0.23379477434643 & 0.30129912349098 & 0.44304589342816 \\ 0.47017792285932 & 0.30129912349098 & 0.87145527971679 & 0.82344103313835 \\ 1.14391695175544 & 0.44304589342816 & 0.82344103313835 & 1.49370406142614 \end{bmatrix}.$$

Figure 3.2 shows us the acceleration produced by the introduction of second-order information. In particular, we observe the asymptotic superlinear convergence. Algorithm 3.2 is run with $\bar{\varepsilon} = 10^{-8}$. Algorithm 3.3 is initialized with $\bar{\varepsilon} = 10^{-3}$, and a dynamic update of the form $\bar{\varepsilon}_{k+1} := 0.9 * \bar{\varepsilon}_k$ is used when $\bar{\varepsilon}_k$ is too large compared with the ε_k of (3.39). This management of $\bar{\varepsilon}$ enables us to enter more rapidly the region of superlinear convergence.

3.7 Perspectives

Depending on the type of problem to be solved, the need for further work can be anticipated in the following directions.

Poor oracles. Despite some isolated experiments reported in section 3.3, the real capabilities of standard bundle methods have not been really investigated so far. Can they solve problems for which other alternatives are impractical or inefficient? How

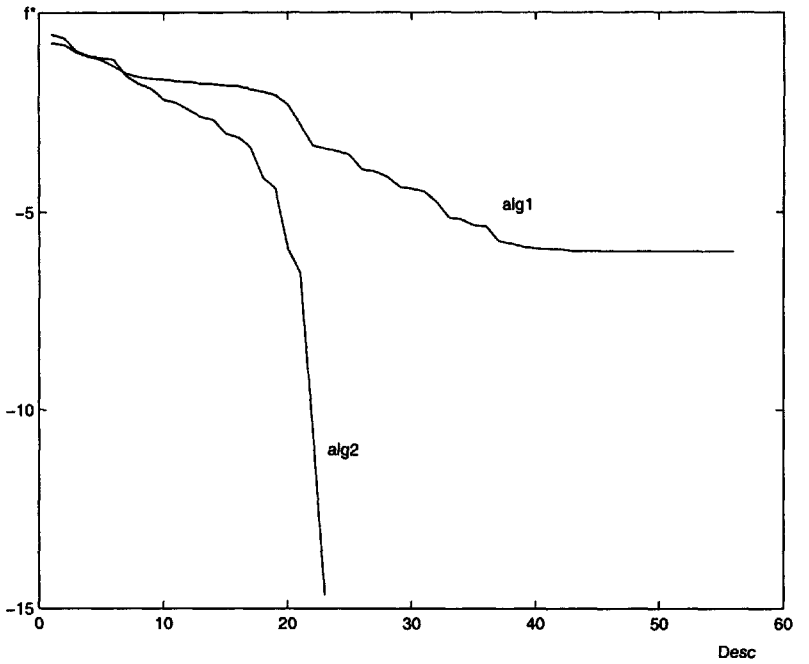


Figure 3.2: Acceleration with second-order information.

large of problems can be solved in practice? To what extent can intermediate oracles be helpful, and how can their additional information be used most efficiently? These questions are not clearly answered yet.

Rich oracles. The real issue underlying the ideas in section 3.4.1 is not completely understood. Are there connections between the dual and primal approaches adopted there? Do these approaches come from a general approximating scheme, or are they just isolated tricks without a firm ground? These questions are important for further developments, for an adequate link with section 3.5, and also for a direct resolution of (3.1).

Another issue is the bulk of work necessary for methods of section 3.5: Computing the $\mathcal{U}$-Hessian appearing in (3.43) really involves heavy computations. The question of approximating this operator by quasi-Newton updates is therefore important for numerical implementations.

On the theoretical side, we mention the question of relating the present nonsmooth analysis with interior points. For example, it should be interesting to relate the $\mathcal{U}$-Hessian with the Hessian of the potential function associated with $\lambda_{\max}$.

This page intentionally left blank

Chapter 4

sdpsol: A Parser/Solver for Semidefinite Programs with Matrix Structure

Shao-Po Wu and Stephen Boyd

4.1 Introduction

This chapter describes a parser/solver for a class of linear matrix inequality (LMI) problems, the so-called max-det problems. A variety of analysis and design problems in control, communication and information theory, statistics, combinatorial optimization, computational geometry, circuit design, and other fields can be expressed as *semidefinite program* (SDP) problems or *determinant maximization* problems (max-det problems). These problems often have matrix structure, which has two important practical ramifications: First, it makes the job of translating the problem into a standard SDP or max-det format tedious, and, second, it opens the possibility of exploiting the structure to speed up the computation.

In this chapter, we describe the design and implementation of `sdpsol`, a parser/solver for SDPs and max-det problems. `sdpsol` allows problems with matrix structure to be described in a simple, natural, and convenient way. Moreover, it allows algorithms to exploit problem structure to gain efficiency. Although the current implementation of `sdpsol` exploits only block-diagonal structure in the solution algorithm, the language and parser were designed with exploiting general matrix structure in mind.

