Semidefinite optimization, a spectral approach

Semidefiniete optimalisatie, een spectrale benadering

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de Rector Magnificus, Prof. dr. W.H. Gispen, ingevolge het besluit van het College voor Promoties in het openbaar te verdedigen op woensdag 5 juni 2002 des middags te 12.45 uur

door

Mischja Alexander van Bossum

geboren op 13 november 1972, te Zaandam

Promotor:	Prof. Dr. H.A. van der Vorst Faculteit der Wiskunde en Informatica Universiteit Utrecht
Co-promotor:	Dr. G.L.G. Sleijpen Faculteit der Wiskunde en Informatica Universiteit Utrecht

Dit onderzoek maakt deel uit van het SWON project *High Performance Methods for Mathematical Optimization* (HPMMO) en wordt gefinancierd door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO). HPMMO is een samenwerkingsverband tussen de Erasmus Universiteit Rotterdam, de Technische Universiteit Delft, de Technische Universiteit Eindhoven en de Universiteit Utrecht.

2000 Mathematics Subject Classification: 15Axx, 26A18, 26A24, 26A51, 30B10, 47A55, 47A75, 65Fxx, 65Kxx, 90Cxx.

Bossum, Mischja Alexander van
Semidefinite optimization, a spectral approach
Proefschrift Universiteit Utrecht – met een samenvatting in het Nederlands.

ISBN 90–393–3041–7

Contents

Introduction

This thesis is about mathematical optimization. Mathematical optimization involves the construction of methods to solve optimization problems, which can arise from real-life problems in applied science, when they are mathematically modeled. Examples come from electrical design, engineering, control theory, telecommunication, environment, finance, and logistics. The mathematical models are relations between variables, which can be discrete or continuous, for example, the amount of men or the time needed for a job, and the relations can be linear equations or more difficult differential equations. The system of equations usually has multiple solutions and we try to find an optimal solution within the set of solutions, for example, maximal profit, minimal time or minimal amount of energy, the objective. The goal will always be to find a solution to the system of equations such that the objective is optimal. The system of equations together with the objective is called an optimization problem.

This thesis deals with semidefinite optimization problems by first introducing linear optimization, also called linear programming, and then expanding the theory to semidefinite optimization, which is also known as semidefinite programming. The semidefinite optimization problem will then be reformulated so that it is suitable for large scale optimization, as will be explained later. Semidefinite programming is a large class of convex optimization problems with convex constraints and these constraints take the form of linear matrix inequalities, where the matrix variables lie in a semidefinite cone. The inequalities induce a bound on the eigenvalues of sums of symmetric or hermitian matrices. Eigenvalues will play an important role if we reformulate the semidefinite optimization problem.

Applications of semidefinite optimization problems can be found in, for example, combinatorial optimization, when separating two sets of points by a minimal ellipsoid [**46**], see Figure 1, and in engineering, when we are interested in the optimal design of structures like bridges or the sizing of transistors [**6**], [**46**], [**47**].

Optimization has been used for centuries, but since the 1950's, at the beginning of the computer technology, it has become more famous. Computers have made it possible to apply algorithms to real life applications, which usually involve a large number of variables and or relations between those variables. The size of the problem can be in the order of thousands or even millions of variables. In certain applications the computations must be completed in a fraction of a second, so apart from powerful computers, efficient algorithms are also needed. In this thesis we will concentrate on the design of an efficient solution method for solving semidefinite programming problems, applicable or extensible to large-scale optimization.

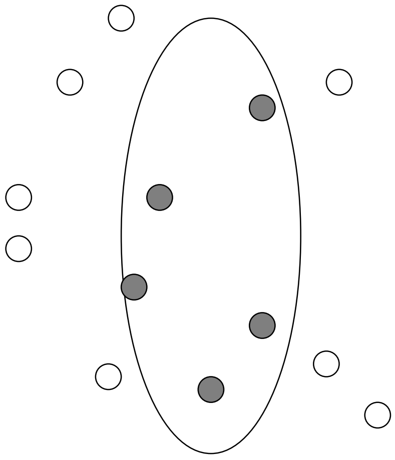
FIGURE 1 A separating ellipsoid.

In 1984 Karmarkar [**24**] initiated the field of interior-point methods for linear programming. These methods are based on a Newton method for solving the optimality conditions of the optimization problem. The optimality conditions are the necessary and sufficient conditions for the solution of the optimization problem to be optimal. Interior-point methods for linear programming have polynomial time behavior strongly dependent on the efficiency of the Newton method [**31**], and the Newton method has quadratic convergence. A couple of years after Karmarkar had published his new method Nesterov and Nemirovskii developed an interior-point method for semidefinite optimization problems that has the same properties as the interior-point method for linear programming [**31**], this method soon got a practical follow-up by Boyd and Vandenberghe [**46**]. One of the differences between semidefinite programming problems and linear programming problems is the number of variables. Semidefinite optimization problems usually are of much larger order, and therefore need more efficient algorithms. Methods that consume less computation time are needed, for example, subspace methods that already exist for solving linear equations, [**5**], [**17**], or eigenvalue problems, [**4**], [**36**]. Subspace methods project the data of a specific problem on a much smaller subproblem that can be solved very fast, and return a good approximation for the solution of the real problem. We do this by transforming the semidefinite programming problem to a convex non-smooth optimization problem by writing its constraint as an eigenvalue constraint, and we develop an algorithm that solves the semidefinite optimization problem for this formulation. The algorithm can be adjusted to apply subspace methods for eigenvalue problems.

In the first section of Chapter 1 some basic linear algebra and analysis is explained. In Section 1.2 a general framework is formulated for convex programming problems in which the linear and the semidefinite programming problems are then fitted. The optimality conditions are described and the chapter is ended with some examples of semidefinite programming problems in Subsection 1.2.4.

In the second chapter we review techniques for the computation of solutions of convex programming problems. The techniques are first given for the general framework developed in Chapter 1, then via the techniques for linear programming problems we fit the semidefinite programming in the general framework. The methods for finding a solution of convex programming problems are called interior-point methods, which are similar to Newton methods. In Section 2.3 more details are given about the implementation of the interior-point methods. In Section 2.4 we propose to project the convex programming problem to a problem of smaller size for the linear and semidefinite programming problems such that not too much information is lost, it becomes clear that for semidefinite programming problems we can formulate a working strategy, although we do not know in general on what space we should project. The last subsection gives sufficient inspiration for more investigation on the projection of semidefinite programming problems onto eigenspaces, which will be done in Chapter 3.

In Chapter 3 a method is introduced for solving semidefinite programming problems, using an eigenvalue formulation. In the first section, Section 3.1, the eigenvalue formulation, properties of a relevant Lagrange multiplier and the derivative of eigenvalue functions are described. In Section 3.2 a method is developed for minimizing the maximum eigenvalue in case the desired eigenvalue is simple. The method is illustrated by some examples in Subsection 3.2.6. In Section 3.3 the method for the simple eigenvalue case is extended to the multiple eigenvalue case. Again the method is illustrated by some examples in Subsection 3.3.5. In Section 3.4 the most common ways of computing eigenvalues are shown, and examples are given of a direct and a reduction method. A subspace principle is developed in Section 3.5 for the methods described in Section 3.2 and 3.3, we illustrate the subspace procedure by some examples in Subsection 3.5.1. The connection between the size of the subspace and the amount of iterations needed to solve the semidefinite optimization problems is explained in Subsection 3.5.2. Two other methods incorporating the computation of the minimal maximum eigenvalue are reviewed in Section 3.6. These methods are the spectral bundle method and Overton's eigenvalues method. Differences and similarities are also included. We conclude Chapter 3 with a discussion and conclusion in Section 3.7.

CHAPTER 1

Notation and modeling

1.1. Basic linear algebra and analysis

The first section contains some basic linear algebra and analysis. The definitions mainly serve to introduce notation, that have been included to make this thesis as self-contained as possible. We will first introduce some well-known theory about sets, mappings, norms, and inner products. We will use these notions to discuss the consequences for the special sets of real and complex numbers. After this the set of symmetric matrices is introduced and, of course, we will discuss some relevant properties of matrices. Finally we describe convex sets and functions, and a method to find the zeros of a function.

Throughout this thesis $:=$ will denote equality by definition.

1.1.1. Sets, orderings, mappings, spaces, norms, and inner products

Sets and subsets. Let X and Y be sets. We will use standard notations $x \in \mathsf{X}$ for "x is an element of X", $\mathsf{X} \subset \mathsf{Y}$ (or $\mathsf{Y} \supset \mathsf{X}$) for "$\mathsf{X}$ is a subset of Y", $\mathsf{X} = \mathsf{Y}$ for "$\mathsf{X} \subset \mathsf{Y}$ and $\mathsf{Y} \subset \mathsf{X}$".

Orderings. An ordering on a set X is a binary relation on X, denoted by $\leq$, which is reflective, transitive, and antisymmetric ($x \leq y$ and $y \leq x$ imply $x = y$). The set X endowed with an ordering is called an *ordered set.* We write $y \geq x$ if $x \leq y$.

Mappings. A mapping (or map) f of a subset D_f of X into Y is denoted by $f : \mathsf{X} \to \mathsf{Y}$. D_f is called the *domain* of f, $\mathsf{Im}\,(f)$ the *image* of f, and $\mathsf{Ker}\,(f) = \{x : f(x) = 0\}$ the *kernel* of f.

Vector Spaces. Let L be a set, K a field. If there are defined an addition $(x, y) \to x + y$ of $\mathsf{L} \times \mathsf{L}$ into L, and a multiplication $(\lambda, x) \to \lambda x$ of $\mathsf{K} \times \mathsf{L}$ into L, that satisfies the well-known eight axioms, then L is a vector space over K. See for example [**40**, p.9].

From now on $\mathsf{K} = \mathbb{C}$, or $\mathsf{K} = \mathbb{R}$.

Ordered Vector Space. A vector space L (over K), with ordering $\leq$, is said to be *ordered*, if the ordering is compatible with its vector structure:

(1) $x \leq y$ implies $x + z \leq y + z$ for all $x, y, z \in \mathsf{L}$,
(2) $x \leq y$ implies $\alpha x \leq \alpha y$ for all $x, y \in \mathsf{L}$ and $\alpha \in \mathsf{K}, \alpha > 0$.

See for example [**40**, p.204].

Norm. A norm $\|\cdot\|$ on a vector space L over K is a K-valued function with the following properties:

(1) $\|\alpha x\| = |\alpha|\|x\|$ for all $\alpha \in \mathsf{K},\ x \in \mathsf{L}$,
(2) $\|x+y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathsf{L}$,
(3) $\|x\| = 0$ implies $x = 0$ and $\|x\| \geq 0$ for all $x \in \mathsf{L}$.

Inner product. Let $\bar{\alpha}$ denote the K conjugate of $\alpha \in \mathsf{K}$. Let L be a vector space over K. An inner product on L is a K-valued function $\langle \cdot, \cdot \rangle$ defined on $\mathsf{L} \times \mathsf{L}$ with the following properties:

(1) $\langle x+y, z\rangle = \langle x, z\rangle + \langle y, z\rangle$ and $\langle \alpha x, y\rangle = \alpha\langle x, y\rangle$ for all $\alpha \in \mathsf{K},\ x, y, z \in \mathsf{L}$,
(2) $\langle x, y\rangle = \overline{\langle y, x\rangle}$ for all $x, y \in \mathsf{L}$.
(3) $\langle x, x\rangle = 0$ implies $x = 0$ and $\langle x, x\rangle \geq 0$ for all $x \in \mathsf{L}$.

If $\langle \cdot, \cdot \rangle$ is an inner product on L then $\|\cdot\|$, defined by $\|x\| = \langle x, x\rangle^{\frac{1}{2}}$, is a norm on L.

One of the most useful relations between the norm and the inner product is the *Cauchy-Schwarz inequality*,

$$|\langle x, y\rangle| \leq \|x\|\|y\|. \tag{1}$$

1.1.2. Vectors

For $\mathsf{K} = \mathbb{R}$ or $\mathsf{K} = \mathbb{C}$, let L denote an n dimensional Euclidean space over K, that is $\mathsf{L} = \mathsf{K}^n$. An element

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathsf{L}$$

is called a *vector*. The *adjoint* or *transpose* or *conjugate transpose* x^* of x is defined as

$$x^* = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}^* = (\bar{x}_1, \ldots, \bar{x}_n),$$

and one has $(x^*)^* = x$. For all $x, y \in \mathsf{L}$, with $x = (\bar{x}_1, \ldots, \bar{x}_n)^*$ and $y = (\bar{y}_1, \ldots, \bar{y}_n)^*$,

$$\langle x, y\rangle := y^* x = \sum_{i=1}^{n} \bar{y}_i x_i, \tag{2}$$

defines the standard inner product. The inner product induces a norm on L:

$$\|x\| = \langle x, x\rangle^{\frac{1}{2}}.$$

Without further indication we will use this inner product and norm on Euclidean spaces.

A vector x is called *orthogonal* to a vector y if $\langle x, y\rangle = 0$. A vector x is called *normal* or *unitary* if $\|x\| = 1$.

Denote by $\mathbb{R}_+$ the set of nonnegative real numbers and $\mathbb{R}_{++}$ the set of positive real numbers. In $\mathbb{R}^n$ there exists a standard ordering. Take $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$, then

$$x \leq y \quad \Leftrightarrow \quad x_i \leq y_i \quad \forall i \in \{1, \ldots, n\}.$$

We will also consider a standard partial ordering

$$x < y \quad \Leftrightarrow \quad (x_i \leq y_i,\ x_i \neq y_i) \quad \forall i \in \{1, \ldots, n\}.$$

By e_i we will denote the i-th standard basis element of L, that is,

$$e_i = (0, \dots, 0, 1, 0, \dots, 0)^*,$$

where 1 is situated on the i-th coordinate of e_i. By $e \in \mathsf{L}$ we will denote the vector of n ones, that is,

$$e = (1, 1, \dots, 1)^*.$$

1.1.3. Matrices

Capitals will denote matrices. We denote by $\mathbb{R}^{m\times n}$ the space of real $m \times n$ matrices and by $\mathbb{C}^{m\times n}$ the space of complex $m \times n$ matrices. Matrices are usually engaged in multiplying vectors. If F is an $m \times n$ matrix and $u \in \mathsf{K}^n$ then $Fu \in \mathsf{K}^m$, so F transforms K^n into K^m. The transformation is linear, that is, $F(\alpha x + \beta y) = \alpha Fx + \beta Fy$, and any linear transformation of K^n into K^m can be represented by some $m \times n$ matrix F. The *adjoint* F^* of F is an $n \times m$ matrix that is uniquely defined by

$$\langle x, F^* u\rangle = \langle Fx, u\rangle, \quad \forall x \in \mathsf{K}^n,\ u \in \mathsf{K}^m.$$

Whenever the product FG is defined, then G^*F^* is defined and

$$(FG)^* = G^*F^*.$$

A matrix (operator) M is called *self-adjoint* if

$$M = M^*.$$

If a self-adjoint M is an element of $\mathbb{R}^{m\times m}$ then M is called *symmetric.* If M is an element of $\mathbb{C}^{m\times m}$ then M is called *Hermitian.* The set of $m\times m$ self-adjoint matrices will be denoted by S^m

An $m \times n$ matrix P with $m \geq n$ is called *orthonormal* if its columns are orthonormal, that is, if

$$P^*P = I.$$

Square orthonormal matrices are called *unitary.* The inner product formula (2) is preserved under the action of orthonormal matrices:

$$\langle Px, Pu\rangle = u^*P^*Px = u^*x = \langle x, u\rangle.$$

Suppose B is an $n \times n$ matrix. Then any element $z \neq 0$ that satisfies

$$Bz = \lambda z,$$

is called an *eigenvector* of B and λ is called an *eigenvalue* of B. The set of eigenvalues of B is called the *spectrum* of B. All eigenvalues of a self-adjoint matrix are real. As a result we may label the eigenvalues of a self-adjoint matrix M in decreasing order:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

The jth largest eigenvalue of M is denoted by $\lambda_j(M)$. Any normalized eigenvector associated with λ_i is denoted by z_i, $Mz_i = \lambda_i z_i$.

A square matrix B is called *invertible* if there exists a matrix C such that

$$BC = CB = I.$$

We will call such a matrix C the inverse of B and denote it by B^{-1}.

For $E, B \in \mathsf{K}^{n\times m}$ the following inner product is defined,

$$\langle E, B\rangle = \operatorname{trace}(B^* E) = \sum_{i=1}^{n}\sum_{j=1}^{m} \bar{b}_{ji} e_{ij},$$

with $(e_{ij})_{ij} := E$ and $(b_{ij})_{ij} := B$. The *trace* of a square matrix is the sum of the diagonal elements. The norm arising from the (trace) inner product is known as the *Frobenius norm.*

$$\|C\| := \|C\|_F = \langle C, C\rangle^{\frac{1}{2}}.$$

These will be our norm and inner product of choice for matrices unless otherwise stated.

A matrix A is called *positive semidefinite* ($A \geq 0$), if A is self-adjoint and

$$\langle x, Ax\rangle \geq 0 \quad \forall x \in \mathsf{K}^n.$$

If in addition A satisfies

$$\langle x, Ax\rangle > 0 \quad \forall x \neq 0,$$

then A is *positive definite* ($A > 0$). The subset of positive semidefinite $n\times n$ matrices is denoted by S^n_+ and the positive definite ones by S^n_{++}. On the space S^n we will use the ordering induced by the notion of (semi)definiteness, that is for $A, X \in \mathsf{S}^n$,

$$\begin{array}{lcl} A \geq X & \Leftrightarrow & A - X \in \mathsf{S}^n_+, \quad \text{and} \\ A > X & \Leftrightarrow & A - X \in \mathsf{S}^n_{++}. \end{array}$$

1.1.4. Matrix factorizations

A matrix $L := (l_{ij})_{ij}$ for which

$$l_{ij} = 0, \qquad \forall i < j,$$

is called a lower triangular matrix. Also, a matrix $U := (u_{ij})_{ij}$ for which

$$u_{ij} = 0, \qquad \forall i > j,$$

is called an upper triangular matrix. A matrix B has an *LU factorization* if there exists a lower triangular matrix L with $l_{ii} = 1$, for all i, and an upper triangular matrix U, such that

$$B = LU.$$

If L and U exist then L and U can be computed by Gauss elimination.

If A is positive definite, then there exists an unique lower triangular matrix L with positive diagonal elements, such that

$$A = LL^*.$$

Such a factorization is called the *Cholesky factorization.*

Any $m \times n$ matrix B can be written as

$$B = QR,$$

with Q an $m \times r$ matrix satisfying $Q^*Q = I$, and with R an upper triangular $r \times n$ matrix with nonnegative diagonal elements, with r the rank of the matrix B. Both Q and R are unique when B has full rank. This factorization is called the *QR factorization.* The QR factorization is the matrix formulation of the Gram-Schmidt orthogonalization process for the columns of B. When B has full rank, then $L := R^*$ is the Cholesky factor of B^*B since

$$LL^* = R^*R = R^*Q^*QR = B^*B.$$

For more information on the QR factorization see, for example, [**36**, p.98].

A self-adjoint $n\times n$ matrix A is *similar* to a diagonal matrix Λ via an orthonormal factorization,

$$A = Z\Lambda Z^* \quad = \quad \sum_{i=1}^{n} \lambda_i z_i z_i^*, \tag{3}$$

and

$$I = ZZ^* \quad = \quad \sum_{i=1}^{n} z_i z_i^*. \tag{4}$$

$Z = (z_1, \ldots, z_n)$ consists of orthonormal vectors that are eigenvectors of A. The diagonal matrix

$$\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

consists of the eigenvalues λ_i of A. This factorization is called the *eigenvector decomposition* of A, see for example [**36**, p.7].

Let N be an $(m+n)\times(m+n)$ block matrix

$$N = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

with $A \in \mathsf{K}^{m\times m}$ invertible, $B \in \mathsf{K}^{m\times n}$, $C \in \mathsf{K}^{n\times m}$ and $D \in \mathsf{K}^{n\times n}$ then,

$$S := D - CA^{-1}B$$

is called the *Schur complement* of A in N, and

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}.$$

The following properties holds if A^{-1} exists and $A > 0$,

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} > 0 \quad \Leftrightarrow \quad C = B^*,\ D = D^* \quad \text{and} \quad S > 0,$$

as well as,

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \geq 0 \quad \Leftrightarrow \quad C = B^*,\ D = D^* \quad \text{and} \quad S \geq 0.$$

Thus the positive (semi) definiteness of a matrix is related to the positive (semi) definiteness of its Schur complement.

1.1.5. Convex sets and cones

Suppose $x_1,\ x_2 \in \mathbb{R}^n$. Then the *line* through these elements can be represented as $x_1 + \tau(x_2 - x_1),\ \tau \in \mathbb{R}$. For values of $\tau \in [0, 1]$ this parameterization corresponds to the *line segment* between x_1 and x_2.

A set $\mathsf{S} \subseteq \mathbb{R}^n$ is said to be *affine* if the line through any two distinct elements in S is contained in S, that is,

$$\forall \tau \in \mathbb{R},\ \forall x_1, x_2 \in \mathsf{S} \quad \Rightarrow \quad x_1 + \tau(x_2 - x_1) \in \mathsf{S}.$$

An element of the form $\tau_1 x_1 + \ldots + \tau_r x_r$ with $\sum_{i=1}^r \tau_i = 1$ is called an *affine combination* of the elements $x_1, \ldots, x_r$.

A set C is said to be *convex* if the line segment through any two elements in C is in C, that is,

$$\forall \tau \in [0, 1],\ x_1, x_2 \in \mathsf{C} \quad \Rightarrow \quad x_1 + \tau(x_2 - x_1) \in \mathsf{C}.$$

Obviously, every affine set is convex. An element of the form $\tau_1 x_1 + \cdots + \tau_r x_r$ with $\sum_{i=1}^r \tau_i = 1$ and $\tau_i \geq 0$ for $i = 1, \ldots, r$ is called a *convex combination* of the elements $x_1, \ldots, x_r$.

A set S is called a *cone* if

$$\forall \tau \geq 0,\ x \in \mathsf{S} \quad \Rightarrow \quad \tau x \in \mathsf{S}.$$

A cone C is convex if

$$\forall \tau_1, \tau_2 \geq 0,\ x_1, x_2 \in \mathsf{C} \quad \Rightarrow \quad \tau_1 x_1 + \tau_2 x_2 \in \mathsf{C}.$$

$\mathbb{R}^n_+$ is a convex cone. The set of positive semidefinite matrices is a convex cone, since if $A \geq 0$ and $B \geq 0$ then $\tau_1 A + \tau_2 B \geq 0$ for all $\tau_1, \tau_2 \geq 0$ because of

$$\langle x, (\tau_1 A + \tau_2 B)x \rangle = \tau_1 \langle x, Ax \rangle + \tau_2 \langle x, Bx \rangle \geq 0.$$

An element of the form $\tau_1 x_1 + \cdots + \tau_r x_r$ with $\tau_1, \cdots, \tau_r \geq 0$ is called a *conic combination* of the elements $x_1, \ldots, x_r$. If x_i, for $i = 1, \ldots, r$, is in a convex cone C then every conic combination of these x_i is in C as well.

Let C be a convex cone. Then the set

$$\mathsf{C}^* := \{ y \mid \langle x, y \rangle \geq 0,\ \forall x \in \mathsf{C} \}$$

is called the *dual cone* of C. A cone is *self-dual* if $\mathsf{C} = \mathsf{C}^*$.

$\mathbb{R}^n_+$ is self-dual. The cone S^n_+ of positive semidefinite matrices is self-dual:

$$\Big(\langle X, Y \rangle \geq 0,\ \forall X \geq 0 \Big) \quad \Leftrightarrow \quad Y \geq 0, \tag{5}$$

since if $Y \not\geq 0$ then there exists an x with $\langle x, Yx \rangle = \langle xx^*, Y \rangle < 0$. Hence, there exists a matrix $X = xx^*$ with $\langle X, Y \rangle < 0$ and that shows that $Y \geq 0$ is a necessary condition for equation (5). Second, $X \geq 0$ if and only if $X = LL^*$, thus

$$\langle X, Y \rangle = \operatorname{trace}(L^* Y L) = \sum_{i=1}^n e_i L^* Y L e_i \geq 0 \quad \text{if} \quad Y \geq 0,$$

which means that $Y \geq 0$ is also a sufficient condition.

For more information on convexity, see for example [**7**].

1.1.6. Functions and derivatives

Let $f : \mathsf{V} \to \mathsf{W}$, where V and W are finite dimensional vector spaces with $\dim \mathsf{V} = n$ and $\dim \mathsf{W} = m$. The function f is said to be (Fréchet) *differentiable* at $x \in \mathsf{D}_f$ if D_f is a neighborhood of x and there exists a linear transformation $D : \mathsf{V} \to \mathsf{W}$ such that, $\mathsf{D}_D = \mathsf{V}$, and

$$\lim_{\|h\| \to 0} \frac{\|f(x+h) - f(x) - Dh\|}{\|h\|} = 0.$$

The transformation D is called the derivative of f at x. We say that f is *differentiable* if f is differentiable at all $x \in \mathsf{D}_f$.

If $\mathsf{W} = \mathbb{R}$ then $g \in \mathsf{V}$, such that $Dh = \langle g, h \rangle$, is called the *gradient* of f at x. The gradient of f at x is often denoted as $\nabla f(x)$. The partial derivative with respect to x_i is denoted by

$$\frac{\partial}{\partial x_i} f \quad \text{and is defined by} \quad \frac{\partial}{\partial x_i} f = \langle \nabla f, e_i \rangle, \quad \forall i = 1, \ldots, n.$$

The affine function of y, given by

$$f(x) + \langle \nabla f(x), (y - x) \rangle,$$

is called the *first order (Taylor) approximation* of f at x.

If $\mathsf{V} = \mathbb{R}^n$ and $W = \mathbb{R}^m$, then D is an $m \times n$ matrix. We also use the notation $\nabla f(x)$ for D^* at x, and also call it the gradient of f at x.

The function $f : \mathsf{V} \to \mathbb{R}$ is said to be *twice differentiable* at $x \in \mathsf{D}_f$ if f is differentiable at a neighborhood of x, ∇f is continuous in x, and $\exists H : \mathsf{V} \to \mathsf{V}$ such that

$$\lim_{\|h\| \to 0} \frac{\|\nabla f(x+h) - \nabla f(x) - Hh\|}{\|h\|} = 0.$$

If H exists, then it is called the *Hessian* of f at x; H is often denoted as $\nabla^2 f(x)$. The *second order (Taylor) approximation* of f at x in y is

$$f(x) + \langle \nabla f(x), (y - x) \rangle + \frac{1}{2} \langle \nabla^2 f(x)(y - x), (y - x) \rangle. \tag{6}$$

In our applications $\mathsf{V} = \mathbb{R}^n$ or $\mathsf{V} = \mathsf{S}^n$, that is, $f : \mathbb{R}^n \to \mathbb{R}$ or $f : \mathsf{S}^n \to \mathbb{R}$. As an example consider the function $f : \mathsf{S}^n \to \mathbb{R}$ is defined by

$$f(X) = \ln \det(X).$$

Clearly f is smooth on S^n_{++}. We determine the derivative as follows:

$$\begin{array}{rcl} f(X+E) & = & \ln \det \Big(X(I + X^{-1}E) \Big) \\ & = & \ln \det(X) + \ln(1 + \operatorname{trace}\big(X^{-1}E\big) + \mathcal{O}(\|E\|^2)) \\ & = & f(X) + \langle X^{-1}, E \rangle + \mathcal{O}(\|E\|^2), \end{array}$$

and we see that $\nabla f(X) = X^{-1}$. Similarly we find the second derivative:

$$\begin{array}{rcl} \nabla f(X+E) & = & \Big(X(I + X^{-1}E) \Big)^{-1} \\ & = & \Big(I - X^{-1}E + F) \Big) X^{-1}, \quad \text{with} \quad \|F\| = \mathcal{O}(\|E\|^2) \\ & = & \nabla f(X) - X^{-1}EX^{-1} + \widehat{F}, \quad \text{with} \quad \|\widehat{F}\| = \mathcal{O}(\|E\|^2). \end{array}$$

Hence, $\nabla^2 f(X)H = -X^{-1}HX^{-1}$.

1.1.7. Convex functions

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be *convex* if

(1) D_f is a convex set and
(2) $\forall x_1, x_2 \in \mathsf{D}_f$ and $\tau \in [0,1]$:

$$f(x_1 + \tau(x_2 - x_1)) \leq f(x_1) + \tau(f(x_1) - f(x_2)).$$

If the inequality is strict for $\tau \in (0,1)$ and $x_1 \neq x_2$, then f is called *strict convex.* We say that f is *concave* if $-f$ is convex.

Suppose that f is differentiable. The gradient of f is a vector in $\mathbb{R}^n$. Then f is convex if and only if D_f is convex and

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \quad \forall x, y \in \mathsf{D}_f.$$

In addition, suppose that f is twice differentiable. Then f is convex if and only if $\nabla^2 f(x) \geq 0$, that is, $\nabla^2 f(x)$ is positive semidefinite.

For more about convex functions, see for example [**37**].

1.1.8. The Newton method

Suppose we want to solve equation

$$h(x) = 0, \tag{7}$$

with $h : \mathbb{R}^n \to \mathbb{R}^n$. Let x^0 be an element in the neighborhood of the solution of equation (7). The following process solves equation (7) approximately. Make a linear approximation to the function h in the element x^0 :

$$h_0(x) = h(x^0) + \langle \nabla h(x^0), (x - x^0) \rangle. \tag{8}$$

The equation $h_0(x) = 0$ is now an approximate equation with solution

$$x^1 = x^0 - (\nabla h(x^0)^*)^{-1} h(x^0).$$

If this process is repeated then the sequence (x^i) follows, where

$$x^{i+1} = x^i - (\nabla h(x^i)^*)^{-1} h(x^i). \tag{9}$$

This process is known as the *Newton(-Raphson) method* and

$$\Delta x^i := x^{i+1} - x^i = -(\nabla h(x^i)^*)^{-1} h(x^i)$$

is called the Newton direction at x^i.

We will use such a Newton approach in the following situation. Suppose we want to minimize a convex function $f : \mathbb{R}^m \to \mathbb{R}$. Determining the minimum is equivalent to solving $\nabla f(x) = 0$, if f is differentiable and the minimum exists, and we apply Newton's scheme to this equation. For ease of reference we give some details.

Take a second order approximation (6) of f at x^0 in x,

$$f_0(x) := f(x^0) + \langle \nabla f(x^0), x - x^0 \rangle + \frac{1}{2} \langle \nabla^2 f(x^0)(x - x^0), (x - x^0) \rangle,$$

and solve $\nabla f_0(x) = 0$:

$$\nabla f_0(x) = \nabla f(x^0) + \nabla^2 f(x^0)(x - x^0) = 0. \tag{10}$$

Notice the similarity between (10) and (8). Henceforth, assume that $\nabla^2 f(x^0)$ is positive definite. Thus f^0 is strictly convex and is minimized by the element x^1 satisfying $\nabla f(x^0) + \nabla^2 f(x^0)(x^1 - x^0) = 0$, or

$$x^1 := x^0 - \left(\nabla^2 f(x^0)\right)^{-1} \nabla f(x^0).$$

We will also use this approach for functions in a space of matrix functions.

1.1.9. Barrier functions

Suppose we want to minimize a function $f : \mathsf{V} \to \mathsf{W}$ subject to some constraint $x \geq 0$, with V and W ordered vector spaces. We have seen how a Newton method works for an unconstrained problem in $\mathbb{R}^n$. Now we explain how to solve this minimization problem subject to a constraint. First rewrite the problem as

$$\text{minimize} \qquad f(x) + \operatorname{ind}(x),$$

where $\operatorname{ind}(x) : \mathsf{V} \to \mathsf{W}$ is the indicator function of the constraint set, that is,

$$\operatorname{ind}(x) = \begin{cases} 0 & \text{if} \quad x \geq 0, \\ +\infty & \text{otherwise.} \end{cases}$$

We would like to construct an algorithm that computes a sequence of points,

$$x^0, x^1, \ldots \in \{x | x > 0\},$$

converging towards x^+ that minimizes the optimization problem. We expect to find the minimum at the boundary of the set $\{x | x \geq 0\}$. The basic idea of barrier methods is to approximate the indicator function with another function $j : \mathsf{V} \to \mathsf{W}$ such that

- j is smooth and convex,
- $\mathsf{D}_j = \{x | x > 0\}$,
- If $x^k \in \mathsf{D}_j$ and (x^k) converges to the boundary of D_j, then $(j(x^k))$ converges to infinity, that is, $j(x)$ grows without bound as x approaches the boundary of the set $\{x | x \geq 0\}$.

Using the barrier function j, for any $\mu > 0$, we consider the problem

$$\text{minimize} \qquad f(x) + \mu j(x). \tag{11}$$

The closer μ is to zero the better μ times the barrier function $j(x)$ approximates $\operatorname{ind}(x)$. One can solve the original problem to any required accuracy solving the unconstrained problem (11) by, for example, a Newton method [**7**, Ch. 6]. A popular barrier function, for $\mathsf{V} = \mathbb{R}^m$ and $\mathsf{W} = \mathbb{R}$, is the logarithmic barrier function, defined as

$$j(x) := \begin{cases} -\sum_{i=1}^m \ln(x_i) & \text{if} \quad x_i > 0, \quad \text{for all } i, \\ +\infty & \text{otherwise.} \end{cases}$$

A popular barrier function, for $\mathsf{V} = \mathsf{S}^n$ and $\mathsf{W} = \mathbb{R}$, is a generalization of the logarithmic barrier function for $\mathsf{V} = \mathbb{R}^m$. It is defined by

$$j(X) := \begin{cases} -\ln\det(X) & \text{if} \quad X > 0, \\ +\infty & \text{otherwise.} \end{cases}$$

These logarithmic barrier functions have properties that can be exploited in the convergence analysis of the Newton method, see for example [**31**].

1.2. Convex programming problems

In this section we will formulate the type of *convex programming problems* that will play a role in the following chapters. A general framework will be given, followed by linear programming problems and semidefinite programming problems. We discuss some practical applications formulated as semidefinite programming problem to show the significance of semidefinite programming problems, and we introduce some problems that later on will be used as test problems for the algorithms in Chapter 3. Usually, we cannot find an analytical solution for the formulated optimization problems, so the problem has to be solved by an iterative algorithm. This is an algorithm that computes a sequence of points $x^0, x^1, \dots$ that converges towards a solution of the optimization problem.

1.2.1. General framework

We consider the following formulation of a convex programming problem. Suppose that we have convex functions f, h with $f : \mathbb{R}^m \to \mathbb{R}$ and $h : \mathbb{R}^m \to \mathbb{R}^n$. We want to solve the following minimization problem.

$$\begin{array}{ll} \min & f(y) \\ \text{subject to} & h(y) \leq 0. \end{array} \tag{12}$$

The function f is called the *objective function* of (12), and $h(y) \leq 0$ represent the *constraints*. If y satisfies $h(y) \leq 0$ then y is said to be *feasible* for (12). The set of feasible elements is the *feasible set* of (12) and y is called *strict feasible* for (12) if $h(y) < 0$. We say that (12) is *solvable* if there exists a y^+ that solves (12). We call y^+ a *solution* of (12) or a *minimizer* of (12) and $f(y^+)$ the *optimal value* of (12).

To solve this minimization problem we eliminate the constraints. First we introduce a so-called vector of *slack* variables $s \in \mathbb{R}^n$ to simplify the inequality constraint in problem (12):

$$\begin{array}{ll} \min & f(y) \\ \text{subject to} & h(y) + s = 0 \\ & s \geq 0. \end{array} \tag{13}$$

Then we place the vector of slack variables via a barrier function in the objective, and this results in the following problem, for $s > 0$:

$$\begin{array}{ll} \min & f(y) - \mu \sum_{i=1}^{n} \ln(s_i) \\ \text{subject to} & h(y) + s = 0. \end{array} \tag{14}$$

By letting μ go to zero, a family of approximate solutions y_μ to (14) converges to a minimizer of the original convex programming problem (12).

The Lagrangian is introduced to characterize a solution of the barrier problem (14). The *Lagrangian* of (14) is

$$\mathcal{L}(x, y, s) := f(y) - \mu \sum_{i=1}^{n} \ln(s_i) + \langle x, h(y) + s \rangle, \tag{15}$$

where $x_i \geq 0,\ i = 1, \dots, n$, are the *Lagrange multipliers* associated with the equality constraints. When looking for a minimum of a function, subject to some equality constraints, one should add to the function the constraints multiplied by undetermined multipliers and seek for the minimum of the resulting sum as if the variables were independent. The resulting equations combined with the constraint equations, will serve to determine all unknowns [**27**]. This rule is called the *Lagrange multiplier rule.* If we express this rule in mathematical formulas, we obtain the following conditions for differentiable functions f and h. The first order optimality conditions for minimizing $\mathcal{L}$ for non-trivial $x, s \geq 0$ are

$$h(y) + s = 0, \qquad (\nabla_x \mathcal{L}(x,y,s) = 0) \tag{16}$$

$$\nabla f(y) + \langle x, \nabla h(y)\rangle = 0, \qquad (\nabla_y \mathcal{L}(x,y,s) = 0) \tag{17}$$

$$x_i s_i - \mu = 0, \quad i = 1, \ldots, n. \quad (\nabla_s \mathcal{L}(x,y,s) = 0) \tag{18}$$

These conditions are also called the *Karush-Kuhn-Tucker (KKT) conditions*, see [**21**]. We will rewrite equation (18) as

$$XSe - \mu e = 0, \quad \text{where} \quad X := \operatorname{diag}(x),\ S := \operatorname{diag}(s) \quad \text{and} \quad e := (1, \cdots, 1)^*.$$

Here $\operatorname{diag}(x)$ is the diagonal matrix with elements $x_i,\ i = 1, \ldots, n$.

If $(x_\mu^+, y_\mu^+, s_\mu^+)$ satisfies conditions (16)-(18), and $x_\mu^+, s_\mu^+ > 0$ then, for $\mu \to 0$, it converges to (x^+, y^+, s^+), where (y^+, s^+) is a solution of optimization problem (13), [**21**, p. 12].

The formulation (12) is a simplification of a general convex programming problem since it does not include equality constraints. In fact the method described in this section and in the next chapter, can easily be extended to all cases. For example, an equality constraint

$$g(y) = 0,$$

for $g : \mathbb{R}^m \to \mathbb{R}^n$, leads to an extra constraint $g(y) = 0$ in (14), an extra term $\langle z, g(y)\rangle$ in the Lagrangian (15), and extra terms in the optimality conditions.

A general framework for matrix spaces (where $\mathbb{R}^m$ and $\mathbb{R}^n$ are replaced by S^m and S^n) is an analogue of this general framework with appropriate ordering, inner product and barrier function.

The part I omitted here and will omit for linear and semidefinite programming as well, are the assumptions that have to be made to find an optimal solution when incorporation (16)-(18). First of all the objective function has to be convex with continuous first and second order derivatives in the interior of the feasible set. The interior of the feasible set has to be nonempty and bounded, as a result (16)-(18) has to be solvable.

1.2.2. Linear programming problems

A convex programming problem is said to be linear if [**49**]:

(1) the objective function f is linear,
(2) the inequality and equality constraints are linear.

The standard form of a *linear programming problem* (LP) is as follows:

$$\text{(p)} \quad \begin{array}{ll} \min & \langle c, x\rangle \\ \text{subject to} & Ax = b \\ & x \geq 0, \end{array}$$

where $c \in \mathbb{R}^n,\ b \in \mathbb{R}^m$, and A an $m \times n$ matrix are given and we have to minimize for $x \in \mathbb{R}^n$. The standard linear programming problem is called the *primal* linear programming problem.

With the standard linear programming problem (p) a *dual* linear programming problem (d) is associated

$$\text{(d)} \qquad \begin{array}{ll} \max & \langle b, y\rangle \\ \text{subject to} & A^*y \le c, \end{array}$$

where we have to minimize for $y \in \mathbb{R}^m$.

A standard assumption for a linear programming problem to be solvable is that $b \perp \mathsf{Ker}\,(A^*)$, because if $b \not\perp \mathsf{Ker}\,(A^*)$ then there is not a finite maximum for (d). Another standard assumption is that $\mathsf{Im}\,(A) = \mathbb{R}^m$.

The dual linear programming problem is of standard form (12), but due to historical reasons (p) is called the standard form.

There exists a duality theory that explains the relationship between (p) and (d), see [**8**] or more recently [**38**]. For example, for any feasible x for (p) and y for (d), we have that

$$\langle b, y\rangle = \langle Ax, y\rangle = \langle x, A^*y\rangle \le \langle c, x\rangle. \tag{19}$$

Hence, any feasible y gives a lower bound $\langle b, y\rangle$ for $\langle c, x\rangle$ and any feasible x an upper bound $\langle c, x\rangle$ for $\langle b, y\rangle$. Moreover, if x^+ solves (p) and y^+ solves (d), then we have $\langle b, y^+\rangle = \langle c, x^+\rangle$. The nonnegative difference $\langle c, x\rangle - \langle b, y\rangle$ between the primal objective value at a primal feasible x and the dual objective value at a dual feasible y is called the *duality gap* for x and y. Dantzig initiated the field of linear programming just after World War II, by formulating linear programming problems and introducing the simplex method for solving them [**8**]. The simplex method is out of scope of this thesis, we will concentrate on the more recently introduced Newton methods in convex optimization.

Primal-dual algorithm. We will first review a framework for the *primal-dual algorithm*, which can be derived from either the primal (p) or the dual programming problem (d), in combination with the primal or dual logarithmic barrier function. Here, we will derive it from the dual linear programming problem (d).

In view of the exposition in the Subsection 1.2.1, we use a vector of slack variables s to simplify the inequality constraint, such that (d) leads to

$$\begin{array}{ll} \max & \langle b, y\rangle \\ \text{subject to} & A^*y + s = c \\ & s \ge 0. \end{array} \tag{20}$$

For this formulation we introduce a barrier function such that for $s > 0$ the dual program (d) leads to the following family of problems associated with (d)

$$\begin{array}{ll} \max & \langle b, y\rangle - \mu \sum_i \ln(s_i) \\ \text{subject to} & A^*y + s = c, \end{array} \tag{21}$$

with $(y^+_\mu, s^+_\mu) \to (y^+, s^+)$ if $\mu \to 0$, where (y^+_μ, s^+_μ) is a solution of (21) and (y^+, s^+) is a solution of (20).

To characterize a solution of (21) the Lagrangian is defined as,

$$\mathcal{L}_{\mathrm{pd}}(x,y,s) := \langle b,y\rangle - \mu\sum_i \ln(s_i) + \langle x,(c-A^*y-s)\rangle. \tag{22}$$

Then the first order optimality conditions for a minimum are, for $x, s > 0$:

$$A^*y + s - c = 0, \qquad (\nabla_x \mathcal{L}_{\mathrm{pd}}(x,y,s) = 0) \tag{23}$$

$$b - Ax = 0, \qquad (\nabla_y \mathcal{L}_{\mathrm{pd}}(x,y,s) = 0) \tag{24}$$

$$x_i s_i - \mu = 0, \quad i = 1,\dots,n. \quad (\nabla_s \mathcal{L}_{\mathrm{pd}}(x,y,s) = 0) \tag{25}$$

If $(x^+_\mu, y^+_\mu, s^+_\mu)$ satisfies conditions (23), (24), and (25), then (y^+, s^+) solves (21) and for $\mu \to 0$ converges to (x^+, y^+, s^+), where (y^+, s^+) is a solution of optimization problem (20) and x^+ solves (p).

The primal-dual algorithm is an iterative method for computing a solution of (p) and (d). It assumes that a feasible (x, y, s) with $x, s > 0$ is known, and takes one Newton iteration for the first order conditions (23), (24), (25), for a fixed value of μ, to find a new triple (x, y, s) which retains feasibility, after some scaling of the Newton directions, while reducing the duality gap:

$$\langle c,x\rangle - \langle b,y\rangle = \langle x,c\rangle - \langle Ax,y\rangle = \langle x, c - A^*y\rangle = \langle x,s\rangle = n\mu.$$

Then the value of μ is reduced and the process is repeated until μ is small enough. This approach will be explained in more detail in Subsection 2.1.2.

Dual algorithm and primal algorithm formulations also exist.

Dual algorithm. The dual algorithm can be derived from the dual problem (d) with a logarithmic barrier formulation slightly different from the one given in (21). The algorithm only iterates and updates the dual variable y and does **not** use a vector of slack variables s.

In order to eliminate the inequality constraints, a logarithmic barrier function is introduced such that the dual problem (d) for $c - A^*y > 0$ leads to

$$\max \quad \langle b,y\rangle + \mu\sum_i \ln(c_i - (A^*y)_i), \tag{26}$$

with $y^+_\mu \to y^+$ if $\mu \to 0$, where y^+_μ is a solution of (26) and y^+ is a solution of (d).

To characterize a solution of (26), a Lagrangian is defined as

$$\mathcal{L}_{\mathrm{d}}(y) := \langle b,y\rangle + \mu\sum_i \ln(c_i - (A^*y)_i).$$

Then the first order optimality condition for minimizing $\mathcal{L}_{\mathrm{d}}$ is:

$$b - \mu\sum_i Ae_i(c_i - (A^*y)_i)^{-1} = 0. \qquad (\nabla_y \mathcal{L}_{\mathrm{d}}(y) = 0) \tag{27}$$

If y^+_μ satisfies conditions (27), then for $\mu \to 0$ it converges to y^+, where y^+ is a solution of optimization problem (d).

The dual algorithm is an iterative method for computing a solution of (d). The procedure is similar to the procedure for the primal-dual algorithm, but here for feasible y and condition (27). This approach will be explained in more detail in Subsection 2.1.4.

Note that if we write $s := c - A^*y$ and $x_i := s_i^{-1}\mu$, and y solves (27), then x, y and s are such that (23)-(25) hold. However, in the primal-dual and dual algorithms the optimality conditions are not solved to full accuracy; only one modified Newton step is applied, and then the primal and primal-dual algorithms are in general not equivalent.

Primal algorithm. The primal algorithm will be derived from the primal problem (p) with a logarithmic barrier function. The algorithm iterates and updates only the primal variable x.

In order to eliminate the constraint $x \geq 0$, we introduce a logarithmic barrier function such that (p) for $x > 0$, leads to

$$\begin{array}{ll} \min & \langle c, x\rangle - \mu \sum_i \ln(x_i) \\ \text{subject to} & Ax = b, \end{array} \tag{28}$$

with $x_\mu^+ \to x^+$ if $\mu \to 0$, where x_μ^+ is a solution of (28) and x^+ is a solution of (p).

To characterize a solution, a Lagrangian for (28) is introduced.

$$\mathcal{L}_\mathrm{p}(x, y) := \langle c, x\rangle - \mu \sum_i \ln(x_i) + \langle y, b - Ax\rangle.$$

The first order optimality conditions for minimizing $\mathcal{L}_\mathrm{p}$ are:

$$c_i - \mu(x_i)^{-1} - (A^*y)_i = 0, \quad i = 1, \ldots, n, \quad (\nabla_x \mathcal{L}_\mathrm{p}(x, y) = 0) \tag{29}$$

$$Ax - b = 0. \qquad (\nabla_y \mathcal{L}_\mathrm{p}(x, y) = 0) \tag{30}$$

If (x_μ^+, y_μ^+) satisfies conditions (29) and (30) then for $\mu \to 0$ it converges to (x^+, y^+), where x^+ is a solution of optimization problem (p).

The primal algorithm is an iterative method for computing a solution of (p). Again we follow the same procedure as for the primal-dual algorithm, but here for feasible $x > 0$ and conditions (29)-(30). This approach will be explained in more detail in Subsection 2.1.3.

Note that if we write $s_i := \mu x_i^{-1}$, that is $x_i s_i = \mu$ for all i, and x, y solve (29)-(30), then x, y, and s satisfy (23)-(25).

As noted, the three different barrier models are equivalent to the optimality conditions (23)-(25) for solving the primal and dual linear programming problem. However, the approaches are different. The primal-dual approach keeps track of the primal and dual variable, and therefore the duality gap is known at every step of the iteration. The duality gap gives a good measure of convergence. The primal- and the dual approach keep track of the primal- and dual variable, respectively. For the primal and the dual algorithm usually more steps are needed to assure appropriate accuracy than for the primal-dual algorithm. The primal-dual method is popular in practice, because of the convenient possibility to measure progress of the convergence.

1.2.3. Semidefinite programming problems

The *semidefinite programming problem* (SDP) is an extension of the linear programming problem in the sense that an SDP can be viewed as an LP with matrix variables and a different inner product. The goal is to minimize the inner product $\langle C, X\rangle$ over X under the constraints $\langle A_i, X\rangle = b_i,\ 1 = 2, \dots, m$, and $X \geq 0$:

$$\begin{array}{lll} \text{(P)} & \min & \langle C, X\rangle \\ & \text{subject to} & \langle A_i, X\rangle = b_i,\ i = 1, \dots, m \quad , \\ & & X \geq 0 \end{array}$$

where C is a given $n \times n$ Hermitian matrix and the matrix X is an $n \times n$ matrix variable. The data matrices A_i are given Hermitian $n \times n$ matrices and the b_i are given scalars. We will call (P) the standard semidefinite programming problem or *primal semidefinite programming problem.*

Associated with (P) is the *dual semidefinite programming problem* (D)

$$\begin{array}{lll} \text{(D)} & \max & \langle b, y\rangle \\ & \text{subject to} & \sum_{i=1}^{m} y_i A_i + S = C \quad , \\ & & S \geq 0 \end{array}$$

where $y \in \mathbb{C}^m$ and $S \in \mathcal{S}^n$ have to be determined. Usually, (P) and (D) are defined as above (see [**1**], [**26**], and [**44**]), but sometimes (P) is written as a maximization problem and called the dual problem, and (D) is written as a minimization problem and called the primal problem (see for example [**46**]).

For simplicity of notation we introduce a linear operator $\mathcal{A} : \mathsf{S}^n \to \mathbb{C}^m$. The operator $\mathcal{A}$ is defined as

$$\mathcal{A}(X) = \begin{bmatrix} \langle A_1, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{bmatrix} \in \mathbb{C}^m \quad \text{for some} \quad A_1, \dots, A_m \in \mathsf{S}^n.$$

This formulation is consistent with the *Riesz* representation theorem [**16**, p.61]. Its adjoint $\mathcal{A}^* : \mathbb{C}^m \to \mathsf{S}^n$ is determined by $\langle \mathcal{A}^*(y), X\rangle = \langle y, \mathcal{A}(X)\rangle$, for all $y \in \mathbb{C}^m$ and for all $X \in \mathsf{S}^n$, and can be represented as

$$\mathcal{A}^*(y) = \sum_{j=1}^{m} y_j A_j \in \mathsf{S}^n \quad \text{for} \quad y \in \mathbb{C}^m.$$

We can now rewrite (P) and (D) as

$$\begin{array}{lllllll} \text{(P)} & \min & \langle C, X\rangle & \qquad & \text{(D)} & \max & \langle b, y\rangle \\ & \text{subject to} & \mathcal{A}(X) = b \quad , & & & \text{subject to} & \mathcal{A}^*(y) + S = C \quad . \\ & & X \geq 0 & & & & S \geq 0 \end{array}$$

The nonnegative difference $\langle C, X\rangle - \langle b, y\rangle$ between the primal objective value at a primal feasible X and the dual objective value at a dual feasible y is called the *duality gap* for X and y.

There is a duality theory that explains the relationship between (P) and (D), see for example [**46**] and [**1**]. There are similarities with the duality theory for the linear programming problem. Analogously to (19) for SDP, we have for feasible X and (y, S) the property that

$$\langle b, y\rangle \leq \langle C, X\rangle.$$

There are also differences. For instance, if for LP either the primal or the dual problem has an optimal solution, then both have optimal solutions and $\langle b, y^+\rangle = \langle c, x^+\rangle$, where x^+ solves (p) and y^+ solves (d). For SDP, the primal problem may be solvable while its dual is infeasible, or it is feasible but $\langle b, y\rangle \neq \langle C, X\rangle$ at optimality [**46**, p.65]. A zero duality gap at optimality is guaranteed if either $X > 0$ is feasible and (y, S) with $S \geq 0$ is feasible, or if $X \geq 0$ is feasible and (y, S) with $S > 0$ is feasible.

As for linear programming we see the relationship between (P) and (D) by inspecting the three barrier formulations of (P) and (D). We will only consider the analogue of the barrier formulation (22) for (D). In order to eliminate the constraint $S \geq 0$ in (D), we will introduce a logarithmic barrier function such that (D), for $S > 0$, leads to

$$\begin{array}{ll} \max & \langle b, y\rangle - \mu \ln(\det(S)) \\ \text{subject to} & \mathcal{A}^*(y) + S = C \end{array}, \tag{31}$$

with $(y^+_\mu, S^+_\mu) \rightarrow (y^+, S^+)$ if $\mu \rightarrow 0$, where (y^+, S^+) is a solution of (D). The singularity at zero of the logarithm keeps the iterates inside the feasible region.

To characterize a solution, the Lagrangian of (31) is introduced:

$$\mathcal{L}_{\mathrm{PD}}(X, y, S) = \langle b, y\rangle - \mu \ln \det(S) + \langle X, C - \mathcal{A}^*(y) - S\rangle. \tag{32}$$

The first order optimality conditions for minimizing (32) are for $X > 0$ and $S > 0$

$$\mathcal{A}^*(y) + S - C = 0 \qquad (\nabla_X \mathcal{L}_{\mathrm{PD}}(X, y, S) = 0), \tag{33}$$

$$\mathcal{A}(X) - b = 0 \qquad (\nabla_y \mathcal{L}_{\mathrm{PD}}(X, y, S) = 0), \tag{34}$$

$$XS = \mu I \qquad (\nabla_S \mathcal{L}_{\mathrm{PD}}(X, y, S) = 0). \tag{35}$$

Note that (35) implies $SX = \mu I$ and consequently $XS + SX = 2\mu I$. The method based on (35) is called the XS-method in [**2**]. For practical computations it is sometimes good to have a symmetric form of (35). We can change (35) to a symmetric form, by a similarity transformation of (35) with some invertible matrix P and by adding its adjoint:

$$P(XS)P^{-1} + P^{-*}(SX)P^* = 2\mu I. \tag{36}$$

This form is symmetric since $X = X^*$ and $S = S^*$ by definition, so that

$$P^{-*}SXP^* = (PXSP^{-1})^*.$$

In [**2**] a method based on (36) with $P = I$ is called the $XS + SX$ method.

The primal-dual algorithm is an iterative method for computing a solution of (P) and (D). It starts with a feasible $(X, S) > 0$ and makes one iteration step of a Newton method for the first order conditions (33)-(35) for a fixed value of μ, in an attempt to find a new triple (X, y, S) that retains feasibility. Then the value of μ is reduced and the process is repeated until convergence. In the next chapter we will use the optimality conditions to solve the SDP problem with a Newton method.

1.2.4. Examples of SDP problems

EXAMPLE 1.2.1. This example is taken from [**46**]. Combinatorial optimization problems can often be formulated as *quadratic optimization problems.* A quadratic optimization problem has a quadratic objective function, and usually quadratic constraints. An example of a (combinatorial) quadratic optimization problem is the *maximum cut* problem:

$$\begin{array}{ll} \min & x^*Qx \\ \text{subject to} & x_i \in \{-1,1\},\ i=1,\dots,k \end{array} \quad . \tag{37}$$

The diagonal entries of the matrix Q are zero; we will denote this by $\mathrm{Diag}(Q) = \mathbf{0}$ and $\mathbf{0} = (0,\dots,0)^*$. Such a combinatorial optimization problem is known to be NP-hard. The notion NP-hard refers to the complexity of the problem, see [**13**, ch.5]. The constraint $x_i \in \{-1,1\}$ can be written as the quadratic equality constraint $x_i^2 = 1$ (or, equivalently, as two quadratic inequalities $x_i^2 \leq 1$ and $x_i^2 \geq 1$, which gives more algorithmic freedom). Combinatorial optimization problems such as (37) allow for SDP *relaxations.* The idea of relaxations is to replace a "difficult" optimization problem by a simpler one whose maximal/minimal value is at least as large/small as the original one. The key observation is that

$$x^*Qx = \langle Qx, x\rangle = \mathrm{trace}\,(Qxx^*) = \langle Q, xx^*\rangle.$$

The rank one matrix $X := xx^*$ is vector x. We can therefore relax $X = xx^*$ to $X \geq 0$.

Note also that $\mathrm{Diag}(X) = e$, where e represents the vector $e := (1,1,\cdots,1)^*$. If we write $A_i = e_ie_i^*$ for $i = 1,\dots,k$, then the constraint $\mathrm{Diag}(X) = e$ is equivalent to the constraint $\langle A_i, X\rangle = 1$ for $i = 1,\dots,k$. We therefore can state the following semidefinite programming formulation for the relaxation of (37):

$$\begin{array}{ll} \min & \langle Q, X\rangle \\ \text{subject to} & \mathrm{Diag}(X) = e \\ & X \geq 0 \end{array} \quad . \tag{38}$$

Because we enlarged the set of feasible solutions, the optimal value of (38) has to be smaller or equal to the optimal value of (37). Goemans and Williamson [**15**] proved that the optimal value of (38) is at most 14% suboptimal for (37). In practice this is a worst case scenario, since optimal values of (38) are usually much closer to the optimal value of (37).

A relaxation of the original optimization problem has at least one of the following properties [**48**, p.24]:

- the set of feasible solutions is enlarged so that one optimizes over a larger set.
- the objective function is replaced by a function that has the same or a larger/smaller value everywhere depending on the problem formulation as maximization or minimization problem.

EXAMPLE 1.2.2. This example is just an alternative formulation for a well know problem. We are interested in the largest eigenvalue of a matrix $A \in \mathsf{S}^n$:

$$Av = \lambda v.$$

This problem can be formulated as a semidefinite programming problem:

$$\text{(39a)}\quad \begin{array}{ll} \max & \langle A, X\rangle \\ \text{subject to} & \operatorname{trace}(X) = 1 \\ & X \succeq 0 \end{array}, \qquad \text{(39b)}\quad \begin{array}{ll} \min & \lambda \\ \text{subject to} & S = \lambda I - A \\ & S \succeq 0 \end{array}.$$

For more information on this problem formulation see [**22**].

EXAMPLE 1.2.3. Let A be an $r \times r$ matrix and n a nonnegative integer. The *Chebyshev polynomial* of degree n for A is the unique monic polynomial p_n^+ such that

$$p_n^+ = \operatorname*{argmin}_{p_n} \|p_n(A)\|. \tag{40}$$

Let $\{B_0, B_1, \dots, B_n\}$ be a linearly independent set of matrices in $\mathbb{C}^{r\times r}$. Then problem (40) is equivalent to the norm minimization problem:

$$\min_{z\in\mathbb{C}^n} \quad \Big\| \sum_{k=1}^{n} z_k B_k - B_0 \Big\|, \tag{41}$$

where $B_0 = A^n$, $B_k = A^{k-1}$, $k = 1, \dots, n$. If z solves (41) then the elements z_k are the coefficients of the Chebyshev polynomial of A. This norm minimization problem can be expressed as a semidefinite program [**46**], [**31**]. If we define

$$K := \sum_{k=1}^{n} z_k B_k - B_0,$$

and use the fact that for $K \in \mathbb{C}^{r\times r}$,

$$\|K\| = \lambda_{\max} \begin{pmatrix} 0 & K \\ K^* & 0 \end{pmatrix}, \tag{42}$$

then the norm minimization problem (41) can be written in the form (39b)

$$\left\{ \begin{array}{ll} \displaystyle\min_{z\in\mathbb{C}^n,\ \lambda\in\mathbb{R}} & \lambda \\ & \\ \text{subject to} & S = \lambda A_{2n+1} + A_0 - \sum_{k=1}^{n}(x_k A_k + y_k A_{n+k}) \\ & S \succeq 0 \end{array} \right. , \tag{43}$$

where $x_k = \operatorname{re}(z_k)$, $y_k = \operatorname{im}(z_k)$, and

$$A_{2n+1} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}, \quad A_0 = \begin{pmatrix} 0 & B_0 \\ B_0^* & 0 \end{pmatrix},$$

$$A_k = \begin{pmatrix} 0 & B_k \\ B_k^* & 0 \end{pmatrix}, \quad A_{n+k} = \begin{pmatrix} 0 & iB_k \\ -iB_k^* & 0 \end{pmatrix}, \quad k = 1, \dots, n.$$

CHAPTER 2

Computational techniques

In this chapter we review some techniques for the computation of solutions for convex programming problems. A popular method in this context is the Newton method applied to the first order optimality conditions, see Section 1.2. A Newton method requires solutions of linear equations, for linear as well as for semidefinite programming problems. For problems of modest size we can solve the linear equations by computing a standard LU-factorization. For larger problems we study sparse variants and subspace methods, because those methods can solve in less computing time when the dimension of the matrices becomes large.

First, we discuss interior-point methods in the general framework for convex optimization problems. In 2.1.2, 2.1.3, and 2.1.4 we apply the interior-point methods to linear programming problems. In Section 1.2 we have seen that the three different barrier formulations for the linear programming problem led to the same optimality conditions. However, we will also see that if we apply Newton's method to the three different problems then the approaches have different properties.

Section 2.2 gives a short discussion on the analogous formulation to LP of the primal-dual interior-point method for SDP. Section 2.3 deals with computational details for the primal-dual interior-point method for LP and SDP. We will see that, especially for SDP, the methods are very inefficient when the problem size is large. We will discuss new ideas for the projection of convex programming problems onto subproblems of smaller size in Section 2.4.

2.1. Interior-point methods for convex programming problems

First, the general framework is introduced. Then the general framework is applied to linear programming problems.

2.1.1. Interior-point methods in the general framework

In Subsection 1.2.1 we introduced a general convex programming problem. Recall from (16)-(18) the first order optimality conditions for x nonnegative and $s > 0$:

$$g(x, y, s) =: \begin{pmatrix} h(y) + s \\ \nabla f(y) + \langle x, \nabla h(y) \rangle \\ SXe - \mu e \end{pmatrix} = 0, \tag{44}$$

where $X := \operatorname{diag}(x)$, $S := \operatorname{diag}(s)$. The basis for a numerical algorithm for computing a solution to (44) is a Newton method. We obtain a Newton method if we substitute $x^k + \Delta x^k$, $y^k + \Delta y^k$, and $s^k + \Delta s^k$ into (44), and neglect second and higher order terms. We use first order approximations of the functions appearing in the KKT conditions:

$$\begin{aligned} \nabla f(y+\Delta y) &= \nabla f(y) + \nabla^2 f(y)\Delta y + \mathcal{O}(\|\Delta y\|^2), \\ h(y+\Delta y) &= h(y) + \nabla h(y)\Delta y + \mathcal{O}(\|\Delta y\|^2). \end{aligned}$$

With the notation:

$$H(x,y) := \nabla^2 f(y) + \langle x, \nabla^2 h(y)\rangle,$$

we obtain the following system:

$$\begin{aligned} \nabla h(y^k)\Delta y^k + \Delta s^k &= -h(y^k) - s^k \ (=: p) \\ H(x^k,y^k)\Delta y^k + \langle \Delta x^k, \nabla h(y^k)\rangle &= -\nabla f(y^k) - \langle x^k, \nabla h(y^k)\rangle \ (=: q) \\ S^k\Delta X^k e + \Delta S^k X^k e &= -S^k X^k e + \mu e \ (=: r) \quad . \end{aligned} \tag{45}$$

The system of equations (45) can be written in matrix form as

$$\begin{pmatrix} 0 & \nabla h(y^k) & I \\ \nabla h^*(y^k) & H(x^k,y^k) & 0 \\ 0 & S^k & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{46}$$

Now it is clear that we have a Newton method for solving $g(x,y,s) = 0$, because $\nabla g(x^k,y^k,s^k)$ equals the matrix at the left hand side of (46), and the Newton direction

$$\begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta s^k \end{pmatrix} = -\nabla g(x^k,y^k,s^k)^{-1} g(x^k,y^k,s^k),$$

is given by (46). This Newton method finds a solution of optimization problem (12), it proceeds iteratively from a strictly feasible initial point (x^0, y^0, s^0) through a sequence of points determined from the search direction $(\Delta x^k, \Delta y^k, \Delta s^k)$ described by (46):

$$\begin{aligned} x^{k+1} &= x^k + \alpha^k \Delta x^k, \\ y^{k+1} &= y^k + \beta^k \Delta y^k, \\ s^{k+1} &= s^k + \beta^k \Delta s^k. \end{aligned}$$

The α^k and β^k are chosen such that $x^{k+1} > 0$ and $s^{k+1} > 0$. Note that if we start with a feasible point, we have that $p = q = 0$ in (46). If we do not have a feasible starting point, then this process leads to a feasible point, if it exists. For infeasible points p and q will measure the infeasibility.

The matrix in equation (46) is not symmetric, but it is easily symmetrized in two steps. First by multiplying the third equation by $(S^k)^{-1}$, which gives

$$\begin{pmatrix} 0 & \nabla h(y^k) & I \\ \nabla h^*(y^k) & H(x^k,y^k) & 0 \\ 0 & I & (S^k)^{-1}X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} p \\ q \\ (S^k)^{-1}r \end{pmatrix}. \tag{47}$$

Since the third equation of (47) can be used to eliminate Δs without producing any off-diagonal fill-in in the remaining subsystem, we write

$$\Delta s^k = (X^k)^{-1}(r - S^k \Delta x^k), \tag{48}$$

and obtain the symmetric reduced system

$$\begin{pmatrix} -(X^k)^{-1}S^k & \nabla h(y^k) \\ \nabla h^*(y^k) & H(x^k,y^k) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} -X^{-1}r + p \\ q \end{pmatrix}. \tag{49}$$

in the second step. In some situations it will be convenient to go one step further and use the lower part of equation (49) for the elimination of the variable Δx^k. Then Δx^k is given by

$$\Delta x^k = (S^k)^{-1}X^k\Big(\nabla h(y^k)\Delta y^k + (X^k)^{-1}r - p\Big).$$

The resulting equation is,

$$\Big(H(x^k,y^k)+\nabla^* h(y^k)(S^k)^{-1}X^k\nabla h(y^k)\Big)\Delta y^k = q+\nabla h^*(y^k)(S^k)^{-1}X^k\Big(p-(X^k)^{-1}r\Big).$$

Note that the matrix at the left hand side is equal to the Schur complement of $H(x^k,y^k)$ in the matrix on the left hand side of (47).

We will now proceed to linear programming problems. First we will describe the primal-dual interior-point method, then the primal method, and finally the dual method.

2.1.2. The primal-dual interior-point method for LP

In Subsection 1.2.2 we introduced linear programming problems. Equations (23)-(25) are the first order optimality conditions for the primal-dual barrier formulation of (p), for reference we repeat them here:

$$g(x,y,s) := \begin{pmatrix} c - A^*y - s \\ b - Ax \\ XSe - \mu e \end{pmatrix} = 0, \tag{50}$$

where $X := \operatorname{diag}(x)$ and $S := \operatorname{diag}(s)$.

Under the assumption that there exist strictly feasible x^0, y^0, and s^0, we apply Newton's method to (50), as we did in the general setting in Subsection 2.1.1. This gives the system

$$\begin{array}{rcl} A^*\Delta y + \Delta s &=& c - A^*y - s \ (=: p), \\ A\Delta x &=& b - Ax \ (=: q), \\ S\Delta x + X\Delta s &=& \mu e - XSe \ (=: r). \end{array} \tag{51}$$

Note that the only quadratic term that had to be neglected is $-\Delta X \Delta S e$ coming from the last equation of (50). The other two KKT conditions are linear and do not lead to high order perturbations.

Again we see (as in (45)) that if we start with a feasible x^0, y^0, and s^0, then $p = q = 0$ in (51). If we start with an infeasible point (x^0, y^0, s^0), then p measures the dual infeasibility and q the primal infeasibility, since p and q denote the violation of the dual and primal constraints, respectively.

Conform the general framework, we write (51) in the same form as (46):

$$\begin{pmatrix} 0 & A^* & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{52}$$

The matrix on the left hand side is symmetrized by multiplying the third row by S^{-1}. We can eliminate the third equation without loosing the diagonal block structure of the matrix. This leads to $\Delta s = X^{-1}(r - S\Delta x)$ and

$$\begin{pmatrix} -SX^{-1} & A^* \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} p - X^{-1}r \\ q \end{pmatrix}. \tag{53}$$

We construct the Schur complement $AS^{-1}XA^*$ of SX^{-1} in (53), which gives

$$\begin{aligned} &(AS^{-1}XA^*)\Delta y = AS^{-1}X(p - X^{-1}r) + q, \\ &\Delta x = S^{-1}X(A^*\Delta y + X^{-1}r - p), \\ &\Delta s = p - A^*\Delta y. \end{aligned} \tag{54}$$

Note that $AS^{-1}XA^*$ is symmetric, since X and S^{-1} are diagonal matrices. As in the general framework 2.1.1, we start a damped Newton iteration process with a strictly feasible initial point (x^0, y^0, s^0): we construct the new estimates

$$\begin{aligned} x^{k+1} &= x^k + \alpha^k \Delta x^k, \\ y^{k+1} &= y^k + \beta^k \Delta y^k, \\ s^{k+1} &= s^k + \beta^k \Delta s^k, \end{aligned}$$

with step sizes α^k, and β^k, such that $x^{k+1} > 0$, and $s^{k+1} > 0$.

For a damped Newton step (the third equation of (50) is not solved exactly), we set the value of μ_{k+1} to to

$$\mu_{k+1} = \rho \frac{\langle x^{k+1}, s^{k+1} \rangle}{n},$$

where $\rho \in (0,1)$ is a given constant. This method is called the *primal-dual interior-point method* for LP.

Very popular in interior-point codes is a predictor corrector method proposed by Mehrotra [**29**]. This is essentially a second order correction derived directly from the first order conditions (50). The difference with the usual primal-dual interior-point method is that the second order term $\Delta X \Delta S e$ is included in (51). Mehrotra proposes to solve first an affine system and then carry out the correction or centering step for the $\Delta X \Delta S e$ term. Suppose that one starts with (strictly feasible) x, y, s, and $\mu = \frac{x^* s}{n}$. Then compute the affine scaling step by solving (52) with right hand side

$$\begin{pmatrix} p \\ q \\ -XSe \ (=: r^{(1)}) \end{pmatrix}.$$

We denote the solution as $(\Delta x^{(1)},\ \Delta y^{(1)},\ \Delta s^{(1)})$.

Construct step sizes α_1 and α_2 such that $x + \alpha_1 \Delta x^{(1)} > 0$ and $s + \alpha_2 \Delta s^{(1)} > 0$, and form

$$\mu_{\text{aff}} = \frac{(x + \alpha_1 \Delta x^{(1)})^*(s + \alpha_2 \Delta s^{(1)})}{n}, \quad \text{and} \quad \sigma = \left(\frac{\mu_{\text{aff}}}{\mu} \right)^3.$$

The σ is a heuristic choice suggested by Mehrotra. We use this to solve (52) with right hand side

$$\begin{pmatrix} 0 \\ 0 \\ \sigma\mu e - \Delta X^{(1)}\Delta S^{(1)} e \end{pmatrix},$$

for some small $\mu \geq 0$. Denote the solution as $(\Delta x^{(2)},\ \Delta y^{(2)},\ \Delta s^{(2)})$. Then add the corrections $\Delta x = \Delta x^{(1)} + \Delta x^{(2)}$, $\Delta y = \Delta y^{(1)} + \Delta y^{(2)}$, and $\Delta s = \Delta s^{(1)} + \Delta s^{(2)}$, compute step lengths, and finally update x, y and s. In practice, this method improves the quality of the approximate solution, compared with the usual primal-dual interior method. This indicates that the neglected second order term may contain important information.

Instead of using one corrector-step, we may also do several repetitions of the corrector step, as is suggested in [**23**]. If we iterate the Mehrotra process then we obtain

$$r^{(j+1)} := \mu e - Xs - \Delta X^{(j)}\Delta s^{(j)}. \tag{55}$$

We now solve (52) with right hand side

$$\begin{pmatrix} p \\ q \\ r^{(j)} \end{pmatrix}. \tag{56}$$

Experiments in [**23**] show that repetitions do not essentially improve the quality of the approximate solution, compared with the one found by Mehrotra's method.

2.1.3. The primal interior-point method for LP

We repeat the first order optimality conditions (29)-(30) for the primal barrier formulation of (p):

$$g(x,y) := \begin{pmatrix} c - \mu X^{-1}e - A^*y \\ b - Ax \end{pmatrix} = 0, \qquad X := \operatorname{diag}(x). \tag{57}$$

The *primal interior-point method* is a damped Newton iteration process just like the primal-dual interior-point method. Now Newton's method is applied to (57). This leads to the following system:

$$\begin{pmatrix} \mu X^{-2} & -A^* \\ -A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} A^*y + \mu X^{-1}e - c \\ Ax - b \end{pmatrix}. \tag{58}$$

Assuming that we start the iteration process with a strictly feasible (x^0, y^0), then we compute the Schur complement μAX^2A^* of μX^{-2} in (58) and express the search direction Δx in terms of this:

$$\Delta x = -\frac{1}{\mu} XPXc + XPe, \tag{59}$$

where

$$P = \Big(I - XA^*(AX^2A^*)^{-1}AX\Big). \tag{60}$$

is the orthogonal projection on $\operatorname{span}(XA^*)^\perp$. Note that we can iterate without explicitly using y; we construct Δy from Δx. We compute the new estimate $x^{k+1} = x^k + \alpha_k \Delta x^k$, where α_k is an appropriate step size such that $x^{k+1} > 0$ and such that progress is made. If the barrier parameter $\mu_k < \varepsilon$ then stop, otherwise, choose $\mu_{k+1} < \rho\mu_k$.

This method is called the *primal* interior-point method because it keeps track of the primal variable x. The method does not give information about the solution of (d) nor about the reduction of the duality gap.