4.1.1 Max-det problems and SDPs

A max-det problem has the form

$$\begin{array}{ll} \text{minimize} & c^T x + \sum_{i=1}^{K} \log\det G^{(i)}(x)^{-1} \\ \text{subject to} & G^{(i)}(x) > 0, \quad i = 1, \ldots, K, \\ & F^{(i)}(x) > 0, \quad i = 1, \ldots, L, \\ & Ax = b, \end{array} \tag{4.1}$$

where the optimization variable is the vector $x \in \mathbf{R}^m$. The matrix functions $G^{(i)} : \mathbf{R}^m \to \mathbf{R}^{l_i \times l_i}$ and $F^{(i)} : \mathbf{R}^m \to \mathbf{R}^{n_i \times n_i}$ are affine:

$$\begin{aligned} G^{(i)}(x) &= G_0^{(i)} + x_1 G_1^{(i)} + \cdots + x_m G_m^{(i)}, \quad i = 1, \ldots, K, \\ F^{(i)}(x) &= F_0^{(i)} + x_1 F_1^{(i)} + \cdots + x_m F_m^{(i)}, \quad i = 1, \ldots, L, \end{aligned}$$

where $G_j^{(i)}$ and $F_j^{(i)}$ are symmetric for $j = 0, \ldots, m$. The inequality signs in (4.1) denote matrix inequalities. We call $G^{(i)}(x) > 0$ and $F^{(i)}(x) > 0$ LMIs in the variable x. Of course the LMI constraints in (4.1) can be combined into one large block-diagonal LMI with diagonal blocks $G^{(i)}(x)$ and $F^{(i)}(x)$.

When $K = 1$ and $G^{(1)}(x) = 1$, the max-det problem (4.1) reduces to the well-known SDP problem:

$$\begin{array}{lll} & \text{minimize} & c^T x \\ (4.2) & \text{subject to} & F^{(i)}(x) > 0, \quad i = 1, \ldots, L, \\ & & Ax = b. \end{array}$$

The max-det problem (4.1) and the SDP (4.2) are convex optimization problems. In fact, LMI constraints can represent many common convex constraints, including linear inequalities, convex quadratic inequalities, and matrix norm and eigenvalue constraints. SDPs and max-det problems arise in many applications; see Alizadeh [5], Boyd, El Ghaoui, Feron, and Balakrishnan [64], Lewis and Overton [247], Nesterov and Nemirovsky [296, section 6.4], Vandenberghe and Boyd [404], and Vandenberghe, Boyd, and Wu [407] for many examples.

SDPs and max-det problems can be solved very efficiently, both in worst-case complexity theory and in practice, using interior-point methods (see [404] and [407]). Current general-purpose solvers such as SP [402], MAXDET [427], SDPT3 [392], SDPA [145], SDPMATH [74], and SDPpack [6] use the standard format (4.2) (and (4.1)) or a simple variation (e.g., its dual form).

4.1.2 Matrix structure

In many SDPs and max-det problems the optimization variables are matrices of various dimensions and structure, e.g., row or column vectors, or symmetric or diagonal matrices. In general, the optimization variables can be collected and expressed as $(X^{(1)}, \ldots, X^{(M)})$, where $X^{(i)} \in \mathbf{R}^{p_i \times q_i}$ and $X^{(i)}$ may have structure (e.g., symmetric, diagonal, upper triangular, etc.). These matrix variables can be vectorized into the single vector variable x that appears in the standard format problems (4.1) and (4.2). To vectorize $X^{(i)}$, we find a basis $E_1^{(i)}, \ldots, E_{m_i}^{(i)}$ such that

$$X^{(i)} = \sum_{j=1}^{m_i} x_j^{(i)} E_j^{(i)},$$

where $x^{(i)} \in \mathbf{R}^{m_i}$ denotes the vectorized $X^{(i)}$. For example, if $X^{(i)} \in \mathbf{R}^{p_i \times q_i}$ has no structure, we have $x^{(i)} = \mathbf{vec}(X^{(i)})$ and $m_i = p_i q_i$; if $X^{(i)} \in \mathbf{R}^{p_i \times q_i}$ is diagonal ($p_i = q_i$), we have $x^{(i)} = \mathbf{diag}(X^{(i)})$ and $m_i = p_i$. Doing this for $i = 1, \ldots, M$, we obtain the vectorized variable $x \in \mathbf{R}^m, m = m_1 + \cdots + m_M$, and

$$x = \left[\begin{array}{ccc} x^{(1)T} & \cdots & x^{(M)T} \end{array} \right]^T.$$

Note that each variable $X^{(i)}$ of the problem corresponds to part of the vectorized variable x. With this correspondence, we can convert the problem to the standard form (4.1) or (4.2), and vice versa.

The following example illustrates this vectorization process. We consider the problem

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}P \\ \text{subject to} & \begin{bmatrix} -A^TP-PA & -PB \\ -B^TP & R \end{bmatrix} > 0, \\ & P = P^T > 0, \\ & R > 0,\ R \text{ diagonal},\ \mathbf{Tr}R = 1, \end{array} \tag{4.3}$$

where A, B are given matrices (i.e., problem data), and the symmetric $P \in \mathbf{R}^{n\times n}$ and diagonal $R \in \mathbf{R}^{k\times k}$ are the optimization variables.

We vectorize P and R as

$$P = \sum_{i=1}^{n}\sum_{j=i}^{n} x_{ij}^{(1)} P_{ij}, \tag{4.4}$$

$$R = \sum_{i=1}^{k} x_i^{(2)} R_i, \tag{4.5}$$

where P_{ij} denotes an $n\times n$ zero matrix except the (i,j) and (j,i) entries are 1; R_i denotes a $k \times k$ zero matrix except the (i,i) entry is 1. Substituting (4.4) and (4.5) for P and R everywhere in the SDP (4.3), we obtain the problem of the form (4.2) in the optimization variable

$$x = \begin{bmatrix} x_{11}^{(1)} & \cdots & x_{nn}^{(1)} & x_1^{(2)} & \cdots & x_k^{(2)} \end{bmatrix}^T \in \mathbf{R}^{n(n+1)/2+k}.$$

4.1.3 Implications of the matrix structure

Clearly it is straightforward but inconvenient to convert an SDP or a max-det problem with matrix structure into the standard form (4.2) or (4.1). This conversion obscures the problem structure and the correspondence between the variables $X^{(i)}$ and the vectorized variable x, which makes it hard to interpret the results after the problem is solved.

Moreover, the problem structure can be exploited by interior-point methods for substantial efficiency gain (see [403] and [67] for examples). To illustrate the idea with a simple example, consider the operation

$$\mathcal{L}(P) = -A^TP - PA$$

that evaluates the $-A^TP - PA$ term in the first matrix inequality of (4.3). $\mathcal{L}(P)$ is an $\mathcal{O}(n^3)$ operation because it involves matrix multiplications of $n \times n$ matrices. However, if we vectorize P as shown in (4.4), then $\mathcal{L}(P)$ becomes

$$\mathcal{L}(P) = \sum_{i=1}^{n}\sum_{j=i}^{n} x_{ij}^{(1)}(-A^TP_{ij} - P_{ij}A),$$

which is an $\mathcal{O}(n^4)$ operation.

4.1.4 sdpsol and related work

In this paper, we describe the design and implementation of **sdpsol**, a parser/solver for SDPs and max-det problems with matrix structure. Problems can be specified in the **sdpsol** language in a form very close to their natural mathematical descriptions. As an

example, the SDP (4.3) can be specified as

```
variable P(n,n) symmetric;
variable R(k,k) diagonal;
[-A'*P-P*A, -P*B;
 -B'*P,     R    ] > 0;
P > 0;
R > 0;  Tr(R) == 1;
minimize  objective = Tr(P);
```

The problem specification is then compiled by the `sdpsol` parser into the standard form and passed to the solution algorithm to be solved. The solution (optimal variables) will again be interpreted by the parser and returned in their specified structure.

Evidently, the `sdpsol` parser/solver automates the tedious and inconvenient step of converting an SDP or a max-det problem into the standard form. It also allows a structural description of the problem such that sophisticated solution algorithms can exploit the problem structure to gain efficiency.

There exist several similar tools that use a specification language to describe and solve certain mathematical programming problems. A well-known example is AMPL [139], which handles linear and integer programming problems.

LMITOOL [126] and the LMI Control Toolbox [151] provide convenient Matlab interfaces that allow the user to specify SDPs with matrix structure (specifically the ones arising in control), using a different approach from the parser/solver described in this paper.

In section 4.2, we describe the design of the specification language. A preliminary implementation of `sdpsol` (version beta) is described in section 4.3. In section 4.4, we give several examples to illustrate various features of `sdpsol`.

4.2 Language design

4.2.1 Matrix variables and affine expressions

The fundamental data structure of the `sdpsol` language is, as in Matlab [266], the matrix. The language provides a Matlab-like grammar, including various facilities to manipulate matrix expressions. For example, it provides commands that construct matrices, operations such as matrix multiplication and transpose, and matrix functions such as trace and inner product. The language also supports assignments; i.e., expressions can be assigned to internal variables that can later be used in the problem specification. Various algorithm parameters, such as tolerance or maximum number of iterations, can be initialized using assignments.

An important extension beyond Matlab is that the `sdpsol` language provides matrix variables and, by extension, matrix expressions formed from matrix variables and matrix constants. Matrix variables of various dimensions and structure can be declared using variable declaration statements. For example, the statements

```
variable P(7,7) symmetric;
variable x(k,1), y;
```

declare a 7×7 symmetric variable P, a $k \times 1$ variable x, and a scalar variable y.

Variables can be used to form *affine expressions*, i.e., expressions that depend affinely on the optimization variables. For example, the expression (assuming P is a variable and

A, D are constant square matrices)

```
A'*P + P*A + D
```

is an affine expression that depends affinely on P. Affine expressions can be used to construct LMI constraints and the objective function of SDPs and max-det problems.

As each affine expression is parsed, `sdpsol` keeps track of its dependence on the variables, adding the dependency information to an internal table for later reference, e.g., by a solver.

4.2.2 Constraints and objective

Various types of (affine) constraints are supported by the language, including matrix inequalities, componentwise inequalities, and equality constraints. Constraints are formed with affine expressions and relation operators, as in

```
A'*P + P*A + D < -1;
diag(P) .> 0;
Tr(P) == 1;
```

which specify the matrix inequality $A^TP + PA + D < -I$, the componentwise inequality $\mathbf{diag}P > 0$, and the equality $\mathbf{Tr}P = 1$.