2.1.4. The dual interior-point method for LP

The dual variant works similar to the primal variant, but now with the optimality conditions for the dual barrier formulation of (d):

$$g(y) := b - \mu \sum_i Ae_i(c_i - (A^*y)_i)^{-1} = 0. \tag{61}$$

For reference we repeat the description from Subsection 2.1.3 applied for y here. Newton's method is applied to (61). This leads to the following system:

$$(AS^{-2}A^*)\Delta y = \frac{1}{\mu}b - AS^{-1}e, \quad \text{with} \quad S := \operatorname{diag}(c - A^*y). \tag{62}$$

Assuming that we started the iteration process with a strictly feasible y^0, we compute the new estimate $y^{k+1} = y^k + \beta_k \Delta y^k$, where β_k is an appropriate step size such that $c - A^*(y^k + \beta_k \Delta y^k) > 0$. We stop if the barrier parameter $\mu_k < \varepsilon$, otherwise, we choose $\mu_{k+1} < \rho\mu_k$.

This method is called the *dual* interior-point method, because it keeps track of the dual variable y. As for the primal interior-point method, it does not give information about the reduction of the duality gap.

2.1.5. Comparison of the interior-point methods

For the comparison of the primal-dual approach with the primal, and dual ones, it is instructive to note that for all approaches we have to solve a system involving a matrix of the form ADA^* for some diagonal matrix D. The matrix D varies per step and per method, but the computational work per step and per method is the same. Two immediate advantages appear when examining the primal-dual method.

(1) The exact current duality gap $\langle c, x\rangle - \langle b, y\rangle = \langle x, s\rangle$ is always known for feasible (x, y, s).
(2) Separate step-lengths α_k and β_k for

$$x^k + \alpha_k \Delta x^k \quad \text{and} \quad (y^k, s^k) + \beta_k(\Delta y^k, \Delta s^k)$$

are allowed, which can significantly reduce the number of iterations to convergence, probably because for both the primal and dual variables the maximum possible step sizes are taken, and therefore the largest reduction per step in objective values of (p) and (d) is attained.

2.2. Interior-point methods for SDP

In the following subsection we will concentrate on the primal-dual interior-point method for SDP. It is an analogue of the primal-dual interior-point method for LP. We only describe the primal-dual variant, because the primal and dual variant are analogous to the primal and dual interior-point method for LP. They can be derived using the same principles as in our derivation of the primal-dual interior-point method in Subsection 2.2.1. Moreover, the primal-dual variant has attractive properties over the primal- or the dual variant, as we argued in the last subsection. It is therefore usually the method of choice.

2.2.1. The primal-dual variant

The primal-dual variant is almost the same as the primal-dual variant for LP, but now for the equations (33)-(35). For reference we repeat them here:

$$g(X,y,S) := \begin{pmatrix} C - \mathcal{A}^*(y) - S \\ b - \mathcal{A}(X) \\ XS - \mu I \end{pmatrix} = 0. \tag{63}$$

Under the assumption that there exist strictly feasible points X^0, y^0, and S^0 we apply Newton's method for (63), as we did in Subsection 2.1.2, for (50). This gives the system

$$\begin{aligned} \mathcal{A}^*(\Delta y) + \Delta S &= C - \mathcal{A}^*(y) - S (=: P), && (64)\\ \mathcal{A}(\Delta X) &= b - \mathcal{A}(X) (=: q), && (65)\\ \Delta X S + X \Delta S &= \mu I - XS (=: R). && (66) \end{aligned}$$

Again note that the only quadratic term that had to be neglected is $-\Delta X \Delta S$ coming from the last row in the matrix in equation (63). The other two optimality conditions in (63) are linear and do not lead to high order perturbations. In general ΔX will not be symmetric, we have to adjust our computations to get a symmetric ΔX, see Subsection 2.3.4. We see, as in (51), that if we start with feasible X^0, y^0, and S^0, that $P = 0$ and $q = 0$.

We start a damped Newton iteration process that needs a strictly feasible starting point. In order to obtain feasible iterates, we compute the new estimates $X^{k+1} = X^k + \alpha^k \Delta X^k$, with α^k an appropriate scaling such that $X^{k+1} > 0$, and $y^{k+1} = y^k + \beta^k \Delta y^k$, with β^k an appropriate scaling such that $S^{k+1} = S^k + \beta^k \Delta S^k > 0$. The parameters α^k and β^k are step sizes.

For a damped Newton step we set μ_{k+1} to

$$\mu_{k+1} = \rho \frac{\langle X^{k+1}, S^{k+1} \rangle}{n},$$

where $\rho \in (0,1)$ is a given constant.

This method is called the *primal-dual interior-point method* for SDP problems.

2.3. Computational details

In this section we discuss some computational details for solving the LP and the SDP problems with primal-dual interior-point methods. Both the LP and the SDP primal-dual interior-point methods need to compute search directions, $\Delta x, \Delta y, \Delta s$ or $\Delta X, \Delta y, \Delta S$, at every iteration step.

In Subsection 2.3.1 we motivate how we will compute the search directions for LP; there are three options. We will see that the best option is to compute Δy from

$$(AS^{-1}XA^*)\Delta y = AS^{-1}X(p - X^{-1}r) + q. \tag{67}$$

and then form

$$\Delta x = S^{-1}X(A^*\Delta y + X^{-1}r - p), \tag{68}$$
$$\Delta s = p - A^*\Delta y. \tag{69}$$

Subsection 2.3.2 describes the practical implementations for solving (67), taking into account the size and sparsity of the matrix $AS^{-1}XA^*$.

Subsection 2.3.3 describes the computational details for SDP. Before we show how $\Delta X, \Delta y$, and ΔS are computed, we introduce notation to bring the correction equations (64)-(66) into a form similar to (67). That makes them more suitable for computations. For SDP primal-dual interior-point computations the practical implementation can be easily described, because we do not have a choice how to solve $\Delta X, \Delta y$, and ΔS. This will be shown in the last part of Subsection 2.3.3.

We finish Section 2.3 with the description of some basic differences of LP and SDP primal-dual interior-point computations, in Subsection 2.3.4.

2.3.1. Computational details for LP

From Subsection 2.1.2 it is obvious how to compute Δy from (67) and then form Δx and Δs, but more options exist. We will state the different options and describe their features. We aim at a symmetric, definite and well-conditioned system to solve.

The complete system. The first option is to compute the vector $(\Delta x, \Delta y, \Delta s)$ from the correction equation

$$\begin{pmatrix} 0 & A^* & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{70}$$

We will refer to the 3×3 block system (70) as the *complete system.* Unfortunately, the *complete system* is non-symmetric, indefinite, and ill-conditioned (see below). System (70) can be symmetrized by multiplying the last row with S^{-1}, but the indefiniteness can not be easily repaired. Therefore we will avoid to solve the complete system and look for a stable, symmetric, and well-defined reduction process.

The reason for the ill-conditioning of (70) is that some components of x and s tend to zero in the iteration process, when x approaches the optimal point (s approaches the associated Lagrange multiplier). In finite precision arithmetic this means that some information may get lost because of cancellation effects, when multiplying with x or s.

The reduced 2×2 *systems.* Freund et al. [**12**] propose a symmetric stable reduction based on two different symmetric 2×2 block matrix factorizations of the complete system (70). The first symmetric 2×2 block matrix factorization is taken from Subsection 2.1.2,

$$\begin{pmatrix} -SX^{-1} & A^* \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} p - X^{-1}r \\ q \end{pmatrix}. \tag{71}$$

with $\Delta s = X^{-1}(r - S\Delta x)$, and the second one is obtained by eliminating Δx from (70):

$$\begin{pmatrix} -SX^{-1} & A^* \\ A & 0 \end{pmatrix} \begin{pmatrix} -XS^{-1}\Delta s \\ \Delta y \end{pmatrix} = \begin{pmatrix} p \\ r - AS^{-1}q \end{pmatrix}, \tag{72}$$

with $\Delta x = S^{-1}(p - X\Delta s)$. The matrices in (71) and (72) are both ill-conditioned for the same reason as (70) is ill-conditioned. We can not see in advance which parts of x and s are vanishing when approaching optimality, therefore a partitioning of x and s will be the first step towards a better understanding of the behavior of the block system.

Partition the vectors x and s into two parts $x_1,\ x_2$ and $s_1,\ s_2$, respectively, such that $x_1 \geq s_1$ and $x_2 < s_2$. For $x_1,\ s_1$ the equation (71) is used and for $x_2,\ s_2$ the equation (72) is used. Following partitioning of x and s, we set the constraint matrix A to

$$A = (A_1 \;\; A_2).$$

Similarly, we partition

$$p = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \quad \text{and} \quad q = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}.$$

Using this partitioning and reordering, we combine the matrices of (71) and (72) into one matrix:

$$K := \begin{pmatrix} -S_1X_1^{-1} & 0 & A_1^* \\ 0 & -S_2X_2^{-1} & A_2^* \\ A_1 & A_2 & 0 \end{pmatrix}.$$

Similarly, we combine the right hand sides and search directions, so that the two systems are combined to one new system, written as $Kz = d$. We have collected the small components in the $(1,1)$ block $-S_1X_1^{-1}$ of K and the large components in the $(2,2)$ block $-S_2X_2^{-1}$ of K. Numerical cancellation is avoided, because for multiplication only the 'large numbers' of x and s (the significant values) are used. The ill-conditioning is reduced by a rescaling matrix Λ:

$$\Lambda = \begin{pmatrix} I & 0 & 0 \\ 0 & S_2^{-1/2}X_2^{1/2} & 0 \\ 0 & 0 & I \end{pmatrix},$$

for which

$$\Lambda K \Lambda = \begin{pmatrix} \varepsilon & 0 & A_1^* \\ 0 & -I & \delta^* \\ A_1 & \delta & 0 \end{pmatrix},$$

where $\varepsilon = -S_1X_1^{-1}$ and $\delta = A_2S_2^{-1/2}X_2^{1/2}$ contains only small numbers. We have to identify a suitable method that can handle problems of the type $\Lambda K \Lambda$. One has to solve $\Lambda K \Lambda z = \Lambda d$. Unfortunately, no effective preconditioners are known for iteratively solving this system, see [**12**].

The Schur complement of the complete system. We will now describe the method that is usually implemented in LP interior-point solvers. We reduce the complete system (70) to (67), with Δx and Δy defined by (68) and (69), respectively.

The matrix $AS^{-1}XA^*$ is symmetric positive definite. Unfortunately, $AS^{-1}XA^*$ is ill-conditioned for the same reason that the complete system is ill-conditioned. However, when computing Δy from (67), the matrix

$$(AS^{-1}XA^*)^{-1}A^*S^{-1}X$$

appears. This matrix remains bounded, independent of the value of $S^{-1}X$, as is shown by Stewart [**43**, p.10]. Therefore (67) is used in practical implementations to compute Δy (and subsequently $\Delta x, \Delta s$). We will proceed with the solution of (67) and describe the methods for solving this linear system in the following subsection.

2.3.2. Practical computation

In this subsection we will discus methods for solving the linear system

$$(AS^{-1}XA^*)\Delta y = AS^{-1}X(p - X^{-1}r) + q. \tag{73}$$

For ease of notation we set $K := AS^{-1}XA^*, x := \Delta y$, and $d := AS^{-1}X(p - X^{-1}r) + q$. We will use different methods for solving $Kx = d$ depending on the size and sparsity of K. We will now describe the methods that exploit properties of K.

For positive definite matrices of low dimension we prefer a Cholesky factorization, as we described in Section 1.1.4. Unfortunately, in practice matrices are not always of low dimension.

Cholesky and reordering. If K is sparse, then a Cholesky factorization usually produces fill-in; that is, some lower triangular locations in the Cholesky factor L contain nonzero elements at locations where the original matrix K has a zero. The amount of fill-in depends on the ordering of the system, so that we can consider the permuted system PKP^* for some fill-in reducing permutation matrix P. For a permutation matrix P, we solve the reordered system

$$(PKP^*)(Px) = Pd.$$

In practice we have to reorder the rows and columns of K before performing the factorization. One of the most effective sparsity preserving orderings for K is called the *minimum degree ordering* [**14**]. At each step, a row and corresponding column interchange is applied to the part of the matrix remaining to be factored, such that the product of the number the number of nonzeros in the pivot row and column is minimal. After n steps of the symbolic factorization the minimum degree ordering is obtained. This helps to reduce the amount of arithmetics that has to be carried out in the Gaussian elimination. However, note that the minimum degree ordering does not take the size of the non-zero matrix-elements into consideration. Therefore, in order to preserve numerical stability, one must take care to avoid small pivot elements. In fact, there should be a balance between minimum fill and numerical stability. A good survey of implementations of minimum degree ordering can be found in, for example, [**14**].

Cholesky and Sherman-Morrison. Most of the interior-point codes use a sparse Cholesky factorization LL^* of the sparse matrix $K := AS^{-1}XA^*$ in every iteration step of the algorithm. To minimize the fill-in, the columns and rows are reordered as we described above for *Cholesky and reordering.* These permutations are determined at the first iteration, because the sparsity pattern remains the same for any positive diagonal matrix $D := S^{-1}X$ in $K = ADA^*$. We will now explain that costly updates of this factor in each step can be largely avoided too.

Consider the situation where only one entry of D changes. Then the new coefficient matrix K' differs from K only by a rank one matrix, so we can write

$$K' = K + uu^*,$$

for a suitable vector u. With the *Sherman-Morrison-Woodbury* formula [**17**, p.50] we have for the inverse of K' that

$$(K + uu^*)^{-1} = K^{-1} - \frac{K^{-1}uu^*K^{-1}}{1 + u^*K^{-1}u}. \tag{74}$$

Since only matrix-vector multiplications are involved in (74), the inverse of K' can be calculated from the inverse of K with only $\mathcal{O}(n^2)$ additional operations. We generalize this idea to the case where K has a very low number of dense columns, say l. Split the relatively dense columns (d) of A from the sparse ones (s), that is, partition

$$A = [A_s A_d], \quad \text{and} \quad D = \begin{pmatrix} D_s & 0 \\ 0 & D_d \end{pmatrix},$$

such that we can rewrite K as

$$K = ADA^* = A_sD_sA_s^* + A_dD_dA_d^* = H + UU^*, \tag{75}$$

where H is the product of sparse matrices and $U = A_dD_d^{1/2}$ is the collection of dense columns multiplied by the corresponding diagonal elements $D_d^{1/2}$. The matrix U will be of low rank, $\operatorname{rank}(U) = l$.

If H is nonsingular, then with the *Sherman-Morrison-Woodbury* formula we have the following form for the inverse of K,

$$K^{-1} = (H + UU^*)^{-1} = H^{-1} - H^{-1}U(I + U^*H^{-1}U)^{-1}U^*H^{-1}. \tag{76}$$

The solution of the given linear system $Kx = d$ is

$$x = [H^{-1}d] - H^{-1}U(I + U^*H^{-1}U)^{-1}U^*[H^{-1}d]. \tag{77}$$

Since U is of low rank, the matrix $I + U^*H^{-1}U$ is square, low dimensional and its computation requires only a small number of back substitutions once $H = LL^*$ is factored. This way of computing the inverse of K is called rank-one updating, since the update in equation (76) can be viewed as l rank-ones updates.

Note that if H is ill-conditioned, then (77) gives an instable method, because of subtracting two almost equal vectors. For this situation we need another strategy that will be described in the next part.

Iterative methods. Before we continue with the problem of solving $Kx = d$ when H is ill-conditioned and K is sparse, we introduce the so-called subspace iteration methods for solving linear systems.

Let us take, for example, the starting vector $x_0 = 0$ as a starting vector for the iteration process, then we define the residual to x_0 by $r^0 := d - Kx_0 = d$. The subspace

$$\mathcal{K}_i(K; r^0) := \{r^0, Kr^0, \dots, K^{i-1}r^0\}$$

is called the *Krylov subspace* of order i generated by K and r^0. Instead of using the induced basis vectors for $\mathcal{K}_i(K; r^0)$, an orthonormal basis is used. The construction of an orthonormal basis is via Arnoldi's approach. Start with

$$v^1 := r^0/\|r^0\|_2.$$

Suppose that we already have an orthonormal basis

$$v^1, \dots, v^j.$$

Define $V_j = (v^1 \cdots v^j)$, that is, the matrix with columns $v^1, \dots, v^j$. Then compute

$$\widetilde{v}^{j+1} := Kv^j - V_j(V_j^* Kv^j), \quad \text{and normalize} \quad v^{j+1} := \widetilde{v}^{j+1}/\|\widetilde{v}^{j+1}\|_2.$$

If we write

$$h_{i,j} := \langle v^i, Kv^j\rangle \quad \text{for} \quad i = 1, \dots j \quad \text{and} \quad h_{j+1,j} := \|\widetilde{v}^{j+1}\|_2,$$

then we obtain the following algebraic expression for the Arnoldi basis

$$KV_i = V_{i+1}H_{i+1,i}. \tag{78}$$

The $(i + 1) \times i$ matrix $H_{i+1,i}$ is an upper Hessenberg matrix with elements $h_{i,j}$. When K is symmetric, then the upper Hessenberg matrix reduces to a tridiagonal matrix. In that case the procedure is called the Lanczos method with Gram-Schmidt orthogonalization. In practice, modified Gram-Schmidt is used, in order to improve the numerical orthogonality [**9**].

We are actually looking for a solution x^i in $\mathcal{K}_i(K; r^0)$ as an approximate to the solution x of $Kx = d$. We can use different approaches to identify a suitable solution x^i. For example

- GMRES determines a minimal residual $\|d - Kx^i\|_2$. This approach is called the minimal residual approach,
- Conjugate Gradients (CG) determines a minimal residual in K^{-1}-norm. This approach is called the Ritz-Galerkin approach.

A good survey of subspace methods for linear systems can be found in [**9**].

Now we return to the problem of solving $Kx = d$, When H is ill-conditioned, then we can avoid using (77) by rewriting (75)

$$(LL^* + UU^*)x = d,$$

matrix properties	method
low dimension	Cholesky
large + sparse	Cholesky + reordering
large + sparse + low rank	Cholesky + reordering + Sherman-Morrison
large + sparse + ill-cond.	CG + preconditioner
large + dense	CG + preconditioner

TABLE 1 How to solve $AS^{-1}XA^*x = d$.

as

$$(I + (L^{-1}U)(L^{-1}U)^*)L^*x = L^{-1}d, \tag{79}$$

and solve this system L^*x with CG in order to get an approximation for L^*x. The approximation for x is then obtained with a back-solve.

When H is not sparse and not of low rank, then (79) is not very useful, since the computation of the factorization $H = LL^*$ is too expensive. In this situation we may solve $Kx = d$ by the CG method with a preconditioner, but unfortunately, no suitable preconditioners are known to us.

We summarize in Table 1 methods that can be used for specific situations solving $Kx = d$.

2.3.3. Computational details for SDP

As for the LP primal-dual interior-point method, we will also describe for SDP how to compute $\Delta X, \Delta y$, and ΔS at every step of the primal-dual interior-point method. Before we give explicit formula's for $\Delta X, \Delta y$, and ΔS we need some notation. We use this new notation to rewrite (64)-(66) into a form as in (70).

For simplicity of notation, we will assume that all matrices involved are real. The extension for complex matrices will be obvious.

Let M be an $m \times n$ matrix and N be a $k \times l$ matrix. The *Kronecker product* or *Tensor product* is a map $\otimes : \mathbb{R}^{m\times n} \times \mathbb{R}^{k\times l} \to \mathbb{R}^{mk\times nl}$ defined as

$$M \otimes N := \begin{pmatrix} m_{11}N & \cdots & m_{1n}N \\ \vdots & & \vdots \\ m_{m1}N & \dots & m_{mn}N \end{pmatrix},$$

where $M = (m_{ij}) \in \mathbb{R}^{m\times n}$.

The *vec operator* is a map $vec : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ defined as

$$vec(M) = \begin{pmatrix} m_1 \\ \vdots \\ m_n \end{pmatrix},$$

where M is an $m \times n$ matrix and $m_1, \dots, m_n$ are its columns. Let *mat* be the inverse of *vec*. Thus the *vec* operator stacks the columns of a matrix in a vector, and *mat* puts a vector into a matrix of appropriate size.

We list some important properties of the Kronecker product.

PROPOSITION 2.3.1 ([**18**]). *Let $K, L, M,$ and N be matrices of appropriate sizes.*

(1) $(M \otimes N)^* = M^* \otimes N^*$,
(2) $(M \otimes N)^{-1} = M^{-1} \otimes N^{-1}$,
(3) $(MK) \otimes (NL) = (M \otimes N)(K \otimes L)$,
(4) $(M \otimes N) vec(K) = vec(NKM^*)$,
(5) $(I \otimes M + K \otimes I) vec(N) = vec(MN + NK^*)$,
(6) $\langle M, N \rangle = \mathrm{trace}\,(N^* M) = vec(N)^* vec(M) = \langle vec(M), vec(N) \rangle$.

The set of symmetric matrices S^n can be identified with vectors in $\mathbb{R}^{\frac{1}{2}n(n+1)}$ via the *svec operator.* The *svec* operator is a map defined as

$$svec(M) := \Big(m_{11}, \sqrt{2}m_{12}, \ldots, \sqrt{2}m_{1n}, m_{22}, \sqrt{2}m_{23}, \ldots, \sqrt{2}m_{2n}, \ldots$$
$$\ldots, m_{(n-1)(n-1)}, \sqrt{2}m_{(n-1)n}, m_{nn}\Big)^*,$$

for $M \in \mathsf{S}^n$. The $\sqrt{2}$ for off-diagonal elements ensures for $M, N \in \mathsf{S}^n$ that

$$\langle M, N \rangle = \mathrm{trace}\,(N^* M) = \langle svec(M), svec(N) \rangle.$$

The *symmetric Kronecker product* $\otimes_s$ is defined for arbitrary square matrices $N, M \in \mathbb{R}^{n \times n}$ by its action on the vector $svec(K)$ for a symmetric matrix $K \in \mathsf{S}^n$,

$$(M \otimes_s N) svec(K) := svec\Big(\frac{1}{2}(NKM^* + MKN^*)\Big).$$

We list also some important properties of the symmetric Kronecker product $\otimes_s$.

PROPOSITION 2.3.2 ([**18**]). *Let M and N be matrices of appropriate sizes.*

(1) $M \otimes_s N = N \otimes_s M$,
(2) $(M \otimes_s N)^* = N^* \otimes_s M^*$,
(3) $(M \otimes_s M)(K \otimes_s L) = \frac{1}{2}(MK \otimes_s NL + ML \otimes_s NK)$,
(4) *$M \otimes_s I$ is symmetric if and only if $M \in \mathbb{R}^{n \times n}$ is symmetric,*
(5) $(M \otimes_s M)^{-1} = M^{-1} \otimes_s M^{-1}$,
(6) *if $M > 0$ and $N > 0$ then $(M \otimes_s N) > 0$.*

With these new definitions we can rewrite (66) as

$$(S \otimes I) vec(\Delta X) + (I \otimes X) vec(\Delta S) = vec(\mu I - XS).$$

With the $XS + SX$-method, that we described in Section 1.2.3 (below (35)), we can rewrite (36), with $P = I$, as

$$(I \otimes_s X) svec(\Delta X) + (I \otimes_s S) svec(\Delta S) = svec\Big(\mu I - \frac{1}{2}(XS + SX)\Big).$$

We will now show the similarity with the optimality conditions (70) for the linear programming problem. This will only be done for the XS-method, that was also described in Section 1.2.3. For the $XS + SX$-method, the same formulation can be used as for the XS-method, with the symmetric Kronecker product ($\otimes_s$) in combination with the *svec* map, instead of the Kronecker product ($\otimes$) with the *vec* map.

Let $x := vec(X),\ s := vec(S),\ \Delta x := vec(\Delta X)$, and $\Delta s := vec(\Delta S)$. Define

$$\mathbf{A} := \begin{pmatrix} vec(A_1)^* \\ \vdots \\ vec(A_m)^* \end{pmatrix},$$

and

$$\mathbf{S} := S \otimes I, \quad \mathbf{X} := I \otimes X \quad \text{and} \quad \mathbf{I} := I \otimes I.$$

Then $\mathbf{A}x = \mathcal{A}(X)$. Hence, one Newton step for (63), that is, for $g(X, y, S) = 0$, can be written as the linear equation

$$\begin{pmatrix} 0 & \mathbf{A}^* & \mathbf{I} \\ \mathbf{A} & 0 & 0 \\ \mathbf{S} & 0 & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} vec(P) \\ q \\ vec(R) \end{pmatrix}. \tag{80}$$

This leads to the analogue of (67)

$$\begin{aligned} (\mathbf{AS^{-1}XA}^*)\Delta y &= \mathbf{AS^{-1}X}(vec(P) - \mathbf{X^{-1}}vec(R)) + q, \\ \Delta x &= \mathbf{S^{-1}}X(\mathbf{A}^*\Delta y + \mathbf{X^{-1}}vec(R) - vec(P)), \\ \Delta s &= vec(P) - \mathbf{A}^*\Delta y. \end{aligned} \tag{81}$$

From Proposition 2.3.1 property (1)-(3) and the fact that X and S are symmetric, we see that the matrix $\mathbf{AS^{-1}XA}^*$ is symmetric. To show that the matrix $\mathbf{AS^{-1}XA}^*$ is positive definite first note that S^{-1} is positive definite and apply Proposition 2.3.1 property (4). Then for all K

$$\begin{aligned} &vec(K)^*(X \otimes S^{-1})vec(K) = vec(K)^* vec(S^{-1}KX^*) = \\ &\text{trace}\left(S^{-1}KX^*K\right) = \text{trace}\left(S^{-1/2}KX^*KS^{-1/2}\right) > 0. \end{aligned}$$

and we conclude that $\mathbf{S^{-1}X} > 0$.

The analogue of $\mathbf{AS^{-1}XA}^*$ for the $XS + SX$-method is not symmetric, but can be shown to be nonsingular if $(XS + SX) > 0$, see [**41**].

For SDP we will always use the system of equations (81) to compute $\Delta X, \Delta y$, and ΔS. The reason is that in contrast to LP the problem size is much larger for SDP, and therefore systems even larger than $\mathbf{AS^{-1}XA}^*$ are computationally too expensive to solve. For example the matrix on the left hand side of system (80) has size $(m + 2n^2) \times (m + 2n^2)$, and the matrix on the left hand side of system (70) has 'only' size $(m + 2n) \times (m + 2n)$.

A difference with LP is that the matrix $\mathbf{AS^{-1}XA}^*$ is dense. The linear equation

$$(\mathbf{AS^{-1}XA}^*)\Delta y = \mathbf{AS^{-1}X}(vec(P) - \mathbf{X^{-1}}vec(R)) + q,$$

is in practice usually solved by a Cholesky or LU-factorization. Therefore, if the number m of matrices is larger than 5000 and if the size of the matrices A_i is larger than $n > 500$, SDP problems cannot be solved by primal-dual interior-point methods on a typical workstation within reasonable computing time.

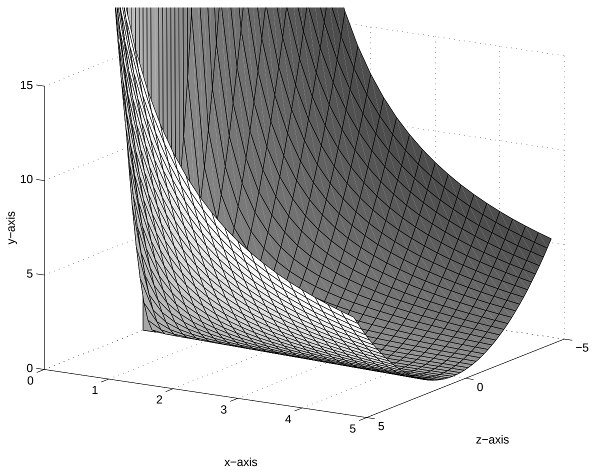

FIGURE 2 Boundary of the cone of S^2_+ matrices represented in $\mathbb{R}^3$.

2.3.4. Comparison of LP and SDP computations

The similarity between the algorithms for LP en SDP is quite clear since we derived the theory for SDP as an analogue of the theory for LP. However, there are important differences:

- The cone of positive semi-definite matrices is not finitely generated,[1] in contrast with the cone for a coordinate-wise ordering (and a non-standard basis) in $\mathbb{R}^n$. Figure 2 shows the boundary of the set of matrices

$$\begin{pmatrix} x & z \\ z & y \end{pmatrix}, \geq 0$$

  represented in $\mathbb{R}^3$, and Figure 3 shows the boundary of a cone in $\mathbb{R}^3$ for a coordinate-wise ordering and a non-standard basis.

  Hence, checking whether $x \geq 0$ is cheaper, in a computational sense, and less complicated, than checking whether a matrix X is positive semi-definite $X \geq 0$.
- There is a difference in choosing the search direction. In the LP case we have diagonal matrices X and S, for which $XS^{-1} = S^{-1}X$ is obvious, while for symmetric non-diagonal matrices, as in the SDP case, in general $XS^{-1} \neq S^{-1}X$. For SDP we do have a choice between symmetry for ΔX, and ΔS, or for XS^{-1}. We usually want to keep the search direction matrices ΔX and ΔS symmetric. To ensure this, (66) in Subsection 2.2.1 can be symmetrized by using the symmetric equation (36) in Subsection 1.2.3. Then (66) becomes

$$P(\Delta XS + X\Delta S)P^{-1} + P^{-*}(S\Delta X + \Delta SX)P^{*} = R,$$

[1]A cone is finitely generated if, for some finite set C, each element of the cone is a positive linear combination of the elements of C.

where

$$R = 2\mu I - P(XS)P^{-1} - P^{-*}(SX)P^{*}.$$

If we choose, for example, $P = I$, then we obtain a search direction called the Alizadeh-Haeberly-Overton (AHO) direction [**2**]. If we choose $P = S^{\frac{1}{2}}$ then we obtain the Monteiro direction [**30**], and

$$P = \left(S^{-\frac{1}{2}}(S^{\frac{1}{2}}XS^{\frac{1}{2}})^{\frac{1}{2}}S^{-\frac{1}{2}}\right),$$

leads to the so-called Nesterov Todd (NT) direction [**32**].

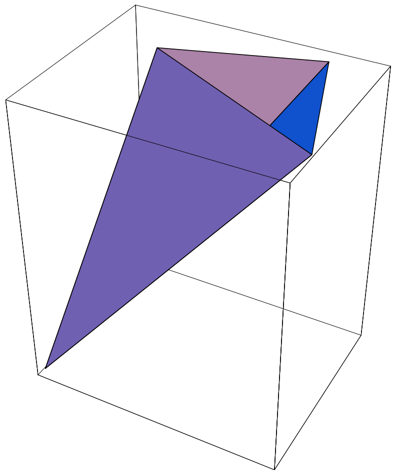

FIGURE 3 Boundary of the cone in $\mathbb{R}^3$ for coordinate-wise ordering and a non-standard basis.

2.4. Projecting the LP and SDP problem

The methods described in the last section work nicely for linear programming problems, but for semidefinite programming problems, the computation of $\mathbf{AS^{-1}XA}$ can be very time and memory consuming, when m and n are large. The most natural way of coping with a large dimension is to project the linear or semidefinite programming problem onto a linear or semidefinite programming problem of smaller size. We will try to identify methods for doing this. The projected linear or semidefinite programming problem can then be solved directly. First we will describe projections for linear programming problems and then for semidefinite programming problems. The framework that we describe, will turn out to be effective for semidefinite programming, and will be the inspiration for the methods discussed in Chapter 3.

2.4.1. LP projections

For $k \ll n$ consider the following maps:

$$P : \mathbb{R}^n \to \mathbb{R}^k, \quad J : \mathbb{R}^k \to \mathbb{R}^n,$$

that, for some (orthonormal) $n \times k$ matrix V, are defined by

$$P(x) = V^* x, \quad J(z) = Vz.$$

For a matrix A related to the standard LP formulation (p), we can formulate the following diagram for P and J,

$$\begin{tikzcd} \mathbb{R}^n \arrow[r, "A"] \arrow[d, shift right, "P"'] & \mathbb{R}^m \arrow[d, shift right, "Id"'] \\ \mathbb{R}^k \arrow[r, "AJ"'] \arrow[u, shift right, "J"'] & \mathbb{R}^m \arrow[u, shift right, "Id"'] \end{tikzcd} .$$

The primal problem (p) can now easily be projected: write $\widetilde{x} = V\bar{x}$, with $\widetilde{x}$ an approximation for x. This leads to

$$\begin{array}{ll} \min & \langle V^* c, \bar{x} \rangle, \\ \text{subject to} & (AV)\bar{x} = b, \\ & \bar{x} \geq 0. \end{array} \tag{82}$$

We do not project the dual problem (d), but use the Lagrangian with respect to the primal problem. Note that this Lagrangian is different from the Lagrangian for the barrier formulation of (p) (see, for example, equation (28)):

$$\begin{aligned} \mathcal{L}(\bar{x}, \widetilde{y}, \widetilde{s}) &\equiv \langle V^* c, \bar{x} \rangle + \langle \widetilde{y}, b - AV\bar{x} \rangle - \langle \widetilde{s}, \bar{x} \rangle \\ &= \langle \widetilde{y}, b \rangle + \langle V^* c - V^* A^* \widetilde{y} - \widetilde{s}, \bar{x} \rangle. \end{aligned}$$

With standard duality theory we obtain the following dual problem for (82):

$$\begin{array}{ll} \max & \langle b, \widetilde{y} \rangle, \\ \text{subject to} & V^* A^* \widetilde{y} + \widetilde{s} = V^* c, \\ & \widetilde{s} \geq 0. \end{array} \tag{83}$$

The observation is that the optimization problem (p) is transformed to an optimization problem of non-standard form (with $\widetilde{x} = V\bar{x}$) :

$$\begin{array}{ll} \min & \langle c, \widetilde{x} \rangle, \\ \text{subject to} & A\widetilde{x} = b, \\ & \bar{x} \geq 0. \end{array}$$

If we want to use the projection procedure for solving (p), a simple embedding technique will not return an $\widetilde{x} \geq 0$, because the projection returns a *high dimensional generated cone* described by $\bar{x} \geq 0$. In Figure 4 we plotted an example of a 6 dimensional cone. We impose the condition $\widetilde{x} \geq 0$ by rewriting (82). The first step is to reformulate the inequality constraint as an equality constraint by introducing an

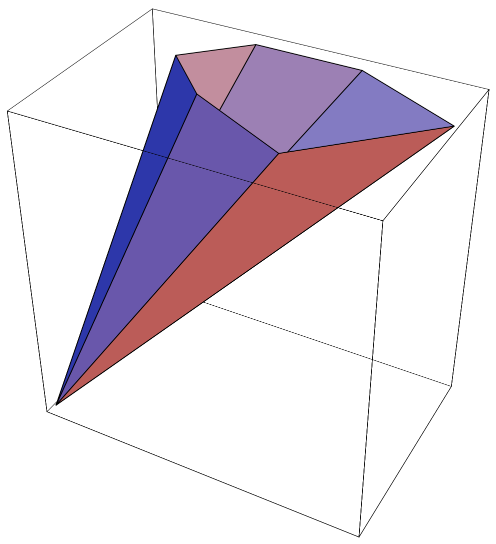

FIGURE 4 Boundary of a 6 dimensional generated cone in $\mathbb{R}^3$.

extra slack variable t:

$$\begin{array}{ll} \text{min} & \langle V^*c, \bar{x}\rangle, \\ \text{subject to} & (AV)\bar{x} = b, \\ & V\bar{x} = t, \\ & t \geq 0. \end{array}$$

The second step is to eliminate the 'free' variable $\bar{x}$, by expressing $\bar{x}$ as the difference of two nonnegative vectors x^+ and x^-, that is, $\bar{x} = x^+ - x^-$, $x^+, x^- \geq 0$. We obtain

$$\begin{array}{ll} \text{min} & \langle V^*c, x^+\rangle - \langle V^*c, x^-\rangle, \\ \text{subject to} & (AV)x^+ - (AV)x^- = b, \\ & Vx^+ - Vx^- = t, \\ & x^+, x^-, t \geq 0. \end{array}$$

This is of standard LP form (p) and (d) with

$$\mathbf{x} = \begin{pmatrix} x^+ \\ x^- \\ t \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} Vc \\ \text{-}Vc \\ 0 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} AV & \text{-}AV & 0 \\ V & \text{-}V & \text{-}I \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

This rewritten standard form of the projected problem is not attractive, because its dimension is even bigger than for the original problem. This is a drawback for the idea of projecting the linear programming problem. We will now discuss how these considerations work out for semidefinite programming problems.