The objective function is specified via an assignment, as in

```
maximize  objective = Tr(P);
```

which assigns $\mathbf{Tr}P$ to the internal variable `objective` and makes maximizing it the objective. A feasibility problem (i.e., a problem that determines whether a set of constraints is feasible or not) is specified without an objective statement.

4.2.3 Duality

We associate with the max-det problem (4.1) the *dual* problem (see [407] for details):

$$
\begin{array}{ll}
\text{maximize} & \displaystyle\sum_{i=1}^{K}\left(\log\det W^{(i)} - \mathbf{Tr}G_0^{(i)}W^{(i)} + l_i\right) - \sum_{i=1}^{L}\mathbf{Tr}F_0^{(i)}Z^{(i)} + b^Tz \\
\text{subject to} & \displaystyle\sum_{i=1}^{K}\mathbf{Tr}G_j^{(i)}W^{(i)} + \sum_{i=1}^{L}\mathbf{Tr}F_j^{(i)}Z^{(i)} + (A^Tz)_j = c_j, \quad j = 1,\ldots,m, \\
& W^{(i)} > 0, \quad i = 1,\ldots,K, \\
& Z^{(i)} > 0, \quad i = 1,\ldots,L,
\end{array}
\tag{4.6}
$$

where the *dual variables* are symmetric matrices $W^{(i)} \in \mathbf{R}^{l_i\times l_i}$ for $i = 1,\ldots,K$, symmetric matrices $Z^{(i)} \in \mathbf{R}^{n_i\times n_i}$ for $i = 1,\ldots,L$, and a vector $z \in \mathbf{R}^p$ (assuming $A \in \mathbf{R}^{p\times m}$ and $b \in \mathbf{R}^p$ in (4.1)). We will refer to (4.1) as the *primal* problem of (4.6). Note that (4.6) is itself a max-det problem.

Similarly, we can associate with the SDP (4.2) the dual

$$
\begin{array}{ll}
\text{maximize} & \displaystyle -\sum_{i=1}^{L}\mathbf{Tr}F_0^{(i)}Z^{(i)} + b^Tz \\
\text{subject to} & \displaystyle\sum_{i=1}^{L}\mathbf{Tr}F_j^{(i)}Z^{(i)} + (A^Tz)_j = c_j, \quad j = 1,\ldots,m, \\
& Z^{(i)} > 0, \quad i = 1,\ldots,L,
\end{array}
\tag{4.7}
$$

which is also an SDP.

The dual problem has several practical uses; e.g., the optimal dual variables give a sensitivity analysis of the primal problem. The most efficient SDP and max-det solvers are primal dual; i.e., they simultaneously solve the primal and dual problems. `sdpsol` automatically forms the dual from the primal problem specification. The dual problem is used by the solver and also, if requested, returns to the user optimal dual variables.

There is a simple relation between the form (size and structure) of the primal constraints and the dual variables (and vice versa). Observe that for each constraint in (4.1) (or (4.2)), there is a dual variable of the same dimensions and structure associated with it. The variables associated with the primal LMI constraints are constrained to be positive definite in the dual problem, while the variables associated with the equality constraints are unconstrained.

Taking the SDP (4.3) as an example, we have that `sdpsol` associates the dual variables

$$\begin{aligned} &Z^{(1)} = Z^{(1)^T} \in \mathbf{R}^{(n+k)\times(n+k)}, \\ &Z^{(2)} = Z^{(2)^T} \in \mathbf{R}^{n\times n}, \\ &Z^{(3)} \in \mathbf{R}^{k\times k}, \quad Z^{(3)} \text{ diagonal}, \end{aligned}$$

with the three matrix inequalities, respectively, and $z \in \mathbf{R}$ with the equality constraint. After some manipulation (which can be easily performed by `sdpsol`), we have the dual problem

$$\begin{array}{lll} \text{maximize} & z & \\ \text{subject to} & \mathbf{Tr}\begin{bmatrix} -A^TP_{ij}-P_{ij}A & -P_{ij}B \\ -B^TP_{ij} & 0 \end{bmatrix} Z^{(1)} + \mathbf{Tr}P_{ij}Z^{(2)} = \delta_{ij} & \\ & \qquad \text{for all } i = 1,\ldots,n \text{ and } j = i,\ldots,n, & (4.8) \\ & \mathbf{Tr}\begin{bmatrix} 0 & 0 \\ 0 & R_i \end{bmatrix} Z^{(1)} + \mathbf{Tr}R_iZ^{(3)} + z = 0, \quad i = 1,\ldots,k, & \\ & Z^{(1)} > 0, \quad Z^{(2)} > 0, \quad Z^{(3)} > 0, & \end{array}$$

where P_{ij}, R_i are given by (4.4) and (4.5), $\delta_{ij} = 1$ if $i = j$, otherwise zero.

`sdpsol` provides an alternate form of constraint specifications, in which the user associates a name with each constraint, which is then used by `sdpsol` to refer to the dual variables. For example, the statements (assuming $P, A, D \in \mathbf{R}^{n\times n}$)

```
constraint lyap   A'*P + P*A + D < -1;
constraint equ    Tr(P) == 1;
```

specify (and name) the constraints $A^TP+PA+D < -I$ and $\mathbf{Tr}P = 1$. `sdpsol` associates with these constraints the dual variables with names

$$\begin{aligned} &\textit{lyap\_dual} \in \mathbf{R}^{n\times n}, \quad \textit{lyap\_dual} \text{ symmetric}, \\ &\textit{equ\_dual} \in \mathbf{R}. \end{aligned}$$

After the solution phase, `sdpsol` returns the optimal values of these dual variables.

4.3 Implementation

We have implemented a preliminary version of the parser/solver, `sdpsol` version beta. The parser is implemented using BISON and FLEX. Two solvers, SP [402] and MAXDET [427], are used to solve the resulting SDPs and max-det problems. Both solvers exploit only block-diagonal structure, so the current version of `sdpsol` is not particularly efficient.

Both SP and MAXDET are primal-dual methods that require a feasible starting primal and dual point. `sdpsol` uses the method described in [404, section 6] to handle

the feasibility phase of the problem; that is, `sdpsol` either finds a feasible starting point (to start the optimization phase) or proves that the problem is infeasible. If there is no objective in the problem specification, `sdpsol` simply solves the feasibility problem only.

A Matlab interface is provided by `sdpsol` to import problem data and export the results. `sdpsol` can also be invoked from within Matlab interactively.

4.3.1 Handling equality constraints

Neither SP nor MAXDET handles problems with equality constraints, so the current implementation of `sdpsol` eliminates them, uses SP or MAXDET to solve the resulting problem (which has no equality constraints), and finally reassembles the variables before reporting the results.

The equality constraints given in the problem specification can be collected and put into the form of $Ax = b$ ($A \in \mathbf{R}^{p\times m}, b \in \mathbf{R}^p$) as given in (4.1) and (4.2). We assume that the rank of A, say, r, is strictly less than m (otherwise the problem has a trivial solution). The matrix A is often not full rank, because `sdpsol` does not detect and remove redundancy in the equality constraints specified. For example, a 2×2 matrix variable X can be constrained to be symmetric by the constraint specification

```
X == X';
```

`sdpsol` interprets the above statement as the following four equality constraints:

$$X_{11} = X_{11}, \quad X_{12} = X_{21}, \quad X_{21} = X_{12}, \quad X_{22} = X_{22}.$$

Evidently one constraint $X_{21} = X_{12}$ is necessary. This will show up as two zero rows in A and two identical rows of A; in particular, A will not be full rank.

Instead of keeping track of such situations, `sdpsol` builds up the matrix A paying no attention to rank. `sdpsol` then performs a complete orthogonal decomposition (see [171]) on A such that

$$AP = Q \begin{bmatrix} R_{11} & 0 \\ 0 & 0 \end{bmatrix} U^T, \tag{4.9}$$

where P is a column permutation matrix (for column pivoting), $R_{11} \in \mathbf{R}^{r\times r}$ is upper triangular, and $Q, U \in \mathbf{R}^{p\times p}$ are orthogonal matrices.

All $x \in \mathbf{R}^m$ satisfying the equality constraints can be expressed as

$$x = PU \begin{bmatrix} R_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q^T b + PU \begin{bmatrix} 0 \\ \widetilde{x} \end{bmatrix} \triangleq x_p + PU \begin{bmatrix} 0 \\ \widetilde{x} \end{bmatrix}$$

$\forall\, \widetilde{x} \in \mathbf{R}^{\widetilde{m}}$ with $\widetilde{m} = m - r$. Defining $B \in \mathbf{R}^{m\times\widetilde{m}}$ by

$$\begin{bmatrix} 0 & B \end{bmatrix} \triangleq PU \begin{bmatrix} 0 \\ I_{\widetilde{m}\times\widetilde{m}} \end{bmatrix},$$

we can express the max-det problem (4.1) as

$$\begin{aligned} \text{minimize} \quad & c^T(x_p + B\widetilde{x}) + \sum_{i=1}^{K} \log\det \widetilde{G}^{(i)}(\widetilde{x})^{-1} \\ \text{subject to} \quad & G^{(i)}(x_p) + \sum_{j=1}^{\widetilde{m}} \widetilde{x}_j \left(\sum_{k=1}^{m} B_{kj} G_k^{(i)} \right) > 0, \quad i = 1, \ldots, K, \\ & F^{(i)}(x_p) + \sum_{j=1}^{\widetilde{m}} \widetilde{x}_j \left(\sum_{k=1}^{m} B_{kj} F_k^{(i)} \right) > 0, \quad i = 1, \ldots, L, \end{aligned} \tag{4.10}$$

where $\widetilde{x} \in \mathbf{R}^{\widetilde{m}}$ is the optimization variables and B_{kj} denotes the (k,j) entry of B. Evidently, the above process reduces the optimization problem (4.1) in $\mathbf{R}^m$ with equality constraints to (4.10) in $\mathbf{R}^{\widetilde{m}}$ without any equality constraint. Similar methods apply to the SDP (4.2).

We should add that eliminating variables destroys the matrix structure that could be exploited by a solver. A future version of `sdpsol` would directly pass the equality constraints to an SDP/max-det solver that handles equality constraints.