2.4.2. SDP projections

For $k \ll n$, we consider the maps:

$$\mathcal{P} : \mathsf{S}^n \to \mathsf{S}^k, \quad \mathcal{J} : \mathsf{S}^k \to \mathsf{S}^n,$$

that, for some orthonormal $n \times k$ matrix V, are defined by

$$\mathcal{P}(X) = V^* X V, \quad \mathcal{J}(H) = V H V^*.$$

For $\ell \ll m$, we consider the maps:

$$P : \mathbb{R}^m \to \mathbb{R}^\ell, \quad J : \mathbb{R}^\ell \to \mathbb{R}^m.$$

that, for some orthonormal $m \times \ell$ matrix W, are defined by

$$P(x) = W^* x, \quad J(y) = W y.$$

For the operator $\mathcal{A}$ involved in the standard SDP formulation (P), we formulate the following diagram for $\mathcal{P}$ and $\mathcal{J}$, P and J,

$$\begin{tikzcd}
\mathsf{S}^n \arrow[r, "\mathcal{A}"] \arrow[d, shift right, "\mathcal{P}"'] & \mathbb{R}^m \arrow[d, shift right, "P"'] \\
\mathsf{S}^k \arrow[r, "P\mathcal{A}\mathcal{J}"'] \arrow[u, shift right, "\mathcal{J}"'] & \mathbb{R}^\ell \arrow[u, shift right, "J"']
\end{tikzcd}.$$

The map $P\mathcal{A}\mathcal{J}$ can be represented as

$$P\mathcal{A}\mathcal{J}(H) = \begin{pmatrix} \langle \widetilde{A}_1, H \rangle \\ \vdots \\ \langle \widetilde{A}_\ell, H \rangle \end{pmatrix},$$

with the matrices

$$\widetilde{A}_i := \sum_{j=1}^{m} w_{ji}\, V^* A_j V, \quad W = (w_{ij})_{ij}, \qquad (i = 1, \ldots, \ell).$$

An important observation is that the projected problem is still a standard SDP problem, because of the property

$$H \geq 0 \quad \Leftrightarrow \quad X = V^* H V \geq 0.$$

This is in contrast to the LP projected problem.

We give an example.

Example 2.4.1. We are interested in the largest eigenvalue $\lambda_{\max}$ of a matrix $A \in \mathsf{S}^n$. The primal (and dual) problem read as:

$$\begin{array}{lllll}
\max & \langle A, X \rangle & \qquad \min & \lambda & \\
\text{subject to} & \operatorname{trace}(X) = 1 & \qquad \text{subject to} & S = \lambda I - A & . \\
 & X \geq 0 & & S \geq 0 &
\end{array}$$

The solution X of the primal problem is given by $X = uu^*$, with u the normalized eigenvector associated with the largest eigenvalue λ_{max}. This λ_{max} is the solution of the dual problem. For $H = VXV^*$, with V orthonormal, the projected problem has the following form:

$$\begin{array}{lllll} \max & \langle V^*AV, H\rangle & \quad \min & \mu & \\ \text{subject to} & \operatorname{trace}(H) = 1 & \quad \text{subject to} & Z = \mu I - V^*AV & . \\ & H \geq 0 & & Z \geq 0 & \end{array}$$

The remaining problem is the construction of V. Although, we will not try to identify an appropriate V for the SDP projection framework, example 2.4.1 gives inspiration for further investigation of the projection idea for eigenspaces, as we will do in the next chapter. While writing this manuscript, work by Oliviera et al. [**33**] came to our attention. They also use the SDP projection framework, made it work for specific SDP approximations of graph partitioning problems.

CHAPTER 3

The eigenvalue approach

In this chapter we introduce a method for solving semidefinite programming problems using an eigenvalue formulation. The approach is based on rewriting a semidefinite constraint as an eigenvalue constraint. It uses a second order approximation of the objective function in combination with a Newton method, so fast convergence may be expected. The method may be, because of the properties of eigenvalues, "sparsified" and therefore could be very attractive for solving large sparse semidefinite optimization problems. For dense matrices the method appears to be not more expensive than standard interior-point methods for semidefinite programming problems. Since the method works with projections onto the space of orthonormal eigenvectors it can be the basis for a subspace algorithm for solving large semidefinite programming problems. The combination with subspace reduction methods lead to a form of the above-mentioned method that is computationally cheaper than the standard primal-dual interior-point method described in Section 2.2.

3.1. Semidefinite programs and eigenvalue computation

In the first section we will describe the eigenvalue formulation, properties of a relevant Lagrange multiplier, and derivatives of eigenvalue functions.

3.1.1. Eigenvalue formulation for (D)

We return to the semidefinite programming problem formulations (P) and (D), defined in Subsection 1.2.3. We make some assumptions on the solvability of the problem, namely we assume that both (P) and (D) are solvable and that at least one of (P) and (D) has a strictly feasible solution. As a consequence we then have a zero duality gap at optimality (see Subsection 1.2.3). Thus for a primal optimal solution X^+ and any dual optimal solution pair (y^+, S^+) we have, that

$$\langle C, X^+\rangle - \langle b, y^+\rangle = X^+S^+ = 0. \tag{84}$$

Since $X^+, S^+ \geq 0$, all eigenvalues of X^+, S^+ are (real and) greater than or equal to zero. Moreover, $X^+S^+ = 0 \Leftrightarrow S^+X^+ = (S^+)^*(X^+)^* = (X^+S^+)^* = 0^* = 0$, and therefore the optimal solutions X^+ and S^+ commute.

We will focus on the problem formulation (D). Problem (D) can be reformulated as an eigenvalue optimization problem by modeling $S \geq 0$ as $\lambda_{\max}(-S) \leq 0$. We assume that the optimal solution of the primal problem (P) is nontrivial: $X^+ \neq 0$. Therefore, the condition $\lambda_{\max}(-S) \leq 0$ can be rewritten as an equality constraint

$\lambda_{\max}(-S) = 0$ since S^+ has to be singular on account of (84). This leads to the following equivalent formulation[2],

$$\text{(DE)} \quad \begin{array}{ll} \min & \langle b, y\rangle, \\ \text{subject to} & \lambda_{\max}(C - \mathcal{A}^*(y)) = 0. \end{array}$$

The Lagrangian for this problem is

$$\mathcal{L}(y, a) = a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y\rangle.$$

The eigenvalue function $y \to \lambda_{\max}(C - \mathcal{A}^*(y))$ is a convex non-smooth function. To show the convexity, we first note that for $A \in \mathsf{S}^n$ the Rayleigh quotient form of $A \to \lambda_{\max}(A)$ is

$$\lambda_{\max}(A) = \max_{x^*x=1} \quad x^*Ax.$$

Now we check the definition (1) and (2) in Section 1.1.7 for convex functions:

(1) S^n is a convex set and
(2) $\forall A_1, A_2 \in \mathsf{S}^n$ and $\tau \in [0, 1]$ such that

$$\begin{array}{rcl} \lambda_{\max}(A_1 + \tau(A_2 - A_1)) & = & \displaystyle\max_{x^*x=1} \quad x^*A_1x + \tau x^*(A_2 - A_1)x \\ & = & \displaystyle\max_{x^*x=1} \quad (1-\tau)x^*A_1x + \tau x^*(A_2)x \\ & \le & \displaystyle\max_{x^*x=1} \quad (1-\tau)x^*A_1x \quad + \quad \max_{x^*x=1} \quad \tau x^*(A_2)x \\ & = & \lambda_{\max}(A_1) + \tau(\lambda_{\max}(A_1) - \lambda_{\max}(A_2)). \end{array}$$

Hence, $\lambda_{\max}$ is indeed convex, and since $y \to C - \mathcal{A}^*(y)$ is affine, we may conclude that $y \to \lambda_{\max}(C - \mathcal{A}^*(y))$ is convex. Non-smoothness will be discussed in Subsection 3.1.2.

The Lagrange multiplier rule, see Subsection 1.2.1, assures that the minimum of the Lagrangian,

$$\text{(E)} \quad \min_{y \in \mathbb{R}^m} \quad a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y\rangle,$$

combined with the constraint equation determines all unknowns, if the function is differentiable. The optimal value of (E) will be equal to the optimal value of (DE). It states that there is a multiplier a^+ for which $\nabla_y \mathcal{L}(y, a) = 0$ in the point (y^+, a^+). Since $\mathcal{L}(y, a)$ is non-smooth, we will go into detail about the meaning of the gradient of $\mathcal{L}(y, a)$ in the following subsection. If $\mathcal{L}(y, a)$ is non-differentiable, in case of multiple eigenvalues, we will concentrate directly on (E). Now, combined with the constraint equation we find the first order conditions for a solution of (DE),

$$\text{(85)} \qquad \begin{array}{ll} \nabla_y \Big(a^+\lambda_{\max}(C - \mathcal{A}^*(y^+)) + \langle b, y^+\rangle\Big) & = 0, \\ \lambda_{\max}(C - \mathcal{A}^*(y^+)) & = 0. \end{array}$$

In general we do not know the correct value for a^+ in advance. However, if

$$\text{(86)} \qquad \exists z : \ \mathcal{A}^*(z) = I, \quad \text{then} \quad a^+ = \langle b, z\rangle.$$

[2]The most natural equivalent formulation of (D) would be to maximize $\langle b, y\rangle$ such that $\lambda_{\max}(\mathcal{A}^*(y) - C) = 0$, but since in literature, [**20**], [**19**],[**34**], [**18**], and [**39**], (DE) is used, we adept this form.

To see this, assume that $\mathcal{A}^*(z) = I$. Then

$$\begin{array}{rlll} \min_{y\in\mathbb{R}^m} \mathcal{L}(y,a) = & \min_{\varepsilon,y\in\mathbb{R}^m} & a\lambda_{\max}(C - \mathcal{A}^*(y+\varepsilon z)) + \langle b, y+\varepsilon z\rangle & = \\ & \min_{\varepsilon,y\in\mathbb{R}^m} & a\lambda_{\max}(C - \mathcal{A}^*(y) - \varepsilon I) + \langle b, y\rangle + \varepsilon\langle b, z\rangle & = \\ & \min_{\varepsilon,y\in\mathbb{R}^m} & a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y\rangle + \varepsilon(\langle b, z\rangle - a). & \end{array} \tag{87}$$

Hence, the minimum of (87) is finite only if $\langle b, z\rangle - a = 0$ (since the optimal value of (E) equals the optimal of value (DE)).

In [**20**] Helmberg and Rendl use formulation (E) to construct a method, the so-called *spectral bundle method*, for solving semidefinite programming problems by projecting on eigenspaces, we will also use (E) to construct our method that projects on eigenspaces. For a comparison between the spectral bundle method and ours, see Subsection 3.6.1.

3.1.2. Derivatives of eigenvalue functions

If we want to solve the new problem formulation (E) by exploiting second order approximations, then we encounter first order and second order derivatives of the function $\lambda_{\max}$, or to be more specific of the function

$$a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y\rangle.$$

This section will be devoted to first and second order derivatives of the eigenvalues of the function $\mathcal{A}^*(y)$.

Recall that $\mathcal{A}^*(y)$ is of the form

$$\mathcal{A}^*(y) = \sum_{i=1}^{m} y_i A_i, \qquad \text{with} \qquad A_i \in \mathsf{S}^n.$$

Denote the eigenvalues of $\mathcal{A}^*(y)$ in by

$$\widehat{\lambda}_1(y) \geq \cdots \geq \widehat{\lambda}_n(y), \tag{88}$$

and the orthonormal eigenvectors by

$$\widehat{u}_1(y), \ldots, \widehat{u}_n(y).$$

The eigenvalue functions $\widehat{\lambda}_i(y)$ are continuous in y [**25**]. The ordering is natural for us since we are interested in $\lambda_{\max}(C - \mathcal{A}^*(y))$. The numbering of the eigenvalues is simple but not always convenient, because the $\widehat{\lambda}_i(y)$ are not necessarily differentiable. For example, consider the graphs of the eigenvalues of $\mathcal{A}^*(y_0 + td)$ as functions of $t \in \mathbb{R}$. The graphs may *cross* each other at some value of t. If such a crossing takes place, the graph of $\widehat{\lambda}_i(C - \mathcal{A}^*(y))$ jumps from one curve to another, making a corner at the crossing point. Summarizing, $\widehat{\lambda}_i(C - \mathcal{A}^*(y))$ depend continuously on y, but not necessarily differentiable. Note that with this ordering

$$\lambda_{\max}(\mathcal{A}^*(y)) := \widehat{\lambda}_1(\mathcal{A}^*(y)).$$

However, if we define an ordering of the eigenvalues $\lambda_i(\mathcal{A}^*(y)) = \lambda_i(y)$ in $y = y_0 + td$ for $t = 0$ by

$$\lambda_1(y_0) \geq \cdots \geq \lambda_n(y_0) \tag{89}$$

and the orthonormal eigenvectors by

$$u_1(y_0), \ldots, u_n(y_0),$$

then the eigenvalues $\lambda_i(\mathcal{A}^*(y_0 + td))$ and the eigenvectors $u_i(\mathcal{A}^*(y_0 + td))$ can be selected to be analytic functions over t [**25**, Thm 6.1, p.120] with in $t = 0$ as prescribed in (89). The ordering depends on y_0 and d. We will call y_0 a *(starting) point*, t the *step size*, and d the *direction*. The analytic functions $\lambda_i(\mathcal{A}^*(y_0 + td))$ over t will be *the eigenvalue curves* of λ_i, and if eigenvalue curves cross in $\bar{t}$ then $\bar{t}$ is the *crossing (point)*. Note that in this situation

$$\lambda_{\max}(\mathcal{A}^*(y_0 + td)) := \max_i(\lambda_i(\mathcal{A}^*(y_0 + td))),$$

which we will refer to as the *maximum eigenvalue curve*. Define

$$\Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \quad \text{and} \quad U := (u_1, \ldots, u_n).$$

If we consider the eigenvalues of $\mathcal{A}^*(y)$ as functions of y, then we will use the ordering (88) for the eigenvalues $\widehat{\lambda}_i(y)$, and denote $\widehat{\lambda}_i(y)$ by $\lambda_i(y)$ (or by $\lambda_i(\mathcal{A}^*(y))$), and the eigenvectors $\widehat{u}_i(y)$ by $u_i(y)$.

If we consider the eigenvalues of $\mathcal{A}^*(y)$ in $y = y_0 + td$ as functions of t, then we will use the ordering (89) defined in $y = y_0$ for the eigenvalues $\lambda_i(y_0 + td)$, and denote the eigenvalues by λ_i in $y = y_0$ and $\lambda_i(\mathcal{A}^*(y_0 + td))$ in $y = y_0 + td$, and the eigenvectors $u_i(y_0)$ by u_i.

Note that $\lambda_i = \lambda_i(y)$ in $y = y_0$.

LEMMA 3.1.1. *Suppose λ_i is simple for all i. Then the pseudo inverse of $\mathcal{A}(y) - \lambda_i(y)I$ at $y = y_0$ is given by*

$$\Big(\mathcal{A}^*(y_0) - \lambda_i I\Big)^\dagger = \sum_{r \neq i} \frac{u_r u_r^*}{\lambda_r - \lambda_i}.$$

Proof. With (3), (4), and the observations that

$$u_r^* U = e_r^* \qquad \text{and} \qquad e_r^*\Big(\Lambda - \lambda_i I\Big)^\dagger = (\lambda_r - \lambda_i)^{-1} e_r^*,$$

we see that,

$$\begin{aligned}
\Big(\mathcal{A}^*(y_0) - \lambda_i I\Big)^\dagger &= U(\Lambda - \lambda_i I)^\dagger U^* \\
&= \sum_{r=1}^{n} u_r u_r^* U(\Lambda - \lambda_i I)^\dagger U^* \\
&= \sum_{r \neq i} \frac{u_r u_r^*}{\lambda_r - \lambda_i}
\end{aligned}$$

∎

LEMMA 3.1.2. *If λ_i is simple, then the formulae for the first and second order partial derivatives at $y = y_0$ are*

$$\frac{\partial \lambda_i(y_0)}{\partial y_k} = u_i^* A_k u_i. \tag{90}$$

$$\frac{\partial^2 \lambda_i(y_0)}{\partial y_k \partial y_l} = 2\mathrm{Re}(\sum_{r \neq i} \frac{u_i^* A_k u_r u_i^* A_l u_r}{\lambda_i - \lambda_r}) \tag{91}$$

Proof. Note that $\frac{\partial}{\partial y_k}(\mathcal{A}^*(y)) = A_k$ and $\frac{\partial^2}{\partial y_k \partial y_l}(\mathcal{A}^*(y)) = 0$. Since we have the eigenvalue assumption

$$(\mathcal{A}^*(y) - \lambda_i(y) I) u_i(y) = 0, \tag{92}$$

Equation (90) can be derived in the following way. We see that

$$0 = \frac{\partial}{\partial y_k}\Big((\mathcal{A}^*(y) - \lambda_i(y) I) u_i(y)\Big) = \tag{93}$$

$$\Big(A_k - \frac{\partial \lambda_i(y)}{\partial y_k} I\Big) u_i(y) + \Big(\mathcal{A}^*(y) - \lambda_i(y) I\Big)\frac{\partial u_i(y)}{\partial y_k}.$$

Multiply on the left side with $u_i^*(y)$, then the second term equals zero by (92), and we are left with

$$u_i^*(y)\Big(A_k - \frac{\partial \lambda_i(y)}{\partial y_k} I\Big) u_i(y) = 0. \tag{94}$$

Equation (90) follows immediately.

For the derivation of (91), we look at the partial derivative $\frac{\partial}{\partial y_l}$ of (94) at $y = y_0$, this leads to:

$$0 = -u_i^*\Big(\frac{\partial^2 \lambda_i(y_0)}{\partial y_k \partial y_l} I\Big) u_i +$$

$$+u_i^*\Big(A_k - \frac{\partial \lambda_i(y_0)}{\partial y_k} I\Big)\frac{\partial u_i(y_0)}{\partial y_l} + u_i^*\Big(A_l - \frac{\partial \lambda_i(y_0)}{\partial y_l} I\Big)\frac{\partial u_i(y_0)}{\partial y_k} + \tag{95}$$

$$+u_i^*\Big(\mathcal{A}^*(y_0) - \lambda_i I\Big)\frac{\partial^2 u_i(y_0)}{\partial y_k \partial y_l} + \tag{96}$$

$$+\Big(\frac{\partial u_i(y_0)}{\partial y_l}\Big)^*\Big[\Big(A_k - \frac{\partial \lambda_i(y_0)}{\partial y_k} I\Big) u_i + \Big(\mathcal{A}^*(y_0) - \lambda_i I\Big)\frac{\partial u_i(y_0)}{\partial y_k}\Big]. \tag{97}$$

The term (96) is zero by assumption (92), the terms (97) sum up to zero because they satisfy (93) if $\frac{\partial u_i(y_0)}{\partial y_l} \neq 0$. We derive from (93) the equality

$$\frac{\partial u_i(y_0)}{\partial y_k} = -\Big(\mathcal{A}^*(y_0) - \lambda_i I\Big)^{\dagger}\Big[A_k - \frac{\partial \lambda_i(y_0)}{\partial y_k} I\Big] u_i.$$

If we plug this in the terms of (95), we see that

$$0 = -u_i^*\Big(\frac{\partial^2 \lambda_i(y_0)}{\partial y_k \partial y_l} I\Big) u_i +$$

$$-u_i^*\frac{\partial}{\partial y_k}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)\Big[\mathcal{A}^*(y_0)-\lambda_i I\Big]^\dagger\frac{\partial}{\partial y_l}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)u_i$$
$$-u_i^*\frac{\partial}{\partial y_l}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)\Big[\mathcal{A}^*(y_0)-\lambda_i I\Big]^\dagger\frac{\partial}{\partial y_k}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)u_i$$
$$=-2\mathrm{Re}\Big[u_i^*\frac{\partial}{\partial y_k}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)\Big[\mathcal{A}^*(y_0)-\lambda_i I\Big]^\dagger\frac{\partial}{\partial y_l}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)u_i\Big].$$

We observe that $u_i^*\Big(\frac{\partial}{\partial y_l}\lambda_i(y_0)I\Big)u_r=0$, if $i\neq r$, and we apply Lemma 3.1.1:

$$\frac{\partial^2}{\partial y_k\partial y_l}\lambda_i(y_0)=$$
$$-2\mathrm{Re}(\textstyle\sum_{r\neq i}u_i^*\frac{\partial}{\partial y_k}\Big(\mathcal{A}(y_0)^*-\lambda_i(y_0)I\Big)\frac{u_ru_r^*}{\lambda_r-\lambda_i}\frac{\partial}{\partial y_l}\Big(\mathcal{A}^*(y_0)-\lambda_i(y_0)I\Big)u_i)=$$
$$-2\mathrm{Re}(\textstyle\sum_{r\neq i}u_i^*A_k\frac{u_ru_r^*}{\lambda_r-\lambda_i}A_lu_i)=$$
$$2\mathrm{Re}(\textstyle\sum_{r\neq i}\frac{u_i^*A_ku_ru_i^*A_lu_r}{\lambda_i-\lambda_r}).$$

□

3.2. A Newton-type method for solving (E): simple eigenvalue case

In this section we introduce an iterative approach for solving (E). Suppose we are in a point y_0 with

$$\phi_+:=a\lambda_{\max}(C-\mathcal{A}^*(y_0))+\langle b,y_0\rangle.$$

We are interested in locating a *descent direction* d, that is a direction d for which

$$a\lambda_{\max}(C-\mathcal{A}^*(y_0+td))+\langle b,y_0+td\rangle<\phi_+, \tag{98}$$

for some step size $t>0$. If td is such that (98) holds, we say that we made *progress* for the *update* td in the process. We construct a, if possible, local second order approximation of the function $a\lambda_{\max}(C-\mathcal{A}^*(y))+\langle b,y\rangle$, derive a descent direction d from this local approximation, and an estimate for the Lagrange multiplier a, and then compute a step size t such that progress is made for td.

The difficulty in minimizing the maximum eigenvalue of a sum of matrices is that the maximum eigenvalue function may not be differentiable in all of its points, since the eigenvalues as functions of y are not differentiable quantities at points were they coincide. The approach that we will introduce for the multiple eigenvalues is more complex than the approach for the simple eigenvalues. The conclusions drawn for the simple eigenvalue case will more clear; we will see that there is more choice for computing, for example, the step size t. Therefore we believe that it is easier for the reader to understand the framework if we assume in this section that the maximum eigenvalue is simple at $y=y_0$, and refer to this section as the simple eigenvalue case. In the next section we will describe the so-called multiple eigenvalue case.

We proceed with the organization of this section. In Subsection 3.2.1 we will describe the computation of a step size, that is, we compute a $t^+>0$ such that for

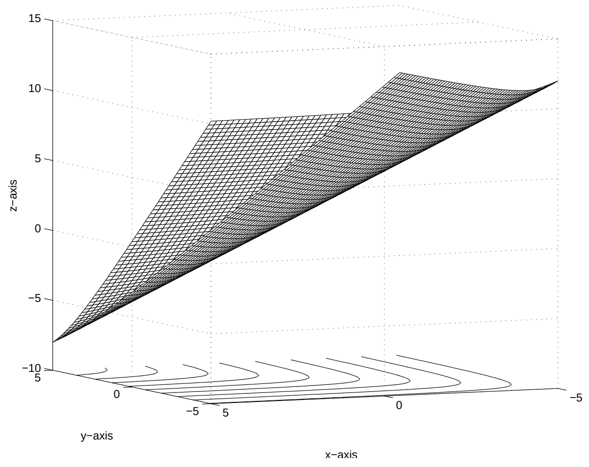

FIGURE 5 $z = \lambda_{\max}(C - xA_1 - yA_2)$.

a given direction d and a given constant for the Lagrange multiplier a

$$t^+ = \underset{t}{\operatorname{argmin}}\ a\lambda_{\max}(C - \mathcal{A}^*(y_0 + td)) + \langle b, y_0 + td\rangle. \tag{99}$$

We start with the computation of the step size instead of the computation of a search direction d, because computing a step size t is much simpler than computing a search direction d. In the first place, because the computation of t involves a 1-dimensional minimization problem, while the computation of d involves an m-dimensional optimization problem. The arguments that involve the solution of the 1-dimensional minimization problem for t will also play an important role in the solution of the m-dimensional minimization problem for the direction d. Moreover, for the computation of t we will assume that we have already a value for the Lagrange multiplier.

We motivate why we need a step size t. The step size t for a Newton update is known. Since any Newton step will be based on a local approximation of the function $\lambda_{\max}$ as a function over t, we do not know in advance if the Newton step size has also global relevance. The behavior of $a\lambda_{\max}(C - \mathcal{A}^*(y_0 + td)) + \langle b, y_0 + td\rangle$ as a function of t may be very wild: the neglected high order terms could be playing an important role, and other eigenvalue curves could cross the eigenvalue curve of λ_1, see Subsection 3.1.2.

If the curve $\lambda_{\max}(C - \mathcal{A}^*(y_0 + td))$ as a function of t has a quadratic term equal to zero we will call d a *linear direction.* So, if the second order term of our approximation is zero we are left with a linear direction for which we do not know what a reasonable value of t is since an optimal t for a linear approximation will probably be minus infinity.

To show that linear directions indeed exist, we give the following example.

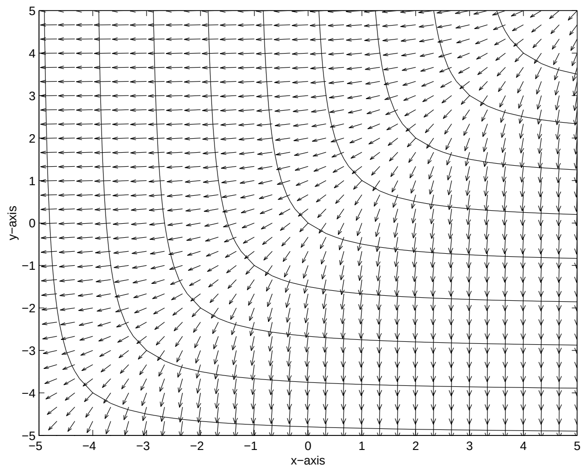

FIGURE 6 Field of gradients of $\lambda_{\max}(C - xA_1 - yA_2)$.

EXAMPLE 3.2.1.

$$C := \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}, \quad A_1 := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad A_2 := \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

We compute

$$z := \lambda_{\max}(C - xA_1 - yA_2) = \lambda_{\max} \begin{pmatrix} -x & -1 \\ -1 & -y \end{pmatrix},$$

which is plotted in Figure 5, and the field of gradients of $\lambda_{\max}(C - xA_1 - yA_2)$ in (x, y), which is plotted in Figure 6. In Figure 6, we see a linear direction from the top right angle to the bottom left angle, this is the trench that can be spotted in Figure 5.

We will compute the step size iteratively and with high accuracy.

Subsection 3.2.2 contains the construction of a second order local approximation of $a\lambda_{\max}(C - \mathcal{A}^*(y_0 + d)) + \langle b, y_0 + d\rangle$ as a function of d. This approximation consists of a linear and a quadratic term, and the optimality condition is a linear equation in d and a. With the properties of this equation an approximated Lagrange multiplier and two candidate search directions d are determined in Subsection 3.2.3. The strategy is based on keeping the next step $y_0 + td$ as near as possible to y_0, by using a and the best possible direction d. The last part of Subsection 3.2.3 consists of some practical notes about implementing the strategies.

From the two candidate search directions in Subsection 3.2.3, we have to select the best one every step of the iterative procedure. Therefore, we motivate in Subsection 3.2.4, which one we will select in what situation.

Our method finds an approximate optimal value for (E) that is also an approximate optimal value for (D), but the argmin $\widehat{y}$ for (E) computed by our eigenvalue method has not by definition the property that $\lambda_{\max}(C - \mathcal{A}^*(\widehat{y})) = 0$, thus is not by definition a solution to (D). Subsection 3.2.5 describes the computation of an approximate optimal y^+, for optimization problem (D).

In Subsection 3.2.6 we illustrate the approach followed in the other subsections, by solving some examples. The conclusions indicate the need for Section 3.3.

3.2.1. Computation of the step size t

For ease of notation we introduce new matrices $\widehat{C}$ and $\widehat{F}$:

$$\widehat{C} := aC - a\mathcal{A}^*(y_0) + \langle b, y_0\rangle I, \qquad \widehat{F} := -a\mathcal{A}^*(d) + \langle b, d\rangle I,$$

and use that $\lambda(A + \alpha I) = \lambda(A) + \alpha$. Note that the term $\langle b, y_0\rangle I$ does not play any role in the minimization process, we included it here for completeness of notation. Suppose that we have computed a direction d and that we have a value for a. We are interested in a step size t such that

$$t = \operatorname*{argmin}_t \lambda_{\max}(\widehat{C} + t\widehat{F}). \tag{100}$$

The first step is: set $t^{\text{it}} = 0$, and construct an eigenvalue decomposition (3) of $\widehat{C}$,

$$\widehat{C} = U\Lambda U^*, \qquad U^*U = I.$$

The matrix Λ is a diagonal matrix $\operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with eigenvalues $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$ of $\widehat{C}$ on its diagonal. The matrix $U = (u_1, \ldots, u_n)$ consists of the orthonormalized eigenvectors of $\widehat{C}$. We use this decomposition to rewrite $\widehat{C} + t\widehat{F}$ as

$$\rho := \lambda_{\max}(\widehat{C} + t\widehat{F}) = \lambda_{\max}(U\Lambda U^* + t\widehat{F}) = \lambda_{\max}(\Lambda + tU^*\widehat{F}U).$$

We define $F = (f_{ij})_{i,j=1}^n := U^*\widehat{F}U$ and proceed with the following partitioning

$$\Lambda + tF =: \left(\begin{array}{c|c} \lambda_1 & 0 \\ \hline 0 & \widehat{\Lambda} \end{array}\right) + t\left(\begin{array}{c|c} \mu_1 & e^* \\ \hline e & \widehat{R} \end{array}\right) = \left(\begin{array}{c|c} \lambda_1 + t\mu_1 & te^* \\ \hline te & R \end{array}\right), \tag{101}$$

where μ_1 is the element f_{11} of the matrix F. The vector e is a column-vector, $e = (f_{21}, \ldots, f_{n1})^*$. The matrix R can be expressed as $R = \widehat{\Lambda} + t\widehat{R}$, with $\widehat{\Lambda} = \operatorname{diag}(\lambda_2, \ldots, \lambda_n)$, and the matrix $\widehat{R}$ is the $(n-1) \times (n-1)$ matrix $(f_{ij})_{i,j=2}^n$. We are looking for the minimal maximum eigenvalue ρ of the matrix $\Lambda + tF$ as a function of t. If the first element of the eigenvector corresponding to ρ is nonzero:

$$\left(\begin{array}{c|c} \lambda_1 + t\mu_1 & te^* \\ \hline te & R \end{array}\right)\begin{pmatrix} 1 \\ z \end{pmatrix} = \rho\begin{pmatrix} 1 \\ z \end{pmatrix}. \tag{102}$$

If in addition ρ is not in the spectrum of R, then this system leads to the following equations,

$$\lambda_1 + t\mu_1 + te^*z = \rho, \tag{103}$$

$$(R - \rho I)z = -te. \tag{104}$$

Note that $\rho = \lambda_1 + t\mu_1$, if $e = 0$. Substituting (104) into (103) gives

$$\rho - (\lambda_1 + t\mu_1) = -t^2e^*(R - \rho I)^{-1}e. \tag{105}$$

To make (105) suitable for numerical computation, we have to eliminate the variable ρ in the right hand side of (105). We first make an approximation of $(R-\rho I)^{-1}$ by expanding it around $\lambda_1 + t\mu_1$:

$$(R-\rho I)^{-1} = (R-(\lambda_1+t\mu_1)I)^{-1} + \mathcal{O}(\rho-(\lambda_1+t\mu_1)). \tag{106}$$

If we substitute this into (105) and take absolute values at both sides, then

$$\begin{aligned} |\rho-\lambda_1-t\mu_1| &= |-t^2e^*(R-(\lambda_1+t\mu_1)I)^{-1}e + \mathcal{O}(t^2|\rho-(\lambda_1+t\mu_1)|)| \\ &\le c_1\cdot t^2 + c_2\cdot t^2|\rho-(\lambda_1+t\mu_1)|, \text{ with } c_i\ge 0,\ i=1,2. \end{aligned}$$

Hence, for $|t|$ small

$$|\rho-(\lambda_1+t\mu_1)| \le ct^2$$

This makes it possible to rewrite (106):

$$(R-\rho I)^{-1} = (R-(\lambda_1+t\mu_1)I)^{-1} + \mathcal{O}(t^2) \quad (t\to 0). \tag{107}$$

We define the matrix Γ as

$$\Gamma := \operatorname{diag}(\lambda_1-\lambda_2,\ldots,\lambda_1-\lambda_n).$$

Since we assumed that λ_1 is simple, we see that Γ is invertible. Furthermore, we want to avoid explicit computation of $(R-(\lambda_1+t\mu_1)I)^{-1}$. This is circumvented by a Neumann series

$$\begin{aligned} (R-(\lambda_1+t\mu_1)I)^{-1} &= (-\Gamma+t\widehat{R}-t\mu_1 I)^{-1} \\ &= -\Gamma^{-1}(I-t(\widehat{R}-\mu_1 I)\Gamma^{-1})^{-1} \\ &= -\Gamma^{-1} - t\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1} + \mathcal{O}(t^2) \quad (t\to 0). \qquad (108) \end{aligned}$$

After substitution of (108) and (107) in (105), we find a third-order approximation of the maximum eigenvalue ρ of $\Lambda + tF$:

$$\rho = \lambda_1 + t\mu_1 + t^2e^*\Gamma^{-1}e + t^3e^*\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1}e. \tag{109}$$

With this local approximation of $\lambda_{\max}(\widehat{C}+t\widehat{F})$ we can approximate t by

$$\widehat{t} = \operatorname*{argmin}_t \Big(\lambda_1 + t\mu_1 + t^2e^*\Gamma^{-1}e + t^3e^*\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1}e\Big). \tag{110}$$

This is equivalent to computing the zeros of the derivative of the function under the minimum in (110),

$$\mu_1 + 2te^*\Gamma^{-1}e + 3t^2e^*\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1}e = 0. \tag{111}$$

The computation of the zeros of (111) can be done analytically:

$$t_{1,2} = \frac{-e^*\Gamma^{-1}e \pm \sqrt{(e^*\Gamma^{-1}e)^2 - 3\mu_1 e^*\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1}e}}{3e^*\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1}e}.$$

To determine whether $\widehat{t}=t_1$ or $\widehat{t}=t_2$ is a minimum of (110), we compute the second derivative of the function under the minimum in (110) and check whether the second derivative is positive in t_1 and t_2 :

$$2e^*\Gamma^{-1}e + 6t_ie^*\Gamma^{-1}(\widehat{R}-\mu_1 I)\Gamma^{-1}e \ge 0 \quad \text{for} \quad i=1,2.$$

We have to check if we made any progress for $\widehat{t}$:

$$\lambda_{\max}(\widehat{C} + \widehat{t}\widehat{F}) < \lambda_{\max}(\widehat{C}),$$

if so, we update $\widehat{C} \leftarrow \widehat{C} + \widehat{t}\widehat{F}$ and $t^{\text{it}} \leftarrow t^{\text{it}} + \widehat{t}$, and start again.

If any eigenvalue curve crosses the eigenvalue curve of λ_1, in say $t = t^+$, then there is a change of eigenvector for λ_1 at $t = t^+$, and we are certain that the approximation of λ_1 is inappropriate for $t \geq t^+$, since the approximation is based on the eigenvector of the maximum eigenvalue at $t = 0$. So, if we did not make any progress we could be in this situation sketched in Figure 7. On the horizontal axis are the values for t. The lines indicate eigenvalue curves, $t \rightarrow \lambda_i(\widehat{C} + t\widehat{F})$. We approximate the maximum eigenvalue curve around $t = 0$ with a second or third order curve, and start in $\circ$ $(t = 0)$. We see in this figure that we did not make any progress in the minimum $\square$ of the "parabola", the actual maximum eigenvalue is $*$, which is larger than $\circ$. At $+$ the eigenvalue curves cross. We have to check if any eigenvalue curve (λ_i, u_i) crosses the eigenvalue curve (λ_1, u_1) for $t \in [0, \widehat{t}]$.

We estimate the crossing of eigenvalue curves by approximating the eigenvalues by linear functions in t, and compute their intersections (at $\widetilde{t}$):

$$\lambda_1 + \widetilde{t}u_1^* F u_1 = \lambda_i + \widetilde{t}u_i^* F u_i \qquad \Rightarrow \qquad \widetilde{t} = \frac{\lambda_1 - \lambda_i}{u_i^* F u_i - u_1^* F u_1}$$

If one of the computed $\widetilde{t}$'s is in the interval $\widetilde{t} \in [0, \widehat{t}]$, we set $\widehat{t} = \widetilde{t}$, and compute if we made any progress for $\widehat{t}$. If we made progress, we update $\widehat{C} \leftarrow \widehat{C} + \widehat{t}\widehat{F}$ and $t^{\text{it}} \leftarrow t^{\text{it}} + \widehat{t}$, and start again.

If we did not make any progress for $\widehat{t}$, we start with $\widehat{t}$ a bisection procedure: take $\widehat{t} = \frac{1}{2}\widehat{t}$, if any progress is made, we update $\widehat{C} \leftarrow \widehat{C} + \widehat{t}\widehat{F}$ and $t^{\text{it}} \leftarrow t^{\text{it}} + \widehat{t}$, and start again, if not we repeat the bisection.

We will continue the process of determining the step size until $\widehat{t} < \varepsilon$, then our approximated step size is t^{it}

Note that the process for locating t is a computationally expensive process, due to the computation of full sets of eigenvalues and eigenvectors. Later on, when we introduce a subspace procedure in Section 3.5, we will see that the computation of full sets of eigenvalues and eigenvectors can be restricted to a low dimensional matrix.