4.4 Examples

In this section, we give several examples to illustrate various features of `sdpsol`.

4.4.1 Lyapunov inequality

As a very simple example, consider a linear system described by the differential equation

$$\dot{x}(t) = Ax(t), \tag{4.11}$$

where $A \in \mathbf{R}^{n\times n}$ is given. The linear system is stable (i.e., all solutions of (4.11) converge to zero), if and only if there exists a symmetric, positive definite P such that the Lyapunov inequality

$$A^TP + PA < 0$$

is satisfied. The problem of finding such a P (if one exists) can be specified in the `sdpsol` language as follows:

```
% Lyapunov inequality
variable P(n,n) symmetric;

A'*P + P*A < 0;
P > 0;
```

In the problem specification, an $n \times n$ symmetric variable P is declared. An affine expression $A^TP + PA$ is formed and is used to specify the Lyapunov inequality. Another LMI, $P > 0$, constrains P to be positive definite. Since no objective is given, `sdpsol` solves the feasibility problem that either finds a feasible P or shows that the problem is infeasible.

Note that this problem can be solved analytically. Indeed, we can solve the linear equation $A^TP + PA + I = 0$ for the matrix P, which is positive definite if and only if (4.11) is stable.

4.4.2 Popov analysis

Consider the mass-spring-damper system with a vibrating base shown in Figure 4.1.

The masses are 1kg each, the dampers have the damper constant 0.5nt/m/s, and the springs are nonlinear: $F_i = \phi_i(x_i)$, where

$$0.7 \le \frac{\phi_i(a)}{a} \le 1.3, \quad i = 1,2,3.$$

The base vibration is described by

$$w_{\text{base}} = \frac{1}{1 + s/0.3}\, w,$$

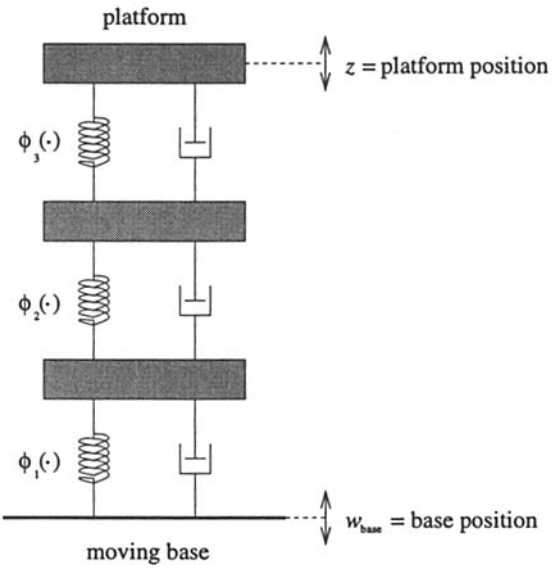


Figure 4.1: Mass-spring-damper system.

where RMS$(w) \leq 1$ but is otherwise unknown. Our goal is to find an upper bound on RMS(z).

The mass-spring-damper system can be described by the Lur'e system (see [64, section 8.1]) with 7 states and 3 nonlinearities

$$\begin{aligned} \dot{x} &= Ax + B_p p + B_w w, \\ z &= C_z x, \\ q &= C_q x, \\ p_i(t) &= \widetilde{\phi}_i(q_i(t)), \quad i = 1,2,3, \end{aligned} \tag{4.12}$$

where $x \in \mathbf{R}^7$, $p \in \mathbf{R}^3$ and the functions $\widetilde{\phi}_i$ satisfy the $[0,1]$ sector condition; i.e.,

$$0 \leq q_i \widetilde{\phi}_i(q_i) \leq q_i^2$$

for $i = 1,2,3$. One method to find an upper bound is to find a γ^2 and a Lyapunov function of the form

$$V(x) = x^T P x + 2\sum_{i=1}^{3} \lambda_i \int_0^{C_{q_i} x} \widetilde{\phi}_i(\sigma) d\sigma, \tag{4.13}$$

where C_{q_i} denotes the ith row of C_q such that

$$\frac{d}{dt} V(x) \leq \gamma^2 w^T w - z^T z$$

for all w, z satisfying (4.12). The square root of γ^2 yields an upper bound on RMS(z).

The problem of finding γ^2 and the Lyapunov function (4.13) can be further relaxed using the $\mathcal{S}$-procedure to the following SDP (see [64, section 8.1.4]):

$$\begin{array}{ll} \text{minimize} & \gamma^2 \\ \text{subject to} & P > 0, \ L \geq 0, \ T \geq 0 \\ & \begin{bmatrix} A^T P + PA + C_z^T C_z & PB_p + A^T C_q^T L + C_q^T T & PB_w \\ B_p^T P + LC_q A + TC_q & LC_q B_p + B_p^T C_q^T L - 2T & LC_q B_w \\ B_w^T P & B_w^T C_q^T L & -\gamma^2 \end{bmatrix} \leq 0, \end{array} \tag{4.14}$$

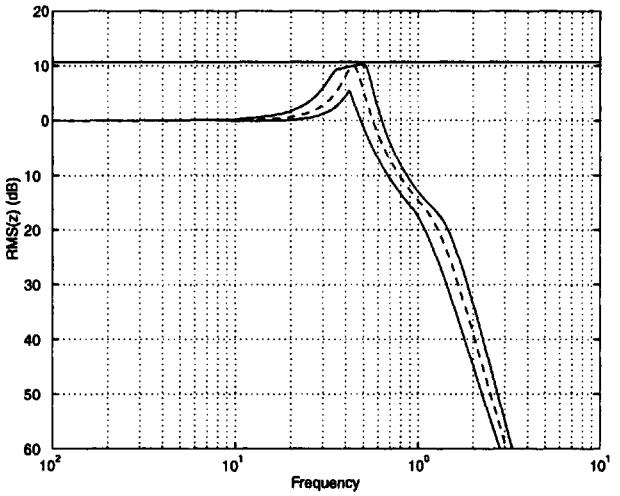


Figure 4.2: RMS(z) of mass-spring-damper system and upper bound.

where $P = P^T \in \mathbf{R}^{7\times 7}$, $L, T \in \mathbf{R}^{3\times 3}$ diagonal, and $\gamma^2 \in \mathbf{R}_+$ are the optimization variables. The square root of the optimal γ^2 of (4.14) is an upper bound of RMS(z) if there exists P, L, T that satisfy the constraints.

The SDP (4.14) can be described using the `sdpsol` language as follows:

```
% Popov analysis of a mass-spring-damper system
variable P(7,7) symmetric;
variable L(3,3), T(3,3) diagonal;
variable gamma_sqr;

P > 0;
L > 0;
T > 0;
[A'*P+P*A+Cz'*Cz,  P*Bp+A'*Cq'*L+Cq'*T,  P*Bw;
 Bp'*P+L*Cq*A+T*Cq,L*Cq*Bp+Bp'*Cq'*L-2*T,L*Cq*Bw;
 Bw'*P,            Bw'*Cq'*L,            -gamma_sqr]<0;

minimize RMS_bound_sqr = gamma_sqr;
```

Again, the specification in the `sdpsol` language is very close to the mathematical description (4.14). The objective of the problem is to minimize `RMS_bound_sqr`, the square of the RMS bound, which is equal to the variable `gamma_sqr`.

Unlike the previous example, this problem of finding an upper bound on RMS(z) has *no* analytical solution. However, one can easily specify and solve the problem using `sdpsol`. We show in Figure 4.2 an envelope of possible RMS(z) obtained via Monte-Carlo simulation and the upper bound obtained via solving (4.14). RMS(z) of the nominal system is shown in the dashed line.

4.4.3 *D*-optimal experiment design

Consider the problem of estimating a vector x from a measurement $y = Ax + w$, where $w \sim N(0, I)$ is the measurement noise. The error covariance of the minimum-variance estimator is equal to $A^\dagger(A^\dagger)^T = (A^TA)^{-1}$. We suppose that the rows of the matrix

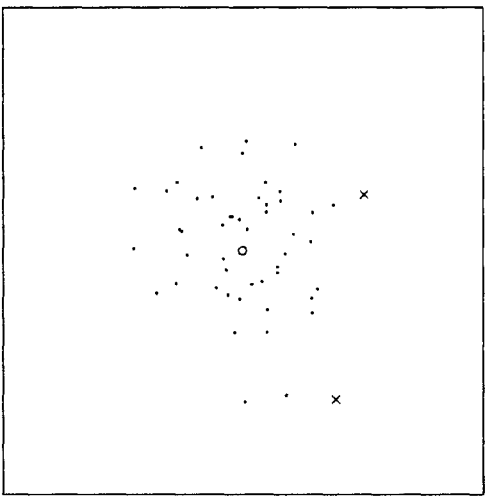


Figure 4.3: D-optimal experiment design.

$A = [a_1 \ \ldots \ a_q]^T$ can be chosen among M possible test vectors $v^{(i)} \in \mathbf{R}^p$, $i = 1, \ldots, M$:

$$a_i \in \{v^{(1)}, \ldots, v^{(M)}\}, \quad i = 1, \ldots, q.$$

The goal of experiment design is to choose the vectors a_i so that the determinant of the error covariance $(A^T A)^{-1}$ is minimized. This is called the D-optimal experiment design.

We can write $A^T A = \sum_{i=1}^M \lambda_i v^{(i)} v^{(i)^T}$, where λ_i is the fraction of rows a_k equal to the vector $v^{(i)}$. We ignore the fact that the numbers λ_i are integer multiples of $1/q$ and instead treat them as continuous variables, which is justified in practice when q is large.