3.2.2. Computation of a direction d

The idea behind computing a direction is the same as the idea behind computing the step size t.

Start again with computing an eigenvalue decomposition of $\widehat{C}$. Then define a partitioning of $-\widehat{\mathcal{A}}^*(d) + b^* dI$ and finally construct an approximation of the maximum eigenvalue in terms of d. The minimization of this approximation is equivalent to a Newton method with some dependency on the Lagrange multiplier a.

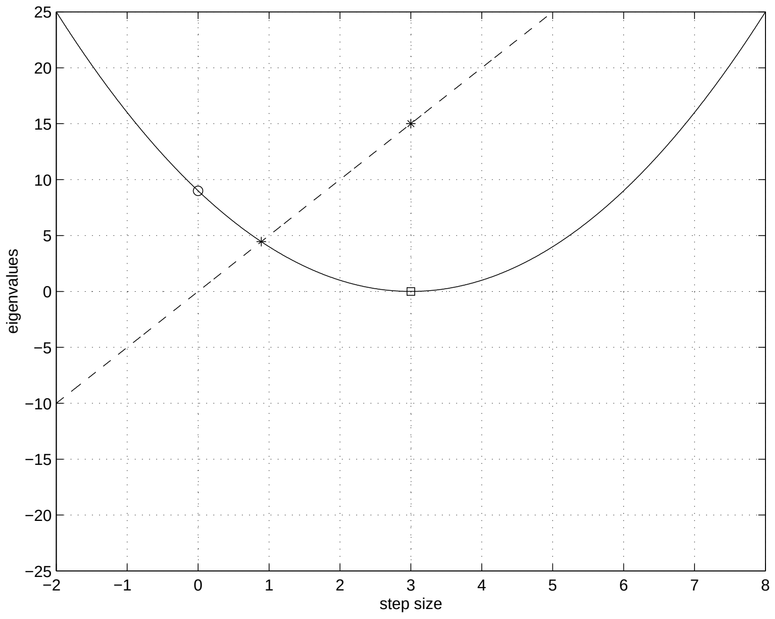

FIGURE 7 Crossing eigenvalue curves.

Suppose we are in the situation as sketched for (98): we have some given y_0, for which $\lambda_{\max}(\widehat{C})$ is simple, and some given $a > 0$,[3] and we search a d such that

$$d = \operatorname*{argmin}_{d} \lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(d) + b^* dI), \tag{112}$$

where

$$\widehat{C} := C - \mathcal{A}^*(y_0) + \frac{1}{a}\langle b, y_0\rangle I. \tag{113}$$

Note that $\widehat{C}$ is slightly different. Since a is part of the problem, we want to keep a visible. We do not know a correct value for a in general, in the next Subsection we will compute an estimate for a based on the philosophy to keep the maximal change of

$$a\lambda_{\max}(C - \mathcal{A}^*(y_0 + td)) + \langle b, y_0 + td\rangle$$

as small as possible and minimize the effect of unwanted directions.

Again in the minimization problem the term $\langle b, y_0\rangle I$ does not play any role, and we could have omitted it, but we kept it for completeness of notation.

The first step is to construct an eigenvalue decomposition of $\widehat{C}$:

$$\widehat{C} = U\Lambda U^* = \sum_{i=1}^{n} \lambda_i u_i u_i^*, \quad \text{with} \quad I = U^*U = \sum_{i=1}^{n} u_i u_i^*.$$

[3]We assume $a > 0$, in our experiments we never encountered $a < 0$, therefore we did not build any procedure for this situation.

We use this decomposition to for

$$a\rho := \lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(d) + b^*dI) = \lambda_{\max}(a\Lambda - aU^*\mathcal{A}^*(d)U + b^*dI).$$

We introduce the following matrix partitioning for

$$-aU^*\mathcal{A}^*(d)U + (b^*d)I := \left(\begin{array}{c|c} -af^*d + b^*d & -a(Ed)^* \\ \hline -aEd & \widehat{R} \end{array}\right). \tag{114}$$

Since it is not immediately clear what the meaning is of the several new characters in expression (114), we will explain this first. Construct an $n \times m$ matrix $F = (f_{ij})_{i,j=1}^{n,m}$ such that,

$$F := (U^*A_1u_1, \dots, U^*A_mu_1).$$

It is now easy to see how f is defined:

$$f = (u_1^*A_1u_1, \dots, u_1^*A_mu_1)^* = F^*e_1.$$

The matrix E is the $(n-1) \times m$ matrix $(f_{ij})_{i=2,j=1}^{n,m}$.

We want the minimal maximum eigenvalue ρ of the matrix $a\Lambda - aU^*\mathcal{A}^*(d)U + (b^*d)I$ as a function of d. If the first element of the eigenvector corresponding to $a\rho$ is nonzero we have a formula for $a\rho$,

$$\left(\begin{array}{c|c} a\lambda_1 - af^*d + b^*d & -a(Ed)^* \\ \hline -aEd & R \end{array}\right)\begin{pmatrix} 1 \\ z \end{pmatrix} = a\rho \begin{pmatrix} 1 \\ z \end{pmatrix},$$

where $R = a\widehat{\Lambda} + \widehat{R}$ with $\widehat{\Lambda} = \operatorname{diag}(\lambda_2, \dots, \lambda_n)$. Note that $te = -Ed$ and $t\mu_1 = (-f + \frac{1}{a}b)^*d$ in the notation of Subsection 3.2.1. If $a\rho$ is not in the spectrum of R, then

$$a\lambda_1 + (-af^* + b^*)d - ad^*E^*z = a\rho, \tag{115}$$

$$(R - a\rho I)z = aEd. \tag{116}$$

A combination of (115) and (116) leads to

$$a(\rho - \lambda_1) = (-af^* + b^*)d - a^2d^*E^*(R - a\rho I)^{-1}Ed. \tag{117}$$

Note that $d \in \mathsf{Ker}(E)$ implies that

$$a\rho = a\lambda_1 + (-af^* + b^*)d, \tag{118}$$

thus $a\rho$ depends linearly on d in this case. To make (117) useful for computation, we eliminate the variable $a\rho$ in the right hand side of (117). With the same arguments as for (106) and (107), we see that

$$(R - a\rho I)^{-1} = (R - a\lambda_1 I)^{-1} + \mathcal{O}(a\|d\|) \quad (\|d\| \to 0). \tag{119}$$

As before

$$\Gamma := \operatorname{diag}(\lambda_1 - \lambda_2, \dots, \lambda_1 - \lambda_n).$$

Furthermore, we want to avoid the explicit computation of $(R - a\lambda_1 I)^{-1}$. This is circumvented by a Neumann series:

$$\begin{aligned}(R - a\lambda_1 I)^{-1} &= -(a\Gamma - \widehat{R})^{-1} \\ &= \frac{-1}{a}\Gamma^{-1}(I - \frac{1}{a}\widehat{R}\Gamma^{-1})^{-1} \\ &= \frac{-1}{a}\Gamma^{-1} - \frac{1}{a^2}\Gamma^{-1}\widehat{R}\Gamma^{-1} + \mathcal{O}(\frac{1}{a^3}\|d\|^3) \quad (\|d\| \to 0). \qquad (120)\end{aligned}$$

By substituting (120) and (119) in (117), we obtain the following expression for the maximum eigenvalue of $a\Lambda - aU^*\mathcal{A}^*(d)U + (b^*d)I$:

$$a\lambda_1 + (-af^* + b^*)d + ad^*E^*\Gamma^{-1}Ed + \mathcal{O}(\|d\|^3) \quad (\|d\| \to 0). \tag{121}$$

The direction d and Lagrange multiplier a in (112) can now be estimated from

$$d = \underset{d}{\operatorname{argmin}} \Big(a\lambda_1 + (-af^* + b^*)d + ad^*E^*\Gamma^{-1}Ed\Big). \tag{122}$$

The optimality condition for (122) is

$$(-af + b) + 2aE^*\Gamma^{-1}Ed = 0 \tag{123}$$

The matrix $E^*\Gamma^{-1}E$ is positive definite, because Γ has only positive elements on the diagonal. In the next subsection we will study this linear system, for example, $E^*\Gamma^{-1}E$ can be singular, which is an interesting situation.

3.2.3. Solving the linear system in (123)

Before solving the linear system

$$E^*\Gamma^{-1}Ed = -\frac{1}{2}(-f + \frac{b}{a}), \tag{124}$$

it is useful to study properties of the matrix $H := E^*\Gamma^{-1}E$. Note that $\mathsf{Ker}\,(H) = \mathsf{Ker}\,(E)$, and that

$$\mathsf{Im}\,(H) = \mathsf{Ker}\,(H^*)^\perp = \mathsf{Ker}\,(H)^\perp.$$

The matrix E is an $n\times(m-1)$ rectangular matrix depending on the order $(n\times n)$ and the amount (m) of the input data A_i, for example, in all combinatorial optimization problems that have been used for testing the spectral bundle method in [**20**] $n = m$.

We can immediately identify two different types of search directions:

- $d \in \mathsf{Ker}\,(E)$,
- $d \perp \mathsf{Ker}\,(E)$.

A search direction $d \in \mathsf{Ker}\,(E)$ is a linear direction, since the quadratic part of (121) is zero (see (118)). Define $f := f_E + f_{E^\perp}, b := b_E + b_{E^\perp}$, where f_E, b_E are the components of f, b in $\mathsf{Ker}\,(E)$ and $f_{E^\perp}, b_{E^\perp}$ the components of $f, b \perp \mathsf{Ker}\,(E)$. The direction of this type that leads locally to the largest decrease of (98) can be computed as

$$d_1 = \frac{af_E - b_E}{\|af_E - b_E\|}. \tag{125}$$

We call this direction the *steepest linear direction.* Unfortunately, we do not always know a beforehand. We consider the linear directions as undesirable, since they are likely to lead to large updates td. We can try to minimize the effect of linear directions by some suitable choice of the Lagrange multiplier. We have two strategies to keep the next step as near as possible to y_0, by using a and the best direction d:

(a) choose a Lagrange multiplier a^+ such that the maximal change of

$$a\lambda_{\max}(C - \mathcal{A}^*(y_0 + td)) + \langle b, y_0 + td\rangle$$

is as small as possible.

(b) choose the Lagrange multiplier a^+ to minimize the *potential dangerous* linear directions: if, for example, there exists an z such that $\mathcal{A}^*(z) = I$ then $z \in \mathsf{Ker}\,(E)$, thus z is purely linear and the eigenvalues curves are all parallel in z. We will call z a *potential dangerous* direction.

Let us take a closer look at strategy (a). The largest change in $a\lambda_{\max}(C - \mathcal{A}^*(y_0 + td)) + \langle b, y_0 + td\rangle$ is for given a and $\mathsf{Ker}\,(E) \neq 0$, equal to:

$$|t| \max_{d \in \mathsf{Ker}(E), \|d\|=1} |(-af^* + b^*)d| + \mathcal{O}(t^2).$$

So, we will concentrate finding an a that minimizes

$$\max_{d \in \mathsf{Ker}(E), \|d\|=1} (-af^* + b^*)d.$$

Because we are looking for a direction d, and not for a specific update td we assumed that $\|d\| = 1$. Since (see (125))

$$\max_{d \in \mathsf{Ker}(E), \|d\|=1} (-af^* + b^*)d = \langle \frac{-af_E + b_E}{\| - af_E + b_E\|}, -af + b\rangle = \| - af_E + b_E\|, \tag{126}$$

we see that $\| - af_E + b_E\|$ is minimal with respect to a if $-af_E + b_E \perp f_E$:

$$a^+ = \operatorname*{argmin}_a \| - f_E + \frac{b_E}{a}\| \qquad \Rightarrow \qquad a^+ = \frac{\langle b_E, f_E\rangle}{\|f_E\|^2}. \tag{127}$$

It is important to check that $\|f_E\|$ is of 'reasonable' size, because a large a^+ can effect the accuracy. In our examples this did never occur.

A direction of the type $d \perp \mathsf{Ker}\,(E)$ can be computed as

$$d_2 = -\frac{1}{2}(E^*_{E^\perp}\Gamma E_{E^\perp})^\dagger(-f_{E^\perp} + \frac{b_{E^\perp}}{a^+}). \tag{128}$$

How this is done is explained in the last part of this Subsection.

For a non-singular E we did not yet propose any strategy. However, for a non-singular E the solution of (121) can be approximated by the solution of the quadratic part of (121) (the Gauss-Newton approach). This should give us a descent direction d:

$$E^*\Gamma^{-1}Ed = \frac{-1}{2}(-f + \frac{b}{a}), \tag{129}$$

Again we follow strategy (a), we choose a, for given d as in (129), such that

$$a^+ = \operatorname*{armin}_a \langle -af + b, d\rangle. \tag{130}$$

To tackle this minimization problem we first solve d with respect to f and b,

$$\begin{array}{rcl} E^*\Gamma^{-1}Ed_f & = & \frac{1}{2}f, \\ E^*\Gamma^{-1}Ed_b & = & \frac{1}{2}b. \end{array}$$

Then determine a, as in (130), such that

$$a = \underset{a}{\operatorname{argmin}} \ \langle -af + b, d_f - \frac{d_b}{a} \rangle \qquad \Rightarrow \qquad a = \sqrt{\frac{b^*d_b}{f^*d_f}}. \tag{131}$$

We finally form $d_2 = d_f - \frac{d_b}{a}$. This completes the description of our strategy, we will now discuss some specific cases.

Specific cases. Note that if $\dim(\mathsf{Ker}\,(E)) = 1$, then

$$b_E - a^+ f_E = 0 \quad \text{and} \quad (-a^+ f + b) \in \mathsf{Ker}\,(E)^\perp = \mathsf{Im}\,\big(E^*\Gamma^{-1}E\big).$$

If $\mathcal{A}^*(z) = I$ then $z \in \mathsf{Ker}\,(E)$, thus if $\dim(\mathsf{Ker}\,(E)) = 1$, then

$$\langle b - a^+ f, z\rangle = 0 \quad \text{and} \quad a^+ = \frac{\langle b, z\rangle}{\langle f, z\rangle}.$$

Now recall that $f^* = (u_1^*A_1u_1, \dots u_1^*A_mu_1)$, whence

$$\langle f, z\rangle = \sum_{j=1}^{m} u_1^*A_ju_1z_j = u_1^*(\sum_{j=1}^{m} A_jz_j)u_1 = 1.$$

Which proves that in this case $a^+ = \langle b, z\rangle$ (see (86)), thus our strategy is consistent with the properties of the Lagrange multiplier a.

For $\dim \mathsf{Ker}\,(E) = 2$, we have an alternative based on strategy (b). We first introduce some notation:

$$-aU^*\mathcal{A}^*(d)U + (b^*d)I := \begin{pmatrix} -af^*d + b^*d & -a(Ed)^* \\ -aEd & \begin{array}{|cc} -ag^*d + b^*d & -a(Gd)^* \\ -aGd & \begin{array}{|c} R \end{array} \end{array} \end{pmatrix}.$$

Recall that

$$f = e_1^*\Big(U^*A_1u_1, \dots, U^*A_ku_1\Big),$$

and we see that

$$g = e_2^*\Big(U^*A_1u_2, \dots, U^*A_ku_2\Big).$$

If we had the property that

$$\exists z : \mathcal{A}^*(z) = I,$$

then the Lagrange multiplier a would be a constant $a^+ = b^*z$. So, if that specific z would exist, then clearly $f^*z = g^*z = 1$, by construction of f and g, and $Ez = 0$. A consequence is that $z \in \mathsf{Ker}\,(E)$. Unfortunately, we do not know if such a z exists, but we can use the above information to motivate the following approach. We would like to drive $(f-g)^*d$ to zero. Therefore, consider the space orthogonal to $(f-g)$:

$$(f-g)^\perp = ((f-g)^\perp \cap \mathsf{Ker}\,(E)) \oplus ((f-g)^\perp \cap \mathsf{Ker}\,(E)^\perp).$$

If $((f-g)^{\perp} \cap \mathsf{Ker}\,(E)) = \{d_0\}$ is 1-dimensional, then $d_0 = z$, and we can compute d_0 immediately, because of two properties for d_0 :

$$(f-g)^{\perp} \in \mathsf{Ker}\,(E) \quad \text{and} \quad (f-g) \perp d_0.$$

The next step is estimating the Lagrange multiplier a from

$$-af^* d_0 + b^* d_0 = 0.$$

This reveals immediately the similarity with the $\dim \mathsf{Ker}\,(E) = 1$ case. Note that we have made the potential dangerous linear direction zero. When we have found a, we can compute the direction d with a quadratic approximation, restricted to the space $\mathsf{H} = ((f-g)^{\perp} \cap \mathsf{Ker}\,(E)^{\perp})$:

$$d = -\frac{1}{2}(E_{\mathsf{H}}^* \Gamma E_{\mathsf{H}})^{-1}(-f_{\mathsf{H}} + \frac{b_{\mathsf{H}}}{a}).$$

We did not encounter the situation $\dim \mathsf{Ker}\,(E) > 2$ in our experiments.

Practical notes. Let us return to the direction of the type $d \perp \mathsf{Ker}\,(E)$ in (128). For practical implementations, the computation of d seems to be quite complicated. We suggest to make a QR-factorization of E^* : $E^* = QR$, with Q an unitary matrix and R an upper triangular matrix. Write $E = LQ$, with $L = R^*$ a lower triangular matrix. We now write $Ed = L(Qd)$, thus $f^*d = F^*(Q^*(Qd)) = (Qf)^*(Qd)$, and (124) then becomes

$$L^* \Gamma^{-1} L(Qd) = -\frac{1}{2}(-Qf + \frac{Qb}{a}).$$

The singularity of E corresponds to a non-trivial kernel of E, which corresponds to columns L_E in L that contain zero elements. The column vectors $Q_{E^{\perp}}$ in Q that correspond to the nonzero elements of L are the vectors that are perpendicular to $\mathsf{Ker}\,(E)$. In practice we would like to use small instead of zero, therefore we used an approach based on error analysis in the QR-factorization and based on numerical experiments. A simpler formulation, in the computational sense, for expression the direction d_2 in (128) is, when using the ideas above:

$$d_2 = -\frac{1}{2} Q_{E^{\perp}} (L_{E^{\perp}}^* \Gamma L_{E^{\perp}})^{-1} (-Q f_{E^{\perp}} + \frac{Q b_{E^{\perp}}}{a^+}).$$

3.2.4. How to select d

The discussion about computing the search direction d in the Subsections 3.2.2 and 3.2.3, implicate that we have two candidates:

(1) the direction d_1 in $\mathsf{Ker}\,(E)$ and $\|d_1\| = 1$,
(2) the direction d_2 in $\mathsf{Ker}\,(E)^{\perp}$,

Each step of our method that approximates the optimal value of (E), we have to select the “best” direction available.

For search direction d_2 we have a very good guess of the progress we will make, since we approximate the eigenvalue curve with a “parabola”. Therefore, if the progress is less than ε, with ε as in the stopping criterion for the change in the iterates,

$$\lambda_{\max}(C - \mathcal{A}^*(y^k)) - \lambda_{\max}(C - \mathcal{A}^*(y^{k+1})),$$

then we select d_1. If this is not the case, we select the *steepest direction.* To find out what the steepest direction is, we compute $\langle b - af, d_1\rangle$ and $\frac{\langle b-af,d_2\rangle}{\|d_2\|}$, and select search direction d_1 if

$$\langle b - af, d_1\rangle > \frac{\langle b - af, d_2\rangle}{\|d_2\|}.$$

We are well aware of the fact that this is a rather weak criterion, but for our tested examples, the chosen search direction seams to work nicely. Future research is necessary. For example, a two-dimensional search to find the optimal combination of d_1 and d_2 could be an interesting solution.

For the selected d we will compute a step size t as explained in Subsection 3.2.2.

3.2.5. Approximating a solution of (D)

Our method finds an approximate optimal value for (E) that is also an approximate optimal value for (D). The argmin $\widehat{y}$ for (E) computed by our eigenvalue method may not be a solution to (D), since we did not try to drive $\lambda_{\max}(C - \mathcal{A}^*(\widehat{y}))$ towards zero, as we should have done according to the optimality conditions (85).

In order to find a solution y^+ of (D), we take the linear approximation of the gradient of $\mathcal{L}(y) = a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y\rangle$, in $\widehat{y}$ (see (121)),

$$0 = \nabla_y \mathcal{L}(\widehat{y}) = -af + b. \tag{132}$$

Starting with $y = \widehat{y}$, we want to drive $a\lambda_{\max}(C - \mathcal{A}^*(y))$ as quickly as possible to zero. Therefore, we select a correction of $\widehat{y}$ in the direction orthogonal to the level set

$$\{y | \lambda_{\max}(C - \mathcal{A}^*(y)) = \lambda_{\max}(C - \mathcal{A}^*(\widehat{y}))\}$$

This direction is given by the gradient of $\lambda_{\max}(C - \mathcal{A}^*(y))$ in $\widehat{y}$, that is,

$$\nabla_y a\lambda_{\max}(C - \mathcal{A}^*(\widehat{y})) = -af = b,$$

because of (132). From $\widehat{y}$ we compute the new iterate $y^1 = \widehat{y} - \bar{\varepsilon} b$, such that

$$a\lambda_{\max}(C - \mathcal{A}^*(\widehat{y})) + (\nabla_y a\lambda_{\max}(C - \mathcal{A}^*(\widehat{y}))(-\bar{\varepsilon} b) = 0, \tag{133}$$

using the first order approximation of $\lambda_{\max}(C - \mathcal{A}^*(\widehat{y} - \bar{\varepsilon} b))$. Then

$$\bar{\varepsilon} = -\frac{a}{\|b\|^2}\lambda_{\max}(C - \mathcal{A}^*(\widehat{y})).$$

In general y^1 will not be a solution to (D), since the linear estimate for $\lambda_{\max}(C - \mathcal{A}^*(\widehat{y} - \bar{\varepsilon} b))$ induces an error of order $\mathcal{O}((\bar{\varepsilon}\|b\|)^2)$. However, we may use this y^1 in our iterative eigenvalue method as a new starting point. If, after running our method with this new starting point we have returned to this procedure, we stop if $|\bar{\varepsilon}|$ is smaller than machine precision, to be sure of an accurate approximation of a solution of (D).

We are now in the position to formulate Algorithm 3.2.1, that summarizes the approach of the last subsections.

```
(0) input
      • a vector $b$, matrices $A_i$ and $C$,
      • an initial point $y^0 \in \mathbb{R}^m$, an $\varepsilon$ for termination, and set
        $k = 0$
(1) search direction compute the eigenvalues of $C - \mathcal{A}^*(y^k)$ and
    compute $d$ (and $a$) following the strategy of Section 3.2
(2) step size compute step size $t$ according to Subsection 3.2.1
(3) update compute $y^{k+1} = y^k + td$
(4) termination compute $\lambda_{\max}(C - \mathcal{A}^*(y^{k+1}))$
      • if $\lambda_{\max}(C - \mathcal{A}^*(y^k)) - \lambda_{\max}(C - \mathcal{A}^*(y^{k+1})) \leq \varepsilon$ goto (5)
      • else $k = k + 1$, goto (1)
(5) solution to (D) compute $y^{k+1} = y^{k+1} - \bar{\varepsilon} b$ (see (133)).
      • If $\bar{\varepsilon} < \varepsilon$ stop
      • else $k := k + 1$, goto (1)
```

Algorithm 3.2.1 The simple eigenvalue case.

3.2.6. Examples to illustrate the Newton method

In this section we apply our approach to some of the SDP problems described in Section 1.2.4. The examples are of low size, and the output that we obtained are printed in Table 2, 3, and 4. The tables are divided into six columns:

- column 1 shows the iteration step k,
- column 2 shows the approximated Lagrange multiplier,
- column 3 shows the step size t computed in point (2) of Algorithm 3.2.1 for normalized d,
- column 4 shows the solution y^{k+1} found after termination in point (4) of Algorithm 3.2.1,
- column 5 shows the step size $\bar{\varepsilon}$ computed in point (5) of Algorithm 3.2.1,
- column 6 shows the approximate objective value after termination in point (4).

The true optimal value, solution and Lagrange multiplier are known beforehand. Therefore, we did not use the value ε of Algorithm 3.2.1 for termination. We stopped if the approximated objective value did not differ more than $1e - 5$ of the known objective value, and the approximated Lagrange multiplier did not differ more than $1e - 12$ of the actual Lagrange multiplier.

Example 3.2.2. The first example is an optimization problem of the form (38), a so-called SDP-relaxation of a Max-Cut problem.

The data for $n = m = 3$ are given by

$$C = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

iter.	a	step size t	y^*	$\bar{\varepsilon}$	obj. value
0	3	1.060660			4.0000000
1	3	0.000000	$[-1/3,\ 2/3,\ -1/3]$	-1.3333	4.0000000
2	3	0.000000	$[1,\ 2,\ 1]$		4.0000000

TABLE 2 Output for Example 3.2.2. Empty spaces in the table are non-applicable outputs.

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and $b = [1 \ \ 1 \ \ 1]^*$. The solution for this problem is $y = [1 \ \ 2 \ \ 1]^*$, with objective $b^*y = 4$. Since $\mathcal{A}^*(z) = I$, for $z = (1,1,1)^*$, the Lagrange multiplier a for this problem is the constant $b^*z = 3$. With the approach of Algorithm 3.2.1, we obtain the output represented in Table 2. For this problem, E of (124) has a 1-dimensional kernel and all directions are selected according to the Newton approach.

After iteration step 0, we have already found the optimal value. Interesting is that the step size for the first update equals almost 1. This shows that our approximation of the maximum eigenvalue function is very accurate. Another observation is that the approximate Lagrange multiplier a is accurate from the first step. We conclude that for this example the strategy for solving the linear equation (124) by constructing the best direction d using the Lagrange multiplier a is working perfectly. The first iteration is necessary to check the stopping criterion, and applies to the approach of point (5) of Algorithm 3.2.1. Note that this approach is also very accurate, although we only used a linear approximation to drive $\lambda_{\max}(C - \mathcal{A}^*(\widehat{y}))$ to zero. The second step is again necessary to check the stopping criterion. This example shows the ideal situation for our approach; all approximations are accurate, therefore the approach seems very promising.

EXAMPLE 3.2.3. We will now illustrate the strategy in Algorithm 3.2.1 for the trivial kernel case of matrix E in (124), that we included in Subsection 3.2.3.

We use the same problem as in Example 3.2.2, to illustrate the strategy, except that we perturb, with parameter $\delta > 0$, and extend the matrices A_i, $i = 1, \ldots, 3$, and C, with one column and row such that $\dim \mathsf{Ker}\,(E) := 0$, for example, C, A_1, A_2 and A_3 will be extended to

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & C & \\ 0 & & \end{bmatrix} + \begin{bmatrix} -\frac{1}{\delta} & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \\ \delta & \delta & \delta & \delta \end{bmatrix},$$

and

$$\begin{bmatrix} \bar{z}_1 & 0 \\ 0 & A_1 \end{bmatrix}, \begin{bmatrix} \bar{z}_2 & 0 \\ 0 & A_2 \end{bmatrix}, \begin{bmatrix} \bar{z}_3 & 0 \\ 0 & A_3 \end{bmatrix},$$

$$\text{with} \quad z, \quad \text{such that} \quad \sum_{i=1}^{3} z_i A_i = I \quad \text{thus,} \quad \bar{z}_i = \frac{z_i}{\langle z, z\rangle}.$$

iter.	a	step size t	y^*	$\bar{\varepsilon}$	obj. value
0	3	0.998000			4.0010000
1	3	0.000000	$[-.998,\ 0.00,\ -.998]$	-1.9990	4.0010000
2	3	0.000000	$[1.001,\ 1.999,\ 1.001]$		4.0010000

TABLE 3 Output for Example 3.2.3. Empty spaces in the table are non-applicable outputs.

The extension of A_i with $\bar{z}_i$ is necessary to guarantee the existence of the same Lagrange multiplier as before for the new enlarged system. With this example we will show that we obtain the same output as for the $\dim \mathsf{Ker}\,(E) = 1$ strategy, in order to illustrate the equivalence of these two strategies for the computation of the direction d. With $\delta = 1e-3$, we obtain the output represented in Table 3. The output in Table 3 is indeed almost the same as the output in Table 2: after iteration 0 we are already in the optimum; iteration 1 is necessary to check the stopping criterion, and to find the y^+ for which $\lambda_{\max}(C - \mathcal{A}^*(y^+)) = 0$. The objective seems to be optimal: $b^*y^+ - \delta = 4$. The approximation for the Lagrange multiplier is already accurate in iteration 0. This reveals that the strategy for the case where E is non-singular seems as promising as for the case where E is singular.

EXAMPLE 3.2.4. This example illustrates that the strategy is not robust. The example is again of the form (38). We take $n = m = 5$:

$$C = \begin{pmatrix} \frac{1}{2} & -\frac{1}{4} & -\frac{1}{4} & 0 & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{2} & 0 & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{4} & 0 & \frac{1}{2} & 0 \\ -\frac{1}{4} & -\frac{1}{4} & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

The 5×5 matrices $A_i,\ i = 1, \dots, 5$ are defined by

$$A_i := e_i e_i^*, \quad i = 1, \dots, 5,$$

thus A_i is filled with zeros, except on the (i,i)-th entry there is a one. The vector b is equal to $b = [1\ 1\ 1\ 1\ 1]^*$. The solution for this problem is $y = [1\ 1\ \frac{3}{4}\ \frac{3}{4}\ \frac{3}{4}]^*$, with objective $b^*y = 4\frac{1}{4}$. The Lagrange multiplier a is the constant $a = 5$, and $\dim \mathsf{Ker}\,(E) = 1$ in iteration 0 and 1, the search directions were selected according to the Newton approach in Subsection 3.2.3. In iteration 2 the $\dim \mathsf{Ker}\,(E) = 2$, and the direction was selected by the alternative approach for $\dim \mathsf{Ker}\,(E) = 2$, in Subsection 3.2.3.

We obtain the output represented in Table 4. We see that the first two iterations converge nicely, then a breakdown occurs in iteration 2. Note that the step sizes printed in column 3 do not equal 1 in this situation, but our quadratic approximation of the search direction in combination with the computation of the Lagrange multiplier is accurate. This indicates that the maximum eigenvalue function is indeed other than a pure quadratic function, but we still have extreme fast convergence. Inspection of the breakdown in iteration 2 reveals that one of the diagonal elements of the matrix $\Gamma = \operatorname{diag}(\lambda_1 - \lambda_2, \dots, \lambda_1 - \lambda_n)$ equals zero, thus Γ is not invertible

iter.	a	step size t	y^*	$\bar{\varepsilon}$	obj. value
0	5	0.250000			4.3750000
1	5	0.111803			4.2500000
2	5	0.000000	$[.15\ .15\ -.1\ -.1\ -.1]$		4.2500000
3	NaN	NaN			NaN

TABLE 4 Output for Example 3.2.4. Empty spaces in the table are non-applicable outputs.

as it should be for the approach of Subsection 3.2.2. We conclude that one of the eigenvalues is multiple. This example shows that we need a strategy for the case that the maximum eigenvalue is multiple, this will be the subject of the next section.

Let us repeat some of the important observations taken from the last three examples. Although the algorithm constructs a local approximation for the maximum eigenvalue, $\lambda_{\max}$, the behavior of the output for the above examples shows fast global convergence: that the algorithm finds the minimum in almost 1 step and that the step size factor equals 1, for two of the three examples, as should be asymptotically with a classic Newton method. This is, of course, a very special situation, in general we do not expect this global behavior. Another observation was that the approximated Lagrange multiplier is very accurate from the first iteration. The approach to approximate a solution to (D) as described in Subsection 3.2.5 is also accurate. We see that our approach seems very promising.

3.3. A Newton-type method for (E): multiple eigenvalue case

Section 3.2 discussed the simple eigenvalue case. As we have seen in the examples difficulty arises when eigenvalues are multiple. Nevertheless, the formulation of an approach with local quadratic properties is still possible. Therefore, we will construct in this section a method for the multiple eigenvalue case. The main idea is the same as for the simple eigenvalue case, namely based on a matrix partitioning like (114); this time not only with respect to u_1 (the eigenvector corresponding to the simple largest eigenvalue) but with respect to $u_1, \ldots, u_l$, the set of eigenvectors corresponding to the multiplicity l of the maximum eigenvalue. We will assume that we have already an approximation of the Lagrange multiplier. In all our experiments it was possible to find a direction for which the maximum eigenvalue function $\lambda_{\max}$ was simple, or the Lagrange multiplier was known beforehand.

In Subsection 3.3.1 we construct a block matrix eigenvalue problem that gives an analogue of the local approximation of $\lambda_{\max}$ in formula (109) for the simple eigenvalue case.

From this local approximation of the function $\lambda_{\max}$ in the direction d, we identify two types of search directions. Based on the properties of the local approximation and the situations that can occur for a solution y^+ of (E).

The first type of search directions are analogous to the search directions for the simple eigenvalue case, and will be discussed in Subsection 3.3.2. These search directions arise from local second order approximations of the maximum eigenvalue function. Within these search directions we can construct again two sub-types of search directions in the same way as we did in Subsection 3.2.2.

The second type of search directions, discussed in Subsection 3.3.3, will lead to, if they exist, a minimal maximum eigenvalue with multiplicity l. We can derive a local first order approximation of $\lambda_{\max}$ in the direction d, and we will shortly point out how to compute a direction d from this first order approximation iteratively, by identifying the combination of two matrices for which the maximum eigenvalue is minimal.

In Subsections 3.3.2 and 3.3.3 we come up with three different search directions. In Subsection 3.3.4 we motivate how we select the "best" search direction every iteration step. In the last part of Subsection 3.3.4, we explain how to determine a new update td using the step size procedure of Subsection 3.2.1. Subsection 3.2.5 is again used to find an approximate solution for (D).

To illustrate the approach for the multiple eigenvalue case we give some examples in Subsection 3.3.5.

3.3.1. Block matrix eigenvalue system

In this section we will construct a block matrix to develop a Newton-type method for solving (E), if the maximum eigenvalue is multiple. We will use the notation of Section 3.2. Let us return to problem (100). We are interested in a second order approximation of the function $\lambda_{\max}(\widehat{C} + t\widehat{F})$, with respect to t. The first step is to construct an eigenvalue decomposition of $\widehat{C}$,

$$\widehat{C} = U\Lambda U^*, \qquad U^*U = I, \qquad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

Suppose that the eigenvalues are ordered in the following way,

$$\lambda_1 = \cdots = \lambda_l > \lambda_{l+1} \geq \cdots \geq \lambda_n.$$

We use Λ_1 for the diagonal matrix $\Lambda_1 := \operatorname{diag}(\lambda_1, \ldots, \lambda_l)$, Λ_2 for the diagonal matrix $\Lambda_2 := \operatorname{diag}(\lambda_{l+1}, \ldots, \lambda_n)$, and Γ for the diagonal matrix $\Gamma := \lambda_1 I - \Lambda_2$. The matrix $U = (u_1, \ldots, u_n)$ consists of the orthonormalized eigenvectors of $\widehat{C}$. With decomposition:

$$\lambda_{\max}(\widehat{C} + t\widehat{F}) = \lambda_{\max}(U\Lambda U^* + t\widehat{F}) = \lambda_{\max}(\Lambda + tU^*\widehat{F}U).$$

Set $F := U^*\widehat{F}U$, and we proceed with the following partitioning:

$$\Lambda + tF := \left(\begin{array}{c|c} \Lambda_1 + tR_1 & tE^* \\ \hline tE & \Lambda_2 + tR_2 \end{array}\right) = \lambda_1 I + \left(\begin{array}{c|c} tR_1 & tE^* \\ \hline tE & -\Gamma + tR_2 \end{array}\right). \tag{134}$$

Compute an eigenvalue decomposition of R_1:

$$R_1 = X\Sigma X^*, \qquad X^*X = I, \qquad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_l).$$

We define the function $\operatorname{diag}_\Sigma(X^* \cdot X) : \mathsf{S}^l \to \mathsf{S}^l$ as

$$\operatorname{diag}_\Sigma(X^*KX)_{ij} = \left\{ \begin{array}{cc} (X^*KX)_{ij} & \text{if} \quad \sigma_i = \sigma_j \\ 0 & \text{everywhere else} \end{array} \right. .$$

Now we can state the following lemma, which will lead to an extension of the approach for the computation of the step size t in Subsection 3.2.1, Equation (110). We will show this after the lemma. This lemma can be derived from [**25**, Ch. 2].