In D-optimal experiment design, λ_i are chosen to minimize the determinant of the error covariance matrix, which can be cast as the max-det problem

$$\begin{array}{lll} \text{minimize} & \log\det\left(\displaystyle\sum_{i=1}^M \lambda_i v^{(i)} v^{(i)^T}\right)^{-1} & \\ \text{subject to} & \lambda_i \geq 0, \ i = 1, \ldots, M, & \\ & \displaystyle\sum_{i=1}^M \lambda_i = 1. & \end{array} \tag{4.15}$$

This problem can be described in the `sdpsol` language as

```
% D-optimal experiment design
variable lambda(M,1);

lambda .> 0;
sum(lambda) == 1;

Cov_inv = zeros(p,p);
for i=1:M;
  Cov_inv = Cov_inv + lambda(i,1)*v(:,i)*v(:,i)';
end;

minimize log_det_Cov = -logdet(Cov_inv);
```

In the specification, an M-vector `lambda` is declared to be the optimization variable. The

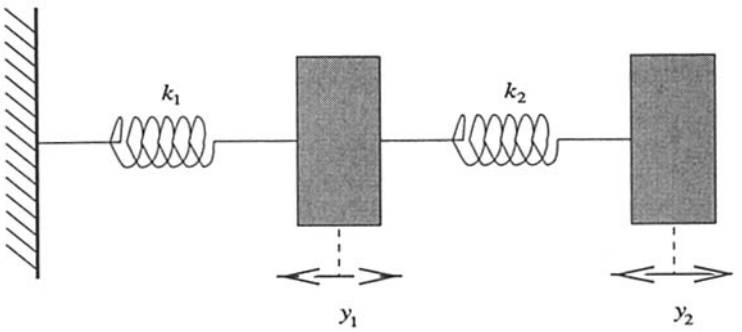


Figure 4.4: Mass-spring system.

(componentwise) inequality constraint specifies that each entry of `lambda` is positive, and the equality constraint says the summation of all entries of `lambda` is 1.

A for-loop is used to construct `Cov_inv`, the inverse of the covariance matrix, to be $\sum_{i=1}^{M} \lambda_i v^{(i)} v^{(i)^T}$. The objective of the optimization problem is to minimize the log determinant of the inverse of `Cov_inv`. An implicit LMI constraint, `Cov_inv` > 0 is added to the problem as soon as the objective is specified. This LMI corresponds to the $G(x) > 0$ term in (4.1), and it ensures that the log-determinant term is well defined.

A numerical example of D-optimal experiment design involving 50 test vectors in $\mathbf{R}^2$ is shown in Figure 4.3. The circle is the origin; the dots are the test vectors that are not used in the experiment; the x's are the vectors that are used. The optimal experiment allocates all measurements to only two test vectors.

4.4.4 Lyapunov exponent analysis

Consider the mass-spring system shown in Figure 4.4 [63, section 7]: Two unit masses and a wall (infinite mass) are connected by nonlinear springs with spring constants that can change instantly over the range of $[1, 2]$. We would like to find an upper bound on the rate of extracting energy from (or pumping energy into) the system by loosening and stiffening the springs.

The system can be described by the differential inclusion

$$\frac{dy}{dt}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_1(t) - k_2(t) & k_2(t) & 0 & 0 \\ k_2(t) & -k_2(t) & 0 & 0 \end{bmatrix} y(t), \qquad 1 \le k_i(t) \le 2 \text{ for } i = 1, 2,$$

where the four extreme cases (i.e., $k_i = 1$ or 2, $i = 1, 2$) are denoted by $\dot{y} = A_j y$, $j = 1, \ldots, 4$. It is shown in [63, 64] that an upper bound is given by the solution of the *generalized eigenvalue problem*:

$$\begin{array}{ll} \text{minimize} & \alpha \\ \text{subject to} & A_j^T P + P A_j - 2\alpha P \le 0, \quad j = 1, \ldots, 4, \\ & \mathbf{Tr} P = 1, \\ & P > 0. \end{array} \tag{4.16}$$

The above problem is a quasi-convex optimization problem and cannot be posed as an SDP or a max-det problem. However, we can solve (4.16) via bisecting an interval of α

and solve, for each α, the SDP feasibility problem

$$\begin{array}{ll} \text{find} & P \\ \text{subject to} & A_j^T P + P A_j - 2\alpha P \leq 0, \quad j = 1, \ldots, 4, \\ & \mathbf{Tr} P = 1, \\ & P > 0 \end{array} \tag{4.17}$$

until the minimum feasible α is found.

We can specify (4.17) in the `sdpsol` language as

```
% Lyapunov exponent analysis via bisection

variable P(4,4) symmetric;

A1'*P+P*A1-2*alpha*P < 0;
A2'*P+P*A2-2*alpha*P < 0;
A3'*P+P*A3-2*alpha*P < 0;
A4'*P+P*A4-2*alpha*P < 0;
Tr(P) == 1;
P > 0;
```

With a simple Matlab script that performs the bisection and checks the feasibility results returned by `sdpsol`, we can solve the problem easily. In fact, since $\alpha \geq 0$ and

$$\alpha \leq \max_{j=1,\ldots,4} \frac{1}{2}\lambda_{\max}(A_j^T + A_j, I) \approx 2.12,$$

we can use bisection over the interval $[0, 2.12]$ and get the optimal bound ≈ 0.33.

4.5 Summary

Using a parser/solver (such as `sdpsol` described in this paper) to solve SDPs and max-det problems has the following advantages:

- Problems with matrix structure can be conveniently specified. The process of converting problems into the standard forms is automatized.
- Problem structure is kept during compilation and can be exploited fully by sophisticated solvers to gain efficiency.
- The solver can be easily updated or changed by modifying its interface with the parser. There is no need to change the problem specifications.

Acknowledgments

The authors would like to thank Lieven Vandenberghe, Laurent El Ghaoui, Paul Dankoski, and Michael Grant for very helpful discussions, comments, and suggestions.

This page intentionally left blank