LEMMA 3.3.1. *Consider the block matrix (eigenvalue) problem*

$$\begin{pmatrix} tR_1 & tE^* \\ tE & -\Gamma + tR_2 \end{pmatrix} \begin{pmatrix} I + tG \\ tZ \end{pmatrix} = \begin{pmatrix} I + tG \\ tZ \end{pmatrix} tD, \tag{135}$$

where E, R_1, R_2 are defined as before and G, Z, D are matrix valued functions depending on t. Then

$$tD = tR_1 + t^2 \mathrm{diag}_\Sigma(X^* E^* \Gamma^{-1} E X) + \mathcal{O}(t^3), \qquad \textit{for} \quad t \to 0. \tag{136}$$

Proof. Assume that $t \neq 0$. After computing the first and second block coordinates of (135), and after dividing by t we obtain the equations

$$R_1 + tR_1 G + tE^* Z = D + tGD, \tag{137}$$

$$(-\Gamma + tR_2)Z + = tZD - E - tEG. \tag{138}$$

Now suppose that Z, G, D are smooth and differentiable for $t \to 0$ ([**25**, Thm 6.1,p.120]), then

$$Z = Z_0 + tZ_1 + \mathcal{O}(t^2),\ G = G_0 + tG_1 + \mathcal{O}(t^2),\ D = D_0 + tD_1 + \mathcal{O}(t^2). \tag{139}$$

We may conclude from (137) that $D_0 = R_1$, and from (138) that $Z_0 = \Gamma^{-1}E$. If we substitute this into (137), then we see that

$$tR_1 G_0 + tE^* \Gamma^{-1} E = tD_1 + tG_0 R_1 + \mathcal{O}(t^2) \quad (t \to 0),$$

so that,

$$R_1 G_0 - G_0 R_1 \approx D_1 - E^* \Gamma^{-1} E.$$

We will now apply the similarity transformation with X and take the "diag_Σ":

$$\mathrm{diag}_\Sigma(X^* R_1 G_0 X - X^* G_0 R_1 X) = \mathrm{diag}_\Sigma(\Sigma X^* G_0 X - X^* G_0 X \Sigma) = 0.$$

From this we conclude that $\mathrm{diag}_\Sigma(X^* D_1 X) = \mathrm{diag}_\Sigma(X^* E^* \Gamma^{-1} E X)$. Thus there exists a G_0 such that $R_1 G_0 - G_0 R_1 = D_1 - E^* \Gamma^{-1} E$. Choose $G_0 = 0$ and use (139) to conclude that

$$\begin{aligned} tD &= tD_0 + t^2 D_1 + \mathcal{O}(t^3) \quad (t \to 0) \\ &= tR_1 + t^2 \mathrm{diag}_\Sigma(X^* E^* \Gamma^{-1} E X) + \mathcal{O}(t^3) \quad (t \to 0). \end{aligned}$$

□

This lemma is an extension of the simple eigenvalue case, because if $l = 1$ then tR_1 coincides with $t\mu_1$ and tE coincides with te in (102), note that diag_Σ reduces to a 1×1 matrix, thus (135) coincides with (102), and we see that the linear and quadratic part of the approximation in (110) follow from (136).

We will use this lemma to get more insight in computing a search direction d for the multiple eigenvalue case (just as we used step size (102) as inspiration for a suitable approximation (110) for the search direction d in the simple eigenvalue case).

Suppose that we know the Lagrange multiplier a explicitly, and that we are in the situation as sketched for (98). We have some given y_0 and we search a d such that (112) is solved. We consider again the matrix $\widehat{C}$ (see 113)),

$$\widehat{C} = C - \mathcal{A}^*(y_0) + \frac{1}{a}\langle b, y_0\rangle I,$$

We represent the problem of computing $\lambda_{\max}$ with respect to the basis $(u_1, \dots, u_n)$,

$$\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(d) + (b^*d)I) = \lambda_{\max}(a\Lambda - aU^*\mathcal{A}^*(d)U + (b^*d)I),$$

and introduce a matrix partitioning, based on (134),

$$a\Lambda - aU^*\mathcal{A}^*(d)U + (b^*d)I = a\lambda_1 I + \left(\begin{array}{c|c} \widehat{R}_1(d) & E(d)^* \\ \hline E(d) & -a\Gamma + R_2(d) \end{array}\right), \tag{140}$$

where

$$\widehat{R}_1(d) = \begin{pmatrix} \widehat{q}_{11}^*d & \cdots & \widehat{q}_{1l}^*d \\ \vdots & & \vdots \\ \widehat{q}_{l1}^*d & \cdots & \widehat{q}_{ll}^*d \end{pmatrix} \quad \text{and} \quad E(d) = (E_1d, \dots, E_ld).$$

It is not difficult to see that the elements of the matrix $\widehat{R}_1(d)$ can be represented as $\widehat{q}_{ij}^*d$, since

$$\widehat{q}_{ij} = \begin{cases} -a(u_i^*A_1u_j, \dots, u_i^*A_mu_j)^* & \text{if} \quad i \neq j \\ -a(u_i^*A_1u_j, \dots, u_i^*A_mu_j)^* + b & \text{if} \quad i = j \end{cases}$$

We find the expression for $E(d)$ in the same way, define

$$F_kd := \Big(-aU^*\mathcal{A}^*(d)U + (b^*d)I\Big)e_k$$

then

$$E_k = (F_k)_{i=l+1,j=1}^{n,l}.$$

We will now apply Lemma 3.3.1 for the matrix on the right hand side of (140) to obtain a suitable approximation for the function $\lambda_{\max}$:

$$\begin{aligned} & a\lambda_1 I_l + \widehat{R}_1(d) + \operatorname{diag}_\Sigma(X^*E(d)^*a\Gamma^{-1}E(d)X) \\ = \; & a\lambda_1 I_l + \begin{pmatrix} \widehat{q}_{11}^*d & \cdots & \widehat{q}_{1l}^*d \\ \vdots & & \vdots \\ \widehat{q}_{l1}^*d & \cdots & \widehat{q}_{ll}^*d \end{pmatrix} + \operatorname{diag}_\Sigma(X^*E(d)^*a\Gamma^{-1}E(d)X). \end{aligned} \tag{141}$$

We would like to have the trace of the matrix $\widehat{R}_1(d)$ as a function of d zero for reasons that will be explained later. To have the trace of a new matrix $R_1(d) = (q_{ij}^*d)_{ij}$, based on the matrix $\widehat{R}_1(d)$, equal to 0, we define

$$p = \frac{1}{l}\sum_i \widehat{q}_{ii} \quad \text{and} \quad q_{ij} = \begin{cases} \widehat{q}_{ij} & \text{if} \quad i \neq j \\ \widehat{q}_{ij} - p & \text{if} \quad i = j \end{cases}.$$

Furthermore, we split p into two parts, $p_1,\ p_2$, such that $p_1 \in \operatorname{span}\{q_{ij}\}$ and $p_2 \perp \operatorname{span}\{q_{ij}\}$. This splitting is convenient, when we divide the optimization problem into two parts at the end of this subsection. Equation (141) gets the following form,

$$a\lambda_1 I_l + p_1^* dI + p_2^* dI + \begin{pmatrix} q_{11}^* d & \cdots & q_{1l}^* d \\ \vdots & & \vdots \\ q_{l1}^* d & \cdots & q_{ll}^* d \end{pmatrix} + \operatorname{diag}_\Sigma(X^* E(d)^* a\Gamma^{-1} E(d) X). \tag{142}$$

Define y^+ as the solution of (E), thus $\lambda_{\max}(C - \mathcal{A}^*(y^+)) = 0$. Then the maximum eigenvalue $\lambda_{\max}(C - \mathcal{A}^*(y^+ + td))$ as a function of t has a minimum in $t = 0$ for any direction d. This minimum can be a minimum of an eigenvalue curve of $C - \mathcal{A}^*(y^+ + td)$ as a function of t, or a crossing of eigenvalue curves of $C - \mathcal{A}^*(y^+ + td)$ as a function of t, in the second case $C - \mathcal{A}^*(y^+ + td)$ for $t = 0$ has a multiple eigenvalue.

We identify two possible types of search directions for equation (142). Set $\mathsf{Q} := \operatorname{span}\{q_{ij}\}$. Then the two types are characterized by

(a) $d \perp \mathsf{Q}$,
(b) $d \in \mathsf{Q}$.

For each part we will develop a different method and a different approximation.

Consider (a), For $d \perp \mathsf{Q}$ equation (142) transforms to

$$a\lambda_1 + p_2^* d + \lambda_{\max}\Big(\operatorname{diag}_\Sigma(X^* E(d)^* a\Gamma^{-1} E(d) X)\Big), \tag{143}$$

For the a search direction determined by (143) we see that we have linear and nonlinear eigenvalue curves as functions of t in the direction d. Therefore, we see that the minimal maximum eigenvalue can either be a crossing of some eigenvalue curves or a minimum for one eigenvalue curve. This is equal to the situations considered in Section 3.2, because we can use the higher order terms of (143) to approximate the eigenvalue curves as functions of t as we will do in Subsection 3.3.2.

For search directions of type $d \in \mathsf{Q}$ equation (142) transforms to

$$a\lambda_1 + p_1^* d + \lambda_{\max}\left(\begin{pmatrix} q_{11}^* d & \cdots & q_{1l}^* d \\ \vdots & & \vdots \\ q_{l1}^* d & \cdots & q_{ll}^* d \end{pmatrix}\right). \tag{144}$$

Consider $\lambda_{\max}(C - \mathcal{A}^*(y^+ + td))$ for $d \in \mathsf{Q}$ as a function of t. The minimum is attained for $t = 0$ where the multiplicity of $\lambda_{\max}$ will probably be l, because:

$$\begin{pmatrix} q_{11}^* d & \cdots & q_{1l}^* d \\ \vdots & & \vdots \\ q_{l1}^* d & \cdots & q_{ll}^* d \end{pmatrix} \tag{145}$$

is symmetric and has by definition a trace equal to zero for all d, thus has positive and negative eigenvalues. Furthermore, we started in y_0, where the maximum eigenvalue had multiplicity l, and want to compute a search direction d such that (144) is minimal, we know that for optimality the maximum eigenvalue of (145) equals zero. Thus the eigenvalue curves of (145) in $y_0 + td$ as functions of t will be increasing and decreasing and have to cross somewhere. Zero trace and zero maximum eigenvalue show that all l eigenvalues have to be zero at optimality. Note that we will not use any information from the higher order terms. We discuss part (b) in Subsection 3.3.3.

3.3.2. Computation of a suitable direction $d \perp \mathsf{Q}$

In this subsection we concentrate on part (a). Let us focus on (143). $d \perp \mathsf{Q}$ implies that $\Sigma := p_2^* dI = R_1$ (tR_1 of (134)), and $X := I$, thus

$$\operatorname{diag}_\Sigma(X^* E(d)^* a \Gamma^{-1} E(d) X) := aE(d)^* \Gamma^{-1} E(d),$$

and (143) transforms to

$$d^+ = \operatorname*{argmin}_{d \perp \mathsf{Q}} \; a\lambda_1 + p_2^* d + a\lambda_{\max}\Big(E(d)^* \Gamma^{-1} E(d)\Big).$$

$\lambda_{\max}(E(d)^* \Gamma^{-1} E(d))$ depends in a complicated way on d, therefore we will approximate it by

$$\frac{1}{l}\operatorname{trace}\big(E(d)^* \Gamma^{-1} E(d)\big).$$

We choose for the trace, because

(1) the trace depends in a controllable way on d,
(2) the trace is l times the average of the eigenvalues. Moreover, all eigenvalues are positive and we know that in the optimum the trace has to be zero,
$$\lambda_{\max}(E(d)^* \Gamma^{-1} E(d)) = 0 \Leftrightarrow \frac{1}{l}\operatorname{trace}\big(E(d)^* \Gamma^{-1} E(d)\big) = 0.$$
Therefore, we hope that if we minimize $\frac{1}{l}$trace we find a useful search direction d,
(3) in the $l = 1$ case, the simple eigenvalue case, this formulation coincides with the Newton approach of Subsection 3.2.2;
(4) in the (to below discussed) examples, we see that this approach does inherit fast convergence.

Note that

$$\frac{1}{l}\operatorname{trace}\big(E(d)^* \Gamma^{-1} E(d)\big) = \sum_{i=1}^{l} e_i^* E(d)^* \Gamma^{-1} E(d) e_i = \sum_{i=1}^{l} d^* E_i \Gamma^{-1} E_i d.$$

For ease of notation we set $H := \frac{1}{l}\sum_{i=1}^{l} E_i^* \Gamma^{-1} E_i$. Then we are left with the following optimization problem

$$\min_{d \perp \mathsf{Q}} p_2^* d + a d^* H d, \tag{146}$$

The argmin of (146) has to satisfy the following optimality condition:

$$0 = \frac{1}{a} p_2 + Hd, \quad (\text{for} \quad d \perp \mathsf{Q}). \tag{147}$$

The matrix H is positive definite by construction. As for the simple eigenvalue case in Subsection 3.2.2, where we had to solve (124), we have to solve a linear equation, (147).

We again immediately identify two possible types of search direction $d \perp \mathsf{Q}$,

- $d \in \mathsf{Ker}\,(H)$,
- $d \perp \mathsf{Ker}\,(H)$.

Note that a search direction $d \in \mathsf{Ker}\,(H)$ is again a linear direction. We will call the steepest linear direction d_1. Thus we can apply the strategies in Subsection 3.2.3 on (147), to minimize the linear directions and to determine the Lagrange multiplier a.

There is a practical difference between solving (124) and (147), because in (147) there is an additional restriction to keep $d \perp \mathsf{Q}$. Therefore, we give the following notes.

Practical notes. We assume that $d \perp \mathsf{Ker}\,(H)$ and use the following strategy. First orthogonalize the vectors q_{ij}. We only have to orthogonalize $\frac{l(l+1)}{2}$ vectors, since by construction $q_{ij} = q_{ji}$.

We orthogonalize by making a QR-factorization $B = QR$ of the $(m \times \frac{l(l+1)}{2})$ matrix B with

$$B := \Big(q_{11}, \dots, q_{1l}, q_{22}, \dots, q_{2l}, \dots, q_{ll}\Big). \tag{148}$$

With this orthogonalization we try to find a solution d^+. Optimality of the optimization problem (146) at d^+ is the same as stating that for all $\varepsilon > 0$, $\varepsilon \ll 1$ and every direction f, $f \perp Q$, $\|f\| = 1$:

$$\begin{aligned} & p_2^*(d+\varepsilon f) + (d+\varepsilon f)^* H (d+\varepsilon f) \geq p_2^* d + d^* H d \quad \forall \varepsilon > 0, \\ \Leftrightarrow \quad & |p_2^* f + 2d^* H f| \leq \varepsilon |f^* H f| \quad \forall \varepsilon > 0, \\ \text{thus} \quad & p_2^* f + 2d^* H f = 0 \quad \forall f \perp Q, \ \|f\| = 1. \qquad (149) \end{aligned}$$

We have to determine a $d \perp Q$ such that (149) holds. This is equivalent to determine a $d \perp Q$ such that

$$\begin{aligned} & f^*(2Hd + p_2) = 0 \quad \forall f \perp Q \\ \Leftrightarrow \quad & \exists y \quad \text{such that} \quad 2Hd + p_2 = Qy \end{aligned}$$

If H is invertible, then

$$d = \frac{1}{2}(H^{-1}Qy - H^{-1}p_2) \tag{150}$$

such that $d \perp Q$. Unfortunately, H will not be invertible. Therefore, we will use the strategy written in the *Practical Notes* part of Subsection 3.2.3, to locate the singular and non-singular part of H, and thus, the part H_E of H in the kernel and $H_{E^\perp}$ perpendicular to the kernel. We will not repeat this here. We do the same splitting for $p_2 = p_2^E + p_2^{E^\perp}$. We proceed with the parts in $\mathsf{Ker}\,(H)^\perp$. Since $Q^*d = 0$, we can write

$$Q^*d = \frac{1}{2}(Q^*(H_{E^\perp}^{-1}Qy - H_{E^\perp}^{-1}p_2^{E^\perp})) = 0 \quad \Leftrightarrow \quad y = (Q^* H_{E^\perp}^{-1} Q)^{-1} Q^* H_{E^\perp}^{-1} p_2^{E^\perp}.$$

By inserting (151) into (150), we find the following expression for the direction d,

$$d := \frac{1}{2}(H_{E^\perp}^{-1} Q (Q^* H_{E^\perp}^{-1} Q)^{-1} Q^* H_{E^\perp}^{-1} p_2^{E^\perp} - H_{E^\perp}^{-1} p_2^{E^\perp}). \tag{151}$$

We will call this direction d_2.

3.3.3. Computation of a suitable direction $d \in \mathsf{Q}$

We will now describe part (b), where we want to determine an appropriate search direction $d \in \mathsf{Q}$ from (144). We use the QR-factorization of the matrix B in (148). The columns r_{ij} of R correspond to the columns q_{ij} of B such that $q_{ij} = Qr_{ij}$, then $q_{ij}^* d = r_{ij}^*(Q^* d)$. Thus we are looking for a $\widehat{d} = Q^* d$, such that $\widehat{d}$ minimizes

$$\min_{\widehat{d} \in \mathsf{Q}, \|\widehat{d}\|=1} \lambda_{\max}\left(\begin{pmatrix} (r_{11}^* + p_1^*)\widehat{d} & \cdots & r_{1l}^*\widehat{d} \\ \vdots & \ddots & \vdots \\ r_{l1}^*\widehat{d} & \cdots & (r_{ll}^* + p_1^*)\widehat{d} \end{pmatrix}\right). \tag{152}$$

We assume that $\|\widehat{d}\| = 1$. This is acceptable because we are in the situation of crossing eigenvalue curves and try to find a search direction d and not an update td for which the length does matter. Define the matrix function $C(y)$ as

$$C(y) := \begin{pmatrix} (r_{11}^* + p_1^*)y & \cdots & r_{1l}^* y \\ \vdots & \ddots & \vdots \\ r_{l1}^* y & \cdots & (r_{ll}^* + p_1^*)y \end{pmatrix},$$

and we write C_{y_0} instead of $C(y_0)$ for fixed $y = y_0$ to express that $C(y_0)$ is a constant matrix.

We will solve the optimization problem (152) by an iterative method. Suppose that we are in a search direction $\widehat{d}^0 \in \mathsf{Q}$ and we are interested in a better search direction $\widehat{d}^1 \in \mathsf{Q}$. Then

$$\widehat{d}^1 = \cos(\phi)\widehat{d}^0 + \sin(\phi)h \quad \text{for some} \quad h \in \mathsf{Q},\ h \perp \widehat{d}^0,\ \|h\| = 1,\ \phi \in [0, i2\pi],$$

and (152) transforms to

$$\min_{h \perp \widehat{d}^0, \phi, \|h\|=1} \lambda_{\max}\Big(C(\cos(\phi)\widehat{d}^0) + C(\sin(\phi)h)\Big). \tag{153}$$

We will not estimate (153) at once, but split it into two parts, first construct h and then ϕ. By concentrating on the situation that $|\phi| \ll 1$, we are able to formulate an approach to find a suitable direction h.

$$\lambda_{\max}\Big(\cos(\phi)C_{\widehat{d}^0} + \sin(\phi)C(h)\Big) \approx \lambda_{\max}\Big(C_{\widehat{d}^0} + \varepsilon C(h)\Big) \approx \lambda_{\max}\Big(C_{\widehat{d}^0} + C(\varepsilon h)\Big).$$

Thus we can approximate h by the method "Computation of the update d" described in Section 3.2.2, then set $h := \frac{h}{\|h\|}$, and then compute ϕ from

$$\phi = \underset{\phi}{\operatorname{argmin}}\ \lambda_{\max}\Big(\cos(\phi)C_{\widehat{d}^0} + \sin(\phi)C_h\Big), \tag{154}$$

and set the new direction $\widehat{d}^1 = \cos(\phi)\widehat{d}^0 + \sin(\phi)h$, for which we can repeat the process, until $\|\widehat{d}^{k+1} - \widehat{d}^k\| \leq \varepsilon$, which is our stopping criterion. We will call the approximated search direction d_3.

Computation of ϕ. We describe how to compute ϕ from (154). Optimization problem (154) can be solved by a method analogue to the method described for the optimization problem (100) in Section 3.2.1. The difference will be that we have two variables $\cos(\phi), \sin(\phi)$ instead of one variable t. For completeness of this thesis we will derive the formula's for the approximation of ϕ. The first step is, as usual, to construct an eigenvalue decomposition (3) of $C_{\widehat{d}^0}$,

$$C_{\widehat{d}^0} = U\Lambda U^*, \qquad U^*U = I.$$

The matrix Λ is a diagonal matrix $\mathrm{diag}(\lambda_1, \ldots, \lambda_l)$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_l$ on its diagonal. The matrix $U = (u_1, \ldots, u_l)$ consists of the orthonormalized eigenvectors. We use this decomposition to rewrite $\lambda_{\max}$ as

$$\begin{aligned}\lambda_{\max}\Big(\cos(\phi)C_{\widehat{d}^0} + \sin(\phi)C_h\Big) &= \lambda_{\max}\Big(\cos(\phi)\Lambda + \sin(\phi)U^*C_hU\Big)\\ &= \lambda_{\max}\Big(\cos(\phi)\Lambda + \sin(\phi)F\Big).\end{aligned}$$

This allows to rewrite (154) as

$$s = \underset{s}{\mathrm{argmin}}\{\lambda_{\max}(c\Lambda + sF) \mid c^2 + s^2 = 1\}.$$

We proceed with the following partitioning:

$$c\Lambda + sF := \left(\begin{array}{c|c} c\lambda_1 + s\mu_1 & se^* \\ \hline se & R \end{array}\right),$$

which is almost the same as partitioning (101). We again obtain a formula for the new minimal maximum eigenvalue ρ:

$$\left(\begin{array}{c|c} c\lambda_1 + s\mu_1 & se^* \\ \hline se & R \end{array}\right)\begin{pmatrix} 1 \\ z \end{pmatrix} = \rho\begin{pmatrix} 1 \\ z \end{pmatrix}. \tag{155}$$

If ρ is not in the spectrum of R, and if the first element of the eigenvector corresponding to ρ is nonzero, this linear system leads to the following equations

$$c\lambda_1 + s\mu_1 + se^*z = \rho, \tag{156}$$

$$(R - \rho I)z = -se. \tag{157}$$

Substitute (157) into (156), then

$$\begin{aligned}\rho &= c\lambda_1 + s\mu_1 - s^2e^*(R - \rho I)^{-1}e\\ &= c\lambda_1 + s\mu_1 - s^2e^*(R - (c\lambda_1 + s\mu_1)I)^{-1}e + \mathcal{O}(s^4) \quad (s \to 0).\end{aligned} \tag{158}$$

Call the diagonal elements of the matrix $F : \mu_1, \ldots, \mu_l$. Define the matrices Γ and N by

$$\Gamma := \mathrm{diag}(\lambda_1 - \lambda_2, \ldots, \lambda_1 - \lambda_l), \quad N := \mathrm{diag}(\mu_1 - \mu_2, \ldots, \mu_1 - \mu_l).$$

Then we have that

$$\begin{aligned}(R - (c\lambda_1 + s\mu_1)I)^{-1} &= (-c\Gamma + s(\widehat{R} - N))^{-1} = -\frac{\Gamma^{-1}}{c}(I - \frac{s}{c}\Gamma^{-1}(\widehat{R} - N))^{-1}\\ &= -\frac{\Gamma^{-1}}{c} + \frac{s}{c^2}\Gamma^{-1}N\Gamma^{-1} + \mathcal{O}(s^2) \quad (s \to 0).\end{aligned} \tag{159}$$

```
(1) initialization choose initial $\widehat{d}^0$, for example $\widehat{d}^0 := \frac{p_1}{\|p_1\|}$, and tolerance $\varepsilon$
(2) update determination compute
        $h = \underset{h \perp \widehat{d}^k}{\operatorname{argmin}}\ \lambda_{\max}(C_{\widehat{d}^k} + C(h))$
    by the method "Computation of the update $d$" described in Section 3.2.2, and set $h := \frac{h}{\|h\|}$.
(3) find new iterate compute the new iterate $\widehat{d}^{k+1}$ by first solving
        $\phi = \underset{\phi}{\operatorname{argmin}}\ \lambda_{\max}\Big(\cos(\phi)C_{\widehat{d}^k} + \sin(\phi)C_h\Big)$,
    and then setting $\widehat{d}^{k+1} = \cos(\phi)\widehat{d}^k + \sin(\phi)h$.
(4) termination
    • if $\|\widehat{d}^{k+1} - \widehat{d}^k\| \le \varepsilon$, set $d := \frac{Q\widehat{d}^{k+1}}{\|Q\widehat{d}^{k+1}\|}$ stop
    • else $k = k + 1$, goto (2)
```

Algorithm 3.3.2

All this formula manipulating was necessary to avoid the explicit computation of the inverse of a non-diagonal matrix R. For c and s we have the following properties: $c^2 = 1 - s^2$, $c = 1 - \frac{1}{2}s^2$. Apply these properties to (159):

$$(R - (c\lambda_1 + s\mu_1)I)^{-1} = -\Gamma^{-1} + s\Gamma^{-1}N\Gamma^{-1} + \mathcal{O}(s^2) \quad (s \to 0),$$

and finally equation (158) gets the following form,

$$\begin{aligned} \rho &= c\lambda_1 + s\mu_1 + s^2 e^*\Gamma^{-1}e - s^3 e^*\Gamma^{-1}N\Gamma^{-1} + \mathcal{O}(s^4) \quad (s \to 0) \\ &= \lambda_1 + s\mu_1 + s^2(e^*\Gamma^{-1}e - \frac{1}{2}\lambda_1) + \mathcal{O}(s^3) \quad (s \to 0). \end{aligned} \tag{160}$$

We have found a local approximation for $\lambda_{\max}(c\Lambda + sF)$. The ϕ that we need for (154) can now be computed by solving the second order approximation of $\lambda_{\max}$:

$$s = \underset{s}{\operatorname{argmin}}\ \lambda_1 + s\mu_1 + s^2(e^*\Gamma^{-1}e - \frac{1}{2}\lambda_1).$$

This optimization problem is similar to (110), and solved following the strategy in Subsection 3.2.1.

We can now formulate an algorithm for solving (152) iteratively. The approach of this subsection is described in Algorithm 3.3.2.

3.3.4. How to select d and, how to compute the step size t.

Subsection 3.3.2 and 3.3.3 describe the computation of a search directions d for three different situations, We repeat them here:

(1) the direction $d_1 \perp \mathsf{Q}$, $d_1 \in \mathsf{Ker}\,(H)$, and $\|d_1\| = 1$,
(2) the direction $d_2 \perp \mathsf{Q}$, and $d_2 \perp \mathsf{Ker}\,(H)$,

(3) the direction $d_3 \in \mathsf{Q}$, and $\|d_3\| = 1$.

From the three different directions we have to select the "best" one, every iteration step of our algorithm. We will first compare the direction d_1 to d_3, both are linear directions; we did not consider high order terms for the computation of d_3, because we do not know how to handle them. Thus both associated eigenvalue curves in t of $\lambda_{\max}$ approach minus infinity, when minimizing in these directions, unless they are 'stopped' by a crossing eigenvalue curve. Thus, it seems reasonable to check whether one of the directions is steeper than the other one, that is, checking if $|p_2^* d_1| > |p_1^* d_3|$ then take d_1 otherwise take d_3. We can call this strategy the strategy of the largest progress. In all the tests we ran, this led to a choice for the direction d_3.

Now suppose that we selected d_3. Then we still have to make a choice between d_2 and d_3. For search direction d_2 we have a very good guess of the progress we will make, since we approximate the eigenvalue curve with a "parabola". So, if the progress is less than ε, with ε as in the stopping criterion for the change in the iterates

$$\lambda_{\max}(C - \mathcal{A}^*(y^k)) - \lambda_{\max}(C - \mathcal{A}^*(y^{k+1})) < \varepsilon,$$

then we select d_3. Otherwise, as we observed in our numerical examples, we select the steepest direction. This is clearly not a very sophisticated strategy, but it is the only one available to us at the time of writing this thesis. In all the examples we tested, this led to a choice for d_2. With this direction our method seems to work fine, see also the discussion in Subsection 3.2.4. Future research is necessary. For example, a two-dimensional search to find the optimal combination of d_2 and d_3 could be an interesting solution.

Another point, that we did not discuss, is the computation of the step size t for the search direction d. For the computation of the step size we use the strategy as discussed in the last part of Subsection 3.2.1. The quadratic or higher order approximation for the step size t given in Subsection 3.2.1 does not apply here, since the matrix Γ is not invertible. Therefore, we cannot use the quadratic approximation of $\lambda_{\max}$ as a function of t for finding effective estimates for t. We will use linear approximations, and try to find eigenvalue curves that cross the eigenvalue curve of λ_1 as a function of t as in Subsection 3.2.1, or use a bisection method starting with a certain t_0, for example $t_0 < 10$, and check whether progress is made . In all the numerical examples this strategy works fine. Although more research is required here.

We are now in a position to write the approach of this section in an algorithm, Algorithm 3.3.3.

3.3.5. Examples to illustrate the effect of Algorithm 3.3.3

In this section we apply our approach to some of the SDP problems described in Section 1.2.4.

The output for the first example is printed in Table 5. This table is divided into 6 columns, like we did in Subsection 3.2.6.

```
(0) input
      • a vector $b$, matrices $A_i$ and $C$,
      • an initial point $y^0 \in \mathbb{R}^m$, an $\varepsilon$ for termination, and set
        $k = 0$
(1) search direction compute the eigenvalues of $C - \mathcal{A}^*(y^k)$ and
    check the multiplicity $l$ of $\lambda_{\max}$
      • if $l = 1$ compute $d$ (and $a$) following the strategy of Sec-
        tion 3.2
      • else compute $d$ (and $a$) following the strategy of Section
        3.3
(2) step size compute step size $t$ according to Subsection 3.2.1
(3) update compute $y^{k+1} = y^k + td$
(4) termination compute $\lambda_{\max}(C - \mathcal{A}^*(y^{k+1}))$
      • If $\lambda_{\max}(C - \mathcal{A}^*(y^k)) - \lambda_{\max}(C - \mathcal{A}^*(y^{k+1})) \leq \varepsilon$ goto (5)
      • else $k = k + 1$, goto (1)
(5) solution to (D) compute $y^{k+1} = y^{k+1} - \bar{\varepsilon}b$ (see (133))
      • If $\bar{\varepsilon} < \varepsilon$ stop
      • else $k := k + 1$, goto (1)
```

ALGORITHM 3.3.3 The multiple eigenvalue case.

For the second example we took different input data for the same optimization problem taken from [**45**], the solutions of the problems as computed in [**45**] are printed in Table 6. The objective values will be used as reference to the output we obtained in Table 7.

The organization of Table 7 is:

- column 1 shows the name of the problem,
- column 2 shows the total amount of iterations to converge,
- column 3 shows how many times point (5) of Algorithm 3.3.3 is applied,
- column 4 shows the approximated Lagrange multiplier,
- column 5 shows the relative error of the approximated objective value compared to the objective value printed in Table 6,
- column 6 shows the approximated minimal maximum eigenvalue,
- column 7 shows the approximated objective value.

The true optimal value, solution and Lagrange multiplier are known beforehand. Therefore, we did not use the value ε of Algorithm 3.3.3 for termination. We stopped if the approximated objective value did not differ more than $1e - 5$ of the known objective value, and the approximated Lagrange multiplier did not differ more than $1e - 12$ of the actual Lagrange multiplier. In practice we will hardly ever encounter an exact multiple eigenvalue $\lambda_1 =: \lambda_2$, because of the approximated and rounded values we use. Therefore, we will call an eigenvalue multiple in our implementation

iter.	a	step size t	y^*	$\bar{\varepsilon}$	obj. value
0	5	0.250000			4.3750000
1	5	0.111803			4.2500000
2	5	0.000000	$[.15\ .15\ -.1\ -.1\ -.1]$	-0.8500	4.2500000
3	5	0.000000	$[1\ 1\ .75\ .75\ .75]$		4.2500000

TABLE 5 Output for Example 3.2.4. Empty spaces in the table are non-applicable outputs.

name	$\|p_8^+(A)\|$
Grcar	1766.31353
Ellipse	7710.27116
Bull's head	1239.41861
Lemniscate1	1.00000000
Lemniscate2	834.738575

TABLE 6 Solutions for the problems of Example 3.3.3 by [**45**] with relative accuracy less than $1e-11$. Thus all digits shown are correct.

of Algorithm 3.3.3 if

$$\frac{\lambda_1 - \lambda_2}{\lambda_1} \leq \varepsilon \quad (\lambda_1 \neq 0).$$

We took $\varepsilon := 1e-08$ for our examples.

EXAMPLE 3.3.2. We consider again Example 3.2.4. Recall that the solution for this problem is $y = [1\ 1\ \frac{3}{4}\ \frac{3}{4}\ \frac{3}{4}]^*$ with objective $b^*y = 4\frac{1}{4}$. The Lagrange multiplier a is a constant $a = 5$. For the improved algorithm we obtain the output in Table 5. With the improved algorithm, we find an optimum, and the optimal value y^+. It takes two iteration steps to find the optimal objective b^*y^+. Iteration step 2 is needed to verify whether we are in an optimum. Until this iteration step, we used the same approach as for Example 3.2.4. When we are in the situation of simple eigenvalues, this approach is the default. In iteration step 3 we had a break-down, when we applied the simple eigenvalue approach. Now the iterates converge nicely. The new approach conveniently handles the multiple eigenvalue that appears in iteration step 3. The termination check of point (4) in Algorithm 3.3.3 is fulfilled and in point (5) $\lambda_{\max}(C - \mathcal{A}^*(\widehat{y}))$ is driven to zero, which is done very accurately as can be concluded from iteration 3, where the iteration process has converged. The nice behavior we observed by the simple eigenvalue approach also extends to the multiple eigenvalue approach. The multiple eigenvalue approach seems very promising.

EXAMPLE 3.3.3. These optimization problems are taken from [**45**]. They involve problems of the type (40): Chebyshev polynomials of matrices A. We use the same size as in [**45**] so that we can compare our results with those in Table 6 (taken from [**45**]). We try to compute $\|p_8^+(A)\|$ for matrices of order (48×48). The sizes in the SDP formulation for these problems are the following: $n = 96, m = 17$. The

Lagrange multiplier is 1 for all problems and the vector b is of size $(2m+1)$, such that $b = [0 \ \cdots 0 \ -1]^*$.

The problem *Grcar:*

$$A = \begin{pmatrix} -1 & 1 & 1 & 1 & & & \\ & -1 & 1 & 1 & \ddots & & \\ & & -1 & 1 & \ddots & 1 & \\ & & & -1 & \ddots & 1 & 1 \\ & & & & \ddots & 1 & 1 \\ & & & & & -1 & 1 \\ & & & & & & -1 \end{pmatrix}.$$

The problem *Ellipse:*

$$A = \begin{pmatrix} 0 & 3 & & & & \\ 2 & 0 & 3 & & & \\ & 2 & 0 & \ddots & & \\ & & 2 & \ddots & 3 & \\ & & & \ddots & 0 & 3 \\ & & & & 2 & 0 \end{pmatrix}.$$

The problem *Bull's head:*

$$A = \begin{pmatrix} 0 & 0 & 1 & .7 & & & \\ 2i & 0 & 0 & 1 & \ddots & & \\ & 2i & 0 & 0 & \ddots & .7 & \\ & & 2i & 0 & \ddots & 1 & .7 \\ & & & 2i & \ddots & 0 & 1 \\ & & & & \ddots & 0 & 0 \\ & & & & & 2i & 0 \end{pmatrix}.$$

The problem *Lemniscate1:*

$$A = \begin{pmatrix} 1 & 1 & & & & & \\ & -1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & -1 & \ddots & & \\ & & & & \ddots & 1 & \\ & & & & & 1 & 1 \\ & & & & & & -1 \end{pmatrix}.$$

The problem *Lemniscate2:*

$$A = \begin{pmatrix} \alpha & & & & & & & \\ 1 & \alpha & & & & & & \\ & 5 & \alpha & & & & & \\ & & 5 & \ddots & & & & \\ & & & \ddots & \alpha & & & \\ & & & & 1 & \alpha & & \\ & & & & & 5 & \alpha & \\ & & & & & & 5 & \end{pmatrix}.$$

with $\alpha = (256/27)^{1/3}$.

The start vector y_0 is taken at random. The results are shown in Table 7, this are average results of several runs to make sure that there is no effect of the starting vector on the converging process. For all of the above-mentioned problems we encounter multiple eigenvalues in the iterates for the maximum eigenvalue. All the problems are solved to the desired precision, as we see from column 5 and 6 of Table 7. Note that this example has complex-valued input data, we corrected our algorithm to handle complex eigenvectors, and to keep directions and eigenvalues real.

As we see in column 2, the amount of iterations needed to get the desired precision of the optimal value, is larger than the amount of iterations observed by the other examples. We observed a lot of steps with a Newton direction for which the step size was, nevertheless, a lot smaller that 1. This reveals that the local quadratic information that was used for the computations of the search direction was far from complete. An explanation is that the input matrices A_i form an ill-conditioned system, although we applied a strategy for better conditioning as explained in [**45**], the matrices stay very sensitive to perturbations: the local behavior of the maximum eigenvalue function is very complicated. Therefore, we think that probably any local strategy will fail for this set of problems. This behavior makes these problems suitable test problems for validation of our method. In view of the large number of iteration steps one may wonder whether it is not better to use a local linear approximation as a default approach instead of the Newton approach for the search direction. The Newton approach is, per step, a lot more expensive than the linear approach. Since we do not know beforehand if we are solving such a difficult problem, we rather take the Newton method as default that can lead to fast convergence. Another reason is that in our examples, linear approximations lead to even worse convergence properties.

Even though we had less good converging speed as for the other examples, because of 'wild local' behavior of the maximum eigenvalue function, our approach works fine.

The most expensive part of the improved Algorithm 3.3.3 is the computation of the complete eigenvalue decomposition of the matrix $C - \mathcal{A}^*(y)$ every iteration

name	#iters	#upd.	a	error	$\|\lambda_{\max}\|$	$\|p_8^+(A)\|$
Grcar	65	13	1	$6.0e-07$	$1.1e-07$	1766.31766
Ellipse	26	6	1	$1.1e-06$	$2.5e-08$	7710.31826
Bull's head	44	3	1	$3.3e-06$	$2.3e-06$	1239.43899
Lemniscate1	31	3	1	$1.7e-09$	$3.1e-06$	1.00000001
Lemniscate2	48	3	1	$4.5e-06$	$9.8e-06$	834.754681

TABLE 7 Results for the problems of Example 3.3.3.

of the algorithm. Specifically for large matrices C and A_i this is a very time consuming operation. Therefore, we will develop a subspace procedure that will only make use of a small number, say 20, of eigenvalues and eigenvectors per iteration of Algorithm 3.3.3. The subspace procedure will be inspired by the projection method for SDP described in Subsection 2.4.2. If we only need a couple of eigenvalues per iteration step, then we can use an iterative method for computing eigenvalues. These methods are computationally not so expensive as direct methods and therefore are attractive for our algorithm when encountering large matrices C and A_i. The following section will describe some methods for computing eigenvalues.

3.4. Computation of eigenvalues

In this section we will describe the most popular ways of computing eigenvalues to give some insight in the computation of eigenvalues in the improved Algorithm 3.3.3, and how we could do better when encountering large matrices.

The computation of eigenvalues of matrices is equivalent with determining the zeros of a polynomial. For general matrices of order 5 or higher there exists no finite algorithm to compute the zeros and therefore the eigenvalues are computed iteratively to a certain precision. There are iterative methods that converge quadratically or even cubic. These are so fast that they are usually referred to as direct methods. For matrices of order n direct methods need $\mathcal{O}(n^3)$ elementary operations to compute the set of eigenvalues and eigenvector to high precision. For storage $\mathcal{O}(n^2)$ amount of memory is needed. For large matrices these methods are to time consuming.

For larger problems reduced matrix techniques come in the picture. These methods reduce the original large system to a system of low size. The eigenvalues of the reduced matrix turn out to be good approximations of the wanted eigenvalues of the original matrix. If all eigenvalues are needed, the reduced methods would also be very time consuming and would therefore be no option. In Subsection 3.4.1 we give an example of a direct method and in Subsection 3.4.2 we give an example of a reduction method.

3.4.1. Direct methods

Direct methods are for example the Power method and the QR method [**17**], [**36**]. We will briefly review the idea of the *QR method.* We first need the definition of a *Schur form* of a matrix.

THEOREM 3.4.1 ([**17**, p.313]). *If $A \in \mathbb{C}^{n\times n}$, then there exists a unitary $Q \in \mathbb{C}^{n\times n}$ such that*

$$Q^*AQ = \widehat{R} = D + N$$

where $D := \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ and N strictly upper triangular. Furthermore Q can be chosen so that the eigenvalues λ_i appear in any prescribed order along the diagonal.

Q is called the matrix of *Schur vectors* of A. If the matrix A is complex, then the matrices $\widehat{R}$ and Q may also be complex. It can be advantageous to work in real arithmetic. Therefore, we define a real Schur form.

THEOREM 3.4.2 ([**17**, p.341]). *If $A \in \mathbb{C}^{n\times n}$, then there exists a orthogonal $Q \in \mathbb{R}^{n\times n}$ such that*

$$Q^*AQ = R = \begin{pmatrix} R_{11} & R_{11} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{mm} \end{pmatrix}$$

where R_{ii} is either a real 1×1 matrix or a real 2×2 matrix having complex conjugate eigenvalues.

An eigenvector x of A corresponding to λ follows from the solution of $Ry = \lambda y$ with $x = Qy$.

We will describe an iterative procedure to compute the Schur form of a matrix A. Suppose that the matrix A has real eigenvalues. Start with a complete set of orthonormalized independent vectors $u_1, \ldots, u_n$. Define $U_0 := (u_1, \ldots, u_n)$. We could for example take $U_0 = I$.

With $A_0 := A$ we compute a QR-factorization of

$$(A_0U_0 =)A = U_1R_1.$$

Set $Q_1 := U_1$. Then the matrix A_1 can be computed as

$$A_1 := Q_1^*A_0Q_1 = R_1Q_1.$$

A_1 is a similarity transformation of the matrix A_0, and hopefully more in the direction of the Schur form of A, than $A_0 = A$ itself.

The next step will be to form AQ_1 and compute a QR-factorization $AQ_1 = U_2R_2$ of this in order to find a new iterate A_2 that is even more in the direction of the Schur form of A, than A_1. But since $A = Q_1R_1$ we see that

$$AQ_1 = Q_1^*R_1Q_1.$$

If we now factor R_1Q_1 as $R_1Q_1 = Q_2R_2$, then we see that

$$AQ_1 = Q_1Q_2R_2. \tag{161}$$

Apparently, Q_2 can be viewed as the orthogonal transformation that corrects Q_1 to U_2. From (161) we conclude that

$$R_1Q_1 = A_1 = Q_1^*AQ_1 = Q_2R_2,$$

```
Start with $A_0 := A$, and $k = 1$,
    (1) factor $A_{k-1} = Q_k R_k$,
    (2) compute $A_k = R_k Q_k$,
    (3) • if converged, then $R_k = D + N$ where $D = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ are the eigenvalues of $A$, $x_i = Q y_i$ are the eigenvectors of $A$, where $y_i$ solve $R_k y_i = \lambda_i y_i$.
        • else $k = k + 1$, goto (1)
```

ALGORITHM 3.4.4 The QR method.

and thus that the factor Q_2 had also been obtained if continued with the iteration with A_1. We set

$$A_2 := Q_2^* A_2 Q_2 = Q_2^* Q_1^* A Q_1 Q_2,$$

and start again. The iterates A_i will converge to the *Schur form* of A, and the product $Q_1 Q_2 Q_3 \cdots Q_i$ of the correction matrices converges to the matrix of *Schur vectors* corresponding to A. This approach is summarized in Algorithm 3.4.4.

3.4.2. Reduction methods

In this section we briefly review one of the iterative methods for computing some of the eigenvalues. Iterative methods are for example Lanczos [**28**], Arnoldi [**3**], Jacobi-Davidson [**42**], described in [**17**], [**36**], [**10**] and [**11**]. We will describe *Jacobi-Davidson.* For a standard eigenvalue problem, Jacobi-Davidson selects an approximate eigenvector from a subspace U that is extended in each step. So instead of a complete set of independent vectors as for direct methods we start with a orthonormal basis $\{u_1, \dots, u_l\}$ of the subspace U with $l \ll n$. Define the matrix U as $U := (u_1, \dots, u_l)$. Each step consists of two parts. In the first part the projected eigenvalue problem

$$(U^* A U - \theta U^* U) v = 0,$$

is solved and a solution (θ, v) is selected. The value θ and $q = Uv$ are an approximate eigenvalue and eigenvector, respectively. The residual is

$$r := (A - \theta I) q, \quad \text{with} \quad \|q\| = 1.$$

We see that $q \perp r$. In the second part the subspace U is extended by a vector $u \perp q$. This vector u has to solve the *correction equation* approximately,

$$q \perp u \quad \text{and} \quad (I - qq^*)(A - \theta I)(I - qq^*) u = -r. \tag{162}$$

The subspace U is extended with the vector u, that is, $\mathsf{U} := \mathrm{span}\{\mathsf{U}, u\}$ such that its matrix representation U is $U = (U, u)$ with $U^* U = I$. The columns of the matrix U are not automatically orthonormal, we have to apply a Gram-Schmidt orthogonalization process on the columns of the matrix U. If the value θ is an eigenvalue, then its eigenvector is contained in $\mathrm{span}\{\mathsf{U}, u\}$, that is, its eigenvector is contained in the subspace U and in the solution of the correction equation (162). If θ is not close to the wanted eigenvalue, then it is often more efficient to replace it by a fixed target close to the wanted eigenvalue. In this way we circumvent that θ converges to an undesired eigenvalue.

The main difference of the direct and the iterative method is the subspace. The subspace projection leads to problems of low size, which are computationally cheap to solve. If only a small amount of eigenvalues are wanted of large system, this is the way to compute them rather cheaply.

3.5. The subspace principle

As we already pointed out, the most expensive part of the Algorithm 3.3.3 is the computation of a full set of eigenvalues and eigenvectors at every iteration step. Now that we introduced in the last subsection a method that is computationally not so expensive for computing some eigenvalues and eigenvectors, we need some ideas for using only a couple of eigenvalues every iteration step, and still converge to an optimum. Therefore, we introduce the following projection inspired by the Subsection 2.4.2. This projection will not reduce the whole of the optimization problem to a reduced problem, but only the eigenvalue constraint $\lambda_{\max}(C - \mathcal{A}^*(y))$.

We explain how to incorporate subspaces in the procedure of Algorithm 3.3.3. Suppose we compute the s largest eigenvalues, $\lambda_1 \geq \ldots \geq \lambda_s$, of the matrix $\widehat{C} := aC - a\mathcal{A}^*(y_0) + \langle b, y_0 \rangle I$ at y_0, and $s \ll n$, with one of the reduction methods for eigenvalue computation in Subsection 3.4.2. Define the $n \times s$ matrix $P := (u_1, \cdots, u_s)$ of the orthonormal eigenvectors corresponding to the s largest eigenvalues u_i of $\widehat{C}$, and define the subspace $\mathsf{U} := \operatorname{span}\{u_1, \cdots, u_s\}$.

For $s \ll n$, we consider the maps:

$$\mathcal{P} : \mathsf{S}^n \to \mathsf{S}^s, \quad \mathcal{J} : \mathsf{S}^s \to \mathsf{S}^n,$$

that are defined by

$$\mathcal{P}(X) = P^* X P, \quad \mathcal{J}(H) = PHP^*.$$

Note that

$$P^* \widehat{C} P = \operatorname{diag}(\lambda_1, \ldots, \lambda_s).$$

Instead of incorporating the search direction procedure of Algorithm 3.3.3 immediately, that is, instead of following the procedure to find an approximation to d^+ from (112),

$$d^+ = \operatorname*{argmin}_d \lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(d) + b^* dI),$$

we introduce the following projection procedure on the subspace U:

$$(163) \qquad (a\widehat{C} - a\mathcal{A}^*(d) + b^* dI) \quad \to \quad (aP^*\widehat{C}P - aP^*\mathcal{A}^*(d)P + b^* dI).$$

Define $\widehat{A}_i := P^* A_i P$ $(i = 1, \ldots m)$, then

$$P^* \mathcal{A}^*(d) P = \sum_{i=1}^m d_i P^* A_i P = \sum_{i=1}^m d_i \widehat{A}_i,$$

and approximate d^+ by $\widehat{d}$

$$\widehat{d} = \operatorname*{argmin}_{d} \lambda_{\max}\Big(a \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_s \end{pmatrix} - a\sum_{i=1}^{m} d_i \widehat{A}_i + b^* d I_s\Big). \tag{164}$$

This equation is of low size s and can be solved relatively quickly by the procedures explained in the Subsections 3.2.2, 3.3.2, and 3.3.3.

The next part of the procedure is to find the appropriate step size for the direction d. Again we will use a system of low size for this. Let us return to problem (100):

$$t^+ = \operatorname*{argmin}_{t} \lambda_{\max}(\widehat{C} + t\widehat{F}).$$

After incorporating the projection onto the subspace U, spanned by orthonormal eigenvectors;

$$\widehat{C} + t\widehat{F} \quad \rightarrow \quad P^*\widehat{C}P + tP^*\widehat{F}P, \tag{165}$$

we are left with finding a step size t such that,

$$\widehat{t} = \operatorname*{argmin}_{t} \lambda_{\max}(P^*\widehat{C}P + tP^*\widehat{F}P). \tag{166}$$

This equation is of low size s and can be solved relatively quickly by the procedures explained in the Subsections 3.2.1.

We have to consider some new situations that can occur. Since we have lost information by projecting on only s eigenvalues instead of using the full set of eigenvectors and eigenvalues. For the full set of eigenvectors and eigenvalues we assumed that there existed a finite solution. Thus whatever search direction is selected, we know that a finite solution will be found for this situation, if the Lagrange multiplier exists. The following situations can occur when solving the subproblem:

(1) the procedure written in Subsection 3.2.1 is not able to come up with a finite step size t, because all the eigenvalue curves of the chosen s eigenvalues are increasing or decreasing. Thus the optimization problem has no finite solution in the subspace U. See Figure 8, Situation 1, here $s = 2$ and, the horizontal axis represent the step size to some computed direction d, and the vertical axis represent the eigenvalues. The solid lines are the eigenvalue curves for the projected matrix $P^*\widehat{C}P + tP^*\widehat{F}P$ as a function of t. And the dashed line is an eigenvalue curve of $\widehat{C} + t\widehat{F}$ that is not included in $P^*\widehat{C}P + tP^*\widehat{F}P$. We start at $t = 0$ in $\circ$, where all eigenvalue curves are decreasing, no finite solution is found. since the crucial crossing eigenvalue curve $--$ is not in the subsystem. We are not able to find the minimum $*$,

(2) the new objective value of the full system has increased, although the objective value of the projected system shows decrease, that is,

$$\lambda_{\max}(aP^*\widehat{C}P - aP^*\mathcal{A}^*(td)P + tb^*dI) < \lambda_1,$$

but

$$\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(td) + tb^*dI) > \lambda_1.$$

This situation gives a too large step in the direction of the update td. See Figure 8, Situation 2, here $s = 1$. We start at $t = 0$ in $\circ$ and approximate the maximum eigenvalue curve with a "parabola", which has minimum $\square$. If we had started with the full set of eigenvalues and eigenvectors, the crucial crossing eigenvalue curve ($--$) would not have been omitted, and we would have spotted that the eigenvalue in $\square$ is not the maximum eigenvalue, actually in $*$ is the maximum eigenvalue which is clearly larger than the maximum eigenvalue in $\circ$.

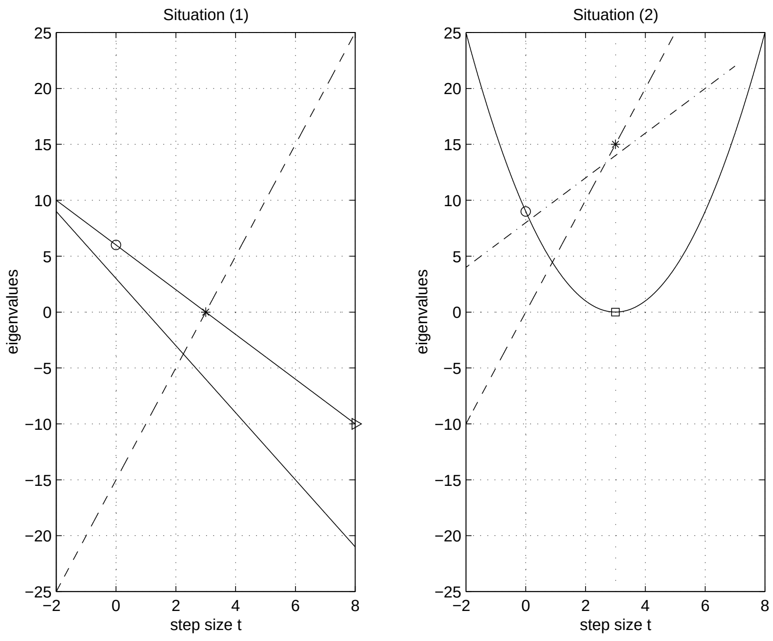

FIGURE 8 Illustration of the problems occurring when solving a projected system.

Situation (1). An infinite step size means that all computed eigenvalues curves are decreasing. If we detect an infinite step size t when running the step size procedure of Subsection 3.2.1, we propose to extend the subspace U by a new vector $v_1 \perp \mathsf{U}$ with v_1 the normalized eigenvector corresponding to the maximum eigenvalue θ_1 of the matrix $\widehat{F} = a\mathcal{A}^*(d) + b^*dI$,

$$\widehat{F}\theta_1 = \theta_1 v_1, \quad \text{and} \quad \|v_1\| = 1.$$

Clearly, the computed v_1 will not automatically be perpendicular to U, we have to apply a Gram-Schmidt orthogonalization process to orthogonalize v_1 with respect to the columns of the matrix P. The new projection matrix $\widehat{P}$ will be $\widehat{P} = (P, v_1)$. This is a natural choice to extent the subspace U, since for large t the matrix $\widehat{F}$ will determine the maximum eigenvalue of $\widehat{C} + t\widehat{F}$, this maximum eigenvalue curve is decreasing for $t \downarrow 0$. Since we assumed that a solution of the problem (D) exists this will guarantee us that t is finite. Therefore, the maximum eigenvalue curve of $t\widehat{F}$ has to cross the maximum eigenvalue curve of $\widehat{C} + t\widehat{F}$ for $t \downarrow 0$. We approximate the step size t by

$$t = \operatorname*{argmin}_t \lambda_{\max}(\widehat{P}^*\widehat{C}\widehat{P} + t\widehat{P}^*\widehat{F}\widehat{P}).$$

Note that a finite t is still not a guarantee that $\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(td) + tb^*dI)$ is smaller than λ_1. If indeed $\lambda_{\max}$ has not decreased we follow the instructions in Situation (2).

Situation (2). Situation (2) is detected when we are computing $\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(td) + tb^*dI)$ at the termination step of Algorithm 3.3.3. We propose to extend the subspace U by a new vector $w_1 \perp \mathsf{U}$ such that w_1 is the normalized eigenvector corresponding to $\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(td) + tb^*dI)$. Again, the computed w_1 will not automatically be perpendicular to U, we have to apply a Gram-Schmidt orthogonalization process to orthogonalize v_1 with respect to the columns of the matrix P. The new projection matrix $\bar{P}$ will be $\bar{P} = [P, w_1]$. And a new step is computed:

$$\widehat{t} = \operatorname*{argmin}_t \lambda_{\max}(\bar{P}^*\widehat{C}\bar{P} + t\bar{P}^*\widehat{F}\bar{P}).$$

This decreases the value of t, but still does not give any guarantee that $\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(\widehat{t}d) + \widehat{t}b^*dI)$ has decreased: we could have missed another eigenvalue curve, see Figure 8, Situation 2, where $s = 1$. The solid line is the eigenvalue curve for the projected problem, the dashed line is the extra eigenvalue curve coming from the extension with v_1 and the dashed $. - -.$ is the eigenvalue curve that determines the minimal maximum eigenvalue of the full system.

Therefore, if it happens again, we repeat the procedure and extend $\operatorname{span}\{\mathsf{U}, w_1\}$ by a new vector $w_2 \perp \operatorname{span}\{\mathsf{U}, w_1\}$ such that w_2 is the normalized eigenvector corresponding to $\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(\widehat{t}d) + \widehat{t}b^*dI)$, etc. We iterate this procedure until t is such that

$$\lambda_{\max}(a\widehat{C} - a\mathcal{A}^*(td) + tb^*dI) \leq \lambda_1. \tag{167}$$

(We have the guarantee that this will happen, because in a finite amount of steps our subspace will be the same as our complete space, for which (167) is true.)

Some extra notes are in place. Every iteration step we start with a new set of s eigenvalues and eigenvectors. Up until now we did not use any information from previous steps. This seems to be feasible and could be an interesting field for further research. The value of s should be selected, such that, the amount of work needed to compute the s eigenvalues does not dominate the total amount of work needed every step of the method.

With these modifications we are able to formulate Algorithm 3.5.5.

```
(0) input
      • a vector $b$, matrices $A_i$ and $C$,
      • an initial point $y^0 \in \mathbb{R}^m$, an $\varepsilon$ for termination, set $k = 0$, and set $s$ at a reasonable value, for example, in our examples $s := 20$,
(1) search direction compute the $s$ largest eigenvalues of $C - \mathcal{A}^*(y^k)$ by a reduction method and perform projection
      $(a\widehat{C} - a\mathcal{A}^*(d) + b^* dI) \quad \rightarrow \quad (aP^*\widehat{C}P - aP^*\mathcal{A}^*(d)P + b^* dI).$
    check the multiplicity $l$ of the maximum eigenvalue
      • if $l = 1$ compute $d$ (and $a$) following the strategy of Section 3.2 for the projected problem,
      • else compute $d$ (and $a$) following the strategy of Section 3.3 for the projected problem,
(2) step size perform projection
      $\widehat{C} + t\widehat{F} \quad \rightarrow \quad P^*\widehat{C}P + tP^*\widehat{F}P$
    compute step size $t$ according to Subsection 3.2.1 for the projected problem,
      • if $t$ is finite follow Situation 2 in Section 3.5, and goto (3)
      • else follow Situation 1 in Section 3.5, and goto (3)
(3) update compute $y^{k+1} = y^k + td$
(4) termination compute $\lambda_{\max}(C - \mathcal{A}^*(y^{k+1}))$
      • if $\lambda_{\max}(C - \mathcal{A}^*(y^k)) - \lambda_{\max}(C - \mathcal{A}^*(y^{k+1})) \leq \varepsilon$ goto (5)
      • else $k = k + 1$, goto (1)
(5) solution to (D) compute $y^{k+1} = y^{k+1} - \bar{\varepsilon}b$ (see (133))
      • If $\bar{\varepsilon} < \varepsilon$ stop
      • else $k := k + 1$, goto (1)
```

Algorithm 3.5.5 The subspace eigenvalue method.

3.5.1. Examples to illustrate the subspace Newton method

The modification with the subspace approach described in Section 3.5 is based on a very simple idea. We would like to apply this to show the possibility to decrease the problem size and therefore the computation time for large optimization problems. Our implementation of the multiple eigenvalue subspace procedure is done only to check the validity of the constructed theoretical framework, for simple and low size testing problems. The current numerical implementation is not applicable to large-scale real-life optimization problems. Therefore, we are not able to check the gain in computation-time, that reduction methods can show in comparison to direct methods for the computation of eigenvalues for large-scale problems. Nevertheless, we can observe the behavior of the iteration process, that is most likely the same for large-scale problems. Therefore, we do some tests with the subspace procedure on the examples we used in Subsection 3.3.5. We used a direct method for the computation of the eigenvalues, since the direct methods are in practice just as fast or even faster than the reduction methods for the problem sizes of the examples that we tested. The results for different examples are printed in Table 8 and 9. The Tables are divided like those in Subsection 3.3.5. Since we know the true optimal value, solution, and Lagrange multiplier we did not use the value ε of Algorithm 3.5.5 for termination, we stopped if the approximated objective value did not differ more than $1e-4$ of the known objective value, and the approximated Lagrange multiplier did not differ more than $1e-12$ of the actual Lagrange multiplier. We will call an eigenvalues multiple in our implementation of Algorithm 3.5.5 if

$$\frac{\lambda_1 - \lambda_2}{\lambda_1} \leq \varepsilon \quad (\lambda_1 \neq 0).$$

We took $\varepsilon := 1e-06$ for our examples.

EXAMPLE 3.5.1. This example is an extension of Example 3.2.2. Our data for $n = m = 100$ is the following,

$$C = \begin{bmatrix} 0 & -1 & 0 & & \\ -1 & 0 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & -1 \\ & & 0 & -1 & 0 \end{bmatrix},$$

The Lagrange multiplier a is equal to $a = 100$, the vector b is of size m such that $b = [1 \ \cdots \ 1]^*$, and the $n \times n$ matrices A_i, $i = 1, \ldots, m$, are filled with zeros except on the (i,i)-th entry there is a one. We used a subspace with $s = 20$ eigenvectors each iteration. The known optimal solution is $b^* y = 198$ with $y = [1 \ 2 \ 2 \ \ldots \ 2 \ 2 \ 1]^*$. The results are printed in Table 8.

Although we observe a slower iteration process, that can be explained by the use of only 20 eigenvectors instead of the complete eigenvalue decomposition, the iteration process is constant. In the first 10 iteration steps, where we find an approximate value with an accuracy of less than 1e-3, although the approximate Lagrange multiplier and approximated solution are not yet accurate. It takes another thirty steps to get these also very accurate. That the Lagrange multiplier is not immediately accurate can be explained by the behavior of the local linear approximation of only 20 eigenvectors. We also tested this example with the approximated Lagrange multiplier set to $a := 100$ during all iteration steps. The output was similar to the output in Table 8. When starting with $s = 100$, the algorithm converges in 3 iterations.

By investigating the eigenvalue curves we discovered that after approximately 15 iterations eigenvalues start clustering near the maximum eigenvalue, the more we approach optimality the more eigenvalues behave like this. The subspace of the 20 eigenvectors is not accurate enough any more, and we see that the iteration process slows down. For higher accuracy of the solution we need to investigate subspaces that can handle the problem of clustering, or even better, that can predict that a specific eigenvalue becomes important in the next iteration. Perhaps, this can be established by using information from previous steps.

#iters	#upd.	a	error	$\lvert\lambda_{\max}\rvert$	obj. value
1	0	96.9	$2.7e-02$	$2.0e+00$	$1.926952e+02$
10	1	99.8	$5.1e-04$	$1.3e-01$	$1.981009e+02$
30	3	100.0	$9.7e-07$	$5.7e-07$	$1.980002e+02$
52	10	100.0	$9.8e-11$	$1.1e-16$	$1.980000e+02$

TABLE 8 Output for Example 3.5.1.

In none of the iteration steps we detected an infinite step size nor an increase of the objective value as described in the two situations of Section 3.5. In the next subsection we will discuss whether there is any dependency of s on m, n, or dependency between s and the convergence rate.

We conclude that the subspace procedure is promising in general, although we see that our approach of using every iteration 20 new eigenvectors to project on can be improved.

EXAMPLE 3.5.2. The optimization problems are those of Example 3.3.3.

We know that multiple eigenvalues can occur as we experienced in previous tests, therefore we used a subspace with $s = 20 + l$ eigenvectors every iteration step. We have every iteration step information of 21 different eigenvalues, thus we can compare the results per iteration step.

The start vector y_0 is again taken at random. The results are printed in Table 9. They are average results of several runs to make sure that there is no effect of the starting vector on the converging process.

Problem Grcar is solved with an equal amount of iteration steps as in Table 7. This is the only problem that did not need more iterations to converge. Usually, we would expect that using a subspace, and therefore less information for the computation of the update, would lead to more iterations. For the Grcar problem, all the important information to compute an accurate update, is stored in the s eigenvectors that form the subspace projection.

For the other problems we observe in Table 9 an increase of the iteration steps compared to the amount of iterations in Table 7. This is the same observation as for Example 3.5.1, it leads to the same conclusions.

What we can not see in Table 8, but what did occur is that most of the problems needed subspace extensions. At the start of the iteration process we encountered often an infinite step size as discussed in situation 1 of Subsection 3.5 occurred a couple of iterations. This can be explained by the fact that the eigenvectors belonging to the 21 largest eigenvalues in the starting point have no connection to those in the optimal solution if the starting points are not near the optimal solution. This difficulty cannot be overcome.

At the end of the iteration process the increase of the objective value, as discussed in situation 2 of Subsection 3.5, occurred quite often. Also we observed that, as for the results in Table 8, that after approximately 15 iterations eigenvalues start clustering near the maximum eigenvalue, the more we approach optimality the more eigenvalues behave like this. Both observations point out that the subspace is not

name	#iters	#upd.	a	error	$\lvert\lambda_{\max}\rvert$	$\lVert p_8^+(A)\rVert$
Grcar	65	10	1	$4.0e-07$	$5.9e-06$	1766.31424
Ellipse	57	5	1	$2.4e-05$	$1.0e-06$	7710.45862
Bull's head	93	6	1	$6.2e-06$	$5.0e-06$	1239.42482
Lemniscate1	42	5	1	$3.2e-07$	$5.1e-06$	1.00000514
Lemniscate2	87	20	1	$6.7e-04$	$9.4e-06$	835.300777

TABLE 9 Results for the problems of Example 3.5.2.

good enough to find accurate updates. Thus, more information than this subspace contains is needed to solve with higher accuracy. We discussed already in Example 3.3.3 that these problems are difficult. The local behavior of the maximum eigenvalue function is very complicated. Thus, when using even less information this does not help to increase the accuracy of the local approximation. Especially for Example Lemniscate2 this is the case, after 15 iterations we have already an accuracy of $1.8e-03$. This explains also the rather slow convergence after 15 steps.

Reconsidering all the observations, we conclude that the results that are shown in Table 9 are actually rather promising. The convergence rate for the full set of eigenvalues and eigenvectors was slow, and it did not slow-down dramatically by using our subspace approach.

3.5.2. Subspace dimension

In this subsection we will discuss the relation between the dimension of the subspace and the amount of iterations needed to solve the optimization problem.

In Table 10 we printed the results for solving Example 3.5.1 with problem size $m=n=120$. We used 6 different subspace sizes, size $s=10,20,40,60,90$, and 120.

In Table 11 we printed the results for solving Example 3.5.1 with subspace size $s=20$ and problem sizes equal to $n=m=30,60,90,120$ and 240.

With the help of the known true optimal values, solutions and Lagrange multipliers, the algorithm is terminated as soon as the Lagrange multiplier, a, has an accuracy of $1e-12$ and the relative error of the objective function with respect to the optimal solution is less than $1e-5$.

In the two tables we recognize linear behavior between the subspace size and the amount of iterations, this is a bit disappointing. If the problem size is doubled than also the amount of iterations is doubled. But, there is also other behavior of the iteration process that is not visible in the Tables 10 and 11. Typically our method shows very fast initial convergence and a strong tailing off effect as the iterates approach the optimal solution. One of the reasons is the clustering of eigenvalues near zero, when approaching the optimal solution. If only a rough guess is needed of the optimal solution, the algorithm finds a guess with relative error $1e-3$ in less than 16 iterations. The convergence speed may then not be quadratic, but the amount of operations needed to compute the solution is very low ($\mathcal{O}(s\cdot n^2)$ versus $\mathcal{O}(n^3)$ per step).

subspace size	#iters	a	error
10	65	120	$8.0e-07$
20	25	120	$2.4e-06$
40	20	120	$1.5e-06$
60	20	120	$8.4e-07$
90	10	120	$8.8e-07$
120	3	120	$1.0e-12$

TABLE 10 Output for Example 3.5.1 with $m = n = 120$ and with relative error $< 1e-5$.

problem size	#iters	a	error
30	8	30	$8.8e-06$
60	17	60	$2.2e-07$
90	25	90	$9.6e-07$
120	25	120	$2.4e-06$
240	67	240	$3.1e-06$

TABLE 11 Output for Example 3.5.1 with subspace size 20, with relative error $< 1e-5$ and with $n = m =$ problem size.

To select the optimal size of the subspace, we have to consider the time needed to compute a set of eigenvalues, the expected total amount of iterations and, the amount of clustered eigenvalues near the minimal maximum eigenvalue. The most expensive part of the algorithm is the computation of eigenvalues every iteration of the algorithm, so the time needed to compute the eigenvalues times the expected amount of iterations should be minimal.

3.6. Other methods for minimizing the maximum eigenvalue

In this section we will briefly discuss two other iterative methods, that rely on minimizing the maximum eigenvalue. The first method is the *spectral bundle method* by Helmberg and Rendl [**20**]. This method is specially designed to solve large scale combinatorial optimization problems of which the relaxation can be written as semidefinite programming problems (D) (cf. Example 1.2.1, Section 1.2.4). The second method is developed by Overton [**35**] and searches for the minimal maximum eigenvalue in magnitude, of a symmetric matrix valued function.

3.6.1. The spectral bundle method

This subsection is based on the article [**20**] by Helmberg and Rendl. The spectral bundle method assumes that (D) can be written as (E)

$$\text{(E)} \quad \min_{y \in \mathbb{R}^m} \quad a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y \rangle,$$

and in addition that the Lagrange multiplier a is explicitly known and constant. They also assume, like we do in Subsection 3.1.1, that both (P) and (D) are solvable and that at least on of (P) and (D) has a strictly feasible solution. Their method consists of two basic parts, the bundle concept and the proximal point idea. The Bundle concept derives information from a bundle (subspace) collected from previous iterates. The proximal point idea determines a new iterate by solving a convex problem with a penalty term, to ensure local minimization. The spectral bundle method is implemented in C by Helmberg, and the code is available at `http://www.zib.de/helmberg`.

Bundle concept. They developed a method for minimizing $f(y)$, where

$$f(y) := a\lambda_{\max}(C - \mathcal{A}^*(y)) + \langle b, y\rangle.$$

Let

$$F(W,y) := a\langle C - \mathcal{A}^*(y), W\rangle + \langle b, y\rangle. \tag{168}$$

Then $f(y)$ can be formulated as the semidefinite program

$$f(y) = \max_W \{F(W,y) : W \geq 0, \operatorname{trace}(W) = 1\}. \tag{169}$$

REMARK 3.6.1. To prove the equivalence, take u_1 the normalized eigenvector corresponding to the largest eigenvalue of $C - \mathcal{A}^*(y)$ in the solution $y = y^+$. Then the argmax of (169) is $W = u_1 u_1^*$.

In order to reduce the size of the optimization problem (169), they consider only W from some subcone V of S^n_+. They start with a local model of the function f at current iterate $y = \widehat{y}$. Let

$$P^*(C - \mathcal{A}^*(\widehat{y}))P = \operatorname{diag}(\lambda_1, \dots, \lambda_k) \quad \text{with} \quad P^*P = I,$$

and represent the eigenvalue decomposition with respect to the k largest eigenvalues $\lambda_1 \geq \cdots \geq \lambda_k$ of $C - \mathcal{A}^*(\widehat{y})$. Define $\mathsf{V} := \{PVP^* : V \geq 0, \operatorname{trace}(V) = 1\}$. P will be called the *bundle*. Now, define the *local minorant* of f by,

$$\widehat{f}(y) := \max_W \{F(W,y) : W \in \mathsf{V}\}.$$

Thus $F(W,y)$ is maximized over a subcone of S^n_+ for $\widehat{f}$. Then

(1) $f(y) \geq \widehat{f}(y), \quad \forall y,$
(2) $f(\widehat{y}) = \widehat{f}(\widehat{y}),$

Proximal point idea. In order to determine a new iterate $y = \widehat{y} + d$ from the current iterate $\widehat{y}$, they solve

$$\min_d \quad \widehat{f}(\widehat{y} + d) + \frac{u}{2}\langle d, d\rangle, \quad \text{for some chosen parameter } u. \tag{170}$$

Thus they minimize, the local minorant $\widehat{f}$, locally around $\widehat{y}$. The term $\frac{u}{2}\langle d, d\rangle$ is some kind of barrier function or penalty function to keep the update d near the current iterate $\widehat{y}$. They call this the *proximal point idea.* The parameter u controls the size of acceptable updates d: they use u to quantify 'the notion of local'.

Substituting the definition of $\widehat{f}$ in expression (170) yields,

$$\min_d \max_{W \in \mathsf{V}} F(W, \widehat{y} + d) + \frac{u}{2}\langle d, d\rangle, \tag{171}$$

Up till now only local information is used from the current iteration point $\widehat{y}$. In the general situation there is available a 'bundle' of vectors from the previous iterates. Those vectors are, for example, the eigenvectors of previous iterates. To use more than only local information from the current iteration point $\widehat{y}$, they collect the interesting vectors in a matrix, and denote by P an orthonormal basis of its span. To keep the dimension of the subspace of all positive semidefinite matrices small, they aggregate some information from previous iterates into a single matrix $\overline{W}$ with trace $\left(\overline{W}\right) = 1$, once the number of columns of P exceeds some limit, and they set

$$\mathsf{V} := \{\alpha\overline{W} + PVP^* : \alpha + \operatorname{trace}(V) = 1, \alpha \geq 0, V \geq 0\}.$$

Expression (171) be simplified, because there is no explicit restriction on d. Note that

$$F(W, \widehat{y} + d) = a\langle C - \mathcal{A}^*(\widehat{y}), W\rangle + \langle b, \widehat{y}\rangle + \langle b - a\mathcal{A}(W), d\rangle,$$

thus $\nabla_d F = b - a\mathcal{A}(W)$. Therefore, they get

$$\begin{aligned}
& \min_d \max_{W\in\mathsf{V}} F(W, \widehat{y} + d) + \frac{u}{2}\langle d, d\rangle, \\
= \quad & \max_{W\in\mathsf{V}} \min_d F(W, \widehat{y} + d) + \frac{u}{2}\langle d, d\rangle, \\
= \quad & \max_{W\in\mathsf{V}, b - a\mathcal{A}(W) + ud = 0} F(W, \widehat{y} + d) + \frac{u}{2}\langle d, d\rangle, \\
= \quad & \max_{W\in\mathsf{V}} a\langle C - \mathcal{A}^*(\widehat{y}), W\rangle + \langle b, \widehat{y}\rangle - \frac{1}{2u}\langle a\mathcal{A}(W) - b, a\mathcal{A}(W) - b\rangle.
\end{aligned}$$

The first equality is interchanging min and max, this is difficult to prove but can be done by using that there exist a finite solution for both (P) and (D). The second equality follows using that for given W the minimum over d is such that $b - a\mathcal{A}(W) + ud = 0$. Since for the inner minimization with respect to d and for given W the first order optimality for

$$\min_d \quad F(W, \widehat{y} + d) + \frac{u}{2}\langle d, d\rangle,$$

is equal to

$$0 = \nabla_d\Big[F(W, \widehat{y} + d) + \frac{u}{2}\langle d, d\rangle\Big] \Leftrightarrow d = \frac{1}{u}(a\mathcal{A}(W) - b). \tag{172}$$

It is now clear that the computation of the update d for (171) will be done in two steps, first solve (172), and then compute d by (172). They motivate that the computation of a solution to (172) can be done by a standard interior-point method for convex programming problems, this is a rather expensive procedure, but still cheaper than the computation of the eigenvalues and eigenvectors every iteration step.

Details. The solution of expression (172) is unique, and is of the form

$$W^* = \alpha^*\overline{W}^k + P^k V^* (P^k)^*.$$

Let $Q\Lambda Q^*$ be an eigenvalue decomposition of V^*. Then the important part of the spectrum of W^* is spanned by the eigenvectors associated with the large eigenvalues of V^*. Now split Q into two parts $Q = [Q_1 Q_2]$, with (corresponding eigenvalues

iter.	$\|d\|$	$\lambda_{\max}$	obj. value
1	0.343	0.85498	4.2749135
3	0.0373	0.85011	4.2505677
6	$3.77e-05$	0.85001	4.2500410
9	$2.36e-05$	0.85000	4.2500158
12	$1.47e-05$	0.85000	4.2500060
15	$1.16e-05$	0.85000	4.2500031

TABLE 12 Output for Example 3.2.4 computed by the spectral bundle method, Algorithm 3.6.6.

also put into two matrices: Λ_1, Λ_2), Q_1 containing the eigenvectors associated to large eigenvalues of V^* and Q_2 the remaining. Now the next P^{k+1} is computed to contain P^kQ_1 and in the other columns at least the eigenvector corresponding to the maximal eigenvalue of $C - \mathcal{A}^*(y^{k+1})$, and the next matrix $\overline{W}^{k+1}$ is defined to be

$$\overline{W}^{k+1} = \alpha^*\overline{W}^k + \frac{P^kQ_2\Lambda_2Q_2^*(P^k)^*}{(\alpha^* + \operatorname{trace}(\Lambda_2))}. \tag{173}$$

The spectral bundle algorithm is now formulated in Algorithm 3.6.6.

We will discuss the properties of the spectral bundle algorithm and compare these with our approach.

The spectral bundle method is a linear approximation method, in the sense that it uses first order information from the maximum eigenvalue function. Therefore, fast or quadratic convergence cannot be expected. This is visible in Table 12, that reports on the solution of Example 3.2.4. This is different from our method that does inherit fast convergence as is clearly visible in the output for the same example in Table 4. The spectral bundle method is especially designed for large-scale combinatorial optimization problems. It is only applicable if the correct Lagrange multiplier is known beforehand, which is usually the case for combinatorial optimization problems. Probably, the most sophisticated part of the spectral bundle method is the subspace procedure. The method does use a penalty function to to keep the update near the last iterate, which is necessary since the approximation of this function is only local. This penalty function approach a different than the step size approach that we use. But the goal of keeping the update near the last iterate is the same.

We apply a step size procedure because of the second order approximations, that give a good guess of the progress that we will make. Linear approximations do lead to large updates, and therefore do not give any guess of the progress.

3.6.2. Overton's maximum eigenvalue minimization

This subsection is based on the article [**35**] by Overton. We will review the method to minimize the maximum eigenvalue, in magnitude, of an affine real and symmetric matrix function. This method can be applied to the optimization problem (E).

```
(1) input an initial point $y^0 \in \mathbb{R}^m$, a normalized eigenvector
    $v^0$ corresponding to $\lambda_{\max}(C - \mathcal{A}^*(y^0))$, an $\varepsilon$ for termination,
    an improvement number $m_L \in (0, \frac{1}{2})$, a weight $u > 0$,
    an upper bound $R \geq 1$ on the number of columns of $P$.
(2) initialization
      • $k = 0$
      • $x^0 = y^0$
      • $P^0 = v^0$
      • $\overline{W}^0 = v^0(v^0)^*$
(3) direction
      • solve
          max          $a\langle C - \mathcal{A}^*(x^k), W\rangle + \langle b, x^k\rangle - \frac{1}{2u}\langle a\mathcal{A}(W) - b, a\mathcal{A}(W) - b\rangle$
          subject to   $W = \alpha\overline{W}^k + P^k V (P^k)^*$
                       $\alpha + \mathrm{trace}\,(V) = 1$
                       $\alpha \geq 0, \quad V \geq 0$
        which gives you the maximum, $W^*$, $V^*$ and $\alpha^*$
      • determine $y^{k+1} = x^k + \frac{1}{u}(a\mathcal{A}(W^*) - b)$
      • decompose $V^*$ : $V^* = Q_1\Lambda_1 Q_1^* + Q_2\Lambda_2 Q_2^*$ with
        rank$(Q_1) \leq R-1$, and $Q_1$ containing the eigenvectors
        associated to large eigenvalues $\Lambda_1$ of $V^*$ and $Q_2$ the
        remaining eigenvectors associated to the remaining
        eigenvalues of $\Lambda_2$.
      • compute
          $\overline{W}^{k+1} = \alpha^*\overline{W}^k + \frac{P^k Q_2 \Lambda_2 Q_2^* (P^k)^*}{(\alpha^* + \mathrm{trace}\,(\Lambda_2))}.$
(4) evaluation
      • compute $\lambda_{\max}(C - \mathcal{A}^*(y^{k+1}))$ and eigenvector $v^{k+1}$
      • construct the columns of $P^{k+1}$ containing an orthonormal
        basis of the space spanned by $P^k Q_1$ and $v^{k+1}$.
(5) termination If $f(x^k) - \widehat{f}^k(y^{k+1}) \leq \varepsilon$ stop that is,
    $a\lambda_{\max}(C - \mathcal{A}^*(x^k)) + \langle b, x^k\rangle - a\langle C - \mathcal{A}^*(y^{k+1}), W^*\rangle -$
    $\langle b, y^{k+1}\rangle \leq \varepsilon$ stop
(6) serious step
      • If $f(y^{k+1}) \leq f(x^k) - m_L(f(x^k) - \widehat{f}^k(y^{k+1}))$ then set
        $x^k = y^{k+1}$ and goto (7)
      • otherwise goto (6)
(7) null step $x^{k+1} = x^k$
(8) increase $k = k + 1$, return to (2)
```

Algorithm 3.6.6 The spectral bundle method.

Formulation. Suppose we want to minimize the maximum eigenvalue of the affine real and symmetric matrix function $C - \mathcal{A}^*(y)$:

$$\min_{y \in \mathbb{R}^m} \psi(y), \tag{174}$$

where

$$\psi(y) = \max_{i \le n} |\lambda_i\Big(C - \mathcal{A}^*(y)\Big)|.$$

We assume that the eigenvalues λ_i are ordered, $\lambda_1 \ge \cdots \ge \lambda_n$. The problem (174) may be written as the following differentiable optimization problem:

$$\begin{array}{ll} \max & \omega \\ \text{subject to} & -\omega \le \lambda_i\Big(C - \mathcal{A}^*(y)\Big) \le \omega, \quad i = 1, \cdots, n, \end{array} \tag{175}$$

or also,

$$\begin{array}{ll} \min & \omega \\ \text{subject to} & \omega I + C - \mathcal{A}^*(y) \ge 0, \\ & \omega I - C + \mathcal{A}^*(y) \ge 0. \end{array} \tag{176}$$

One goal of the algorithm that will be described here is adjusting the multiplicity of the extreme eigenvalues while obtaining a reduction of the objective $\psi(y)$. A correct value for the multiplicity of the maximum eigenvalue in the optimal solution is required. Write

$$\begin{array}{lll} \min\limits_{\omega \in \mathbb{R}, y \in \mathbb{R}^m} & \omega & \\ \text{subject to} & \lambda_i\Big(C - \mathcal{A}^*(y)\Big) = \omega, & i = 1, \dots t, \\ & \lambda_i\Big(C - \mathcal{A}^*(y)\Big) = -\omega, & i = n - s + 1, \dots n, \end{array} \tag{177}$$

where as a consequence of the minimization process, it is established that $\omega = \max(\lambda_1, -\lambda_n)$, with

$$\omega = \lambda_1 = \cdots = \lambda_t > \lambda_{t+1} \ge \cdots \ge \lambda_{n-s} > \lambda_{n-s+1} = \cdots = \lambda_n = -\omega. \tag{178}$$

Thus t and s are the multiplicities of the maximum and minimum eigenvalue respectively. Let $\{q_1(y), \dots q_n(y)\}$ be a corresponding set of orthonormal eigenvectors to $\lambda_1, \dots \lambda_n$, and let $Q_1 := (q_1, \cdots, q_t)$ and $Q_2 := (q_{n-s+1}, \cdots, q_n)$.

Optimality conditions. The following theorem states the optimality condition for solving (174).

THEOREM 3.6.2. [**35**, Thm. 3.2] *A necessary and sufficient condition for y to solve (174) is that there exist matrices U and V of dimension $t \times t$ and $s \times s$, respectively, with $U = U^* \ge 0$ and $V = V^* \ge 0$, such that*

$$\text{trace}\,(U) + \text{trace}\,(V) \;=\; 1, \tag{179}$$

$$\text{trace}\,((Q_1^* A_k Q_1)U) - \text{trace}\,((Q_2^* A_k Q_2)V) \;=\; 0, \quad k = 1, \dots, m, \tag{180}$$

where t, s, Q_1, Q_2 are defined by (178).

For a proof, see the proof of Theorem 3.2 in [**35**].

The matrices U, V will play the role of Lagrange multipliers as will become clear later. An application of theorem 3.6.2 on problem (177) under the assumption that t and s are known leads to

$$\begin{array}{lll} & \min\limits_{\omega\in\mathbb{R}, y\in\mathbb{R}^m} & \omega \\ (181) & \text{subject to} & \omega I_t - Q_1^*(y)(C-\mathcal{A}^*(y))Q_1(y) = 0, \\ & & \omega I_s + Q_2^*(y)(C-\mathcal{A}^*(y))Q_2(y) = 0. \end{array}$$

We can apply a Newton method to this nonlinear problem (see [**35**, p.261]). Then the appropriate subproblem to solve at each step of the Newton method, is the quadratic program:

$$\begin{array}{lll} & \min\limits_{\omega\in\mathbb{R}, y\in\mathbb{R}^m} & \omega + \frac{1}{2}d^*Wd \\ (182) & \text{subject to} & \omega I_t - Q_1^*(y)(C-\mathcal{A}^*(y+d))Q_1(y) = 0, \\ & & \omega I_s + Q_2^*(y)(C-\mathcal{A}^*(y+d))Q_2(y) = 0. \end{array}$$

where W should be the Hessian, with respect to y, of the Lagrangian of the optimization problem (181),

$$\begin{array}{rcl} \mathcal{L}(\omega, y, U, V) & = & \omega - \langle U, \omega I_t - Q_1^*(y)(C-\mathcal{A}^*(y))Q_1(y)\rangle + \\ & & -\langle V, I_s + Q_2^*(y)(C-\mathcal{A}^*(y))Q_2(y)\rangle, \end{array}$$

where $U=U^*,\ V=V^*$. We see that the first order optimality conditions $\nabla_\omega \mathcal{L}=0$, and $\nabla_y\mathcal{L}=0$, for y to solve (181), are that there exist symmetric matrices U and V such that the equations in Theorem 3.6.2 hold. It can be shown [**35**] that the exact expression for W is the following,

$$W_{jk} = \langle U, G_1^{j,k}\rangle - \langle V, G_2^{j,k}\rangle,$$

with

$$\begin{array}{rcl} G_1^{j,k} & = & 2Q_1(y)^* A_k \overline{Q}_1(y)\Big(\omega I_{n-t} - \Lambda_1(y)\Big)^{-1}\overline{Q}_1(y)^* A_j Q_1(y) \\ G_2^{j,k} & = & -2Q_2(y)^* A_k \overline{Q}_2(y)\Big(\omega I_{n-s} + \Lambda_2(y)\Big)^{-1}\overline{Q}_2(y)^* A_j Q_2(y) \end{array}$$

The columns of $\overline{Q}_i(y)$ consist of all eigenvectors $q_1(y), \ldots, q_n(y)$ that are not columns of $Q_i(y)$ $(i=1,2)$. The matrix $\Lambda_i(y)$ is a diagonal matrix with entries $\lambda_1, \ldots, \lambda_n$ except those corresponding to $Q_i(y)$ $(i=1,2)$.

Since each step of the Newton method we have to solve (182), we need estimates of the matrices U and V, this is for example done by minimizing the 2-norm of the residual of (179), (180). When solving (182) it is likely that the initially largest eigenvalue is reduced way below the others. Therefore, incorporate into (182) the following inequality constraints

$$(183) \qquad -\omega \le q_i^*(y)(C-\mathcal{A}^*(y+d))q_i(y) \le \omega \quad \text{for} \quad t+1 \le i \le n-s,$$

on the other eigenvalues. A reasonable strategy is to increase t by the number of constraints which are at their upper bound, and to increase s by the number at their

iter.	y^*	obj. value
1		6.541381
2		4.817767
3	$[0.0\ 2.0e-15]$	1.000000

TABLE 13 Output for Example 3.6.3 computed by Overton's method, Algorithm 3.6.7, initial $y^* = [1\ \ 2]$. Empty spaces in the table are unknown values.

```
(0) input an initial point $y^0 \in \mathbb{R}^m$, starting multiplication estimates $t, s$,
(1) initial evaluation compute $\lambda_i(y^k)$ and $q_i(y^k)$
(2) direction derivation Solve the optimization problem (182) with (183) and (184). And use some estimates of $U$ and $V$ to define $W$.
      • if (182) becomes infeasible goto (3b)
      • if $\|d\| \leq \varepsilon$ goto (4)
(3) evaluation compute $\lambda_i(y^k + d)$ if $\psi(y^k + d) < \psi(y^k)$
    (3a) increase $t$ and $s$, by the number of upper and lower bounds which are active in (183). and set $y^{k+1} = y^k + d$, double $\rho$, and goto (2)
         else
    (3b) reduce $t$ and $s$ to some acceptable values, and divide $\rho$ by 2 and goto (2)
(4) termination
      • If $U \geq 0$ and $V \geq 0$ then stop
      • else obtain a reduction by adjusting $U$ (cf. [35, Section 5]), goto (2)
```

ALGORITHM 3.6.7 Overton's eigenvalue method.

lower bound. However, some caution should be used, since if s and t become too large problem (181) will become infeasible (see [**35**, p.262] for details).

We insist that a reduction of ω is obtained each step. The reduction is obtained by incorporating into (182) a bound constraint on d,

$$|d_i| \leq \rho, \quad 1 = i, \dots m \tag{184}$$

where ρ is during the iteration process adjusted.

EXAMPLE 3.6.3. This example is a little bit different from the other examples we have seen up till now, since it is not solving a problem of the form (D). We would like to minimize the maximum eigenvalue, in magnitude, of the following function.

$$\min_{y \in \mathbb{R}^m} \max_{1 \leq i \leq n} |\lambda_i\Big(C - \mathcal{A}^*(y)\Big)|,$$

iter.	y^*	obj. value
1	$[4.3 \quad 0.37]$	5.792694
2	$[4.4 \quad -0.63]$	5.047499
3	$[1.0e-15 \quad -2.2e-16]$	1.000000

TABLE 14 Output for Example 3.6.3, initial $y^* = [1 \;\; 2]$.

We take $m = n = 2$ and

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A_1 = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -1 & -3 \\ -3 & -4 \end{pmatrix}.$$

This example is taken from [**35**], and will be used for comparing our method to Overton's approach. We did not have the code of Overton's method, therefore we adjusted our algorithm for this problem for which the minimal maximum eigenvalue in magnitude equals the minimal maximum eigenvalue. The output of Table 14 is from our method. The output of Overton's approach is copied from [**35**] in Table 13.

Overton's eigenvalue method is a quadratic method, it inherits quadratic convergence as can be observed from Table 13. This table shows the solution process for Example 3.6.3. Our method does also have fast convergence for the same example as is visible in Table 14. A drawback of Overton's eigenvalue method is that it needs beforehand information of the multiplicity of the minimal maximum eigenvalue. It is also, in this form, not applicable to large-scale optimization problems, especially the computation of a complete eigenvalue decomposition every step of the algorithms is computationally too expensive. However, it should be noted that is was never designed to solve large scale optimization problems.

3.7. Discussion and conclusions

In this chapter we created a new method for solving semidefinite programming problems, by means of an eigenvalue method. The semidefinite constraint is lifted into the objective function, when written as an eigenvalue constraint. The method is based on a second order approximation of the maximum eigenvalue function. It can handle both simple and multiple maximum eigenvalues. The method should especially be interesting for large semidefinite programming problems, that is, if the problem contains large matrix variables or a huge number of constraints for which classical interior-point methods, see Section 2.2, grow very slow and consume a huge amount of memory. We discussed in Subsection 2.2.1, the primal-dual interior-point method for SDP. This method requires in each iteration the factorization of the dense matrix $\mathbf{AS^{-1}XA}^*$ in system (81), of size equal to the number of constraints, and it requires one to three factorizations of the positive semidefinite matrix variables, to find an appropriate step size, see Section 2.2.1. Therefore, if the amount m of matrices is larger than 5000 and if the size of the matrices A_i is larger than $n > 500$, SDP problems cannot be solved with the primal-dual interior-point method on a typical workstation in reasonable computing time. The method we developed in Chapter 3 uses a subspace procedure, which reduces the size of the matrix variables dramatically. As a consequence for the reduced problems, all computations can be done without restriction on the memory or computation time when setting the subspace size to a small number. The most expensive part of Algorithm 3.5.5 is the computation of a small amount, say s, of eigenvalues every iteration. The value of s is considerably smaller than m, the amount of constraints, and n, the size of the matrix variable in the semidefinite programming problem. By using an appropriate iterative eigensolver for the computation of the eigenvalues we expect a large gain in the computing time and for memory requirements. In Tables 2, 5, and 14 we have seen that our algorithm inherits fast convergence. With the subspace procedure we lost the fast convergence, which was to be expected, since we lost information by restricting the optimization problem to a low dimensional subspace.

In Section 3.6 we described two other approaches for solving semidefinite programming problems incorporating eigenvalues: the spectral bundle method and Overton's eigenvalue method. The difference between the spectral bundle method and our method is that the spectral bundle method is a linear method and has therefore no fast convergence properties. It also needs the correct value of the Lagrange multiplier beforehand. It therefore restricts the set of semidefinite programming problems (to semidefinite programming problems with a fixed trace of the primal matrix variable X, see [**20**]). The most attractive part of the spectral bundle method is the use of a sophisticated subspace procedure, this is in contrast to the rather basic one that we used in Algorithm 3.5.5. More research is needed to develop a sophisticated subspace procedure for Algorithm 3.5.5, in order to get a less tailing off effect, that should give a reductions in the total amount of iterations used to solve the semidefinite programming problems.

The other method described in Section 3.6 is Overton's eigenvalue method, this method is a quadratic method, but is not applicable to large scale semidefinite programming problems, since it uses a full eigenvalue decomposition every iteration step and is therefore not less expensive than the classical interior-point methods. Another problem is that it needs the multiplicity of the maximum eigenvalue of the solution of the semidefinite programming problem beforehand.

Concluding, we have developed a promising method with 'quadratic' convergence properties. Further work should be done in the following directions to make the proposed method more efficient and applicable to large instances.

In the first place the algorithm should be implemented in a stable manner and sparse form, so that it can handle large inputs.

At each iteration step of our approach, for the simple eigenvalue case we need to select one direction out of the two directions, and for the multiple eigenvalue case one out of the three directions. Our method to select the direction, described in Subsection 3.2.4 and 3.3.4, does work fine in our tests, but further research is necessary. For example, a two-dimensional search to find the optimal combination of two directions could be an interesting option.

Further investigations into an efficient subspace procedure is needed, for example, to collect 'interesting' information from previous iterates, like the one described for the spectral bundle method in Subsection 3.6.1.

Another point of investigation is the iterative eigensolver. The eigensolver should be tuned for the matrices that appear during the iteration process. For example, we can consider starting vectors for the eigensolver from previous iterates or construct a preconditioner for structured matrices appearing in the process, designed for the specific eigensolver.

Bibliography

[1] F. Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal on Optimization **5** (1995), no. 1, 13–51.

[2] F. Alizadeh, J.-P. Haeberly, and M. L. Overton, *Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results*, SIAM Journal on Optimization **8** (1998), 746–768.

[3] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math. **9** (1951), 17–29.

[4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. A. van der Vorst, *Templates for the solution of algebraic eigenvalue problem: A practical guide*, SIAM Publications, 2000.

[5] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, C. Romine, and H. A. van der Vorst, *Templates for the solution of linear systems : Building blocks for iterative methods*, SIAM Publications, 1993.

[6] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*, SIAM Studies in Applied Mathematics, vol. 15, Society of Industrial and Applied Mathematics (SIAM), Philadelphia, PA 19101, USA, 1994.

[7] S. Boyd and L. Vandenberghe, *Convex optimization*, Course Reader Nonlinear Programming 1997-8, UCLA, 1997.

[8] G. B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, N.J., 1963.

[9] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Numerical linear algebra for high-performance computers*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[10] D. R. Fokkema, *Subspace methods for linear, nonlinear, and eigen problems*, Ph.D. thesis, Mathematical Institute, Utrecht University, October 1996.

[11] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Accelerated inexact Newton schemes for large systems of nonlinear equations*, SIAM Journal on Scientific Computing **19** (1998), no. 2, 657–674.

[12] R. W. Freund and F. Jarre, *A QMR–based interior–point algorithm for solving linear programs*, Mathematical Programming **76** (1997), 183–210.

[13] M. R. Garey and D. S. Johnson, *Computers and intractability*, W. H. Freeman and Co., San Francisco, Calif., 1979, A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[14] A. George and J. W. H. Liu, *The evolution of the minimum degree ordering algorithm*, SIAM Review **31** (1989), no. 1, 1–19.

[15] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM **42** (1995), no. 6, 1115–1145.

[16] S. Goldberg and I. Gohberg, *Basic operator theory*, Birkhauser Boston, 1980.

[17] G. H. Golub and C. F. Van Loan, *Matrix computations*, third ed., Johns Hopkins Studies in the Mathematical Sciences, The Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[18] C. Helmberg, *Semidefinite programming for combinatorial optimization*, Habilitations-schrift 00-34, Konrad–Zuse Zentrum für Informationstechnik Berlin, D–14195 Berlin–Dahlem, Germany, October 2000.

[19] C. Helmberg and K. C. Kiwiel, *A spectral bundle method with bound*, ZIB Preprint SC–99–37, Konrad–Zuse Zentrum für Informationstechnik Berlin, D–14195 Berlin–Dahlem, Germany, December 1999.

[20] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM Journal on Optimization **10** (2000), no. 3, 673–696 (electronic).

[21] D. den Hertog, *Interior point approach to linear, quadratic and convex programming*, Mathematics and Its Applications, Kluwer Academic Publishers, Netherlands, 1994.

[22] B. Jansen, C. Roos, and T. Terlaky, *Interior point methods, a decade after Karmarkar—a survey, with application to the smallest eigenvalue problem*, Statist. Neerlandica **50** (1996), no. 1, 146–170.

[23] F. Jarre and M. Wechs, *Extending Mehrotra's corrector for linear programs*, Preprint 219, Mathematische Institute der Universität Würzburg, Am Hubland, D–97074 Würzburg, Germany, December 1996, (Revised October 1997).

[24] N. K. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica (1984), no. 4, 373–395.

[25] T. Kato, *Perturbation theory for linear operators*, Grundlehren der mathematischen Wissenschaften ; 132, Springer, Berlin, Germany, 1980 2nd ed.

[26] E. de Klerk, *Interior point methods for semidefinite programming*, Ph.D. thesis, Faculty of Technical Mathematics and Informatics, TU Delft, NL–2600 GA Delft, The Netherlands, 1997.

[27] J. L. Lagrange, *Mećanique analytique*, Paris, France, 1788.

[28] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Research Nat. Bur. Standards **45** (1950), 255–282.

[29] S. Mehrotra, *On the implementation of a primal–dual interior point method*, SIAM Journal on Optimization **2** (1992), no. 4, 575–601.

[30] R. D. C. Monteiro, *Primal-dual path-following algorithms for semidefinite programming*, SIAM Journal on Optimization **7** (1997), no. 3, 663–678.

[31] Yu. E. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.

[32] Yu. E. Nesterov and M. J. Todd, *Primal-dual interior-point methods for self-scaled cones*, SIAM Journal on Optimization **8** (1998), no. 2, 324–364 (electronic).

[33] S. Oliviera, D. Stewart, and T. Soma, *Solving semidefinite programming in subspace*, Technical report, University of Iowa, December 2001.

[34] F. Oustry, *A second-order bundle method to minimize the maximum eigenvalue function*, Mathematical Programming **89** (2000), no. 1, Ser. A, 1–33.

[35] M.L. Overton, *On minimizing the maximum eigenvalue of a symmetric matrix*, SIAM Journal on Matrix Analysis and Applications **9** (1988), 256–268.

[36] B. N. Parlett, *The symmetric eigenvalue problem*, Prentice-Hall Series in Computational Mathematics, Prentice–Hall, Englewood Cliffs, N.J. 07632, 1980, Reissued with revisions by SIAM, Philadelphia, 1998.

[37] J. Renegar, *A mathematical view of interior-point methods in convex optimization*, School of Operations Research and Industrial Engineering, Cornell University, June 1999, Unpublished.

[38] C. Roos, T. Terlaky, and J.-Ph. Vial, *Theory and algorithms for linear optimization*, first ed., John Wiley and Sons Ltd,, Chichester, England, 1997.

[39] R. Saigal, L. Vandenberghe, and H. Wolkowicz (eds.), *Handbook on semidefinite programming and applications*, first ed., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[40] H. H. Schaefer, *Topological vector spaces*, first ed., Macmillan Series in Advanced Mathematics and Theoretical Physics, The Macmillan Company, New York, 1966.

[41] M. Shida, S. Shindoh, and M. Kojima, *Existence of search directions in interior-point algorithms for the sdp and the monotone sdlcp*, Research Reports on Information Sciences, Ser. B : Operations Research B–310, Department of Information Sciences, Tokyo Institut e of Technology, Oh–Okayama, Meguro–ku, Tokyo 152, Japan, 1996.

[42] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM Journal on Matrix Analysis and Applications **17** (1996), no. 2, 401–425.

[43] G. W. Stewart, *An iterative method for solving linear inequalities*, Technical Report CS-TR-1833, University of Maryland, College Park, April 1987.

[44] J. F. Sturm, *Primal–dual interior point approach to semidefinite programming*, Ph.D. thesis, Tinbergen Institute, Amsterdam, The Netherlands, 1997.

[45] K.-C. Toh and L. N. Trefethen, *The Chebyshev polynomials of a matrix*, SIAM Journal on Matrix Analysis and Applications **20** (1999), no. 2, 400–419 (electronic).

[46] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review **38** (1996), no. 1, 49–95.

[47] L. Vandenberghe and S. Boyd, *Applications of semidefinite programming*, Applied Numerical Mathematics: Transactions of IMACS **29** (1999), no. 3, 283–299.

[48] L. A. Wolsey, *Integer programming*, first ed., John Wiley and Sons Ltd., New York, USA, 1998.

[49] S. J. Wright, *Primal-dual interior-point methods*, first ed., SIAM, Philadelphia, PA, 1997.

Index

Samenvatting

Het belangrijkste onderwerp van dit proefschrift is semidefiniete optimalisatie. Semidefiniete optimalisatie bestaat uit het analyseren en oplossen van optimalisatiemodellen die een ingewikkeldere structuur hebben dan de algemeen bekende lineaire programmeringsproblemen (LP). Semidefiniete optimalisatieproblemen (SDP) komen bijvoorbeeld voort uit de wereld van de telecommunicatie, logistiek, elektrotechniek en constructie-bouw. Stel dat er een brug moet worden gebouwd. Er volgt een ontwerp en men wil graag weten welk deel van de constructie dikke balken nodig heeft en hoe de kabels zo gespannen kunnen worden dat er zo min mogelijk staal nodig is. Maar de brug moet ook heel blijven in geval van storm, sterke stroming van de rivier en tijdens de spits. Het mathematisch model van zo'n praktijkprobleem bestaat over het algemeen uit een grote hoeveelheid variabelen en is daardoor niet binnen redelijke tijd met een computer door te rekenen. Voor dit soort problemen is een efficïente *oplosmethode* nodig, ook wel algoritme genoemd. In dit proefschrift wordt een algoritme gepresenteerd dat toepasbaar en uitbreidbaar is voor grootschalige SDP. Het basisprincipe bestaat uit het verkleinen van het SDP, zodanig dat er niet te veel informatie verloren gaat en binnen redelijke tijd een benaderde oplossing kan worden berekend. Het hier gepresenteerde algoritme is vrij eenvoudig en doorstaat tests met veelbelovend resultaat.

In hoofdstuk 1, sectie 1.1 wordt basis lineaire algebra en analyse beschreven. In sectie 1.2 wordt een algemeen raamwerk geschetst voor convexe optimalisatieproblemen. Vanuit dit raamwerk worden LP en SDP beschreven. Hierna komen de optimaliteitscondities aan bod en wordt er afgesloten met een paar voorbeelden van SDP.

In het tweede hoofdstuk wordt gekeken naar bestaande technieken om convexe optimalisatieproblemen op te lossen. Deze technieken worden eerst weer beschreven binnen het algemene raamwerk en daarna gelijk toegepast op LP in sectie 2.1. In sectie 2.2 komen de technieken voor SDP aan bod. De technieken zijn Newton methoden en heten binnen de optimalisatiegemeenschap inwendige-puntmethoden. Sectie 2.3 gaat verder in op de implementatie van inwendige-puntmethoden en in sectie 2.4 wordt een voorstel gedaan om SDP te projecteren op kleinschalige SDP. Deze sectie opent de weg naar de projectie van SDP op eigenruimten. Eigenruimten zullen in hoofdstuk 3 een belangrijke rol spelen.

Sectie 3.1 bevat de introductie van een eigenwaardeformulering van een SDP. Verder wordt een relevante Lagrange multiplier bepaald en worden afgeleiden van eigenwaardefuncties geïntroduceerd. In sectie 3.2 wordt een methode beschreven om de maximale eigenwaarde te minimaliseren van sommen van symmetrische matrices, als deze eigenwaarde enkelvoudig is. De methode wordt geïllustreerd aan de hand van voorbeelden. Sectie 3.3 bevat de uitbreiding van laatstgenoemde methode voor een meervoudige eigenwaarde. Ook deze aanpak wordt geïllustreerd door voorbeelden op te lossen. Sectie 3.4 beschrijft de meest gebruikelijke oplosmethoden voor eigenwaardeproblemen. Deze zullen worden toegepast in sectie 3.5, waarin tevens een deelruimteprocedure beschreven wordt die bedoeld is voor de methoden van de secties 3.2 en 3.3. Conclusies over de afmeting van de deelruimten, een beschrijving van twee bestaande methoden om de maximale eigenwaarde te minimaliseren en de vergelijking met de nieuwe methode komen hier ook aan bod.

In hoofdstuk 3 is een aanpak geschetst om SDP efficiënt op te lossen met behulp van een eigenwaardemethode. Deze methode bezit de mooie eigenschap dat hij "sneller" is dan de gebruikelijke primal-dual inwendige-puntmethode voor SDP. De primal-dual inwendige-puntmethode, beschreven in hoofdstuk 2, factoriseert namelijk iedere iteratiestap een matrix met een afmeting die gelijk is aan het aantal constraints. Tevens benut de inwendige-puntmethode één tot drie factorisaties van matrices ter grootte van de inputmatrices, om een geschikte stapgrootte te bepalen. Als het aantal constraints groter is dan 5000 en de afmeting van de inputmatrices groter is dan 500 kunnen de SDP niet meer in een redelijke tijd worden opgelost met behulp van de primal-dual inwendige-puntmethode. De eigenwaardemethode maakt daarentegen gebruik van een deelruimteproces dat de afmetingen van het SDP probleem drastisch reduceert. Voor het gereduceerde probleem kunnen alle operaties in een korte tijd worden uitgevoerd. Het meeste werk van deze aanpak is dat er iedere iteratiestap s eigenwaarden en eigenvectoren berekend moeten worden. In het algemeen zal s een stuk kleiner zijn dan het aantal constraints. Daarom is er een grote tijdwinst ten opzichte van de primal-dual inwendige-puntmethode te verwachten, als gebruik gemaakt wordt van een geschikte oplosmethode voor eigenwaardeproblemen. Het nadeel van de deelruimteaanpak is dat vrijwel zeker snelle convergentie verloren gaat.

Dit proefschrift leidt tot de volgende conclusies en aanbevelingen. De beschreven methode kan snel convergentie-gedrag vertonen, maar verliest dit gedrag als deelruimteprocedures worden toegevoegd. Er moet nader gekeken worden naar het keuzeproces voor een geschikte zoekrichting. Bovendien is nader onderzoek nodig naar efficiëntere deelruimteprocedures en moeten iteratieve methoden voor eigenwaardeberekening verder worden aangepast voor de specifieke gestructureerde matrices die voorkomen in SDP toepassingen. Daarbij moet gedacht worden aan startvectoren en preconditionering. Verder zal de ontwikkelde methode goed moeten worden geïmplementeerd, zodat hij direct toepasbaar is op grootschalige SDP.

Dankwoord

Geen enkel proefschrift wordt afgerond zonder de hulp van begeleiders en collega's. Als eerste wil ik graag mijn co-promotor Gerard Sleijpen bedanken voor al de tijd die hij discussierend en verbeterend met mij heeft moeten doorbrengen. Zonder zijn support geen proefschrift. Mijn promotor Henk van der Vorst, hem wil ik bedanken voor de ruimte die hij mij vier jaar heeft geboden om dit onderzoek te kunnen verrichten. De leescommissie, K. Aardal, E. Balder, R. Mattheij en C. Roos, voor de tijd die ze aan de beoordeling van dit proefschrift hebben besteed. Wat betreft het luisteren naar mijn voordrachten en het aanvullen en verbeteren van mijn hersenspinsels bedank ik de gehele numerieke sectie voor hun ondersteuning.

De reden dat ik in die vier jaar niet ben opgelost in het stof op mijn bureau is voornamelijk te danken aan de altijd goed gemutste collega's van de zevende verdieping. Daarbij denk ik in het bijzonder aan: Arno, Barbara, Bob, Daan, Lennaert, Luis, Martijn, Menno en Theo, hoewel Barbara voornamelijk de zesde onveilig maakt en Daan het KNMI. De koffiekamerbezetting op de zevende voor de gezellige tussenstops.

De volgende reeks is er verantwoordelijk voor dat ik buiten wiskundige zaken ook nog bleef functioneren. Mike en Marijke bedankt voor jullie 29 jaar durende steun. Emiel en Len, bedankt dat jullie willen paranimfen. Simone bedankt voor je onuitputtelijke steun en je lieve woordjes. Hopelijk ben ik vanaf het ter perse gaan van dit proefschrift weer te genieten. Tot besluit wil ik alle vrienden en familie bedanken, die buiten bovengenoemde opsomming vallen en zeker niet vergeten mogen worden.

Curriculum Vitae

Mischja van Bossum werd geboren op 13 november 1972 in Zaandam. Vanaf 1985 bezocht hij het Sint Michaël College te Zaandam, waar hij in 1991 het diploma Atheneum-B behaalde. Aansluitend studeerde hij wiskunde aan de Vrije Universiteit in Amsterdam, waar hij zich specialiseerde in de algebra, Lie en Hopf algebra's bij prof. dr. E.J. Ditters. In 1997 behaalde hij zijn doctoraal diploma wiskunde.

In februari 1998 kwam hij in dienst van de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) om als Onderzoeker in Opleiding te werken aan de Universiteit Utrecht. Binnen het project High Performance Methods for Mathematical Optimization van het NWO verrichtte hij wetenschappelijk onderzoek onder leiding van prof. dr. H.A. van der Vorst en onder directe begeleiding van dr. G.L.G. Sleijpen. Dit proefschrift is het resultaat van het onderzoek.

In het kader van zijn aanstelling organiseerde hij in 2001 de conferentie High Performance Optimization Techniques 6 te Utrecht.

Mischja van Bossum was born on november the 13th 1972 in Zaandam. From 1985 he visited the Sint Michaël College in Zaandam, where he graduated in 1991. In 1991 he started studying mathematics at the Vrije Universiteit in Amsterdam, where he specialized in Lie and Hopf algebras under supervision of prof. dr. E.J. Ditters. In 1997 he received his Master's degree in mathematics.

In February 1998 he got a PhD position financed by "de Nederlandse Organisatie voor Wetenschappelijk Onderzoek" at Utrecht University. Within the project High Performance Methods for Mathematical Optimization he did scientific research under supervision of prof. dr. H.A. van der Vorst and under direct supervision of dr. G.L.G. Sleijpen. This thesis contains the results of his research.

During his stay at Utrecht University he organized the international conference High Performance Optimization Techniques 6 in Utrecht.